Efficient and Invariant Regularisation with Application to Computer Graphics

A thesis submitted to the School of Information Technology and Electrical Engineering, the University of Queensland, for the degree of Doctor of Philosophy.

Christian James Walder, B.E. (elec) Hons. January 2008

Supervisors: Brian Lovell, Bernhard Schölkopf, Olivier Chapelle
Examiners: Stéphane Canu, Hans-Peter Seidel

Except where acknowledged in the customary manner, the material presented in this thesis is to the best of my knowledge original and has not been submitted in whole or part for a degree in any university.

Signed:

Dated:

Acknowledgements

This thesis and the time spent developing it have benefited from the support of many kind and inspirational people. First I would like to thank my advisory team: Brian Lovell, Olivier Chapelle and Bernhard Schölkopf. I received help on many levels, including concrete scientific input, strategic direction and inspiration, and the freedom to pursue my own interests.

The initial time I spent at the beautiful University of Queensland St Lucia campus was a pleasure. It is crucial to be excited about ones PhD topic, and for me this excitement was mainly sparked by the people at the University of Queensland. In particular I would like to give praise to my fellow students at the time: David Mackinnon, Ben Appleton, Emanuel Zelniker, Richard Davis, Simon Clode, Carlos Leung, Nianjun Liu, Shaokang Chen and Stefan Lehmann, as well as the more senior Peter Kootsookos, Vaughan Clarkson and Kurt Kubik.

Continuing chronologically, I am indebted to Brian and Kurt for helping to arrange what turned out to be a life changing visit to the Fachhochschule für Technik in Stuttgart, where Michael Hahn helped me to find my feet and also to get some first lecturing experience.

A quantum leap in my knowledge and understanding occurred when Bernhard Schölkopf kindly allowed me to make a lengthy visit to his group at the Max Planck Institute in Tübingen. The people in the group, as well as the constant stream of strong visitors who pass through it, help to create a stimulating atmosphere in which one cannot help but absorb ideas, practically by osmosis. Those responsible are too numerous to mention so I ask to be excused for merely drawing particular attention to Jan Eichhorn, Florian Steinke, Arthur Gretton, Wolf Kienzle, Kwang In Kim, Matthias Hein, Valentin Schwamberger (thanks for the LaTeX help!), Jakob Macke, Mingrui Wu, Cheng Soon Ong, Thomas Navin Lal, Carl Rasmussen, Matthias Seeger and Gunnar Rätsch.

The late Charles Bennett can take most of the credit for my funding, thanks to the bequest left by him to the University of Queensland, who then passed some of it on to me. I also thank the University of Queensland for an additional and generous scholarship which allowed me to travel to Germany. Research is highly valued and very well funded in Germany. The Deutsche

Akademischer Austausch Dienst kindly funded my stay for the first year or so in Germany. The Max Planck Society subsequently funded my last half a year or so, and also covered various work related trips — including my attendence of the 2004 Machine Learning Summer School on Berder Island. Finally, in the context of this paragraph it would be negligent to omit my generous and supportive parents who — in addition to their unconditional love and support — also provided me with timely aid of a more tangible nature.

Abstract

This thesis develops the theory and practise of reproducing kernel methods. Many functional inverse problems which arise in, for example, machine learning and computer graphics, have been treated with practical success using methods based on a reproducing kernel Hilbert space perspective. This perspective is often theoretically convenient, in that many functional analysis problems reduce to linear algebra problems in these spaces. Somewhat more complex is the case of conditionally positive definite kernels, and we provide an introduction to both cases, deriving in a particularly elementary manner some key results for the conditionally positive definite case.

A common complaint of the practitioner is the long running time of these kernel based algorithms. We provide novel ways of alleviating these problems by essentially using a non-standard function basis which yields computational advantages. That said, by doing so we must also forego the aforementioned theoretical conveniences, and hence need some additional analysis which we provide in order to make the approach practicable. We demonstrate that the method leads to state of the art performance on the problem of surface reconstruction from points.

We also provide some analysis of kernels invariant to transformations such as translation and dilation, and show that this indicates the value of learning algorithms which use conditionally positive definite kernels. Correspondingly, we provide a few approaches for making such algorithms practicable. We do this either by modifying the kernel, or directly solving problems with conditionally positive definite kernels, which had previously only been solved with positive definite kernels. We demonstrate the advantage of this approach, in particular by attaining state of the art classification performance with only one free parameter.

Contents

Chapter 1

Introduction

This thesis is most generally concerned with the regularised solution of functional inverse problems — the estimation of a function based on some observational data. Here regularisation refers to the controlling of the complexity of the solution, in order to obtain what in everyday language might be referred to as a smoother or in some sense simpler solution. Regularisation is not a new concept — one could well interpret the act of a physicist proposing a mathematical model of nature as an example of it. In this case, the physicist might prefer, say, polynomials of the lowest possible order, provided that they are flexible enough to explain the data at hand.

One example scenario in which regularisation is often benificial is that of regression — the problem of inferring from a set of pairs $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$ a function which describes the underlying relationship. In this thesis we will consider two slightly different settings:

1. Supervised learning — in this case the y_i are binary class labels rather than real numbers. For example, the $\boldsymbol{x}_i$ could represent data pertaining to a particular patient and the y_i whether that patient responded well to a certain medication.

2. Surface fitting, in which we are not provided with the y_i's, but only the $\boldsymbol{x}_i$'s. Here we wish to infer a function, the zero level set of which approximates well the $\boldsymbol{x}_i$. This is useful in computer graphics problems in which one wishes to reconstruct a surface from a set of point samples.

It is always necessary to make assumptions in order to deal with such problems. The methodology of Bayesian inference is particularly lucid in this respect — after we have observed an event E the probability of our hypothesis H being true is given by

$$p(H) = \frac{p(E|H)p(H)}{p(E)}, \tag{1.1}$$

which already assumes that we know the *a priori* probability of our hypothesis, $p(H)$. In the present thesis will be inferring functions, and so the hypothesis H will correspond to the choice of a particular function, and $p(H)$ will implement the regularisation we have already referred to, by taking on a smaller value for less smooth functions. In this functional estimation setting, $p(H)$ is a what we shall refer to as a *regularisation functional*.

Kernel methods have arisen as a powerful tool for such problems. It turns out that choosing regularisation functionals related to the reproducing kernel Hilbert space (r.k.h.s.) norm of the function is often convenient. In many cases it allows seemingly difficult functional analysis problems to be converted into simple finite dimensional linear algebra problems. Moreover, these norms often lead to sensible priors distributions over functions, as evidenced by good practical performance on a wide range of problems. This thesis is heavily rooted in the study of kernel methods.

In one of the main components the thesis we derive efficient approximations to kernel methods. This addresses the common complaint that kernel based algorithms are computationally expensive — typically leading to cubic time complexity in the number of data points. On the other hand such methods tend to scale comparably well with the dimensionality of the space of interest. To facilitate application to computer graphics problems involving many points in few dimensions, we propose alternatives which behave somewhat conversely to the computational complexity just described.

Additionally, we consider the invariance of kernel methods to such transformation groups as rotation and dilation. In some problems it is reasonable to assume that a particular function is no more or less likely than, say, a rotated copy of it. We provide some basic analysis into the nature of kernel methods which lead to these properties. We then focus on a specific combination of invariances not previously considered in machine learning algorithms, namely algorithms which are at once rotation, translation and dilation invariant. Such algorithms have useful practical properties, not the least of which being that one may often dispense with certain degrees of freedom which would otherwise be manifested as so-called free parameters of the algorithm.

How to read this Thesis

Writing a thesis inevitably involves a trade off between being fully comprehensibile to the relatively uninitiated, and succinct enough to avoid boring more advanced readers. In an attempt at compromise we have tried to avoid excess verbiage, while providing pointers to further reading where appropriate. As usual these pointers come in the form of the references to the relevant publications. Italicised keywords also provide starting points for exploration

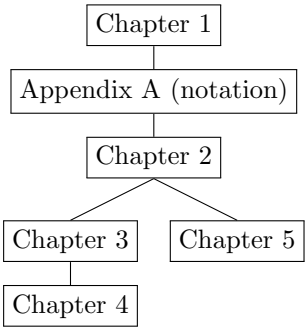


Figure 1.1: Chapter dependencies.

into broader areas.

We now provide a minimal overview of the material contained in each Chapter. A diagram of the dependencies between chapters is given in Figure 1.1. Before proceeding, we would like to note that it may be wise of the reader to peruse the mathematical notation listed in section A.1 before reading the subsequent chapters. The abbreviations on the other hand will be (re)introduced the first time they are used in each new chapter.

Chapter 2

Here we provide a semi-formal introduction to the concept of ill-posedness, taking the solution of a linear system as an example. Next we consider positive definite (p.d.) kernel methods, in particular giving three ways of arriving at the algorithm known as kernel ridge regression (k.r.r.), and providing a novel generalisation of the representer theorem 2.2.7. We then turn to methods based on conditionally positive definite (c.p.d.) kernels, providing what we believe to be a particularly simple way of understanding the main ideas. The chapter culminates with a novel elementary proof of the representer theorem for the c.p.d. case, theorem 2.2.7.

Chapter 3

We develop ways of approximating kernel methods when the number of points is large and the dimensionality relatively low. In particular we show in example 3.1.4 how to analytically compute the thin-plate spline norm on a multi-scale compactly supported function basis. We also give expressions more useful for higher dimensional machine learning problems — that is, the inner product between two Gaussian functions of arbitrary dilation, in the r.k.h.s. of another arbitrary Gaussian kernel, as per theorem 3.1.2. This allows the best approximation of a normal Gaussian kernel based algorithm to be made within the span of a multi-scale Gaussian function basis.

Chapter 4

This Chapter applies the fast approximation scheme of the previous chapter to the problem of implicit surface reconstruction. We provide three algorithms. The first is a simple generalisation of the support vector machine (s.v.m.) which takes surface points as well as additional labelled interior/exterior points as input. The second is a state of the art method which uses surface point/normal pairs as input. The third method requires only a sampling of the surface, which as it turns out makes the problem significantly more difficult.

Chapter 5

Here we begin by discussing the notion of transformation invariance, and providing in theorem 5.2.1 the necessary and sufficient conditions for a p.d. kernel to enjoy such invariances. We show that there exist no p.d. kernels (but rather only c.p.d. ones) which are translation, rotation and dilation invariant. We demonstrate that the thin-plate spline enjoys these invariances, and set about applying it in one setting where it has not already been applied, namely the s.v.m. classifier. We provide two main options for doing this. The first is to construct p.d. kernels from c.p.d. ones using theorem 5.4.2. The second is Algorithm C.1, which solves the s.v.m. optimisation problem for the c.p.d. case. This second option leads to a classification algorithm with only one regularisation parameter which as we demonstrate, performs as well on a real problems as the widely used Gaussian kernel method, which has an additional length scale parameter.

Chapter 2

Background Theory

This introductory chapter consists of mostly non original material, with the few exceptions noted in the text. Our main goals in this chapter are to

1. Make the remainder of the thesis reasonably self contained by providing the necessary background theory on the solution of ill-posed problems, kernel methods and implicit surface reconstruction.
2. Provide a common framework in which to understand the theory and algorithms which we present later on — in particular we will motivate Tikhonov regularisation probabilistically as a *maximum a posteriori* (m.a.p.) estimate.

The beginning of the chapter also aims to provide some background on and motivation for the study of machine learning, which we touched on in the introduction. Although difficult to define in general, the component of the machine learning field with which we are mainly concerned is that of inductive inference based on observational data. The idea is most easily conveyed by an example, and so it is worth reiterating now in greater detail the one we gave in the introductory chapter.

Let's say that a particular drug can have either a negative or a positive effect on those who take it, depending on their genetic makeup. We would like to determine whether the drug is likely to have a beneficial effect on a new patient. A machine learning approach to this problem would involve first measuring and recording information about the genetic makeup of all of those patients who have previously been administered with the drug, along with whether or not it was benificial in each case. Then, machine learning tools can be applied to determine the likelihood of success on a new patient, given similar measurements of his or her genetic information, along with all of the historical data. Clearly machine learning is closely related to the field of statistics. The name comes from the fact that one aims to create machines which can automate the *scientific method*, and hence achieve what would be referred to by some as *learning*.

2.1 Ill Posed Problems

The machine learning field is often concerned with ill-posed problems — problems that are not *well-posed* [TA77]. We shall improve on this statement directly, but for clarity we begin with the loose but widely used

Definition 2.1.1. A **well posed problem** is one for which

1. A solution exists.
2. The solution is unique.
3. The solution depends continuously on the data in some reasonable topology.

The study of ill-posed problems may seem unworthy of attention — after all it is not even clear what it could mean to "solve" them. Nonetheless, since the present thesis is largely concerned with such problems, we begin by motivating their study.

One clear motivation is the fact that any task which aims to model the physical world is inherently ill-posed — every phenomenon can be explained by more than one model. Given that this is the case, how can we prefer a particular one? Naturally one can only give such a preference according to some criterion, for example simplicity or usefulness. This necessity for making *prior assumptions* is an unavoidable property of inference tasks in general, which manifests itself in the idea of *Occam's razor* [RG01], and the work of Popper for example, who prefers scientific theories which are more easily falsified by counter evidence [Pop68]. As we mentioned in chapter 1, this is also well summarised by (1.1).

So, although we began by indicating that machine learning is often concerned with the study of ill-posed problems, it is already apparent that this is a category into which all of the physical sciences fall. It is therefore more precise to say that machine learning is the study of *automated* methods of dealing with such problems. Roughly speaking, the physical sciences propose solutions to ill-posed problems whereas machine learning studies the act of doing so. It is merely in this sense that machine learning has been referred to as *the science of sciences* by Vladimir Vapnik. Next, we become more specific about how one may go about dealing with an ill-posed problem, beginning with a simple example.

2.1.1 Solving a Linear System

The material in this section not only provides a concrete illustration of the concept of ill-posedness, but will also eventually lead us to the kernel methods which are central to this thesis. Consider the problem of finding $\boldsymbol{w}$ such that

$$X\boldsymbol{w} = \boldsymbol{y} \tag{2.1}$$

where $X \in \mathbb{R}^{m\times n}$ and $\boldsymbol{w}, \boldsymbol{y}$ are vectors of appropriate dimension. In general this is an ill-posed problem. Depending on the nature of X and $\boldsymbol{y}$, this problem may have infinitely many solutions, a unique one, or none at all. Let us consider the first possibility, which can occur if X is square and singular, *e.g.*

$$X = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \boldsymbol{y} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{2.2}$$

In this case (2.1) is satisfied for all $\boldsymbol{w} \in \left\{ \begin{pmatrix} w \\ -w \end{pmatrix} : w \in \mathbb{R} \right\}$, demonstrating the ill-posedness of the problem. As a first step in dealing with this ill-posedness, we begin by constructing a problem which for the X and $\boldsymbol{w}$ of (2.2) yields a unique solution which satisfies (2.1). In particular, we take the *minimum-norm* solution

$$\arg \min_{\boldsymbol{w}\in\{\boldsymbol{v}:X\boldsymbol{v}=\boldsymbol{y}\}} \|\boldsymbol{w}\|_{\mathbb{R}^n}, \tag{2.3}$$

which which can be easily calculated by imposing stationarity, and in the case of our example is equal to the zero vector. Now consider the more problematic case $m > n$, so that in general no solution exists. It may be natural to choose the *least squares* solution

$$\arg \min_{\boldsymbol{w}\in\mathbb{R}^n} \|X\boldsymbol{w} - \boldsymbol{y}\|_{\mathbb{R}^m} = \left(X^\top X\right)^{-1} X^\top \boldsymbol{y}, \tag{2.4}$$

since if a unique solution to the original problem exists then it is given by (2.4), and if not then $X\boldsymbol{w}$ is at least close to $\boldsymbol{y}$ in some sense. Of course, if infinitely many solutions to (2.4) exist then we could construct a well posed problem to find one by using *e.g.* a similar minimum-norm approach as we applied to (2.1).

In the case where, for a given X and $\boldsymbol{y}$, no solution to (2.1) exists (such a situation could arise in practice due to *e.g.* measurement noise), then (2.4) can be interpreted probabilistically. Supposing that the elements of $\boldsymbol{y}$ are independent and identically distributed (i.i.d.) according to

$$[\boldsymbol{y}]_i \,|\, [X]_{i,:}, \boldsymbol{w} \sim \mathcal{N}([X]_{i,:}\,\boldsymbol{w}, \sigma_y^2), \tag{2.5}$$

then

$$p(\boldsymbol{y}|X, \boldsymbol{w}) = \prod_{i=1}^{m} p([\boldsymbol{y}]_i \,|\, [X]_{i,:}, \boldsymbol{w}) \propto \exp\left(-\sigma_y^{-2} \|X\boldsymbol{w} - \boldsymbol{y}\|_{\mathbb{R}^m}^2\right), \tag{2.6}$$

so that all we can make is the awkward statement that, loosely speaking, (2.4) chooses the $\boldsymbol{w}$, given which, given X and given (2.5), $\boldsymbol{y}$ is most likely. This is typically referred to as the *maximum likelihood* solution. However

short of resorting to improper distributions[1], we cannot make a more natural statement such as *given some assumptions* (2.4) *chooses the most likely* $\boldsymbol{w}$.

To make such a statement we can instead employ Bayes' rule, and choose as our solution the *maximum a posteriori* (m.a.p.) estimate

$$\begin{aligned}\boldsymbol{w}_{\text{MAP}} &= \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|X,\boldsymbol{y}) \\ &= \arg\max_{\boldsymbol{w}} \frac{p(\boldsymbol{y}|X,\boldsymbol{w})p(\boldsymbol{w}|X)}{p(\boldsymbol{y}|X)} \\ &= \arg\max_{\boldsymbol{w}} p(\boldsymbol{y}|X,\boldsymbol{w})p(\boldsymbol{w}|X),\end{aligned} \tag{2.7}$$

and if we choose the *a priori* distribution

$$p(\boldsymbol{w}|X) = \mathcal{N}(\boldsymbol{0}, \Sigma_w) \tag{2.8}$$

then we have

$$\begin{aligned}\boldsymbol{w}_{\text{MAP}} &= \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|X) \prod_{i=1}^{m} p([\boldsymbol{y}]_i \,|\, [X]_{i,:}, \boldsymbol{w}) \\ &= \arg\max_{\boldsymbol{w}} \exp\left(-\boldsymbol{w}^\top \Sigma_w^{-1} \boldsymbol{w} - \sigma_y^{-2} \left\|X\boldsymbol{w} - \boldsymbol{y}\right\|_{\mathbb{R}^m}^2\right) \\ &= \arg\min_{\boldsymbol{w}} \left(\boldsymbol{w}^\top \left(\sigma_y^2 \Sigma_w^{-1}\right) \boldsymbol{w} + \left\|X\boldsymbol{w} - \boldsymbol{y}\right\|_{\mathbb{R}^m}^2\right) \qquad (2.9) \\ &= \left(X^\top X + \sigma_y^2 \Sigma_w^{-1}\right)^{-1} X^\top \boldsymbol{y}. \qquad (2.10)\end{aligned}$$

Note the similarity between (2.10) and (2.4). The solution (2.10) is clearly unique since although $X^\top X$ is possibly only positive semi-definite, $\sigma_y^2 \Sigma_w^{-1}$ is necessarily strictly positive definite. Now we have two interpretations of $\boldsymbol{w}_{\text{MAP}}$:

1. (*cf.* (2.9)) it balances between minimising the squared error and keeping the term $\boldsymbol{w}^\top \left(\sigma_y^2 \Sigma_w^{-1}\right) \boldsymbol{w}$ small. This is a *regularisation perspective* in which we refer to $\boldsymbol{w}^\top \left(\sigma_y^2 \Sigma_w^{-1}\right) \boldsymbol{w}$ as the *regulariser*.
2. (*cf.* (2.7)) it is the m.a.p. estimate of $\boldsymbol{w}$ under the given noise model (2.5) and *a priori* distribution (2.8) on $\boldsymbol{w}$. This is a probabilistic view in which we refer to the distribution on $\boldsymbol{w}$ given by (2.8) as a *prior*

Both views are equivalent in the sense that they lead to the same estimate of $\boldsymbol{w}$. Let us now return to the original thread of our argument. We said that modelling the physical world is an ill-posed problem and that individual models can be preferred only according to some criterion chosen *a priori*, not in any absolute sense. This final example of the m.a.p. estimate demonstrates this, the criterion which led to $\boldsymbol{w}_{\text{MAP}}$ being

[1] To be precise, we can interpret the maximum likelihood solution as an m.a.p. estimate only if the prior distribution (*i.e.* the $p(\boldsymbol{w})$ which arises shortly in the text) is constant. But constant functions on $\mathbb{R}^n$ are not integrable and hence not proper probability distributions.

- The prior (2.8) that we placed on $\boldsymbol{w}$.
- The chosen noise model (2.5).
- The decision to take the m.a.p. estimate.

At this point it may not be obvious what solving a linear system has to do with modelling the physical world (*i.e.* modelling relationships between observed data). We now proceed to make this more clear.

2.1.2 Linear (-in-the-parameters) Models

As it will be useful later on, let us introduce the following vague

Definition 2.1.2. Given a set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} \subset \mathcal{X} \times \mathcal{Y}$, the task of inferring a function $f : \mathcal{X} \to \mathcal{Y}$ which should in some sense explain well the relationship between the $\boldsymbol{x}_i$ and y_i shall be referred to as the **regression problem**. If we enforce $f(\boldsymbol{x}_i) = y_i$ for all $i = 1 \ldots m$ then we refer to f as an **interpolant**, otherwise (*i.e.* if errors are allowed) as an **approximant**.

For example, an engineer has repeatedly measured various quantities from an electric circuit (represented as a vector in $\mathbb{R}^m$), along with a corresponding output voltage (represented as a number in $\mathbb{R}$), and wishes to infer a relationship between them which will aid in using the circuit.

If the engineer guesses that the unknown function f is a linear one but that the measurements are corrupted with Gaussian noise, *i.e.*

$$f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{w} + \epsilon,$$

where $\epsilon \sim \mathcal{N}(0, \sigma_y^2)$, then it is easy to see that we have returned to the scenario of the previous section but with $[\boldsymbol{y}]_i = y_i$ and $[X]_{:,i} = \boldsymbol{x}_i^\top$. As we have seen, the engineer cannot avoid imposing some prior distribution on $\boldsymbol{w}$ in order to make an probabilistically reasonable estimate of it based on the measured $(\boldsymbol{x}_i, y_i)$ pairs.

A linear model is rather restrictive however, and one may often prefer to consider a more complex model. A strikingly elegant way of doing this is to project the vectors $\boldsymbol{x}_i$ non-linearly into another space, and then apply the linear model as before, so that

$$f(\boldsymbol{x}) = \phi(\boldsymbol{x})^\top \boldsymbol{w} + \epsilon,$$

where $\phi : \mathbb{R}^m \to \mathbb{R}^p$ and $\boldsymbol{w} \in \mathbb{R}^p$. The analysis in the previous section can now be repeated but with $[\Phi]_{:,i} \triangleq \phi(\boldsymbol{x}_i)^\top$ in place of X to yield

$$\boldsymbol{w}_{\text{MAP}} = \left(\Phi^\top \Phi + \sigma_y^2 \Sigma_w^{-1}\right)^{-1} \Phi^\top \boldsymbol{y}.$$

Now, with a little algebra — the matrix inversion lemma (A.1) being useful — one can derive an expression for the mean of the posterior distribution at a test point $\boldsymbol{t}$. Since the posterior is Gaussian this corresponds to the m.a.p. estimate, and is given by

$$\begin{aligned} E[f(\boldsymbol{t})] &= \phi(\boldsymbol{t})^\top \boldsymbol{w}_{\text{MAP}} \\ &= \phi(\boldsymbol{t})^\top \Sigma_w \Phi^\top \left(\Phi \Sigma_w \Phi^\top + \sigma_y^2 I\right)^{-1} \boldsymbol{y}. \end{aligned}$$

Which is an interesting expression since the required matrix inversion is of size $n \times n$, irrespective of the dimension of the feature space p. Moreover, since ϕ only appears in the form of *e.g.* $\phi(\boldsymbol{x})\Sigma_w\phi(\boldsymbol{x}')^\top$, if we have access to a *kernel* or *covariance* function

$$k_{\phi,\Sigma_w}(\boldsymbol{x}, \boldsymbol{x}') \triangleq \phi(\boldsymbol{x})^\top \Sigma_w \phi(\boldsymbol{x}') \tag{2.11}$$

then the mapping ϕ need never be computed explicitly, and the overall computational costs will typically be independent of the dimensionality p of the feature space.

The kernel function which we have arrived at is useful for a variety of algorithms. A similar motivation for using such functions is the manoeuvre often referred to as the *kernel trick* (see *e.g.* [SS02]) — the fact that if a computation on a set $\{\boldsymbol{x}_i\}$ can be carried out using only the scalar products $\boldsymbol{x}_i^\top \boldsymbol{x}_j$, then it can also be carried rather easily on the set $\{\phi(\boldsymbol{x}_i)\}$, provided that we have some function $k(\boldsymbol{x}_i, \boldsymbol{x}_j) \triangleq \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)$. This is obviously identical to (2.11) but with $\Sigma_w = I$. The fact that numerous important algorithms such as principal components analysis and the support vector machine (s.v.m.) fall into this category has led to a great deal of interest in kernel based algorithms.

2.2 Kernel Methods

At the end of the last section we saw that given some mapping $\phi : \mathbb{R}^m \to \mathbb{R}^p$ and a function $k_\phi(\boldsymbol{x}, \boldsymbol{x}') \triangleq \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}')$, we can evaluate the m.a.p. estimate of a linear model $f(\boldsymbol{x}) = \phi(\boldsymbol{x})^\top \boldsymbol{w}$ (under certain Gaussian assumptions), regardless of the dimensionality p of the *feature space*, which may even be infinite. We now take a different path to this conclusion, which is based on Tikhonov regularisation in an reproducing kernel Hilbert space (r.k.h.s.) To motivate this new path consider the following

Definition 2.2.1. We refer to finding the minimiser of the sum of a norm along with some other (data dependent) terms, *i.e.*

$$f^* = \arg\min_{f \in \mathcal{F}} \|f\|_{\mathcal{F}}^2 + c(f) \tag{2.12}$$

as the **Tikhonov regularised** variant of the related problem

$$\arg\min_{f\in\mathcal{F}} c(f).$$

We note that there is a simple probabilistic motivation for Tikhonov regularisation — if $c(f)$ is the negative log-likelihood of some some observational data $\mathcal{D}$ given f, and $\|f\|^2_{\mathcal{F}}$ is the negative log-likelihood of f, *i.e.*

$$p(f) = \exp\left(-\|f\|^2_{\mathcal{F}}\right)$$

and

$$p(\mathcal{D}|f) = \exp\left(-\|c(f)\|^2_{\mathcal{F}}\right),$$

then

$$\begin{aligned}
\arg\min_{f\in\mathcal{F}} \|f\|^2_{\mathcal{F}} + c(f) &= \arg\max_{f\in\mathcal{F}} \exp\left(-\|f\|^2_{\mathcal{F}}\right)\exp\left(-c(f)\right) \\
&= \arg\max_{f\in\mathcal{F}} p(f)p(\mathcal{D}|f) \\
&= \arg\max_{f\in\mathcal{F}} \frac{p(f)p(\mathcal{D}|f)}{p(\mathcal{D})} \\
&= \arg\max_{f} p(f|\mathcal{D})
\end{aligned}$$

is the m.a.p. estimate of f given $\mathcal{D}$. Of course for this interpretation to be possible it is necessary that both $p(f)$ and $p(\mathcal{D}|f)$ are probability distributions, and as such need to satisfy some technical requirements such as integrability. We shall see that for problems of the form (2.12), taking $\mathcal{F}$ to be an r.k.h.s. is a convenient choice, and in many cases (*i.e.* for many *loss functions* c) leads to tractable algorithms for computing f^*. Our analysis of r.k.h.s.'s will culminate in a theorem (namely theorem 2.2.7), which is directly related to the above motivating example of Tikhonov regularisation.

2.2.1 Reproducing Kernel Hilbert Spaces

We now provide a basic overview of what are for our purposes the key aspects of the r.k.h.s. structure. For further details we recommend *e.g.* [SS02, HB04, Wen04].

Definition 2.2.2. Let $\mathcal{X}$ be a non-empty set. A function $k : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ is **positive definite** (positive definite (p.d.)) (respectively **positive semi-definite** or positive semi-definite (p.s.d.)) if for all $m \in \mathbb{N}$, all pairwise distinct sets $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\} \subseteq \mathcal{X}$ and all $\boldsymbol{\alpha} \in \mathbb{R}^m \setminus \mathbf{0}$, the expression

$$\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j k(\boldsymbol{x}_i, \boldsymbol{s}_j)$$

is positive (non-negative).

Definition 2.2.3. Let $\mathcal{F}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the **reproducing kernel (r.k.)** for $\mathcal{F}$ if

1. $k(\cdot, \boldsymbol{x}) \in \mathcal{F}$ for all $\boldsymbol{x} \in \mathcal{X}$.
2. $f(\boldsymbol{x}) = \langle f, k(\cdot, \boldsymbol{x}) \rangle_{\mathcal{F}}$ for all $f \in \mathcal{F}$ and $\boldsymbol{x} \in \mathcal{X}$.

Clearly the r.k. is unique — if we let both k_1 and k_2 be r.k.'s then

$$\langle f, k_1(\cdot, \boldsymbol{x}) - k_2(\cdot, \boldsymbol{x}) \rangle_{\mathcal{F}} = f(\boldsymbol{x}) - f(\boldsymbol{x}) = 0,$$

and if we put $f = k_1(\cdot, \boldsymbol{x}) - k_2(\cdot, \boldsymbol{x})$ then we have $\|k_1(\cdot, \boldsymbol{x}) - k_2(\cdot, \boldsymbol{x})\|_{\mathcal{F}} = 0$ and that $k_1 = k_2$. We now give a simple characterisation of a Hilbert space of functions with an r.k. — that is, an r.k.h.s. For the proof of this theorem we first need the following closely related

Theorem 2.2.4 (Riesz Representation Theorem)**.** *Let $\mathcal{H}$ be a Hilbert space and $t : \mathcal{H} \to \mathbb{C}$ be linear and continuous. There exists some $g \in \mathcal{H}$ such that for all $f \in \mathcal{H}$,*

$$t(f) = \langle f, g \rangle_{\mathcal{H}}.$$

For a proof, see *e.g.* [Wen04]. Using this result it is easy to prove the following

Theorem 2.2.5. *If $\mathcal{H}$ is a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$ then the following statements are equivalent,*

1. *The point evaluation functionals are continuous.*
2. *$\mathcal{H}$ has an r.k.*

Proof. Suppose that (1) is true. By Riesz' representation theorem we find that for every $\boldsymbol{x} \in \mathcal{X}$ there exists a $k_{\boldsymbol{x}} \in \mathcal{H}$ such that $\delta_{\boldsymbol{x}}(f) = \langle f, k_{\boldsymbol{x}} \rangle_{\mathcal{H}}$ for all $f \in \mathcal{H}$, and therefore $k(\boldsymbol{x}, \boldsymbol{y}) \triangleq k_{\boldsymbol{x}}(\boldsymbol{y})$ is the r.k. in $\mathcal{H}$. Now suppose (2) is true so that k is the r.k.Then $\delta_{\boldsymbol{x}} = \langle \cdot, k(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}}$, and since the inner product is continuous, so is $\delta_{\boldsymbol{x}}$. □

There is a one-to-one correspondence between r.k.h.s.'s and p.d. kernel functions. More formally, we have

Theorem 2.2.6. *To every r.k.h.s. there corresponds a p.d. r.k. and conversely given a p.d. kernel k on $\mathcal{X} \times \mathcal{X}$ we can construct a unique r.k.h.s. of real-valued functions on $\mathcal{X}$ with k as its r.k..*

Proof. Let $\boldsymbol{x}_1, \ldots \boldsymbol{x}_m \subseteq \mathcal{X}$ and $\alpha_1, \ldots \alpha_m \subseteq \mathbb{R}$ be arbitrary. Defining $f = \sum_{i=1}^{m} \alpha_i k(\cdot, \boldsymbol{x}_i)$ and using the reproducing property we have that

$$0 \leq \|f\|^2 = \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

and therefore k is p.d. The proof of the converse part is merely technical, and a simple proof can be found in [Wah90]. □

The representer theorem, widely celebrated within the machine learning community, says that the function minimising an r.k.h.s. norm along with some penalties associated with the function value at various points (as in kernel ridge regression (kernel ridge regression (k.r.r.)) for example) is a sum of kernel functions at those points (we have in fact already observed this phenomenon, *cf.* Equation (2.11) and the surrounding comments).

The theorem as we present it is a novel generalisation of [SHS01] (using the same proof idea) with equivalence if we choose all L_i to be identity operators. Note however that the case of general linear operators was in fact dealt with already in [Wah90] (which merely states the earlier result in [KW71]) – but only for the case of a specific loss function c. Hence the following theorem, which was originally stated in [WSC06], combines the two frameworks:

Theorem 2.2.7 (Representer theorem)**.** *Denote by $\mathcal{X}$ a non-empty set, by k an r.k. with r.k.h.s. $\mathcal{H}$, by Ω a strictly monotonic increasing real-valued function on $[0,\infty)$, by $c : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ an arbitrary cost function, and by $L_1, \dots L_m$ a set of linear operators $\mathcal{H} \to \mathcal{H}$. Each minimiser $f \in \mathcal{H}$ of the regularised risk functional*

$$c((L_1 f)(\boldsymbol{x}_1), \dots (L_m f)(\boldsymbol{x}_m)) + \Omega(||f||^2_{\mathcal{H}}) \tag{2.13}$$

admits the form

$$f = \sum_{i=1}^{m} \alpha_i L_i^* k_{\boldsymbol{x}_i}, \tag{2.14}$$

where $k_{\boldsymbol{x}} \triangleq k(\cdot, \boldsymbol{x})$ and L_i^* denotes the adjoint of L_i.

Proof. Decompose f into

$$f = \sum_{i=1}^{m} \alpha_i L_i^* k_{\boldsymbol{x}_i} + f_\perp$$

with $\alpha_i \in \mathbb{R}$ and $\langle f_\perp, L_i^* k_{\boldsymbol{x}_i} \rangle_{\mathcal{H}} = 0$, for each $i = 1 \dots m$. Due to the reproducing property we can write, for $j = 1 \dots m$,

$$\begin{aligned}
(L_j f)(\boldsymbol{x}_j) &= \langle (L_j f), k(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{H}} \\
&= \sum_{i=1}^{m} \alpha_i \langle L_j L_i^* k_{\boldsymbol{x}_i}, k(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{H}} + \langle (L_j f_\perp), k(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{H}} \\
&= \sum_{i=1}^{m} \alpha_i \langle L_j L_i^* k_{\boldsymbol{x}_i}, k(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{H}}.
\end{aligned}$$

Thus, the first term in (2.13) is independent of $f_\perp$. Moreover, it is clear due to orthogonality that if $f_\perp \neq 0$ then

$$\Omega\left(\left\|\sum_{i=1}^{m}\alpha_i L_i^* k_{\boldsymbol{x}_i} + f_\perp\right\|_{\mathcal{H}}^2\right) > \Omega\left(\left\|\sum_{i=1}^{m}\alpha_i L_i^* k_{\boldsymbol{x}_i}\right\|_{\mathcal{H}}^2\right),$$

so that for any fixed $\alpha_i \in \mathbb{R}$, Equation 2.13 is minimised when $f_\perp = 0$. □

Let us mention in passing a possibly more intuitive but certainly less general and less rigorous way of seeing the above result, as mentioned in [Cha07]. Consider the functional

$$\mathcal{O}(f) = \frac{1}{2}\langle f, f\rangle_{\mathcal{H}} + \sum_i c_i\left(f(\boldsymbol{x}_i)\right).$$

If $\mathcal{O}$ is differentiable in f (in the sense of the functional or Gâteaux derivative from functional analysis [RS80]) then we have at its minimum the stationarity condition

$$\begin{aligned} 0 = \frac{\partial}{\partial f}\mathcal{O}(f) &= f + \frac{\partial}{\partial f}\sum_i c_i\left(f(\boldsymbol{x}_i)\right) \\ &= f + \sum_i c_i'\left(f(\boldsymbol{x}_i)\right)\frac{\partial}{\partial f}\langle f, k(\cdot,\boldsymbol{x}_i)\rangle_{\mathcal{H}} \\ &= f + \sum_i c_i'\left(f(\boldsymbol{x}_i)\right)k(\cdot,\boldsymbol{x}_i), \end{aligned} \tag{2.15}$$

and therefore $f = -\sum_i c_i'\left(f(\boldsymbol{x}_i)\right)k(\cdot,\boldsymbol{x}_i)$, which is a similar result to the representer theorem. It is also interesting to note that vanishing derivatives c_i' lead to sparse solutions in the sense that many of the basis functions $k(\cdot,\boldsymbol{x}_i)$ have a multiplicative coefficient of zero — this is the case with the s.v.m. for example (see 2.2.2). We now give an example of how convenient the r.k.h.s. structure can be in combination with the representer theorem.

2.2.2 Some Kernel Based Algorithms

Now that we have seen how convenient and powerful the r.k.h.s. structure is, it should come as no surprise that there exist several widely used kernel based algorithms. This is indeed the case, and [SS02] provides many such examples. We provide details on just three of these: kernel ridge regression, Gaussian processes and the support vector machine.

Kernel Ridge Regression

Given an r.k.h.s. of functions $\mathcal{H} : \Omega \to \mathbb{R}$ with r.k. k, and a set of points $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m) \subset \Omega \times \mathbb{R}$, the algorithm known as *kernel ridge regression* (k.r.r.) solves for the function $f^* \triangleq \arg\min_{f \in \mathcal{H}} \mathcal{O}[f]$ where the functional $\mathcal{O}$ (the objective) is given by

$$\mathcal{O}[f] = \sigma_y^2 \|f\|_{\mathcal{H}}^2 + \sum_{i=1}^{m} \left(f(\boldsymbol{x}_i) - y_i\right)^2 . \tag{2.16}$$

To solve for f^* in closed form, we can use the representer theorem to write $f^* = \sum_{i=1}^m \alpha_i^* k(\cdot, \boldsymbol{x}_i)$, and then the reproducing property to solve for

$$\begin{aligned} \boldsymbol{\alpha}^* &= \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^m} \boldsymbol{\alpha}^\top K \boldsymbol{\alpha} + \|K\boldsymbol{\alpha} - \boldsymbol{y}\|^2 \\ &= \left(K + \sigma_y^2 I\right)^{-1} \boldsymbol{y} \end{aligned} \tag{2.17}$$

where we have defined $[K]_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Note that if we choose $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k_{\phi,\Sigma_w}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ (*cf.* (2.11)), then f^* is identical to the m.a.p. estimate of the linear-in-the-parameters model we derived subsection 2.1.2. One might say that the reason we can solve for f^* so easily is the fact that we can use the representer theorem to switch from a problem in $\mathcal{H}$ to one in $\mathbb{R}^m$.

We now present another means of solving (2.16), which we believe is not only interesting in its own right, but also instructive[2]. The method is a very direct way of essentially ploughing through from (2.16) to a closed form expression for f^*. We begin by defining

$$\begin{aligned} T_n : \mathcal{H} &\to \mathbb{R} \\ f &\to \langle f, k(\cdot, \boldsymbol{x}_n) \rangle_{\mathcal{H}} , \end{aligned}$$

as well as

$$\begin{aligned} T : \mathcal{H} &\to \mathbb{R}^m \\ f &\to \left[T_1 f, T_2 f, \ldots T_m f\right]^\top , \end{aligned}$$

so that we can write

$$\begin{aligned} \mathcal{O}[f] &= ||Tf - \boldsymbol{y}||_{\mathbb{R}^m}^2 + \sigma_y^2 ||f||_{\mathcal{H}}^2 \\ &= \langle Tf - \boldsymbol{y}, Tf - \boldsymbol{y} \rangle_{\mathbb{R}^m} + \sigma_y^2 \langle f, f \rangle_{\mathcal{H}} \\ &= \langle f, T^*Tf \rangle_{\mathcal{H}} + ||\boldsymbol{y}||_{\mathbb{R}^m}^2 - 2 \langle f, T^*\boldsymbol{y} \rangle_{\mathcal{H}} + \sigma_y^2 \langle f, f \rangle_{\mathcal{H}} , \end{aligned}$$

[2]This part draws heavily on personal communication with Cheng Soon Ong, as well as the lectures by Stephán Canu at the Machine Learning Summer School, Berder Island 2004.

and due to convexity we have the following stationarity relation for f^*:

$$\begin{aligned} \frac{\partial}{\partial f}|_{f=f^*}\mathcal{O}[f] &= 0 \\ 2T^*Tf^* - 2T^*\boldsymbol{y} + \sigma_y^2 2f^* &= 0 \\ f^* &= (T^*T + \sigma_y^2 I_{\mathcal{H}})^{-1}T^*\boldsymbol{y}. \end{aligned} \tag{2.18}$$

To characterise the adjoint T^* as well as TT^* and T^*T we begin by rearranging

$$\begin{aligned} < Tf, \boldsymbol{\alpha} >_{\mathbb{R}^m} &= \sum_{n=1}^{m} \alpha_n \left\langle f, k(\cdot, \boldsymbol{x}_n) \right\rangle_{\mathcal{H}} \\ &= \left\langle f, \sum_{n=1}^{m} \alpha_n k(\cdot, \boldsymbol{x}_n) \right\rangle_{\mathcal{H}} \\ &= \left\langle f, T^*\boldsymbol{\alpha} \right\rangle_{\mathcal{H}}, \end{aligned}$$

so that we can write

$$\begin{aligned} T^* : \mathbb{R}^m &\to \mathcal{H} \\ \boldsymbol{\alpha} &\to \sum_{n=1}^{m} \alpha_n k(\cdot, \boldsymbol{x}_n), \end{aligned}$$

as well as

$$\begin{aligned} T^*T : \mathcal{H} &\to \mathcal{H} \\ f &\to T^* \left[T_1 f, T_2 f, \ldots T_m f\right]^\top \\ &\to \sum_{n=1}^{m} f(\boldsymbol{x}_n) k(\cdot, \boldsymbol{x}_n), \end{aligned}$$

and finally

$$\begin{aligned} TT^* : \mathbb{R}^m &\to \mathbb{R}^m \\ \boldsymbol{\alpha} &\to T \sum_{n=1}^{m} \alpha_n k(\cdot, \boldsymbol{x}_n) \\ &\to K\boldsymbol{\alpha}, \end{aligned}$$

(recall that $[K]_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$). Now, to deal with the inversions in $\mathcal{H}$ (*cf.* (2.18)) we have, similarly to the matrix inversion lemma,

$$(T^*T + \sigma_y^2 I_{\mathcal{H}})^{-1} = I_{\mathcal{H}}/\sigma_y^2 - T^*(I_{\mathbb{R}^m} + TT^*/\sigma_y^2)^{-1}T/\sigma_y^4.$$

To verify this for the generally infinite dimensional case let us assume that

the required inverses exist so that we can compute directly

$$(T^*T + \sigma_y^2 I_{\mathcal{H}})(I_{\mathcal{H}}/\sigma_y^2 - T^*(I_{\mathbb{R}^m} + TT^*/\sigma_y^2)^{-1}T/\sigma_y^4) = \tag{2.19}$$
$$I_{\mathcal{H}} + \underbrace{T^*T/\sigma_y^2}_{X} - \underbrace{T^*(I_{\mathbb{R}^m} + TT^*/\sigma_y^2)^{-1}T/\sigma_y^2}_{Y} - \underbrace{T^*TT^*(I_{\mathbb{R}^m} + TT^*/\sigma_y^2)^{-1}T/\sigma_y^4}_{Z}$$

but

$$\begin{aligned} X &= T^*(I_{\mathbb{R}^m} + TT^*/\sigma_y^2)(I_{\mathbb{R}^m} + TT^*/\sigma_y^2)^{-1}T/\sigma_y^2 \\ &= T^*(I_{\mathbb{R}^m} + TT^*/\sigma_y^2)^{-1}T/\sigma_y^2 + T^*TT^*(I_{\mathbb{R}^m} + TT^*/\sigma_y^2)^{-1}T/\sigma_y^4 \\ &= Y + Z \end{aligned}$$

so the right hand side of (2.19) is indeed just $I_{\mathcal{H}}$, the identity operator in $\mathcal{H}$, and we can indeed write

$$\begin{aligned} f^* &= (I_{\mathcal{H}}/\sigma_y^2 - T^*(I_{\mathbb{R}^m} + TT^*/\sigma_y^2)^{-1}T/\sigma_y^4)T^*\boldsymbol{y} \\ &= T^*(\boldsymbol{y}/\sigma_y^2 - (I_{\mathbb{R}^m} + TT^*/\sigma_y^2)^{-1}TT^*\boldsymbol{y}/\sigma_y^4) \\ &\triangleq \sum_{n=1}^{m} \alpha_n k(\cdot, \boldsymbol{x}_n). \end{aligned}$$

where, putting $K = TT^*$, one may verify with a little algebra that

$$\begin{aligned} \boldsymbol{\alpha} &= \boldsymbol{y}/\sigma_y^2 - (I_{\mathbb{R}^m} + K/\sigma_y^2)^{-1}K\boldsymbol{y}/\sigma_y^4 \\ &= (K + \sigma_y^2 I_{\mathbb{R}^m})^{-1}\boldsymbol{y}, \end{aligned}$$

is identical to the $\boldsymbol{\alpha}^*$ we had before.

Gaussian Processes

We have already arrived at k.r.r. from the perspective of

1. An m.a.p. estimate of a linear-in-the-parameters model (*cf.* subsection 2.1.2).
2. A Tikhonov regularised least squares solution (*cf.* 2.2.2).

We now briefly mention one last way of looking at it — this time from the perspective of a Gaussian process (g.p.) For further details on g.p.'s, we recommend *e.g.* [Mac98, RW06].

The key idea is placing a zero mean *Gaussian process prior* on f, which states that for any $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \Omega$, the vector $\boldsymbol{f}_x \triangleq (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_m))^\top$ is multivariate Gaussian according to

$$\boldsymbol{f}_x \sim \mathcal{N}(\boldsymbol{0}, K_{xx}),$$

where $[K_{xx}]_{i,j} \triangleq k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The function $k : \Omega \times \Omega \to \mathbb{R}$ is known in the g.p. literature as the *covariance function*, and defines the prior distribution

over functions. Next we place a zero-mean i.i.d. Gaussian noise model on the observed value $y_{\boldsymbol{z}}$ at $\boldsymbol{z}$, *i.e.*

$$y_{\boldsymbol{z}}|f \sim \mathcal{N}(f(\boldsymbol{z}), \sigma_y^2).$$

Altogether the joint distribution between the vector of observations

$$\boldsymbol{y}_{\boldsymbol{x}} \triangleq (y_{\boldsymbol{x}_1}, \ldots, y_{\boldsymbol{x}_m})^\top$$

and the value of the function at some test point $\boldsymbol{t}$ is therefore distributed according to

$$\begin{pmatrix} \boldsymbol{y}_{\boldsymbol{x}} \\ f(\boldsymbol{t}) \end{pmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{pmatrix} \left(K_{xx} + \sigma_y^2 I\right) & \boldsymbol{k}_{xt} \\ \boldsymbol{k}_{xt}^\top & k_{tt} \end{pmatrix}\right),$$

where $[\boldsymbol{k}_{xt}]_i \triangleq k(\boldsymbol{x}_i, \boldsymbol{t})$ and $k_{tt} = k(\boldsymbol{t}, \boldsymbol{t})$. Now employing (A.1) and (A.7) we have for the posterior

$$f(\boldsymbol{t})|\boldsymbol{y}_{\boldsymbol{x}} \sim \mathcal{N}\left(\boldsymbol{k}_{xt}^\top \left(K_{xx} + \sigma_y^2 I\right)^{-1} \boldsymbol{y}_{\boldsymbol{x}}, k(\boldsymbol{t}, \boldsymbol{t}) - \boldsymbol{k}_{xt}^\top \left(K_{xx} + \sigma_y^2 I\right)^{-1} \boldsymbol{k}_{xt}\right),$$

and it is clear by comparison with (2.17) that the mean of the posterior is identical to the k.r.r. solution.

Support Vector Machines

The detailed discussion of k.r.r. which we have just presented serves well to illustrate the concepts behind kernel based algorithms. Let us now mention the s.v.m. — a now standard algorithm for supervised learning, said to have resulted from statistical learning theory research [Vap95]. The s.v.m. is in fact very similar to logistic regression, a classical statistical technique [DHS00]. Given a labelled classification data set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m) \subset \Omega \times \{1, -1\}$, the s.v.m. algorithm assigns to query point $\boldsymbol{x}$ the label $\text{sign}(f_{\text{svm}}(\boldsymbol{x}))$, where

$$f_{\text{svm}} = \arg\min_{f \in \mathcal{H}} \lambda \|f\|_{\mathcal{H}}^2 + \sum_{i=1}^{m} \max(0, y_i f(\boldsymbol{x}_i) - 1)^p. \tag{2.20}$$

Where typically $p \in \{1, 2\}$, although the choice $p = 1$ often leads to modest computational advantages [SS02]. Regardless of the value of p however, the s.v.m. has the attractive property that the solution will often be sparse in the sense that many of the coefficients α_i in the expansion (2.14) vanish. The reason for this is the fact that the functional derivatives in f of the terms $\max(0, y_i f(\boldsymbol{x}_i) - 1)$ vanish for $y_i f(\boldsymbol{x}_i) \geq 1$. The fact that this results in sparsity can immediately be seen from (2.15).

The parameter λ controls the regularisation in the s.v.m.Note that the function f_{svm} can be interpreted as a m.a.p. estimate in a manner similar to the discussion at the beginning of section 2.2 — for details see [Sol00].

2.2.3 Kernels and Regularisation Operators

We have already said that the functional (2.16) minimised in k.r.r. can be understood intuitively as trading between the goodness of fit to the data as measured by $\sum_{i=1}^{m} (f(\boldsymbol{x}_i) - y_i)^2$, and the complexity of the function as measured by the norm $\|f\|_{\mathcal{H}}^2$ — but what do we mean by complexity? One way of gaining more insight is to begin by assuming that we can write

$$\|f\|_{\mathcal{H}}^2 = \langle \psi f, \psi f \rangle_{L_2}, \tag{2.21}$$

for some *regularisation operator* $\psi : \mathcal{H} \to L_2$ and some suitable space L_2 of functions — the idea is to then analyse and interpret the form of ψ. The present Section clarifies the connection between ψ and the r.k. k of $\mathcal{H}$. Key to this analysis is the concept of a Green's function, which we now briefly introduce.

Green's Functions

Presently we provide a minimal introduction to the concept of a Green's function — for a more comprehensive account see *e.g.* [Roa70]. Consider the problem of solving for $f \in \mathcal{F}$ the equation

$$Pf = u$$

where $P : \mathcal{F} \to \mathcal{F}$ and $u \in \mathcal{F}$ are given. This *operator inversion* problem can be arbitrarily difficult to solve. One way of attempting to find a solution is to guess that

$$f = P^{-1}u$$

where P^{-1} is informal notation for the operator satisfying $P^{-1}Pf = f$ for all $f \in \mathcal{F}$. Now, if P is a differential operator, a natural guess for P^{-1} is an integral operator. Thus we assume a general integral transformation with kernel g, as defined by

$$\left(P^{-1}u\right)(\boldsymbol{x}) = \int_{\boldsymbol{x}\in\Omega} g(\boldsymbol{x}, \boldsymbol{y}) u(\boldsymbol{y}) \mathrm{d}\boldsymbol{y},$$

where $g : \Omega \times \Omega \to \mathbb{R}$ is referred to as the Green's function (after George Green) associated with the operator P. According to our definition of P^{-1}, we can write

$$\left(PP^{-1}u\right)(\boldsymbol{x}) = u(\boldsymbol{x}) = \int_{\boldsymbol{y}\in\Omega} Pg(\boldsymbol{x}, \boldsymbol{y}) u(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} \triangleq \langle Pg(\boldsymbol{x}, \cdot), u \rangle_{\mathcal{F}}, \tag{2.22}$$

where we have defined along the way the scalar product on $\mathcal{F}$. If we now assume that there lies in $\mathcal{H}$ a function δ satisfying

$$u(\cdot) = \langle \delta(\cdot - \boldsymbol{x}), u \rangle_{\mathcal{F}}$$

for all u — as is the case when $\mathcal{F} = L_2(\mathbb{R}^d)$ for example, whereupon δ is the Dirac function — we can read off from (2.22) the more familiar definition of the Green's function g of P, *i.e.*

$$Pg(\boldsymbol{x}, \boldsymbol{y}) = \delta(\boldsymbol{x} - \boldsymbol{y}).$$

A great deal of work has been done to solve equations of the above form, usually with the aim of solving differential equations which model physical systems (see *e.g.* [Roa70]). It turns out that some of this work can aid in the understanding of kernel methods, due to the connection we make clear in the following section.

Kernels are Green's Functions

Following [Wah90, GJP93, SS02], we combine the reproducing property with our previous definition of the regularisation operator ψ to attain

$$f(\boldsymbol{x}) = \langle k(\cdot, \boldsymbol{x}), f \rangle_{\mathcal{H}} = \langle \psi k(\cdot, \boldsymbol{x}), \psi f \rangle_{L_2} = \langle \psi^* \psi k(\cdot, \boldsymbol{x}), f \rangle_{\mathcal{H}},$$

and so by comparison with (2.22), k is the Green's function of $\psi^*\psi$ in $\mathcal{H}$. It is important to note that the regularisation operator ψ is not uniquely determined by the kernel function, as we demonstrate in the following

Example 2.2.8 Regularisation operator of the linear kernel
Consider the linear kernel on $\mathbb{R}^d \times \mathbb{R}^d$ defined by $k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y}$. The r.k.h.s. $\mathcal{H}$ of this kernel is given by

$$\mathcal{H} = \left\{ \boldsymbol{w}^\top \cdot, \forall \boldsymbol{w} \in \mathbb{R}^d \right\}.$$

If we define

$$\left\langle \boldsymbol{w}_1^\top \cdot, \boldsymbol{w}_2^\top \cdot \right\rangle_{\mathcal{H}} = \boldsymbol{w}_1^\top \boldsymbol{w}_2, \tag{2.23}$$

then it follows that the regularisation operator

$$\begin{aligned} \psi_1 \ : \ & \mathcal{H} \to \mathcal{H} \\ & f \to \psi_1 f = f \end{aligned}$$

satisfies $\langle \psi_1 k(\cdot, \boldsymbol{x}), \psi_1 f \rangle_{\mathcal{H}} = f(\boldsymbol{x})$ as required. there exists a similarly defined identity regularisation operator for any r.k.h.s. By varying the space which the operator maps to however, we can attain more insightful results. Hence consider the following alternative operator

$$\begin{aligned} \psi_2 \ : \ & \mathcal{H} \to \mathbb{R}^d \\ & f \to \psi_2 f = \nabla f. \end{aligned}$$

This is also a valid choice, since if we let $g = \boldsymbol{w}^\top \cdot$ we have

$$\begin{aligned}\langle \psi_2 k(\cdot, \boldsymbol{x}), \psi_2 g \rangle_{\mathcal{H}} &= \langle \boldsymbol{x}, \boldsymbol{w} \rangle_{\mathbb{R}^d} \\ &= \boldsymbol{w}^\top \boldsymbol{x} \\ &= g(\boldsymbol{x}),\end{aligned}$$

as required. Moreover this choice leads to the insight that using the r.k.h.s. of the linear kernel in a Tikhonov regularised setting is equivalent to penalising the gradient on the space of linear functions.

Translation Invariant Kernels

Let's now consider a more specific class of kernels, namely the translation invariant ones that can be written

$$k(\boldsymbol{x}, \boldsymbol{y}) \triangleq \phi(\boldsymbol{y} - \boldsymbol{x}).$$

Such kernels are particularly amenable to analysis from a regularisation perspective [SS02]. As we shall see, the Fourier transform is a useful tool for analysing translation invariant regularation operators. By first using (2.21) along with the reproducing property, then the translation invariance of the L_2 inner product, and finally our definition of convolution (see A.1), we may write

$$\begin{aligned}\phi(\boldsymbol{y} - \boldsymbol{x}) &= \langle \psi\phi(\cdot - \boldsymbol{x}), \psi\phi(\cdot - \boldsymbol{y}) \rangle_{L_2} \\ &= \langle \psi^*\psi\phi(\cdot + \boldsymbol{y} - \boldsymbol{x}), \phi(\cdot) \rangle_{L_2} \\ &= \left[\psi^*\psi\phi(\cdot) \otimes \phi(\cdot)\right](\boldsymbol{y} - \boldsymbol{x}),\end{aligned}$$

and now taking Fourier transforms and using (A.3) we have

$$\mathcal{F}_{\boldsymbol{r}}\left[\phi(\boldsymbol{r})\right](\boldsymbol{\omega}) = \mathcal{F}_{\boldsymbol{r}}\left[\psi^*\psi\phi(\boldsymbol{r}) \otimes \phi(\boldsymbol{r})\right](\boldsymbol{\omega}). \tag{2.24}$$

Now, it turns out that every translation invariant operator ψ which is bounded and linear can be expressed as a Fourier multiplier [Hör60], hence we may write

$$\mathcal{F}_{\boldsymbol{y}}\left[\psi^*\psi\phi(\boldsymbol{y}) \otimes \phi(\boldsymbol{y})\right](\boldsymbol{\omega}) = \Psi^2(\boldsymbol{\omega})\left(\mathcal{F}_{\boldsymbol{y}}\left[\phi(\boldsymbol{y})\right](\boldsymbol{\omega})\right)^2. \tag{2.25}$$

Where we have used the sloppy notation $\Psi^2(\boldsymbol{\omega}) = \mathcal{F}_{\boldsymbol{x}}\left[\psi^*\psi\delta(\boldsymbol{x})\right](\boldsymbol{\omega})$. Indeed, for operators that are a linear combination of differential operators this can be seen directly from the Fourier transform identity

$$\mathcal{F}_{\boldsymbol{x}}\left[\frac{\partial}{\partial\left[\boldsymbol{x}\right]_j} f(\boldsymbol{x})\right](\boldsymbol{\omega}) = \left(2\pi\left[\boldsymbol{\omega}\right]_j \imath\right)\mathcal{F}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\right](\boldsymbol{\omega}).$$

The expression (2.25) leads a concise expression relating translation invariant regularisation operators to the kernels they induce, *i.e.*

$$\mathcal{F}_{\boldsymbol{y}}\left[\phi(\boldsymbol{y})\right](\boldsymbol{\omega}) = \Psi(\boldsymbol{\omega})^{-2}.$$

In summary, we have assumed that the kernel is a reproducing one and translation invariant, that the regulariser is a Fourier multiplier, and used the Fourier transform as a tool for *deconvolution*. To summarise in mathematical terms, we can write

$$\begin{aligned}\langle \psi f, \psi\phi(\cdot - \boldsymbol{x})\rangle_{L_2} &= [\psi f \otimes \psi\phi](\boldsymbol{x}) \\ &= \mathcal{F}_{\boldsymbol{\omega}}^{-1}\left[\mathcal{F}_{\boldsymbol{x}}[f(\boldsymbol{x})](\boldsymbol{\omega})\Psi^2(\boldsymbol{\omega})\Phi(\boldsymbol{\omega})\right](\boldsymbol{x}) \\ &= f(\boldsymbol{x}),\end{aligned}$$

which succinctly demonstrates the basic notions.

Gaussian Kernel

The Gaussian is probably the most widely used p.d. kernel within the domain of machine learning, often prescribed as a reasonable "default choice" [HCL03]. The kernel is given by

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(\|\boldsymbol{x} - \boldsymbol{y}\|^2 / (2\sigma^2)\right). \tag{2.26}$$

One way of interpreting the regularisation properties of this kernel function is to derive the operator $\psi^*\psi$ of which it is the Green's function. To this end we apply our previous analysis of translation invariant kernels, in a manner which turns out to be rather similar to the procedure outlined in [YG89]. The ϕ we defined previously is in this case given by $\phi(\boldsymbol{y}) = \exp(\frac{-\|\boldsymbol{y}\|^2}{2\sigma^2})$, and so from (A.8) we have

$$\Phi(\boldsymbol{\omega}) \triangleq \mathcal{F}_{\boldsymbol{x}}[\phi(\boldsymbol{x})](\boldsymbol{\omega}) = \mathcal{F}_{\boldsymbol{y}}\left[\exp\left(\frac{-\|\boldsymbol{y}\|^2}{2\sigma^2}\right)\right](\boldsymbol{\omega}) = \sigma^d \exp\left(\frac{-\|\boldsymbol{\omega}\|^2 \sigma^2}{2}\right).$$

Now, using the series

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!},$$

as well as

$$\|\boldsymbol{\omega}\|^2 = \frac{-1}{(2\pi)^2} \sum_{p=1}^{d} \left(2\pi\omega_p\sqrt{-1}\right)^2$$

we can therefore derive that

$$\begin{aligned}\Psi^2(\boldsymbol{\omega}) = \Phi(\boldsymbol{\omega})^{-1} &= \sigma^{-d} \exp\left(\frac{\|\boldsymbol{\omega}\|^2 \sigma^2}{2}\right) \\ &= \sigma^{-d} \sum_{n=0}^{\infty} \left(\frac{\|\boldsymbol{\omega}\|^2 \sigma^2}{2}\right)^n / n! \\ &= \sigma^{-d} \sum_{n=0}^{\infty} \left(\frac{-\sum_{p=1}^{d} (2\pi\imath\omega_p)^2 \sigma^2}{8\pi^2}\right)^n / n!\end{aligned} \tag{2.27}$$

Note that (2.27) already provides a very intuitive interpretation of the regulariser induced by the Gaussian kernel, *i.e.* as a Fourier domain *filter* which attenuates high frequencies. To get an explicit expression for $\psi^*\psi$, we use (A.4) and (2.25) along with the above expression to get

$$\begin{aligned}\psi^*\psi f(\boldsymbol{x}) &= \sigma^{-d}\sum_{n=0}^{\infty}\left(-\frac{\sigma^2}{8\pi^2}\sum_{p=1}^{d}\partial^2_{[\boldsymbol{x}]_p}\right)^n\frac{f(\boldsymbol{x})}{n!}\\ &= \sigma^{-d}\sum_{n=0}^{\infty}\left(-\frac{\sigma^2}{8\pi^2}\right)^n\frac{\left(\nabla^{2n}f\right)(\boldsymbol{x})}{n!},\end{aligned}$$

where ∇^{2n} is the Laplacian raised to the n-th power, as implicitly defined by the above expressions.

Thin-plate Spline

Another widely used regularisation operator ψ — well studied in [Wah90] — is defined implicitly by the expression

$$\langle\psi f,\psi f\rangle_{L_2}=\sum_{i_1=1}^{d}\cdots\sum_{i_m=1}^{d}\int_{\boldsymbol{x}\in\mathbb{R}^d}\left(\frac{\partial}{\partial\left[\boldsymbol{x}\right]_{i_1}}\cdots\frac{\partial}{\partial\left[\boldsymbol{x}\right]_{i_m}}f(\boldsymbol{x})\right)^2 d\mu(\boldsymbol{x}) \qquad (2.28)$$

This is the m-th order thin-plate spline regulariser in $\mathbb{R}^d$ — so named because for the case $m = d = 2$, it can be regarded as an approximation of the bending energy of a thin metal plate which occupies the manifold in $\mathbb{R}^3$ given by $\left\{\left(\left[\boldsymbol{x}\right]_1,\left[\boldsymbol{x}\right]_2,f(\boldsymbol{x})\right)^\top:\boldsymbol{x}\in\mathbb{R}^2\right\}$. In practice the above function norm is typically utilised as a regulariser with $m = 2$ in two and three dimensions, whereas in four dimensions $m = 3$ is more common.

It is easy to see that in the notation of the previous expression this regulariser corresponds to the Fourier domain multiplier

$$\Psi^2(\boldsymbol{\omega}) = (2\pi)^2(-1)^m\left\|\boldsymbol{\omega}\right\|^{2m}.$$

It turns out that, because the thin-plate spline regularisation operator ψ has a non-empty null-space, the corresponding kernel function is only *conditionally* positive define (conditionally positive definite (c.p.d.)). We define and discuss the c.p.d. case in the following sub-section, for now note that, ignoring a multiplicative constant, the thin-plate kernel is given by [Duc77]

$$k(\boldsymbol{x},\boldsymbol{y}) = \begin{cases}(-1)^{m-(d-2)/2}\left\|\boldsymbol{x}-\boldsymbol{y}\right\|^{2m-d}\log(\left\|\boldsymbol{x}-\boldsymbol{y}\right\|) & \text{if } d\in 2\mathbb{N},\\ (-1)^{m-(d-1)/2}\left\|\boldsymbol{x}-\boldsymbol{y}\right\|^{2m-d} & \text{if } d\in(2\mathbb{N}-1),\end{cases} \qquad (2.29)$$

which is c.p.d. with respect to $\pi_{m-1}(\mathbb{R}^d)$, the set of d-variate polynomials of degree at most $m-1$.

2.2.4 Conditionally Positive Definite Kernels

In the last Section we alluded to c.p.d. kernel functions – these are given by the following

Definition 2.2.9. A continuous function $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is **conditionally positive definite** with respect to (w.r.t.) the linear space of functions $\mathcal{P}$ if, for all $m \in \mathbb{N}$, all $\{\boldsymbol{x}_i\}_{i=1\ldots m} \subset \mathcal{X}$, and all $\boldsymbol{\alpha} \in \mathbb{R}^m \setminus \{\mathbf{0}\}$ satisfying $\sum_{j=1}^m \alpha_j p(\boldsymbol{x}_j) = 0$ for all $p \in \mathcal{P}$, the following holds

$$\sum_{j,k=1}^{m} \alpha_j \alpha_k \phi(\boldsymbol{x}_j, \boldsymbol{x}_k) > 0. \tag{2.30}$$

Due to the positivity condition (2.30) — as opposed one of non negativity — we are referring to c.p.d. rather than conditionally positive *semi*-definite kernels. The c.p.d. case is more technical than the p.d. case. We provide a minimalistic discussion here — for more details we recommend *e.g.* [Wen04]. To avoid confusion, let us note in passing that while the above definition is quite standard (see *e.g.* [Wen04, Wah90]), many authors in the machine learning community use a definition of c.p.d. kernels which corresponds to our definition when $\mathcal{P} = \{1\}$ (*e.g.* [SS02]) or when $\mathcal{P}$ is taken to be the space of polynomials of some fixed maximum degree (*e.g.* [SSM98]). Let us now adopt the notation $\mathcal{P}^\perp(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$ for the set

$$\left\{ \boldsymbol{\alpha} \in \mathbb{R}^m : \sum_{i=1}^{m} \alpha_i p(\boldsymbol{x}_i) = 0 \text{ for all } p \in \mathcal{P} \right\}.$$

The c.p.d. kernels of Definition 2.2.9 naturally define a Hilbert space of functions as per

Definition 2.2.10. Let $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a c.p.d. kernel w.r.t. $\mathcal{P}$. We define $F_\phi(\mathcal{X})$ to be the Hilbert space of functions which is the completion of the set

$$\left\{ \sum_{j=1}^{m} \alpha_j \phi(\cdot, \boldsymbol{x}_j) : m \in \mathbb{N}, \boldsymbol{x}_1, .., \boldsymbol{x}_m \in \mathcal{X}, \boldsymbol{\alpha} \in \mathcal{P}^\perp(\boldsymbol{x}_1, .., \boldsymbol{x}_m) \right\},$$

which due to the definition of ϕ we may endow with the inner product

$$\left\langle \sum_{j=1}^{m} \alpha_j \phi(\cdot, \boldsymbol{x}_j), \sum_{k=1}^{n} \beta_k \phi(\cdot, \boldsymbol{y}_k) \right\rangle_{F_\phi(\mathcal{X})} = \sum_{j=1}^{m} \sum_{k=1}^{n} \alpha_j \beta_k \phi(\boldsymbol{x}_j, \boldsymbol{y}_k). \tag{2.31}$$

Note that ϕ is not the r.k. of $F_\phi(\mathcal{X})$ — in general $\phi(\boldsymbol{x}, \cdot)$ does not even lie in $F_\phi(\mathcal{X})$. In [Wen04] (Chapter 10, *Native Spaces*), a space is constructed

related to $F_\phi(\mathcal{X})$ but which does have an r.k., but this is out of the present scope. It does follow immediately from the definition however that

$$\left\langle f, \sum_{j=1}^{m} \alpha_j \phi(\cdot, \boldsymbol{x}_j) \right\rangle_{F_\phi(\mathcal{X})} = \sum_{j=1}^{m} \alpha_j f(\boldsymbol{x}_j). \tag{2.32}$$

For the remainder of this Section we develop a c.p.d. analog of the representer theorem. We begin with

Lemma 2.2.11. *Let $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a c.p.d. kernel w.r.t. $\mathcal{P}$ and $p_1, \ldots p_r$ a basis for $\mathcal{P}$. For any $\{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_m, y_m)\} \subset \mathcal{X} \times \mathbb{R}$, there exists an $s = s_{F_\phi(\mathcal{X})} + s_{\mathcal{P}}$ where $s_{F_\phi(\mathcal{X})} = \sum_{j=1}^{m} \alpha_j \phi(\cdot, \boldsymbol{x}_j) \in F_\phi(\mathcal{X})$ and $s_{\mathcal{P}} = \sum_{k=1}^{r} \beta_k p_k \in \mathcal{P}$, such that $s(\boldsymbol{x}_i) = y_i, i = 1 \ldots m$.*

A simple and elementary proof (which shows (5.14) is solvable when $\lambda = 0$), is given in section B.2. Note that although such an interpolating function s always exists, it need not be unique. The distinguishing property of the interpolating function is that the norm of the part which lies in $F_\phi(\mathcal{X})$ is minimum.

Definition 2.2.12. Let $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a c.p.d. kernel w.r.t. $\mathcal{P}$. We use the notation $P_\phi(\mathcal{P})$ to denote the projection $F_\phi(\mathcal{X}) \oplus \mathcal{P} \to F_\phi(\mathcal{X})$.

Note that $F_\phi(\mathcal{X}) \oplus P_\phi(\mathcal{P})$ is a direct sum since $p = \sum_{j=1}^{m} \beta_i \phi(\boldsymbol{z}_j, \cdot) \in \mathcal{P} \cap F_\phi(\mathcal{X})$ implies

$$\|p\|^2_{F_\phi(\mathcal{X})} = \langle p, p \rangle_{F_\phi(\mathcal{X})} = \sum_{i=1}^{m} \sum_{j=1}^{n} \beta_i \beta_j \phi(\boldsymbol{z}_i, \boldsymbol{z}_j) = \sum_{j=1}^{m} \beta_j p(\boldsymbol{z}_j) = 0.$$

For the remainder of this Section we shall develop a c.p.d. analog of theorem 2.2.7, the representer theorem. This is slightly more involved than the p.d. case. As a first step consider the following

Lemma 2.2.13. *Let $f = \sum_{j=1}^{m} \alpha_j \phi(\cdot, \boldsymbol{x}_j) \in F_\phi(\mathcal{X})$. For all $g \in F_\phi(\mathcal{X})$ satisfying*

$$f(\boldsymbol{x}_j) = g(\boldsymbol{x}_j), j = 1 \ldots m, \tag{2.33}$$

$\|f\|_{F_\phi(\mathcal{X})} \leq \|g\|_{F_\phi(\mathcal{X})}$.

Proof. Due to (2.32), (2.33) and the Cauchy-Schwarz inequality, we have

$$\begin{aligned}\|f\|^2_{F_\phi(\mathcal{X})} &= \langle f, f-g+g\rangle_{F_\phi(\mathcal{X})} \\ &= \langle f, f-g\rangle_{F_\phi(\mathcal{X})} + \langle f, g\rangle_{F_\phi(\mathcal{X})} \\ &= \sum_{j=1}^{m} \alpha_j (f-g)(\boldsymbol{x}_j) + \langle f, g\rangle_{F_\phi(\mathcal{X})} \\ &= \langle f, g\rangle_{F_\phi(\mathcal{X})} \\ &\leq \|f\|_{F_\phi(\mathcal{X})} \|g\|_{F_\phi(\mathcal{X})}\end{aligned}$$

□

At first it seems that one could derive a representer theorem directly from the above lemma. This is not the case however since the constraint that the coefficients $\boldsymbol{\alpha}$ of functions of the form

$$\sum_{j=1}^{m} \alpha_j \phi(\cdot, \boldsymbol{x}_j)$$

lie in $\mathcal{P}^\perp(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$ means that we cannot choose the coefficients to satisfy the constraints

$$f(\boldsymbol{x}_i) = y_i, i = 1 \ldots m,$$

for arbitrary values $y_1, \ldots y_m$. Instead we must consider the space $F_\phi(\mathcal{X}) \oplus \mathcal{P}$, as indicated by the following lemma — our proof of which seems to be novel and particularly elementary.

Lemma 2.2.14. *Denote by $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a c.p.d. kernel w.r.t. $\mathcal{P}$ and by $p_1, \ldots p_r$ a basis for $\mathcal{P}$. Consider an arbitrary function $s = s_{F_\phi(\mathcal{X})} + s_\mathcal{P}$ with $s_{F_\phi(\mathcal{X})} = \sum_{j=1}^{m} \alpha_j \phi(\cdot, \boldsymbol{x}_j) \in F_\phi(\mathcal{X})$ and $s_\mathcal{P} = \sum_{k=1}^{r} \beta_k p_k \in \mathcal{P}$. $\|P_\phi(\mathcal{P})s\|_{F_\phi(\mathcal{X})} \leq \|P_\phi(\mathcal{P})f\|_{F_\phi(\mathcal{X})}$ holds for all $f \in F_\phi(\mathcal{X}) \oplus \mathcal{P}$ satisfying*

$$f(\boldsymbol{x}_i) = s(\boldsymbol{x}_i), i = 1 \ldots m. \tag{2.34}$$

Proof. Let f be an arbitrary element of $F_\phi(\mathcal{X}) \oplus \mathcal{P}$. We can always write

$$f = \sum_{j=1}^{m} (\alpha_i + \overline{\alpha}_i) \phi(\cdot, \boldsymbol{x}_j) + \sum_{l=1}^{n} b_l \phi(\cdot, \boldsymbol{z}_l) + \sum_{k=1}^{r} c_k p_k.$$

If we define[3] $[P_x]_{i,j} = p_j(\boldsymbol{x}_i)$, $[P_z]_{i,j} = p_j(\boldsymbol{z}_i)$, $[\Phi_{xx}]_{i,j} = \phi(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $[\Phi_{xz}]_{i,j} = \phi(\boldsymbol{x}_i, \boldsymbol{z}_j)$, and $[\Phi_{zx}]_{i,j} = \phi(\boldsymbol{z}_i, \boldsymbol{x}_j)$, then the condition (2.34) can hence be written

$$P_x \boldsymbol{\beta} = \Phi_{xx} \overline{\boldsymbol{\alpha}} + \Phi_{xz} \boldsymbol{b} + P_x \boldsymbol{c}, \tag{2.35}$$

[3] Square brackets w/ subscripts denote matrix elements, and colons denote entire rows or columns.

and the definition of $F_\phi(\mathcal{X})$ requires that *e.g.* $\boldsymbol{\alpha} \in \mathcal{P}^\perp(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$, hence implying the constraints

$$P_x^\top \boldsymbol{\alpha} = \mathbf{0} \quad \text{and} \quad P_x^\top (\boldsymbol{\alpha} + \overline{\boldsymbol{\alpha}}) + P_z^\top \boldsymbol{b} = \mathbf{0}. \tag{2.36}$$

The inequality to be demonstrated is then

$$L \triangleq \boldsymbol{\alpha}^\top \Phi_{xx} \boldsymbol{\alpha} \leq \begin{pmatrix} \boldsymbol{\alpha} + \overline{\boldsymbol{\alpha}} \\ \boldsymbol{b} \end{pmatrix}^\top \underbrace{\begin{pmatrix} \Phi_{xx} & \Phi_{xz} \\ \Phi_{zx} & \Phi_{zz} \end{pmatrix}}_{\triangleq \Phi} \begin{pmatrix} \boldsymbol{\alpha} + \overline{\boldsymbol{\alpha}} \\ \boldsymbol{b} \end{pmatrix} \triangleq R. \tag{2.37}$$

By expanding

$$R = \underbrace{\boldsymbol{\alpha}^\top \Phi_{xx} \boldsymbol{\alpha}}_{=L} + \underbrace{\begin{pmatrix} \overline{\boldsymbol{\alpha}} \\ \boldsymbol{b} \end{pmatrix}^\top \Phi \begin{pmatrix} \overline{\boldsymbol{\alpha}} \\ \boldsymbol{b} \end{pmatrix}}_{\triangleq \Delta_1} + 2 \underbrace{\begin{pmatrix} \boldsymbol{\alpha} \\ \mathbf{0} \end{pmatrix}^\top \Phi \begin{pmatrix} \overline{\boldsymbol{\alpha}} \\ \boldsymbol{b} \end{pmatrix}}_{\triangleq \Delta_2},$$

it follows from (2.36) that $P_x^\top \overline{\boldsymbol{\alpha}} + P_z^\top \boldsymbol{\beta} = \mathbf{0}$, and since Φ is c.p.d. w.r.t. $\begin{pmatrix} P_x^\top & P_z^\top \end{pmatrix}$ that $\Delta_1 \geq 0$. But (2.35) and (2.36) imply that $L \leq R$, since

$$\Delta_2 = \boldsymbol{\alpha}^\top \Phi_{xx} \overline{\boldsymbol{\alpha}} + \boldsymbol{\alpha}^\top \Phi_{xz} \boldsymbol{b} = \underbrace{\boldsymbol{\alpha}^\top P_x}_{=0} (\boldsymbol{\beta} - \boldsymbol{c}) - \boldsymbol{\alpha}^\top \Phi_{xz} \boldsymbol{b} + \boldsymbol{\alpha}^\top \Phi_{xz} \boldsymbol{b} = 0. \qquad \square$$

Using these results it is now easy to prove an analog of the representer theorem for the p.d. case.

Theorem 2.2.15 (Representer theorem for the c.p.d. case)**.** *Denote by $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a c.p.d. kernel w.r.t. $\mathcal{P}$, by Ω a strictly monotonic increasing real-valued function on $[0, \infty)$, and by $c : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ an arbitrary cost function. There exists a minimiser over $F_\phi(\mathcal{X}) \oplus \mathcal{P}$ of*

$$W(f) \triangleq c\left(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_m)\right) + \Omega\left(\|P_\phi(\mathcal{P}) f\|^2_{F_\phi(\mathcal{X})}\right) \tag{2.38}$$

which admits the form $\sum_{i=1}^m \alpha_i \phi(\cdot, \boldsymbol{x}_i) + p$, where $p \in \mathcal{P}$.

Proof. Let f be a minimiser of W. Let $s = \sum_{i=1}^m \alpha_i \phi(\cdot, \boldsymbol{x}_i) + p$ satisfy $s(\boldsymbol{x}_i) = f(\boldsymbol{x}_i), i = 1 \ldots m$. By lemma 2.2.11 we know that such an s exists. But by lemma 2.2.14 $\|P_\phi(\mathcal{P}) s\|^2_{F_\phi(\mathcal{X})} \geq \|P_\phi(\mathcal{P}) f\|^2_{F_\phi(\mathcal{X})}$. As a result, $W(s) \leq W(f)$ and s is a minimizer of W with the correct form. $\square$

Chapter 3

Fast Approximation Methods

In the previous chapter we saw how Tikhonov regularisation can be used to deal with ill-posed problems, and we motivated the regularised solution as an *maximum a posteriori* (m.a.p.) estimate. We also saw that for Tikhonov regularised function estimation problems, if we assume that the function lies in an reproducing kernel Hilbert space (r.k.h.s.), then we can invoke the representer theorem and conveniently work in the span of kernel functions at the given data points (*cf.* (2.14)).

It is already clear that the representer theorem is indispensable if we seek the globally optimal function the from a high or even infinite dimensional r.k.h.s. $\mathcal{H}$. Moreover, if the dimension of the input space d is large in comparison to the number of training points m, then it is convenient that the computational complexity is chiefly dependent on the latter. While this may often be the case in machine learning problems, it is rarely the case in surface estimation problems we consider in chapter 4. In those problems one typically has $d \lesssim 5$ but m up to the order of 10^7. But if we naively apply *e.g.* kernel ridge regression (k.r.r.) with a fully supported kernel function, then we will suffer a cubic time complexity in m due to the requisite matrix inversion (*cf.* (2.17)). We are therefore forced to either seek a different approach, or resort to approximations.

In subsection 4.1.1 we will see that kernel methods based on fully supported kernels have highly attractive properties. Motivated by all of this, in the present chapter we concern ourselves with methods of approximating such kernel methods which scale well in m, if not so well in d. The main part of the chapter is section 3.1, where we develop a new method based on compactly supported basis functions. For the sake of comparison we then discuss the well known fast multipole method (f.m.m.) approach in section 3.2.

3.1 Decoupling Regulariser and Function Basis

As we saw in chapter 2, the solution to the Tikhonov regularisation problem in an r.k.h.s. $\mathcal{H}$ with reproducing kernel (r.k.) k, where the data dependent term (*i.e.* the c in (2.12)) depends on $f(\boldsymbol{x}_1), \ldots f(\boldsymbol{x}_m)$, always takes the form

$$f = \sum_{i=1}^{m} \alpha_i k(\cdot, \boldsymbol{x}_i). \tag{3.1}$$

This is convenient since the norm of the function — one of the terms minimised in the Tikhonov regularisation setting — has the form

$$\|f\|_{\mathcal{H}}^2 = \langle f, f \rangle_{\mathcal{H}} = \left\langle \sum_{i=1}^{m} \alpha_i k(\cdot, \boldsymbol{x}_i), \sum_{j=1}^{m} \alpha_j k(\cdot, \boldsymbol{x}_j) \right\rangle_{\mathcal{H}} = \sum_{i,j=1}^{m} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j),$$

where the last step is due to the reproducing property. In the following, we shall depart from the function basis implied by (3.1). Instead we opt for an ad-hoc basis not related to k or $\mathcal{H}$, but rather chosen for its computational advantages. By doing so we will also be forced to depart from the convenience of the above expression for the norm. Nonetheless, as we shall demonstrate it is possible to do this and it does lead to useful approximations.

Before proceeding we would like to address the natural question of why we do not simply use a kernel function with compact support — after all this would allow similar computational advantages without requiring that we abandon the convenience of the above expression for the function norm. At the very end of chapter 2 we gave a brief survey to support the claim that compactly supported kernels and local processing methods in general have undesirable properties — presently we give an intuitive explanation as to why this is.

Compactly Supported Kernels

The main problem with compactly supported kernels (and any kernel — such as the Gaussian — which converges to a constant value as the two input points become further apart) is that the corresponding regularisers are somewhat poor for geometrical problems. Such regularisers draw the function towards some nominal constant as one moves away from the data, thereby implementing the non-intuitive behaviour of regularising the constant function and making interpolation impossible.

Moreover, since the support of the kernel function must be comparable to the size of the smallest details one wishes to capture from the data, interpolation is impossible on scales larger than that. To overcome this one may try fitting the function with basis functions of large support, to interpolate unsampled regions, followed by fitting to the residual errors with basis

functions of diminishing support, to capture smaller details. A number of authors have proposed this — for a representative example see *e.g.* [OBS03]. The final result is a final function that is the sum of the various intermediate functions, say $f = \sum_{i=1}^{n} g_i$. Since regularisation is done on each scale separately however, the overall regulariser is, roughly speaking, something like the expression $\sum_{i=1}^{n} ||g_i||^2_{\mathcal{H}_i}$. Unless the individual $\mathcal{H}_i$ are orthogonal, this is not a sensible regulariser — one simple example demonstrating this being the case $g_1 + g_2 = 0$, since even this arguably simplest of functions corresponds to (supposed) regularisation term $||g_1||^2_{\mathcal{H}_1} + ||g_2||^2_{\mathcal{H}_2}$ that could be arbitrarily large.

3.1.1 Restricting the Set of Available Functions

Let us now begin with the main thread of the present chapter. Pivotal in our strategy will be the use of compactly supported basis functions whose width and density are selected adaptively to the complexity of the function we wish to estimate. Thus, we propose forcing our estimated function f to take the form:

$$f(\cdot) = \sum_{k=1}^{p} \pi_k f_k(\cdot), \tag{3.2}$$

where the individual basis functions are

$$f_k(\cdot) = \phi_r(||\cdot - \boldsymbol{v}_k||/s_k),$$

for some compactly supported function $\phi_r : \mathbb{R}^+ \to \mathbb{R}$ with support $[0, 1)$. The $\boldsymbol{v}_k$ and s_k can be interpreted as the basis function centres and dilations (or scales), respectively.

We wish to minimise a Tikhonov regularised risk function (*e.g.* the k.r.r. objective given by (2.12)) within the span of (3.2). The key to doing this is to note that as we discussed in subsection 2.2.3, the regulariser (function norm) can be written as $\|f\|^2_{\mathcal{H}} = \langle \psi f, \psi f \rangle_{L_2}$. The point of the approximation can be seen by first substituting (3.2) into the k.r.r. objective (2.16), leading to

$$\begin{aligned}
\mathcal{O}[f] &= \sigma_y^2 \|f\|^2_{\mathcal{H}} + \sum_{i=1}^{m} \left(f(\boldsymbol{x}_i) - y_i\right)^2 \\
&= \sigma_y^2 \left\langle \sum_{j=1}^{p} \pi_j f_j, \sum_{k=1}^{p} \pi_k f_k \right\rangle_{\mathcal{H}} + \sum_{i=1}^{m} \left(\sum_{k=1}^{p} \pi_k f_k(\boldsymbol{x}_i) - y_i \right)^2 \\
&= \sigma_y^2 \boldsymbol{\pi}^\top F_\psi \boldsymbol{\pi} + \|F_x \boldsymbol{\pi} - \boldsymbol{y}\|^2_{\mathbb{R}^m}
\end{aligned} \tag{3.3}$$

where $[F_x]_{i,k} \triangleq f_k(\boldsymbol{x}_i)$, and

$$\begin{aligned}[F_\psi]_{k,k'} &\triangleq \langle f_k, f_{k'} \rangle_{\mathcal{H}} \\ &= \langle \psi f_k, \psi f_{k'} \rangle_{L_2}\end{aligned}$$

where ψ is the regularisation operator associated with $\mathcal{H}$ as per subsection 2.2.3. The optimal coefficients $\boldsymbol{\pi}^*$ are implied by stationarity:

$$\begin{aligned}\frac{1}{2}\frac{\partial}{\partial \boldsymbol{\pi}}\mathcal{O}[f] = 0 &= \sigma_y^2 F_\psi \boldsymbol{\pi}^\top + F_x^\top \left(F_x \boldsymbol{\pi}^\top - \boldsymbol{y}\right) \\ \rightarrow \boldsymbol{\pi}^* &= \left(\sigma_y^2 F_\psi + F_x^\top F_x\right)^{-1} F_x^\top \boldsymbol{y}. \end{aligned} \tag{3.4}$$

It is easy to see that both F_ψ and $F_x^\top F_x$ have the same sparsity pattern — the j,k-th element can only be non-zero if the f_j and f_k have overlapping support, *i.e.*

$$\|\boldsymbol{v}_j - \boldsymbol{v}_k\| \leq s_j + s_k. \tag{3.5}$$

The coefficients that we need to determine are therefore given by a sparse p-dimensional positive semi-definite linear system, which can be constructed efficiently by simple code that takes advantage of software libraries for fast nearest neighbour type searches (see *e.g.* [MPL00]). To actually solve the system we can use one of the several standard and widely available *conjugate gradient* type algorithms [GV96].

In the later subsection 3.1.4 we present an algorithm — effective for implicit surface reconstruction — which constructs the function basis (*i.e.* the $(\boldsymbol{v}_j, s_j)$ pairs). Presently we turn to the computation of F_ψ.

3.1.2 Computing the Regularisation Matrix

We now come to the crucial point of computing F_ψ, which can be thought of as the regulariser in (3.2). To build the sparse matrix F_ψ, a fast range search library (*e.g.* [MPL00]) can be used to identify the non-zero entries (*i.e.* those satisfying (3.5)). The next step is evaluating the terms

$$\langle \psi f_j(\cdot), \psi f_k(\cdot) \rangle_{L_2},$$

preferably with a closed form expression. We now provide two approaches for doing this. The first is a Fourier domain method, and the second a more direct method which is better suited to evaluation by numerical methods.

Fourier Domain Method

Let us begin by defining the function

$$\begin{aligned}\phi : \mathbb{R}^d &\rightarrow \mathbb{R} \\ \boldsymbol{x} &\rightarrow \phi(\boldsymbol{x}) = \phi_r(\|\boldsymbol{x}\|).\end{aligned}$$

As we mentioned in subsection 2.2.3 (*cf.* (2.25) and surrounding comments), many regularisers ψ are Fourier multipliers, *i.e.* they admit the form

$$\mathcal{F}_{\boldsymbol{y}}\left[\psi\phi(\boldsymbol{y})\right](\boldsymbol{\omega}) = \Psi(\boldsymbol{\omega})\mathcal{F}_{\boldsymbol{y}}\left[\phi(\boldsymbol{y})\right](\boldsymbol{\omega}). \tag{3.6}$$

For these regularisers, the Fourier domain is very convenient in deriving expressions for F_ψ — the term we require is given by

$$\begin{aligned}
\left\langle \psi f_j(\cdot), \psi f_k(\cdot)\right\rangle_{L_2} &= \left\langle \psi\phi((\cdot-\boldsymbol{v}_j)/s_j), \psi\phi((\cdot-\boldsymbol{v}_k)/s_k)\right\rangle_{L_2} \\
&= \left\langle \psi\phi(\cdot/s_j), \psi\phi((\cdot-\boldsymbol{v}_k+\boldsymbol{v}_j)/s_k)\right\rangle_{L_2} \\
&= \left[\psi\phi(\cdot/s_j)\otimes\psi\phi(\cdot/s_k)\right](\boldsymbol{v}_j-\boldsymbol{v}_k) \\
&= \mathcal{F}^{-1}_{\boldsymbol{\omega}}\left[\mathcal{F}_{\boldsymbol{x}}\left[\psi\phi(\boldsymbol{x}/s_j)\right](\boldsymbol{\omega})\mathcal{F}_{\boldsymbol{x}}\left[\psi\phi(\boldsymbol{x}/s_k)\right](\boldsymbol{\omega})\right](\boldsymbol{v}_j-\boldsymbol{v}_k) \\
&= \mathcal{F}^{-1}_{\boldsymbol{\omega}}\left[\Psi(\boldsymbol{\omega})^2(s_js_k)^d\Phi(s_j\boldsymbol{\omega})\Phi(s_k\boldsymbol{\omega})\right](\boldsymbol{v}_j-\boldsymbol{v}_k)
\end{aligned} \tag{3.7}$$

where $\Phi(\boldsymbol{\omega}) \triangleq \mathcal{F}_{\boldsymbol{x}}\left[\phi(\boldsymbol{x})\right](\boldsymbol{\omega})$. Assuming that $\Psi(\boldsymbol{\omega})$ is radially symmetric, then all of the required Fourier and inverse Fourier transforms above are also radially symmetric — and the above expression may be solved in closed form using (A.2) along with a table of integrals or a computer algebra package.

Example 3.1.1 Gaussian Basis / Gaussian Regulariser
The regulariser in Fourier multiplier form which corresponds to the Gaussian kernel with scale parameter σ is given by

$$\Psi(\boldsymbol{\omega}) = \sigma^{-d/2}\exp\left(\left\|\boldsymbol{\omega}\right\|^2\sigma^2/4\right)$$

(cf. (2.27) *and surrounding comments). If we take a Gaussian basis*

$$\phi_r = \exp(-(\cdot)^2/2) \tag{3.8}$$

(this is clearly not compactly supported — we consider it in this example because it makes the mathematics easier), then the Fourier transform

$$\Phi(\boldsymbol{\omega}) = \exp\left(-\left\|\boldsymbol{\omega}\right\|^2/2\right),$$

is also Gaussian, as is the regularisation term. If we assume the condition

$$\sigma^2 < s_j^2 + s_k^2, j = 1\ldots p, k = 1\ldots p,$$

or equivalently that

$$s_j > \sigma/\sqrt{2}, j = 1\ldots p, \tag{3.9}$$

then the required Fourier transforms exist and we have from (3.7) *and the*

Fourier transform dilation lemma (A.6) *that*

$$\begin{aligned}\langle \psi f_j(\cdot), \psi f_k(\cdot) \rangle_{L_2} &= \left(\frac{s_j s_k}{\sigma}\right)^d \mathcal{F}_{\boldsymbol{\omega}}^{-1}\left[\exp\left(\frac{\|\boldsymbol{\omega}\|^2\left(\sigma^2 - s_j^2 - s_k^2\right)}{2}\right)\right](\boldsymbol{v}_j - \boldsymbol{v}_k) \\ &= \left(\frac{s_j s_k}{\sigma}\right)^d \left(s_j^2 + s_k^2 - \sigma^2\right)^{-d/2} \exp\left(\frac{\|\boldsymbol{v}_j - \boldsymbol{v}_k\|^2}{2\left(\sigma^2 - s_j^2 - s_k^2\right)}\right).\end{aligned} \tag{3.10}$$

Note that

- *For* $s_j = s_k = \sigma$ (3.10) *reduces to a Gaussian with parameter* σ.
- *For* $s_j = \sigma$ *it reduces to a Gaussian with scale parameter* s_k, *evaluated at* $\boldsymbol{v}_k$, *as required by the reproducing property.*
- *The condition* (3.9) *is no restriction, since basis functions violating this condition do not have a finite r.k.h.s. norm.*

Figure 3.2 provides a visual depiction of the relationship between the scale parameter of a Gaussian function and its norm in the r.k.h.s. of a Gaussian kernel. We demonstrate the use of (3.10) *as a regulariser in Figure 3.1. Use of the expression in constructing sparse kernel machines using a multi-scale Gaussian basis is the subject of ongoing experimental work.*

The above example is neatly summarised by the following

Lemma 3.1.2. *Let*

$$g(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\sigma}) \triangleq |2\pi\, diag\,(\boldsymbol{\sigma})|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{y})^\top diag\,(\boldsymbol{\sigma})^{-1}(\boldsymbol{x} - \boldsymbol{y})\right), \tag{3.11}$$

where $\boldsymbol{x}, \boldsymbol{y}$ *and* $\boldsymbol{\sigma} \in \mathbb{R}^d$, *and let* $\mathcal{H}$ *be the r.k.h.s. with reproducing kernel* $g(\cdot, \cdot, \boldsymbol{\sigma})$. *If the conditions* $\boldsymbol{\sigma}_i > \frac{1}{2}\boldsymbol{\sigma}$ *and* $\boldsymbol{\sigma}_j > \frac{1}{2}\boldsymbol{\sigma}$ *are satisfied component-wise, then*

$$\langle g(\cdot, \boldsymbol{v}_i, \boldsymbol{\sigma}_i), g(\cdot, \boldsymbol{v}_j, \boldsymbol{\sigma}_j) \rangle_{\mathcal{H}} = g(\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{\sigma}_i + \boldsymbol{\sigma}_j - \boldsymbol{\sigma}). \tag{3.12}$$

If either condition is not satisfied, then the corresponding function on the left hand side is not in $\mathcal{H}$.

Which we prove directly in Appendices B.1.1 and B.1.2. It is interesting to compare this with the following famous result.

Theorem 3.1.3. *[Aro50] The function* f *belongs to the r.k.h.s.* $\mathcal{H}$ *with r.k.* k *if and only if there exists an* $\epsilon > 0$ *such that*

$$R_\epsilon(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y}) - \epsilon f(\boldsymbol{x}) f(\boldsymbol{y}),$$

is positive definite (p.d.), in which case

$$\|f\|_{\mathcal{H}}^2 = \inf\left\{1/\epsilon : R_\epsilon \text{ is p.d.}\right\}.$$

In particular, applying the above theorem 3.1.3 to the result of theorem 3.1.2 allows us to make the following statement. If $\sigma_1 > \sigma/\sqrt{2}$, the function $R_\epsilon(\boldsymbol{x}, \boldsymbol{y})$ given in this case by

$$\exp\left(-\left\|\boldsymbol{x}-\boldsymbol{y}\right\|^2/(2\sigma^2)\right) - \epsilon\exp\left(-\left(\left\|\boldsymbol{x}-\boldsymbol{v}_1\right\|^2 + \left\|\boldsymbol{y}-\boldsymbol{v}_1\right\|^2\right)/(2\sigma_1^2)\right)$$

is p.d. for all $\boldsymbol{v}_1 \in \mathbb{R}^d$ and all

$$\epsilon \leq 1/\left\|f_1\right\|_{\mathcal{H}}^2 = \left(\frac{\sigma_1^2}{\sigma}\right)^{-d}\left(2\sigma_1^2-\sigma^2\right)^{d/2}.$$

The case of Gaussian regulariser/Gaussian basis is probably the easiest one to handle from a mathematical point of view, due to the fact that the Fourier transform of the Gaussian is itself Gaussian, as is the product of two Gaussians. It turns out that the same analysis can be done in closed form for *e.g.* the exponential kernel

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-a\left\|\boldsymbol{x}-\boldsymbol{y}\right\|\right),$$

but the resulting expressions are messier, so we won't bother with them here. Although the case which we will now move onto is messier still, it is of interest due to its practical value.

Example 3.1.4 B_3-spline Basis / Thin-Plate Regulariser
For our numerical work with implicit surfaces we chose for ϕ_r the B_3-spline function (see Figure 3.3)

$$\phi_r(r) = \sum_{n=0}^{4}\frac{(-1)^n}{d!}\binom{n}{d+1}\left(r+\left(\frac{d+1}{2}-n\right)\right)_+^d, \tag{3.13}$$

although this choice is rather inconsequential as we ensure that the regulariser is unrelated to the function basis — any smooth compactly supported basis function could be used. In order to achieve the same interpolating properties as the thin-plate spline, we choose the thin-plate regulariser for ψ (see (2.28)). The easiest way to determine the corresponding Fourier multiplier $\Psi(\boldsymbol{\omega})$ is to write out for the implicit form

$$\begin{aligned}
&\left\langle \psi f(\cdot-\boldsymbol{r}), \psi g(\cdot)\right\rangle_{L_2}\\
&= \left[\sum_{i_1=1}^{d}\cdots\sum_{i_m=1}^{d}\left(\frac{\partial}{\partial x_{i_1}}\cdots\frac{\partial}{\partial x_{i_m}}f(\cdot)\right)\otimes\left(\frac{\partial}{\partial x_{i_1}}\cdots\frac{\partial}{\partial x_{i_m}}g(\cdot)\right)\right](\boldsymbol{r})\\
&= \mathcal{F}_{\boldsymbol{\omega}}^{-1}\left[\sum_{i_1=1}^{d}\cdots\sum_{i_m=1}^{d}\mathcal{F}_{\boldsymbol{x}}\left[\left(\frac{\partial}{\partial x_{i_1}}\cdots\frac{\partial}{\partial x_{i_m}}f(\boldsymbol{x})\right)\right](\boldsymbol{\omega})\mathcal{F}_{\boldsymbol{x}}\left[\left(\frac{\partial}{\partial x_{i_1}}\cdots\frac{\partial}{\partial x_{i_m}}g(\boldsymbol{x})\right)\right](\boldsymbol{\omega})\right](\boldsymbol{r})\\
&= \mathcal{F}_{\boldsymbol{\omega}}^{-1}\left[\sum_{i_1=1}^{d}\cdots\sum_{i_m=1}^{d}(2\pi\imath\omega_{i_1})^2\cdots(2\pi\imath\omega_{i_m})^2\,\mathcal{F}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\right](\boldsymbol{\omega})\mathcal{F}_{\boldsymbol{x}}\left[g(\boldsymbol{x})\right](\boldsymbol{\omega})\right](\boldsymbol{r})\\
&= \mathcal{F}_{\boldsymbol{\omega}}^{-1}\left[(2\pi\imath\left\|\boldsymbol{\omega}\right\|)^{2m}\,\mathcal{F}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\right](\boldsymbol{\omega})\mathcal{F}_{\boldsymbol{x}}\left[g(\boldsymbol{x})\right](\boldsymbol{\omega})\right](\boldsymbol{r}),
\end{aligned}$$

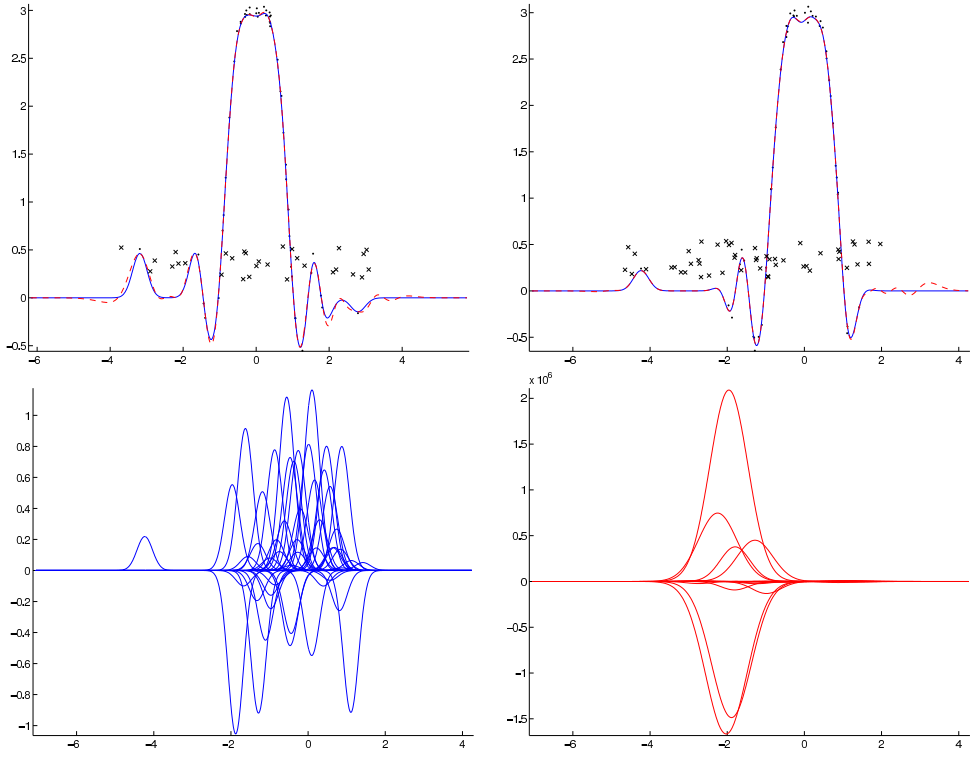


Figure 3.1: For $p = 30$ (top-left) and $p = 50$ (top-right), we take 50 random noisy samples of a sinc function, denoted by black dots. The blue line is the k.r.r.approximation based on these samples, using the $\sigma = 0.2$ Gaussian kernel (*cf.* (2.26)), call it k, and noise parameter $\sigma_y = 0.1$ (*cf.* (2.16)). The dashed-red line is the minimiser of the k.r.r.objective within the span of a randomly constructed multi-scale Gaussian function basis of p individual basis functions (*cf.* (3.3) and Example 3.1.1) such that $f = \sum_{j=1}^{p} \pi_j \exp((\cdot - v_j)^2/(2s_j^2))$ — the (v_j, s_j) pairs are denoted as black crosses on the top row. Depicted on the bottom left are the constituent $\alpha_i k(\cdot, x_i)$ terms from the k.r.r.solution $\sum_{i=1}^{50} \alpha_i k(\cdot, x_i)$ taken from the top-right plot. The bottom-right plot similarly shows the constituent $\pi_j \exp((\cdot - v_j)^2/(2s_j^2))$ terms from the top-right plot. Naturally the $p = 50$ approximation is closer to the exact solution. The large coefficients π_j (evident in the bottom-right plot) are allowed due to the cancellation effects but would not be allowed under a weight decay scheme. Zooming in on the top row reveals that the approximations tend to oscillate about the optimal solution.

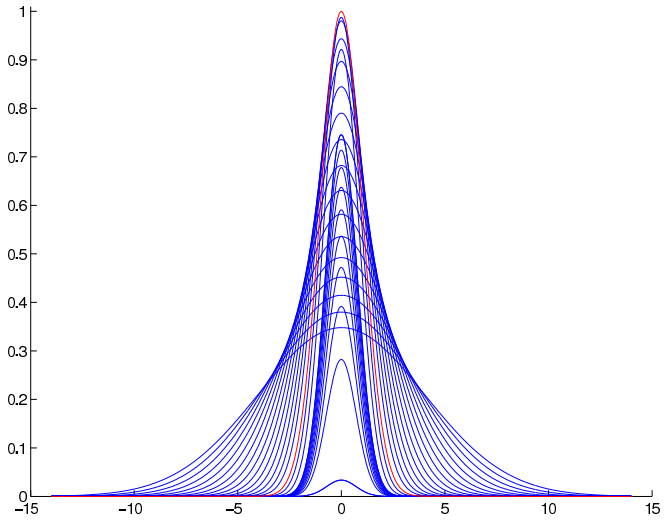


Figure 3.2: A family of unit norm Gaussians. Let $\mathcal{H}$ be the r.k.h.s. with r.k. $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|_{\mathbb{R}}^2 / 2)$. Each curve above depicts a function of the form $f(\boldsymbol{x}) = c \exp(-\|\boldsymbol{x}\|_{\mathbb{R}}^2 / (2\alpha^2))$ where c was chosen according to (3.10) such that $\|f\|_{\mathcal{H}} = 1$, *i.e.* we put $c = \alpha^{-2d} \left(2\alpha^2 - 1\right)^{d/2}$ where $d = 1$. The red curve represents the $\alpha = 1$ case, and hence corresponds to the largest c value of the curves, namely $c = 1$.

from which we can read off

$$\Psi^2(\boldsymbol{\omega}) = (2\pi\imath \|\boldsymbol{\omega}\|)^{2m} .$$

where Ψ^2 *is an informal notation for* $\mathcal{F}_{\boldsymbol{x}}[\psi^* \psi \delta(\boldsymbol{x})]$. *In this case we were able to solve* (3.7) *only for the three dimensional case — fortunate as this is probably the most useful case in computer graphics etc. The resulting expression is rather unwieldy however so we only give an implementation of it in the C language in section C.1, as well as a plot in Figure 3.3.*

Direct Exploitation of Symmetry

The previous Fourier transform oriented approach will not always lead to integrals which are known in closed form — for example in Example 3.1.4 we could not find a closed form expression for the four dimensional case. As such we are forced to resort to numerical integration methods, for which it can be useful to make the following simplifications which as we shall see basically consist of writing out the limits of integration of the intersection of two circles.

Firstly, since the basis functions f_i are radial, and provided that the regularisation operator ψ is translation invariant, then for any $d \geq 3$ we may use symmetry to reduce the problem to a two dimensional integral.

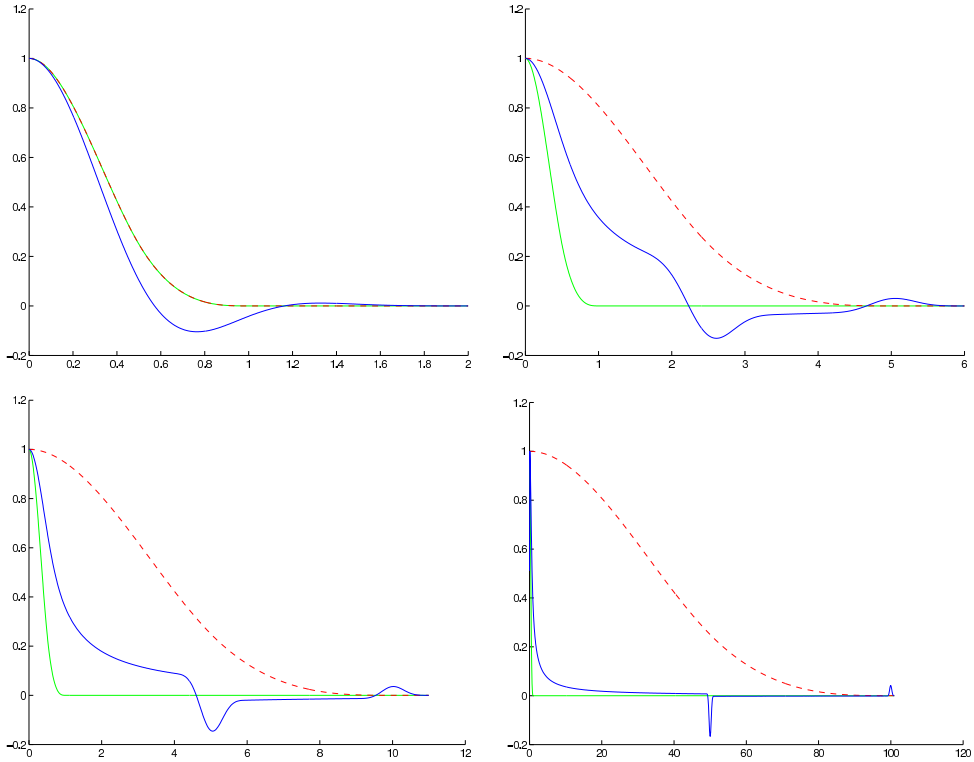


Figure 3.3: One of the most useful achievements of the present thesis is probably the derivation of closed form expressions for the above functions — the expression for the blue line is given in section C.1. The green and dashed-red lines are $\phi_r(x/s_j)$ and $\phi_r(x/s_k)$ plotted against x, where ϕ_r is the B_3-spline of (3.13). The support parameter s_k is, from top-left to bottom-right, 1, 5, 10 and 100, whereas s_j fixed at 1. The blue lines denote the inner product $\langle \psi f_j, \psi f_k \rangle_{L_2}$, where ψ is the $m = 2$ thin-plate regularisation operator and $f_i = \phi_r(\|\cdot - \boldsymbol{v}_i\|_{\mathbb{R}^3} / s_i)$, as per Example 3.1.4. The blue lines are plotted as a function of the distance between the maxima of the two basis functions in $\mathbb{R}^3$, *i.e.* the horizontal axis denotes $\|\boldsymbol{v}_j - \boldsymbol{v}_k\|_{\mathbb{R}^3}$.

Assuming without loss of generality that

$$f_i(\cdot) = g_r(\|\cdot\|)$$

as well as

$$f_j(\cdot) = h_r(\|\cdot - \boldsymbol{d}_1\|)$$

where

$$\boldsymbol{d}_1 = (\|\boldsymbol{v}_i - \boldsymbol{v}_j\|, 0, \ldots, 0)^\top \in \mathbb{R}^d,$$

and then by symmetry we have

$$\langle \psi f_i, \psi f_j \rangle_{L_2} \;\propto\; \int_{x_1=-\infty}^{\infty} \int_{x_2=-\infty}^{\infty} x_2^{d-1} Q(x_1, x_2, d_1) \mathrm{d}x_1 \mathrm{d}x_2 \;\triangleq\; I$$

where

$$Q(x_1, x_2, d_1) \triangleq (\psi f_i)\left(\left\|(x_1, x_2, 0 \ldots, 0)^\top\right\|\right)(\psi f_j)\left(\left\|(x_1 - d_1, x_2, 0 \ldots, 0)^\top\right\|\right).$$

Since the f_i are smooth and compactly supported, the above integral can be computed rather easily using standard numerical integration techniques. Moreover we can take advantage of the known supports of f_i and f_j to decompose the integral into a more manageable form. Assuming without loss of generality that $s_i, s_j > 0$ and $s_i \geq s_j$, two cases must be considered:

1. That the support of f_i is a subset of that of f_j.
2. That the two supports overlap only partly.

The first case occurs when $d_1 + s_j \geq s_i$, whereupon we have

$$\begin{aligned} I =& 2 \int_{z=-s_j}^{x_{\text{int}}} \int_{x_2=0}^{x_g} x_2^{d-1} Q(x_1, x_2, d_1) \mathrm{d}x_1 \mathrm{d}x_2 \\ &+ 2 \int_{x_1=x_{\text{int}}}^{s_i} \int_{x_2=0}^{x_f} x_2^{d-1} Q(d_1 - z, x_2, d_1) \mathrm{d}x_1 \mathrm{d}x_2, \end{aligned}$$

where we have defined

$$\begin{aligned} x_{\text{int}} &= \frac{d_1^2 + s_i^2 - s_j^2}{2r}, \\ x_f &= \sqrt{s_i^2 - x_1^2}, \end{aligned}$$

and

$$x_g = \sqrt{s_j^2 - (x_1 - r)^2}.$$

The second case occurs when $d_1 + s_j < s_i$, in which case we have

$$I = 2 \int_{x_1=r-s_j}^{r+s_j} \int_{x_2=0}^{x_f} x_2^{d-1} Q(x_1, x_2, d_1) \mathrm{d}x_1 \mathrm{d}x_2.$$

3.1.3 Interpretation as a Gaussian Process

So far, we have motivated our approach as the minimiser of a Tikhonov regularised objective function within the span of some ad-hoc function basis. If we consider these quantities to be log-likelihoods then we can take a probabilistic view — in fact, using ideas from [QCR05] we now demonstrate that the approximation we have developed is equivalent to inference in an exact Gaussian process (g.p.) with a degenerate[1] covariance function depending on the choice of function basis.

Placing a multivariate Gaussian prior over the coefficients in (3.2), namely $\boldsymbol{\pi} \sim \mathcal{N}(\mathbf{0}, F_\psi^{-1})$, we see that f obeys a zero mean g.p. prior — writing $[\boldsymbol{f}_x]_i = f(\boldsymbol{x}_i)$ and denoting expectations by $E[\cdot]$ we have for the covariance

$$\begin{aligned} E\left[\boldsymbol{f}_x \boldsymbol{f}_x^\top\right] &= F_x E\left[\boldsymbol{\pi}\boldsymbol{\pi}^\top\right] F_x^\top \\ &= F_x F_\psi^{-1} F_x^\top \end{aligned}$$

Now, assuming an independent and identically distributed (i.i.d.) Gaussian noise model with variance σ^2, defining $[\boldsymbol{f}_t]_j = f_j(\boldsymbol{t})$ (don't be confused by the fact that $\boldsymbol{f}_t$ and $\boldsymbol{f}_x$ are do not have analogous meaning), and retaining our previous definition of F_x, we can immediately write the joint distribution between the observation at a test point $\boldsymbol{t}$, that is $y_t \sim \mathcal{N}(f(\boldsymbol{t}), \sigma^2)$ and the vector of observations at the $\boldsymbol{x}_i$, namely $\boldsymbol{y}_x \sim \mathcal{N}(f_x, \sigma^2 I)$, which is

$$p(\boldsymbol{y}_x, y_t) = \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \left(F_x F_\psi^{-1} F_x^\top + \sigma^2 I\right) & F_x F_\psi^{-1} \boldsymbol{f}_t \\ \boldsymbol{f}_t^\top F_\psi^{-1} F_x & \left(\boldsymbol{f}_t^\top F_\psi^{-1} \boldsymbol{f}_t + \sigma^2 I\right) \end{pmatrix}\right).$$

The posterior distribution is therefore itself Gaussian,

$$p(y_t|\boldsymbol{y}_x) \sim \mathcal{N}\left(\mu_{y_t|\boldsymbol{y}_x}, \Sigma_{y_t|\boldsymbol{y}_x}\right).$$

Using (A.7) for the marginals of the multivariate Gaussian followed by the Matrix inversion lemma (A.1) we can derive mean of the posterior

$$\begin{aligned} \mu_{y_t|\boldsymbol{y}_x} &= \left(F_x F_\psi^{-1} \boldsymbol{f}_t\right)^\top \left(F_x F_\psi^{-1} F_x^\top + \sigma^2 I\right)^{-1} \boldsymbol{y} \\ &= \boldsymbol{f}_t^\top \left(\sigma^2 F_\psi + F_x^\top F_x\right)^{-1} F_x^\top \boldsymbol{y}. \end{aligned}$$

By comparison with (3.4) we can see that the mean of the posterior distribution is identical to the approximate regularised solution. For the corresponding posterior variance we have

$$\begin{aligned} \Sigma_{y_t|\boldsymbol{y}_x} &= \left(\boldsymbol{f}_t^\top F_\psi^{-1} \boldsymbol{f}_t + \sigma^2\right) - \left(\boldsymbol{f}_t^\top F_\psi^{-1} F_x\right)\left(F_x F_\psi^{-1} F_x + \sigma^2 I\right)^{-1}\left(F_x F_\psi^{-1} \boldsymbol{f}_t\right) \\ &= \sigma^2 \boldsymbol{f}_t^\top \left(\sigma^2 F_\psi + F_x^\top F_x\right)^{-1} \boldsymbol{f}_t + \sigma^2. \end{aligned}$$

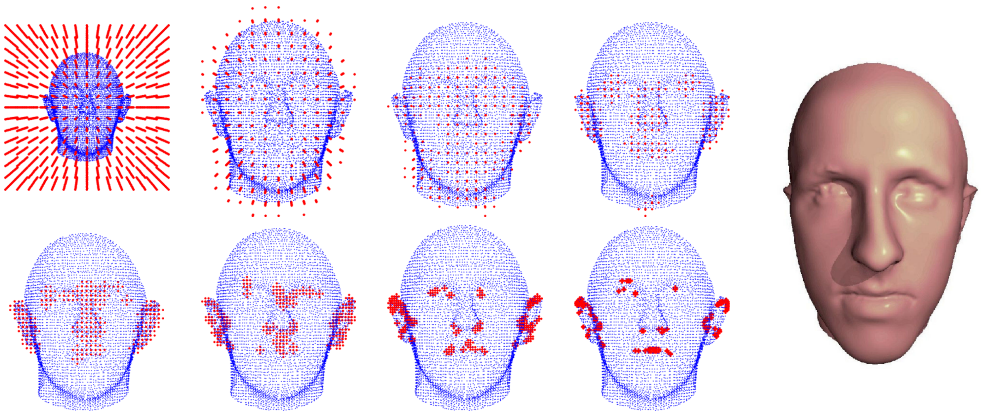


Figure 3.4: An input data set taken from a light based depth scanner (blue) with the basis centres produced by Algorithm 3.1 (red) (basis function support decreasing from top-left to bottom-right), and a rendering of the fit surface using the technique described in section 4.3 (which appears elongated due to the difference in image plane projection method). The parameters of Algorithm 3.1 were $\epsilon = 1/50$, $s = 1/5$, $t = 13/10$ and $r = 1/3$.

3.1.4 Construction of the Function Basis

The success of the above scheme hinges on the ability to construct a good function basis, *i.e.* one that

1. Spans a set of functions containing good solutions to *e.g.* the k.r.r. system given by (2.16) (in particular having a support which covers a given region of interest — here assumed to be the unit hyper-box).

2. Leads to fast solution of (3.4).

3. Can be constructed and evaluated quickly.

The construction of the function basis is rather problem dependent — we now consider the implicit surface reconstruction problem of Definition 4.1.1. To this end, we make use of some standard ideas for 3D point cloud simplification [PGK02]. The main idea that we use is that the closer a set of points within a given region is to being linear, the less information is required to sufficiently describe the shape within that region. The way in which we apply this idea for the present task is described precisely by the pseudo-code of Algorithm 3.1. The basic idea can be seen more easily in Figure 3.4 however, and the remainder of this sub Section could be safely skipped at first reading.

Note that although Algorithm 3.1 is only intuitively justified, we will later minimise a well justified objective function (*e.g.* (2.12)) within the span of the resultant basis functions. The results of the overall algorithm therefore

[1] Degenerate in the sense that the corresponding r.k.h.s. is finite dimensional

Algorithm 3.1: $\mathcal{B} = \text{MakeBasis}\,(\mathcal{X}, \epsilon, s, t, r)$
input:
data $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^d\}_{1 \le i \le m}$ (scaled to lie within the unit hyper-box)
curvature tolerance $\epsilon \in [0, 1]$
initial basis support $s \in \mathbb{R}^+$
basis support decrement ratio $t > 1$
basis grid width to basis support ratio $r \in [0, 1]$
output:
basis function (centre,support) pairs $\mathcal{B} = \{(\boldsymbol{v}_k, s_k) \in \mathbb{R}^d \times \mathbb{R}^+\}$

```
X̃ ← X
B ← {(v, s) : v ∈ grid({x : ‖x‖_∞ ≤ 1}, rs)}
while X̃ ≠ ∅ do
  s ← s/t
  for all x ∈ X̃ do
    if curvature({x' ∈ X : ‖x' − x‖_2 < s}) < (ε/d) then
      B ← B ∪ {(v, s) : v ∈ grid({x}, rs)}
      X̃ ← X̃ \ x
    end if
  end for
end while
```

do not depend too strongly on the output of Algorithm 3.1, in the sense that adding more basis functions can only make the overall solution better. The algorithm makes use of two auxiliary functions, *curvature* and *grid*. The first of these functions, $curvature(\mathcal{X})$, used also in [PGK02], returns the ratio of the smallest to the sum of all of the eigenvalues of the covariance matrix of the set $\mathcal{X} = \{\boldsymbol{x}_i \ \in \mathbb{R}^d\}_{1 \le i \le p}$. If we define the covariance matrix as $C = DD^\top$ where

$$D = \left[\boldsymbol{x}_1 - \overline{\mathcal{X}}, \boldsymbol{x}_2 - \overline{\mathcal{X}}, \ldots, \boldsymbol{x}_p - \overline{\mathcal{X}}\right],$$

and $\overline{\mathcal{X}}$ is the empirical mean of $\mathcal{X}$, then letting the diagonal matrix $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots \lambda_d)$ and the ortho-normal matrix V be the solutions to the eigen-system $CV = \Lambda V$, the function returns

$$\text{curvature}(\mathcal{X}) = \frac{\min_{i \in 1 \ldots d} |\lambda_i|}{\sum_{j=1}^{d} |\lambda_j|},$$

which is a real number in $[0, 1/d]$ that is smaller for point sets that lie closer to a linear manifold. The second function, $grid(\mathcal{S}, w)$ returns points from a grid of spacing w immediately surrounding the set $\mathcal{S} \subset \mathbb{R}^d$, that is

$$\text{grid}(\mathcal{S}, w) = \text{dilation}(\mathcal{S}, w) \bigcap \left\{w\boldsymbol{z} : \boldsymbol{z} \in \mathcal{Z}^d\right\},$$

where $\text{dilation}(\mathcal{S}, w)$ is the union of the set of hyper-spheres centered at all elements of $\mathcal{S}$ (in other words, a *dilation* of $\mathcal{S}$):

$$\text{dilation}(\mathcal{S}, w) = \left\{\boldsymbol{x} \in \mathbb{R}^d : \left(\min_{\boldsymbol{x}' \in \mathcal{S}} \|\boldsymbol{x} - \boldsymbol{x}'\|\right) \le w\right\}.$$

3.2 Fast Multipole Method

We believe that the scheme presented in Section 3.1 is useful for a wide range of problems, particularly in computer graphics. Specifically, it can be applied to any problem involving large amounts of fairly low-dimensional data (say, $d \lesssim 4$) in which it is necessary to estimate a regularised function that either satisfies some set of constraints or minimises some cost function.

A very different and important approximation scheme based on the f.m.m. [BG97] has been applied to the implicit surface reconstruction problem in [CBC+01] however, so we now take the time to review this approach. The f.m.m. is an elegant algorithm which, among other accolades, won its authors the *Steele prize for seminal contribution to research* in 2001. It is useful for the implicit surface reconstruction problem because it allows the k.r.r. solution to be constructed and evaluated quickly, although it was originally designed for solving n-body problems in computational physics. The algorithm efficiently approximates to an arbitrary precision summations of the form

$$s(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i). \tag{3.14}$$

Obviously this allows fast evaluation of a k.r.r. solution, but what about constructing this solution? To do this it is necessary to solve the system

$$\boldsymbol{y} = (K + \sigma^{-2} I)\boldsymbol{\alpha}, \tag{3.15}$$

(*cf.* (2.17)) for the coefficients vector $\boldsymbol{\alpha} \in \mathbb{R}^m$. For large m, solving the system directly is impossible due to the $O(m^3)$ time and $O(m^2)$ memory requirements [GV96]. By employing a conjugate gradients (see *e.g.* [GV96]) type of iterative solver however, it is possible to obtain an approximate solution in a more efficient manner. This is because because such iterative solvers require only matrix-vector products — which essentially involves computing sums of precisely the form which the f.m.m. is designed to efficiently approximate. Indeed, the f.m.m. approximation allows one to evaluate such summations to an arbitrary precision in time $O(\log(m))$, or in some cases $O(1)$ — an improvement over the naive $O(m)$. Thus, an approximate solution of the system can be attained in time $O(cm\log(m))$, where c is the number of iterations required by the conjugate gradients solver. For the remainder of the present chapter we provide a short introduction the f.m.m. Our goal is not to provide a thorough understanding as this is beyond the present scope — instead we aim to provide only those details necessary to understand the relative pros and cons of the f.m.m. in comparison to the fast scheme we developed in section 3.1 — for a more thorough exposition we recommend [BG97] as a starting point.

3.2.1 The Basic Idea — Unipole Expansion

Assume that we can expand k in the form

$$k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{q=1}^{r} \phi_q(\boldsymbol{x})\psi_q(\boldsymbol{y}) + R_r(\boldsymbol{x}, \boldsymbol{y})$$

where R_r is a series remainder term which tends to zero for $\|\boldsymbol{x} - \boldsymbol{y}\|_2 \to \infty$ or for $r \to \infty$. In the f.m.m. literature this is known as a *far-field* expansion of k about the *source* $\boldsymbol{y}$. We can then rewrite (3.14) in the form

$$\begin{aligned} s(\boldsymbol{x}) &= \sum_{i=1}^{m} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) \\ &= \sum_{i=1}^{m} \alpha_i \sum_{q=1}^{r} \phi_q(\boldsymbol{x})\psi_q(\boldsymbol{x}_i) + \sum_{i=1}^{m} \alpha_i R_r(\boldsymbol{x}, \boldsymbol{x}_i) \\ &= \sum_{q=1}^{r} \phi_q(\boldsymbol{x}) \underbrace{\sum_{i=1}^{m} \alpha_i \psi_q(\boldsymbol{x}_i)}_{\triangleq \beta_q} + \sum_{i=1}^{m} \alpha_i R_r(\boldsymbol{x}, \boldsymbol{x}_i). \end{aligned} \tag{3.16}$$

Hence, by precomputing each the β_q at a cost of $O(mr)$, the approximation

$$\tilde{s}(\boldsymbol{x}) = \sum_{q=1}^{r} \beta_q \phi_q(\boldsymbol{x})$$

can be computed in $O(r)$ time. Moreover we have the error bound

$$|s(\boldsymbol{x}) - \tilde{s}(\boldsymbol{x})| \leq \|\boldsymbol{\alpha}\|_1 \max_{i=1\ldots m} |R_r(\boldsymbol{x}, \boldsymbol{x}_i)|,$$

which is small if $\boldsymbol{x}$ is sufficiently far from the sources $\boldsymbol{x}_i$. This is known as a *unipole* expansion of k for the given set of sources $\boldsymbol{x}_i$. The unipole expansion leads to an efficient approximation when $\boldsymbol{x}$ is sufficiently far from the sources for the expansion to be truncated at a level $r \ll m$ while still retaining an acceptable bound on the error.

For the case of k.r.r. however, we need to evaluate (3.14) at each of the $\boldsymbol{x}_i$, which in the context of the f.m.m. means that the evaluation points will typically be close to some of the centres, requiring a more elaborate approach. This involves a space subdivision scheme which is the subject of the next section.

3.2.2 Space Subdivision and the Multipole Expansion

To obtain the best efficiency/accuracy trade-off, the unipole scheme needs to be applied in an adaptive manner — if a subset of the sources are far from

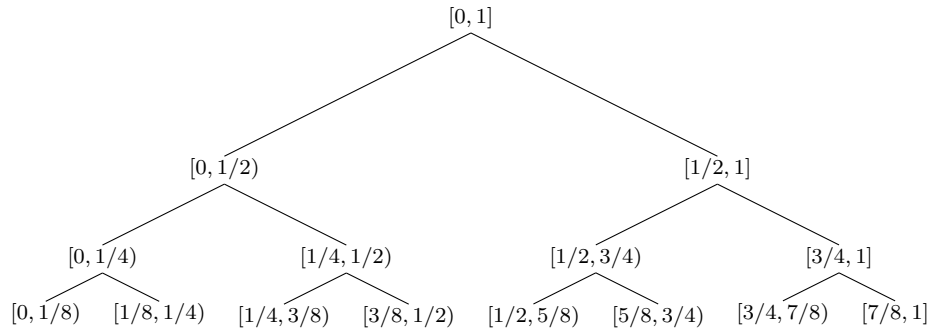


Figure 3.5: A binary tree structure induced by a uniform subdivision of the unit interval — in the parlance of the f.m.m.each node of the tree represents one panel.

the evaluation point, they can be replaced by a unipole expansion. Nearer points need to be grouped into smaller sets if they are to be replaced by unipole expansions, and some very near points must be evaluated in the direct manner.

Typically this is done by defining a hierarchical subdivision of the space and for each subdivision pre-computing the unipole expansion for those sources lying within it. A simple example is given in Figure 3.5, which depicts a uniform subdivision of the unit interval. Let us refer to the nodes of the tree as *panels*. With each panel T we associate the part of s due to the sources in T by writing with a somewhat sloppy notation

$$s_T(\boldsymbol{x}) = \sum_{x_i \in T} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i).$$

The truncated far-field approximation of s_T is denoted by $\tilde{s}_T$. Given such a tree, the cost of pre-computing the far-field expansions for all of the panels directly is $O(rm \log(m))$, but we shall improve on this in the next section.

Let us now present a simple approximation scheme which takes advantage of the space subdivision, based on the idea of well-seperatedness. We say that a point is *well-separated* from a panel T if it has a distance to T of at least $\mathrm{diam}(T)$, and that a panel U is well separated from T if all of the points in U are well separated from T. Now, to evaluate the approximation at a point $\boldsymbol{x}$, we take the leaf-node panel $T_{\boldsymbol{x}}$ which contains $\boldsymbol{x}$, as well as all of the spatial (leaf node) neighbours of $T_{\boldsymbol{x}}$, and compute the exact terms s_T for each of these panels, summing the results. Then, only if a panel is well-separated from $T_{\boldsymbol{x}}$ do we use its far-field approximation. The procedure is explained more succinctly by Algorithm 3.2.

For example, applying Algorithm 3.2 to the subdivision of Figure 3.5, for

Algorithm 3.2:
$\tilde{s}(\boldsymbol{x}) = \text{EvaluateFMM}(\boldsymbol{x}, T)$
input:
evaluation point $\boldsymbol{x}$
panel T
output:
f.m.m.approximation $\tilde{s}(\boldsymbol{x})$

```
if T is well separated from x then
  return s̃_T(x)
else
  if T is a leaf then
    return s_T(x)
  else
    call EvaluateFMM with x and each of the children of T
    return the sum of the results
  end if
end if
```

an evaluation point $x \in [3/8, 1/2)$ leads to the approximation

$$\begin{aligned}\tilde{s}(x) =& s_{[3/8,1/2)}(x) + s_{[1/4,3/8)}(x) + s_{[1/2,5/8)}(x) \\ &+ \tilde{s}_{[0,1/8)}(x) + \tilde{s}_{[1/8,1/4)}(x) + \tilde{s}_{[5/8,3/4)}(x) + \tilde{s}_{[3/4,1]}(x).\end{aligned}$$

Let's now estimate the computational time complexity of Algorithm 3.2. If we choose the number of levels m of the binary space subdivision to be $\approx \log_2(m)$ then each leaf panel contains $O(1)$ sources. Since we need evaluate only three such leaf nodes by the direct method the total cost for the direct part is $O(1)$. For the far-field part, we need evaluate at most 3 panels of each level of the tree — otherwise the panels would be well separated at the next level down the tree. The cost for the far-field part is therefore $O(r \log(m))$, where r is the number of terms in the truncated series. Since r dictates the precision of the approximation, we can say that for a fixed precision the complexity is $O(\log(m))$.

Algorithm 3.2 is very simple, and although it does afford an order of magnitude computational advantage in comparison with the direct method, a number of improvements are possible, which we survey in the next section.

3.2.3 Improvements

Data Structure.

An obvious improvement can be made to the spatial data structure, which should not be constructed uniformly. Instead the structure should refine to finer subdivisions where there is more data, and the leaf nodes should be shrunk to the minimum size which contains the corresponding sources. Various structures such as *bd*-trees have been used — for a review see *e.g.* [LMGY04].

Tree Construction.

Rather than constructing the far-field coefficients for each of the panels individually this can be done in the following more efficient manner. First, the far field expansions for the leaf nodes are constructed normally. Then, for panels in subsequent levels of the tree the far-field expansions are constructed based only on the far-field expansions the child panels, along with the sources within the panel itself. Obviously some more analytical results (*i.e.* in addition to the forms of ϕ, ψ and R_r in (3.16)) are required for this. The complexity of this new tree construction algorithm depends on the how the far-field expansions are communicated to the parents, which introduces a dependency on the dimension of the space d as well as the truncation order r. Fixing d and r however, leads to a complexity of $O(m)$ [BG97].

Efficient Evaluation.

If evaluation time is more important than the time required to construct the tree, it is possible to sacrifice the latter for the former. In particular, for each panel one may conglomerate the far-field terms of all well separated panels into a single approximation valid only for the cell itself. This setup procedure now costs $O(m \log(m))$ — but remarkably the evaluation involves the $O(1)$ part for evaluating the near panels, and now only $O(1)$ for all of the far-field terms [BPT06]. This is the method which has been employed to excellent effect in the FastRBF$^{\text{TM}}$ toolbox [CBC^{+}01].

Dual Trees.

It is not necessary to treat each evaluation point separately. Savings may be possible by constructing a tree-like spatial data structure for the evaluation points and then treating groups of evaluation points at the same time. In the context of f.m.m. this seems to be a new idea which has yet to be properly tested and understood. For more information we refer the reader to [LGM06] which, although lacking in theoretical insight and computational complexity estimates, seems to be the only work available to date.

3.3 Comparing the Two

For the problem of solving large thin-plate spline k.r.r. problems, the inherent computational difficulty is due to the fact that, because the kernel function does not decay with distance (*cf.* (2.29)), all points interact strongly with one another. Both methods overcome this by local processing strategies. The new method does this by constructing compactly supported basis functions, and then transmitting the regularisation information in a domino

like manner by way of the inner products which are non-zero for overlapping basis functions only. The f.m.m. transforms the problem to a local one by way of the method described in the last part of the previous section, entitled *Efficient Evaluation*.

We now compare the f.m.m. to the method of section 3.1 (which we refer to as the new method). A plus (minus) sign indicates a factor favourable for the new method (the f.m.m.).

+ Compression

For the new method we need only store the basis centers (p points in $\mathbb{R}^d$) along with the coefficients $\pi_1, \ldots \pi_m$ and dilations $s_1, \ldots, s_m$. In fact, only a small number of unique dilations are produced by Algorithm 3.1, making the storage cost for the $s_1, \ldots, s_m$ negligible. In contrast, the f.m.m. approach retains all of the input points. In our experiments with implicit surfaces, the number of basis functions p was typically considerably less than the number of input points, leading to a compression of the data — see Table 4.1.

+ Functionals easy to apply

With the new method, various functionals can be applied to the function (*e.g.* the gradient operator, as used heavily in section 4.3) with minimal additional implementation effort. The non-trivial part is done by the range search algorithm, for which many well developed software libraries exist. Hence, evaluating (3.2) can be done efficiently by using a range search library to identify the contributing basis functions, over which the summation is computed directly. Modifying this to give the gradient of the function, for example, is no more difficult than implementing the gradient computation for a single basis function. In contrast, the f.m.m. method requires a far greater additional effort.

+ Simplicity

The new method is both conceptually simpler and easier to implement. This statement can be justified in various ways. First, we manage to explain the new method fairly thoroughly in a little over ten pages in section 3.1. In contrast, explaining and deriving the necessary tools for the f.m.m. approach for thin-plate splines in $\mathbb{R}^3$ requires some forty odd pages in [BPT06], and that only for a special case and without discussing preconditioning. Note that in the f.m.m. approach to k.r.r. with the thin-plate spline, in addition to the f.m.m. machinery itself it is necessary to employ preconditioning strategies due to the notoriously ill-conditioned nature of the spline interpolation equations, further adding to the overall complexity [MBC99]. Additional

evidence of the difficulty of implementing the f.m.m. approach is that despite its effectiveness, it currently only exists in the proprietary FastRBF^{TM} toolbox.

- Generality

The new method is specific to regularised interpolation problems, and is not useful for computing sums of the form (3.14). Moreover, the effectiveness of the scheme is somewhat limited by the capacity to derive closed form expressions for the regulariser inner products (*cf.* Example 3.1.4). We say somewhat, because in practice it is rather straightforward to construct a numerical approximation for this integral and then cache the resulting values for later use. Such a caching scheme is effective for bases constructed by Algorithm 3.1 for example, as there are only a finite number of unique distances and dilations to consider.

- Error Bounds

A great feature of the f.m.m. is the ability for the user to specify the desired accuracy. No such guarantee is made by the new method. For work on interpreting the bound on the evaluation error provided by the f.m.m. when coupled with the approximate solver of the k.r.r. linear system, see *e.g.* [SS03, FWML06].

Chapter 4

Implicit Surface Reconstruction

In section 3.1 of the previous chapter, we developed an approximation to kernel ridge regression (k.r.r.) which, in comparison to the naive method, scales better with the number of data points, but worse with respect to the input space dimension. As such the technique is especially well suited to a wide range of computer graphics problems. In the present chapter we consider one such problem — the one which motivated us to develop the approximation method — namely that of implicit surface reconstruction.

The chapter begins with the necessary background to the problem in section 4.1, and goes on to present three algorithms for solving the problem. We summarise these three algorithms in subsection 4.1.5 before presenting them individually in Sections 4.2 to 4.4.

4.1 Background

4.1.1 Surface Reconstruction

The problem of reconstructing (or inferring) a surface from a set of points frequently arises in computer graphics. The problem as we will approach it is stated more precisely in the following

Definition 4.1.1. The **surface reconstruction problem** consists of inferring a co-dimension one[1] manifold $\mathcal{M} \subset \mathbb{R}^d$ from a finite sampling

$$\mathcal{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_m \in \mathcal{M}\},$$

given in the form of corresponding noisy observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$. We say that the problem is **with normals** if one is also given a set of vectors $\boldsymbol{n}_1, \ldots, \boldsymbol{n}_m \subset \mathbb{R}^m$, with the property that $\boldsymbol{n}_i$ is the consistently oriented normal vector of $\mathcal{M}$ at $\boldsymbol{s}_i$, observed with noise.

[1] *i.e.* a manifold of dimensionality one less than that of the space in which it lies.

The surface reconstruction problem poses various theoretical and practical problems. Obviously, the problem is ill-posed, necessitating the use of prior knowledge (such as *e.g.* a prior distribution over manifolds $\mathcal{M}$). The noisiness of the input data varies greatly depending on how it was acquired (see subsection 4.1.2), as does the distribution of the samples $\mathcal{S}$. In particular, significant regions may be completely unsampled, necessitating what the computer graphics community refers to as *hole filling*. Finally, in computer graphics problems one is typically faced with a large number of samples (currently, say, tens of millions), precluding the use of algorithms which scale poorly in m.

4.1.2 Data Acquisition

Numerous methods of sampling physical surfaces are now available, which we now briefly review in order to better understand the data they provide. Before proceeding let us make a short note on the availability of normal vectors.

Given a point cloud, it is usually possible to estimate the normal vector of the scanned surface at a given point in an un-oriented manner (*i.e.* up to a sign change), based on the local neighbourhood of points. Of course, this requires that the sampling is sufficiently dense and free of noise for the surface to vary smoothly between samples. As we shall see, given this un-oriented normal vector, it is usually possible to take advantage of knowledge of the scanning setup in order to orientate the normal vectors, *e.g.* such that they all point outwards. In particular this poses no problem in computer graphics scanning setups in which the manifold is sampled sufficiently densely and with sufficiently low noise. For example, if a laser light is shined on the subject, then the direction of the beam indicates the orientation of the surface.

Passive Optical Scanning

The least expensive methods tend to be the passive optical ones, as they usually only require one or more digital cameras. Perhaps the most common approach is to mimic the human stereoscopic vision system. This can be done with two cameras along with a *correspondence finding* algorithm which attempts to locate points in both images that correspond to the same location on the subject. Given such correspondences one may then solve for a 3D location by the process of *triangulation*.

Less popular are silhouette based methods which typically photograph the subject from various angles, and then determine the shape using some *space carving* algorithm — naturally concave subjects are problematic for this approach.

Active Optical Scanning

Both normal and laser light can be used actively for scanning. One widespread approach is to shine a laser onto the subject, take a picture from a different angle, and then use the process of triangulation to solve for the location being illuminated. This highly accurate method is popular for constructing models of small (of the order of a few metres) objects.

Alternatively, *time of flight* methods scan the subject with a *laser range finder* — a radar like device which determines the distance to the object based on the round-trip time of a pulse of light. This method is less accurate than the laser triangulation method, but has the advantage of allowing a greater distance to the subject, and is particularly useful for *e.g.* aerially scanning the Earth's surface.

Active systems also exist that use normal (non-laser) light. The principle is similar to the laser triangulation method, except that the scanning is typically not done on a point by point basis. For example, one may project a pseudo-random light pattern onto the subject and then apply image processing and triangulation. This can be useful for scanning moving objects such as a person's face.

Mechanical Contact Scanning

Contact methods are also used, primarily in an industrial manufacturing setting. The basic idea is very simple and involves placing contact elements on the subject and measuring the location based on the displacement of the contact element. This method tends to be very accurate but rather slow and intrusive, and may damage the subject.

4.1.3 Implicit Surfaces

In order to solve the surface reconstruction problem, it is necessary to have a surface representation. Although piecewise linear representations (*i.e.* triangulated meshes) are very widely used, many other representations exist and have their own particular advantages and disadvantages. We will be mainly concerned with one particular representation, namely the implicit surface, although various other representations exist — these include *e.g.* Bernstein-Bézier representations within a Delaunay triangulated tessellation [BBX95], as well as point based representations that analyse the point cloud directly and locally as needed at rendering time [ABCO$^+$01].

Before proceeding to discuss the implicit surface representation we would like to point out that there exists a class of algorithms which are often referred to as implicit or level-set methods but that are, contrary to the definition we will give shortly, defined numerically on a grid rather than by a continuous analytic function [Set98, WB98]. Although we will not concern

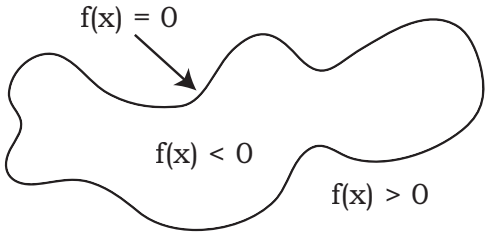


Figure 4.1: An implicit surface representation with embedding function f, of a curve in 2D.

ourselves further with these methods we do not wish to suggest that they are not useful or interesting.

Implicit Surface Representation

Implicit surfaces (or simply *implicits*) are defined by way of a continuous *embedding function* $f : \mathbb{R}^d \to \mathbb{R}$. As depicted in Figure 4.1, the surface is the zero level set of f, *i.e.*

$$f^{-1}(0) = \left\{ \boldsymbol{x} \in \mathbb{R}^d : f(\boldsymbol{x}) = 0 \right\}.$$

We shall assume that $f^{-1}(0)$ is of bounded, and we shall refer to the set

$$f^{-1}(\mathbb{R}^-) = \left\{ \boldsymbol{x} \in \mathbb{R}^d : f(\boldsymbol{x}) < 0 \right\}$$

as being *interior* to the surface (we also assume that $f^{-1}(\mathbb{R}^-)$ also has finite measure). The implicit surface representation has the useful properties that

- The surface is by construction *water-tight*, which is particularly useful for the surface reconstruction problem.
- Pathological cases aside, $\text{sign}(f(\boldsymbol{x}))$ indicates whether $\boldsymbol{x}$ is interior to the surface.
- $(\nabla f)(\boldsymbol{x})$ is the normal direction of the surface at $\boldsymbol{x}$.
- Set operations between implicits (*i.e.* constructive solid geometry) are convenient, *e.g.* $\max(f_1, f_2)$ implements the intersection operation.

Next we discuss ways of solving the surface reconstruction problem using an implicit representation.

4.1.4 Implicit Surface Reconstruction

The implicit surface reconstruction problem is typically transformed into a classical regression problem of the type readily solved by *e.g.* k.r.r. (see

2.2.2) — the regressed function being the embedding function of the implicit. These methods typically construct as an intermediate step a set $(\boldsymbol{x}'_1, y_1), \ldots, (\boldsymbol{x}'_n, y_n) \subset \mathbb{R}^d \times \mathbb{R}$ of (input, value) pairs (as input for the regression algorithm) from the given set $(\boldsymbol{x}_1, \boldsymbol{n}_1), \ldots, (\boldsymbol{x}_m, \boldsymbol{n}_m) \subset \mathbb{R}^d \times \mathbb{R}^d$ of (surface sample, surface normal) pairs. For example, given unit length surface normal vectors, the FastRBF$^{\text{TM}}$ algorithm [CBC+01] (which we introduce shortly) constructs a regression problem with the following set of (input, value) pairs:

$$\bigcup_{i=1}^{m} \left\{ (\boldsymbol{x}_i, 0), (\boldsymbol{x}_i + d\boldsymbol{n}_i, y_{i+}), (\boldsymbol{x}_i - d\boldsymbol{n}_i, -y_{i-}) \right\},$$

for some heuristically chosen scalar d. The targets y_{i+} and y_{i-} are taken to be the distance to the nearest data point. These manufactured *off-surface* points may contradict one another however, and further heuristics are employed by FastRBF$^{\text{TM}}$ to detect and discard such cases.

It is possible however to use the normal vectors directly — indeed the direct use of normal vectors in a fast approximation to k.r.r. is one of the key contributions of the present thesis (see section 4.3). In any case, the problem eventually boils down to solving a large regression type problem, most commonly in $\mathbb{R}^3$. We now survey the most important means of doing this.

Partition of Unity

The idea here is to construct a set of local approximating functions and then blend them together to obtain a global approximating function. To this end, consider a bounded set $\Omega \subset \mathbb{R}^d$ and a set of compactly supported functions $w_i : \mathbb{R}^d \to \mathbb{R}^+, i = 1 \ldots q$ with the property

$$\Omega \subset \bigcup_{i=1}^{q} \text{interior}\left(\Omega_i\right)$$

where $\Omega_i \triangleq \text{supp}(w_i)$. We can define another set of functions

$$\varphi_i \triangleq \frac{w_i}{\sum_{k=1}^{q} w_k}, i = 1 \ldots q$$

which by construction have the property that

$$\sum_{i=1}^{q} \varphi_i(\boldsymbol{x}) = 1 \tag{4.1}$$

for all $\boldsymbol{x} \in \Omega$. The name *partition of unity* stems from this property (4.1). Now, given a set of local approximating functions $f_1, \ldots, f_q$ such that f_i

approximates the given data on Ω_i, we can use the φ_i to blend them together to produce an approximating function f on Ω given by

$$f = \sum_{i=1}^{q} \varphi_i f_i.$$

This scheme has been successfully applied to the implicit surface problem using *e.g.* weighted least squares polynomial fitting to construct the f_i (see *e.g.* [OBA+03, SOS04]). The method is perhaps the fastest and most scalable to date, but is only applicable to benign (low noise, densely sampled) data — we shall elaborate on these shortcomings directly in our comparison with the kernel based approach.

Kernel Methods

The implicit surface methods closest to the main one developed in the present thesis (see section 4.3) are those based on a Tikhonov regularisation framework within the reproducing kernel Hilbert space (r.k.h.s.) of a fully-supported kernel function such as the thin-plate spline. This line of research seems to have begun with the *Variational Implicits* proposed by Turk and O'Brien in 1999 [TO99]. Such methods produce excellent results, but in their basic form suffer from a cubic computational fitting cost in the number of points. Their undisputed effectiveness has led researchers to look for ways to overcome the computational problems, however. Presently, two main options exist.

The first of these uses compactly supported kernel functions, leading to fast algorithms that are easy to implement [MYC+01]. Unfortunately however, such a solution is suitable for benign data sets only. As noted in [CBC+01], compactly supported kernels "yield surfaces with many undesirable artifacts in addition to the lack of extrapolation across holes".

Kernel methods based on fully supported kernel functions seem to represent the state of the art in terms of their ability to handle noisy or missing data. This conclusion is confirmed in one of the most important works on the partition of unity approach [OBA+03], wherein it was noted that the "partition of unity method is more sensitive to the quality of input data [than] approximation and interpolation techniques based on globally-supported radial basis functions" — a conclusion corroborated by the results within a different paper from the same group [OBS03].

The second means of overcoming the aforementioned computational problems does not suffer from the same problems as compactly supported kernels, as demonstrated by the FastRBF$^{\text{TM}}$algorithm [CBC+01]. This method uses the Fast Multipole Method (fast multipole method (f.m.m.)) [GR97] (see section 3.2) to overcome the computational problems without having to

abandon the fully supported kernel. The resulting method is non-trivial to implement however and to date exists only in the proprietary FastRBF$^{\text{TM}}$ package.

The main new method which we shall propose in section 4.3 — based on the approximation scheme of section 3.1 — lies somewhere between local and global processing. Although we use only locally supported basis functions, the regulariser we employ is globally defined. It could be said that in doing so we attempt to achieve the cliched best of both worlds: by applying them in this manner compactly supported basis functions can lead to high quality implicit surface reconstruction results, but with considerably less implement than the f.m.m. based method. section 4.3 is an attempt to bring the reader to the same conclusion.

4.1.5 Overview of the Rest of the Chapter

The remainder of the chapter presents three algorithms for the surface reconstruction problem as we defined it in subsection 4.1.1:

1. A simple starting point which generalises the support vector machine (s.v.m.) classifier, such that some additional points (the surface points) are constrained to lie near the zero level set of the continuous decision function (section 4.2, based on our work in [WLK03]).

2. A state of the art method where we demonstrate the fast approximation developed in section 3.1. The method is essentially k.r.r. with gradient constraints for the surface normal vectors (section 4.3, based on our work in [WSC06]).

3. An experimental idea which avoids the requirement for surface normal vectors at the cost of solving a more difficult problem — this is k.r.r. with additional terms to *e.g.* push the function away from zero (section 4.4, based on our work in [WCS05]).

The three methods can all be considered improvements on

$$\arg\min_{f\in\mathcal{F}} \sigma^2 \|f\|_{\mathcal{H}}^2 + \sum_{i=1}^{m} f(\boldsymbol{x}_i)^2$$

... which is fortunate as the solution to the above is trivially the zero function! Thus, one may view the methods as three different means of avoiding this triviality. Note that we have denoted the hypothesis space of functions by $\mathcal{F}$ rather than $\mathcal{H}$ since the present chapter is not always concerned with r.k.h.s.'s — in particular we deviate in section 4.4.

For the remainder of the chapter we stick to the notation of Definition 4.1.1, *i.e.* $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_m \subset \mathbb{R}^d$ denote surface points and $\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots \boldsymbol{n}_m \subset \mathbb{R}^d$

denote the corresponding surface normals which, in Sections 4.2 and 4.3, are also assumed to be given.

4.2 An SVM-like Method

Let us assume that in addition to the given set of surface points, we have a set $\{(\boldsymbol{z}_j, y_j)\}_{1\leq j\leq n}$ of labelled *off-surface* points $\boldsymbol{z}_j \in \mathbb{R}^d$ with associated labels $y_j \in \{1, -1\}$, where $\boldsymbol{z}_j$ is interior to the surface of interest if $y_j = 1$, and exterior if $y_j = -1$. These off-surface point/label pairs can be taken as, *e.g.*

$$(\boldsymbol{x}_i + d\boldsymbol{n}_i, 1)\,, (\boldsymbol{x}_i - 1\boldsymbol{n}_i, -d)\,,$$

for some heuristically chosen scalar d. For example, one may fix d to some value sufficiently smaller that the smallest geometric detail of the target manifold, and then discard those off-surface point/label pairs which violate the constraint that

$$\min_{\boldsymbol{x}\in\{\boldsymbol{x}_1,\ldots\boldsymbol{x}_m\}} \|\boldsymbol{x}_i - \boldsymbol{x}\| < 0.9\, d.$$

For further details see *e.g.* [SSB05, CBC+01].

Regardless of where it comes from, our proposed method of utilising this information is to perform *hard-margin* s.v.m. classification of the $\boldsymbol{z}_j$ with a squared error penalty on the value of the function at the $\boldsymbol{x}_i$, *i.e.* we take the limit $\lambda \to \infty$ of

$$f_a = \arg\min_{f\in\mathcal{H}} \sigma^2 \|f\|^2_{\mathcal{H}} + \sum_{i=1}^{m} f(\boldsymbol{x}_i)^2 + \lambda \sum_{j=1}^{n} \max(0, y_j f(\boldsymbol{z}_j) - 1)^2 \qquad (4.2)$$

When we originally proposed this method, we suggested implementing the optimisation based on the Lagrangian dual of the above problem [WLK03]. Presently we take a simpler path toward deriving a practicable problem formulation. First, due to theorem 2.2.7 (the representer theorem), we have

$$f_a(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, \boldsymbol{x}_i) + \sum_{j=1}^{n} \beta_j k(\cdot, \boldsymbol{z}_j).$$

Now, due to the limit $\lambda \to \infty$, we can rewrite f_a as the solution to the following problem, in which the last term of (4.2) has been rewritten as a constraint

$$\min_{f\in\mathcal{H}} \ \sigma^2 \|f\|^2_{\mathcal{H}} + \sum_{i=1}^{m} f(\boldsymbol{x}_i)^2$$
$$\text{subject to } y_i f(\boldsymbol{z}_i) \geq 1, \quad i = 1 \ldots n$$

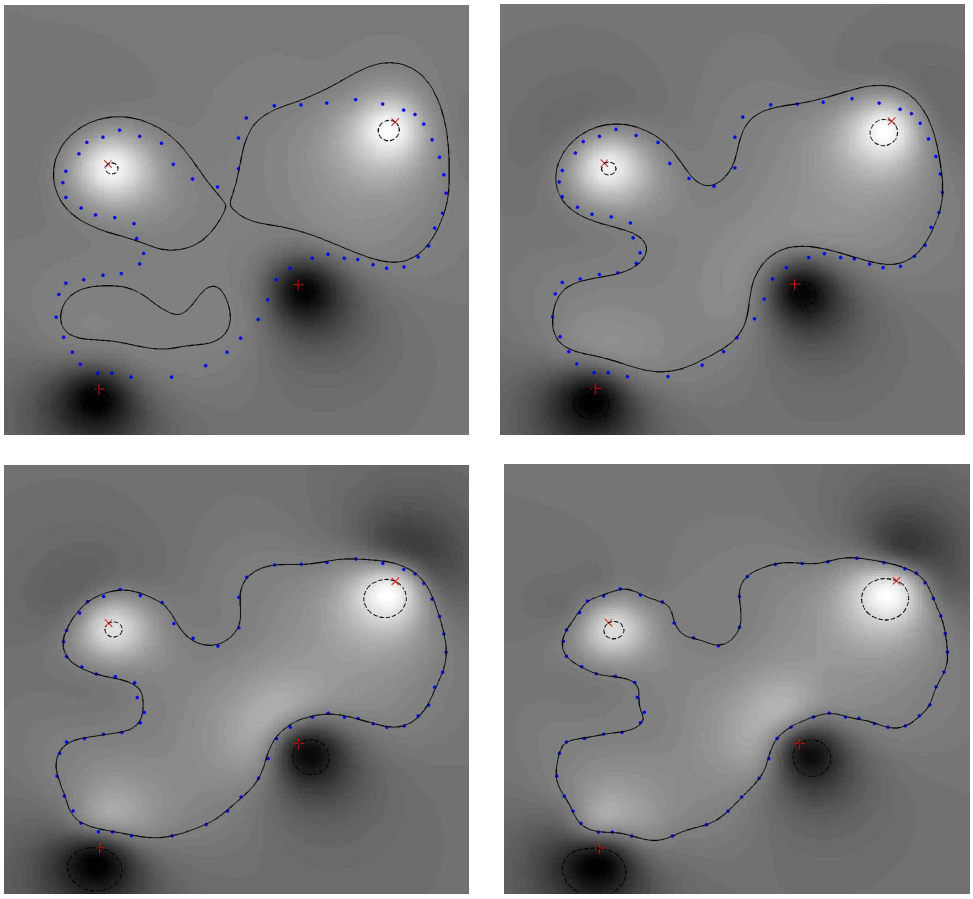


Figure 4.2: A rendering of f_a (*cf.* (4.2)) where the $\boldsymbol{x}_i$ are blue dots, the $\boldsymbol{z}_j$ are pluses and crosses for y_j plus or minus one respectively, the colour represents the value of f_a. The solid line denotes the zero level, the dashed lines the ± 1 levels. A gaussian kernel function was used, and the parameter σ decreases from top-left to bottom-right.

which is just an $(m+n)$-dimensional *quadratic programme* in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, *i.e.*

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^m,\ \boldsymbol{\beta}\in\mathbb{R}^n} \sigma^2 \begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \begin{pmatrix}K_{xx} & K_{xz}\\ K_{zx} & K_{zz}\end{pmatrix}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} + \left\| \begin{pmatrix}K_{xx}\\ K_{xz}\end{pmatrix}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}\right\|^2_{\mathbb{R}^m}$$
$$\text{subject to } \operatorname{diag}(\boldsymbol{y})\begin{pmatrix}K_{zx} & K_{zz}\end{pmatrix}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} \geq \mathbf{1},$$

where $[K_{xz}]_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{z}_j)$, *etc.* This problem can be solved by a variety of quadratic programme solvers. Note that the advantages of switching to the dual formulation as per [WLK03] are not especially compelling, as demonstrated in more recent work on primal optimisation methods for the s.v.m. [Cha07, KD05].

4.2.1 Experiments and Discussion

This is reasonable approach — in fact, the treatment of the off-surface points could be construed as an improvement over the more typical regression based methods (*cf.* subsection 4.1.1), in the sense that one need not assign regression target values. Moreover, those points for which the constraints are well satisfied, *i.e.* the set $\{\boldsymbol{z}_j : |f_a(\boldsymbol{z}_j)| > 1\}$, do not affect the solution and therefore do not impinge on the complexity of the function as measured by the norm $\|f\|^2_{\mathcal{H}}$. Moreover, the absence of these points in the final solution makes the resultant function quicker to evaluate.

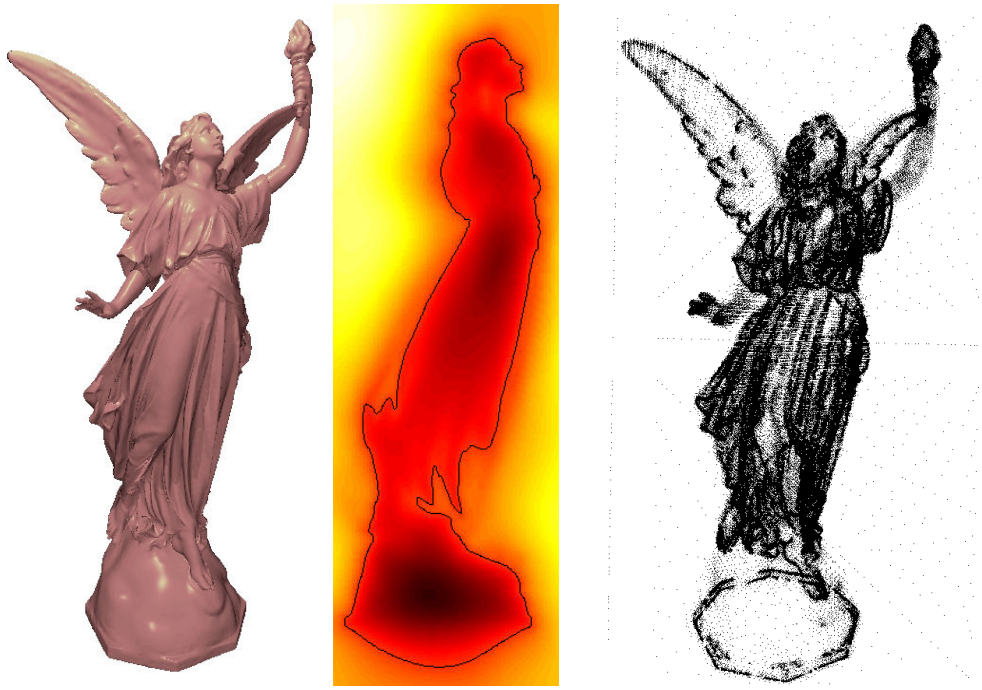
The first main problem however, is that in its basic form the time complexity of the algorithm is cubic in m — making it infeasible for realistic problem sizes. Although this issue may be overcome by the approximation scheme of section 3.1, we prefer to first correct the other main problem with the method — namely the need to construct off-surface points. This is precisely what we focus on in section 4.3.

Before proceeding however, it is interesting to compare the algorithm with the standard hard-margin s.v.m. classifier. In this perspective the surface points can be considered training examples that an expert has labelled *too hard to classify*. As we noted in [WLK03], an application for the algorithm as a data classifier is *e.g.* in the domain of handwriting recognition, in which a human could label some sample characters as being, say, *either an eight or a six*. Although this may not seem an especially promising idea, surprisingly, the algorithm has in fact since been used for a similar purpose. We are referring here to the so-called *UniverSVM* in which the $\boldsymbol{x}_i$ are points which belong to neither of the two classes to be classified — for further details see [WCS+06].

4.3 Direct Incorporation of Normal Vectors

As noted in subsection 4.1.2, most current geometrical scanning methods yield surface normal vectors at each of the surface points. We already mentioned in subsection 4.1.1 however, that all of the presently available implicit surface reconstruction methods based on Tikhonov regularisation in an r.k.h.s. require off-surface points — as does the method we suggested in the previous section 4.2.

Presently we remedy this by showing how one may incorporate normals directly without having to move away from the powerful r.k.h.s. framework. The method we propose is suggested by the fact that the normal direction of the implicit surface is given by the gradient of the embedding function — thus normal vectors can be incorporated by regression with gradient targets.



(a) **(b)** **(c)**

Figure 4.3: **(a)** Rendered implicit surface model of *Lucy*, constructed from 14 million points with normals. **(b)** A planar slice that cuts the nose — colour represents the value of the embedding function and the black line its zero level. **(c)** A black dot at each of the 364,982 compactly supported basis function centres which, along with the corresponding dilations and magnitudes, define the implicit.

We therefore propose instead to solve for

$$f_b \triangleq \arg\min_{f \in \mathcal{H}} \|f\|_{\mathcal{H}}^2 + \sigma_y^{-2} \sum_{i=1}^{m} f(\boldsymbol{x}_i)^2 + \sigma_n^{-2} \sum_{i=1}^{m} \|(\nabla f)(\boldsymbol{x}_i) - \boldsymbol{n}_i\|_{\mathbb{R}^d}^2 , \quad (4.3)$$

which uses the given surface point/normal pairs $(\boldsymbol{x}_i, \boldsymbol{n}_i)$ directly. As ought to be clear after reading chapter 2, this can be interpreted probabilistically as a *maximum a posteriori* (m.a.p.) estimate with certain Gaussian assumptions about the value and the gradient of the function at the given surface points. We now show how to solve for f_n in explicit form.

Exact Solution

We begin by giving the explicit solution for f_b without any derivation — we will give the derivation directly (*i.e.* in lemma 4.3.1) for a slightly more general case.

From theorem 2.2.7, we already know that we need only solve for m coefficients α_i as well as a further md coefficients β_{lj} to obtain the optimal

solution

$$f_n(\boldsymbol{x}) = \sum_i^m \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + \sum_{i=1}^m \sum_{l=1}^d \beta_{li} k_l(\boldsymbol{x}_i, \boldsymbol{x}), \tag{4.4}$$

where we have defined

$$k_l(\boldsymbol{x}_i, \boldsymbol{x}) \triangleq \frac{\partial}{\partial [\boldsymbol{x}_i]_l} k(\boldsymbol{x}_i, \boldsymbol{x}),$$

the partial derivative of k in the l-th component of its first argument. The coefficients $\boldsymbol{\alpha}$ and $\boldsymbol{\beta_l}$ of the solution are found by solving

$$\mathbf{0} = (K + \sigma_y^2 I)\boldsymbol{\alpha} + \sum_{l=1}^d K_l \boldsymbol{\beta}_l \tag{4.5}$$

along with, for $m = 1 \dots d$

$$N_m = K_m \boldsymbol{\alpha} + (K_{mm} + \sigma_n^2 I)\boldsymbol{\beta_k} + \sum_{\substack{l=1 \\ l \neq m}}^d K_{lm} \boldsymbol{\beta_l}, \tag{4.6}$$

where we've defined the following matrices and vectors:

$$\begin{array}{cc} [N_l]_i = [\boldsymbol{n}_i]_l; & [\boldsymbol{\alpha}]_i = \alpha_i \\ [\boldsymbol{\beta}_l]_i = \beta_{li}; & [K]_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ [K_l]_{i,j} = k_l(\boldsymbol{x}_i, \boldsymbol{x}_j)\ ; & [K_{lm}]_{i,j} = k_{lm}(\boldsymbol{x}_i, \boldsymbol{x}_j). \end{array}$$

Note that the k_{lm} are the second derivatives of $k(\cdot, \cdot)$, defined similarly to the first. Rather than deriving these expressions in a direct manner, it is simpler to consider the more general

Lemma 4.3.1. *Denote by $\mathcal{X}$ a non-empty set, by k a reproducing kernel (r.k.) with r.k.h.s. $\mathcal{H}$, by $(\boldsymbol{z}_1, y_1) \dots (\boldsymbol{z}_p, y_p) \subset \mathcal{X} \times \mathbb{R}$ a training set, by $\sigma_1, \dots \sigma_p \subset \mathbb{R}$ a set of noise variances, and by $L_1, \dots L_p$ a set of linear operators $\mathcal{H} \to \mathcal{H}$. The minimiser $f \in \mathcal{H}$ of the Tikhonov regularised risk functional*

$$\mathcal{O} \triangleq \|f\|_{\mathcal{H}}^2 + \sum_{i=1}^p \sigma_i^{-2} \left((L_i f)(\boldsymbol{z}_i) - y_i \right)^2, \tag{4.7}$$

is given by

$$f = \sum_{i=1}^p \alpha_i (L_i^* k)(\cdot, \boldsymbol{z}_i), \tag{4.8}$$

where, if we define $\Sigma = diag(\boldsymbol{\sigma})$ as well as — in sloppy notation,

$$[K_L]_{j,i} = \langle k(\cdot, \boldsymbol{z}_j), (L_j L_i^* k)(\cdot, \boldsymbol{z}_i) \rangle_{\mathcal{H}},$$

the coefficients $\boldsymbol{\alpha}$ of the solution are given by

$$\boldsymbol{\alpha} = \left(K_L + \Sigma^2 \right)^{-1} \boldsymbol{y}. \tag{4.9}$$

Proof. The expansion (4.8) follows directly from theorem 2.2.7. Putting it into (4.7) yields

$$\mathcal{O} = \boldsymbol{\alpha}^\top K_L \boldsymbol{\alpha} + \left\| \Sigma^{-1} \left(K_L \boldsymbol{\alpha} - \boldsymbol{y} \right) \right\|_{\mathbb{R}^m}^2 , \tag{4.10}$$

which, since it is convex in $\boldsymbol{\alpha}$, is minimised at the stationary point satisfying

$$\frac{1}{2} \frac{\partial}{\partial \boldsymbol{\alpha}} \mathcal{O} = 0 = K_L \boldsymbol{\alpha} + \left(\Sigma^{-1} K_L \right)^\top \left(\Sigma^{-1} \left(K_L \boldsymbol{\alpha} - \boldsymbol{y} \right) \right) ,$$

the explicit solution of which is given by (4.9). □

Given lemma 4.3.1, it is trivial to solve our original problem (4.3) — if we rewrite the gradient terms therein using partial derivatives, *i.e.* we make the substitution

$$\left\| \left(\nabla f \right) \left(\boldsymbol{x}_i \right) - \boldsymbol{n}_i \right\|_{\mathbb{R}^d}^2 = \sum_{j=1}^{d} \left(\left(\frac{\partial}{\partial x_j} f \right) \left(\boldsymbol{x}_i \right) - \left[\boldsymbol{n}_i \right]_j \right)^2$$

where the $x_1, \ldots, x_d$ represent the components of $\mathbb{R}^d$, then it is clear that (4.3) is merely a special case of lemma 4.3.1 with (assuming the obvious dimensionalities of the following matrices and vectors),

$$\begin{aligned} p &= m \left(d + 1 \right) , \\ \boldsymbol{y} &= \begin{pmatrix} \mathbf{0}^\top & N_{1:} & \cdots & N_{d:} \end{pmatrix}^\top , \\ \begin{pmatrix} L_1 & \cdots & L_m \end{pmatrix} &= \begin{pmatrix} \begin{pmatrix} I & \cdots & I \end{pmatrix} & D_1 & \cdots & D_d \end{pmatrix} , \end{aligned}$$

and

$$\begin{pmatrix} \boldsymbol{z}_1 & \cdots & \boldsymbol{z}_m \end{pmatrix} = \begin{pmatrix} X & \cdots & X \end{pmatrix} ,$$

and where we have defined

$$\begin{aligned} \left[X \right]_{:,i} &= \boldsymbol{x}_i , \\ \left[N \right]_{:,i} &= \boldsymbol{n}_i , \end{aligned}$$

and

$$D_j = \begin{pmatrix} \frac{\partial}{\partial x_j} & \cdots & \frac{\partial}{\partial x_j} \end{pmatrix} .$$

The method we have developed so far in the present section 4.2 represents a straightforward way of solving one of the two main problems with the previous method of section 4.3, namely the need to manufacture off-surface points. The computational problems remain however, as the computational time complexity of solving for the required coefficients is $O((md)^3)$ due to the matrix inverse, and so we now apply the fast approximation scheme of section 3.1.

Fast Approximate Solution

One of the main advantages of the approximation scheme in comparison to the f.m.m. for example, is that it is straightforward to solve problems involving more complex functionals than the evaluation functional. In the present case this is particularly useful due to the gradients in (4.3). For convenience we recall here the form of our approximate solution, as given previously by (3.2), *i.e.*

$$f = \sum_{k=1}^{p} \pi_k f_k, \tag{4.11}$$

as well as the expression for the norm of the above function,

$$\|f\|_{\mathcal{H}}^2 = \boldsymbol{\pi}^\top F_\psi \boldsymbol{\pi}.$$

Once again it is simpler to derive the expression for the optimal vector $\boldsymbol{\pi}$ for the more general case. Putting the above into (4.7) leads to the expression

$$\mathcal{O} = \boldsymbol{\pi}^\top F_\psi \boldsymbol{\pi} + \left\| \Sigma^{-1} \left(F_L \boldsymbol{\pi} - \boldsymbol{y} \right) \right\|^2$$

where $[F_L]_{i,k} \triangleq L_i f_k(\boldsymbol{x}_i)$ and $\Sigma = \operatorname{diag}(\boldsymbol{\sigma})$ as before. The optimal coefficients $\boldsymbol{\pi}$ are those satisfying the stationarity condition

$$\frac{1}{2} \frac{\partial}{\partial \boldsymbol{\pi}} \mathcal{O}[f] = 0 = F_\psi \boldsymbol{\pi} + F_L^\top \Sigma^{-2} F_L \boldsymbol{\pi} - F_L^\top \Sigma^{-2} \boldsymbol{y}$$

the solution of which is

$$\boldsymbol{\pi} = \left(F_\psi + F_L^\top \Sigma^{-2} F_L \right)^{-1} F_L^\top \Sigma^{-2} \boldsymbol{y}. \tag{4.12}$$

For the special case of (4.3), if we write $[F_x]_{i,k} \triangleq f_k(\boldsymbol{x}_i)$ as before, and $[F_{xj}]_{i,k} \triangleq \left(\frac{\partial}{\partial x_j} f_k \right)(\boldsymbol{x}_i)$, then the optimal solution is given by

$$\boldsymbol{\pi} = \left(F_\psi + \sigma_y^{-2} F_x^\top F_x + \sigma_n^{-2} \sum_{j=1}^{d} F_{xj}^\top F_{xj} \right)^{-1} \left(\sigma_n^{-2} \sum_{j=1}^{d} F_{xj} N_{j:}^\top \right). \tag{4.13}$$

Clearly the statements we made about the sparsity of the linear system (3.4) also hold for the above case. Also note that if F_x is too big to fit into the computer memory, it is possible to utilise the fact that

$$\begin{pmatrix} A_1 & \cdots & A_m \end{pmatrix} \begin{pmatrix} A_1 & \cdots & A_m \end{pmatrix}^\top = \sum_{i=1}^{m} A_i A_i^\top,$$

by computing chunks of F_x, multiplying and then updating a running total of $F_x^\top F_x$. Moreover it is straightforward to add this to F_ψ *in place*, since the

sparsity patterns are the same. Hence, the overall memory requirement is approximately only that of F_ψ.

It turns out that in combination with the B_3-spline basis plus thin-plate spline regulariser (*cf.* Example 3.1.4), the overall algorithm is rather efficient, as we demonstrate in the next sub-section.

4.3.1 Experiments and Discussion

Before we demonstrate the method on various problems, let us explain how we ray-trace the implicit surfaces.

Ray Tracing

To render 3D implicit surfaces we use a ray tracer, the usual choice for high quality 3D graphics. To ray trace a given object one need only be able to find the first intersection (if one exists) between the object and a light ray with given starting point $\boldsymbol{x}_0$ and direction $\boldsymbol{n}$. This is particularly simple for the present case — as the embedding function is smooth and differentiable everywhere the following iterative second order approximation line search for the zero set converges rather quickly, and in our experiments took, on average, approximately 2-3 iterations per intersection search.

Note that we found using a first order approximation to be slower. This is probably due to the fact that a considerable part of the time needed to compute the Hessian of the function is taken by the range search which finds the contributing basis functions. Since these need to be found in order to compute the function value and gradient for a first order method anyway, it is not too much more expensive to use a more accurate second order approximation which converges in fewer iterations.

At each iteration we make the second order approximation about $\boldsymbol{x}_0$, *i.e.*

$$\tilde{f}(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{x}_0)^\top A(\boldsymbol{x} - \boldsymbol{x}_0) + \boldsymbol{w}^\top(\boldsymbol{x} - \boldsymbol{x}_0) + b,$$

where b, $\boldsymbol{w}$ and A are available analytically, and given by

$$\begin{aligned} b &= f(\boldsymbol{x}_0) \\ w &= (\nabla f)(\boldsymbol{x}_0) \\ [A]_{i,j} &= \left(\frac{\partial^2}{\partial [\boldsymbol{x}]_i \, \partial [\boldsymbol{x}]_j} f \right)(\boldsymbol{x}_0)/2. \end{aligned}$$

We then solve with respect to λ the equation

$$\tilde{f}(\boldsymbol{x}_0 + \lambda \boldsymbol{n}) = \lambda^2 \boldsymbol{n}^\top A \boldsymbol{n} + \lambda \boldsymbol{w}^\top \boldsymbol{n} + b = 0,$$

which is a quadratic function of λ with roots

$$\lambda^* = \frac{2t_c}{-t_b \pm \sqrt{t_b^2 - 4t_a t_c}},$$

where we have defined

$$t_a = \boldsymbol{n}^\top A \boldsymbol{n}; \quad t_b = \boldsymbol{w}^\top \boldsymbol{n}; \quad t_c = b.$$

We take the lambda of smallest magnitude, since we seek the *first* intersection. Note that if the system cannot be solved for real λ then the real part of λ^* minimises $\left(\tilde{f}\left(\boldsymbol{x}_0 + \lambda \boldsymbol{n}\right)\right)^2$. The implementation of our ray tracer is rather simple and not optimised, so we do not provide timing results in the following subsection 4.3.1, but merely note in passing that the ray traced images in Figure 4.8 each took of the order of a two minutes to calculate.

2D Examples

In order to show how the embedding function behaves (rather than just its zero set) we include a two dimensional example in Figure 4.4. Note that although the function basis includes various scales of basis function at irregular locations, the regularisation scheme outlined in section 3.1 leads to a solution that is well behaved over the entire support of the function. Further evidence of the efficacy of the regulariser is evident in the video accompanying [WSC06], which shows the performance on a rather complex fractal-like data set.

3D Examples

Here we fit models to several 3D data sets of up to 14 million data points. For each model, the timing of the various stages of the fitting process is given in Table 4.1. The following general observations can be made regarding this table:

- High compression ratios can be achieved, in the sense that a relatively small number of basis functions can represent the shape.
- The fitting time scales rather well, from 38 seconds for the Stanford Bunny with 35 thousand points to 4 hours 23 minutes for the Lucy statue with 14 million points. Moreover, after accounting for the different hardware the times seem to be similar to those of the f.m.m. approach [CBC+01].

Some example ray traced images of the resulting models are given in figures 4.3 and 4.7, and the well-behaved nature of the implicit over the entire 3D volume of interest is shown for the Lucy data-set in the accompanying video.

In practice the system is extremely robust — we achieved excellent results without any parameter adjustment — larger values of σ_y and σ_n in (4.3) simply lead to the smoothing effect shown in Figure 4.5. Missing and noisy data are also handled gracefully, as shown in figures 4.6 and 4.8.

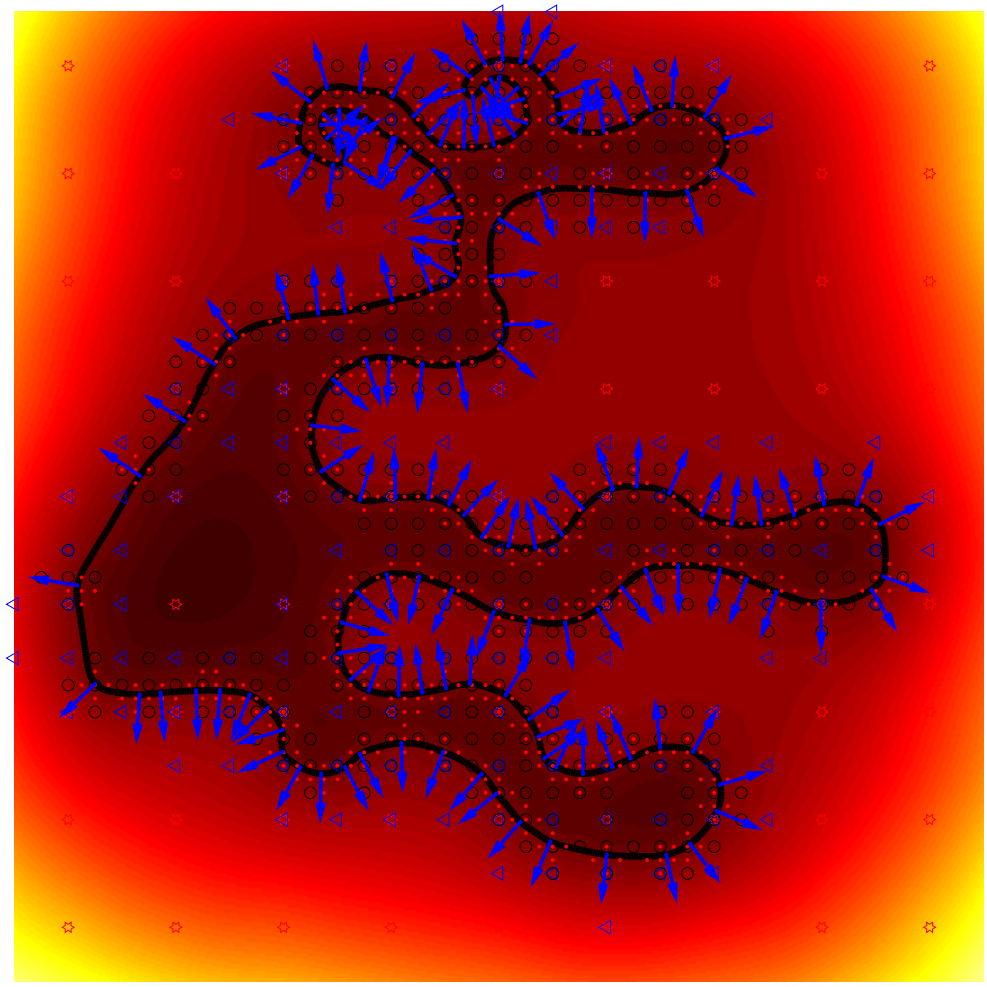


Figure 4.4: An example solution in two dimensions. Data points are depicted as dots with normals vectors as arrows, the zero set as a black contour and the embedding function value as the background colour. All of the basis function centers of various scales (produced by Algorithm 3.1 with the parameter setting $\epsilon = 0$ and a lower limit on the basis support) are plotted with various markers corresponding to the basis function support (*e.g.* the stars have the largest and the red dots the smallest support). Note that although the basis consists of an ad-hoc set of multi-scale, compactly supported basis functions, the overall solution is well behaved due to the computation of the inner products depicted and explained in *e.g.* Figure 3.3. Also note that it is precisely this algorithm which scales up to handle tens of millions of input point/normal pairs, as per Figure 4.3.

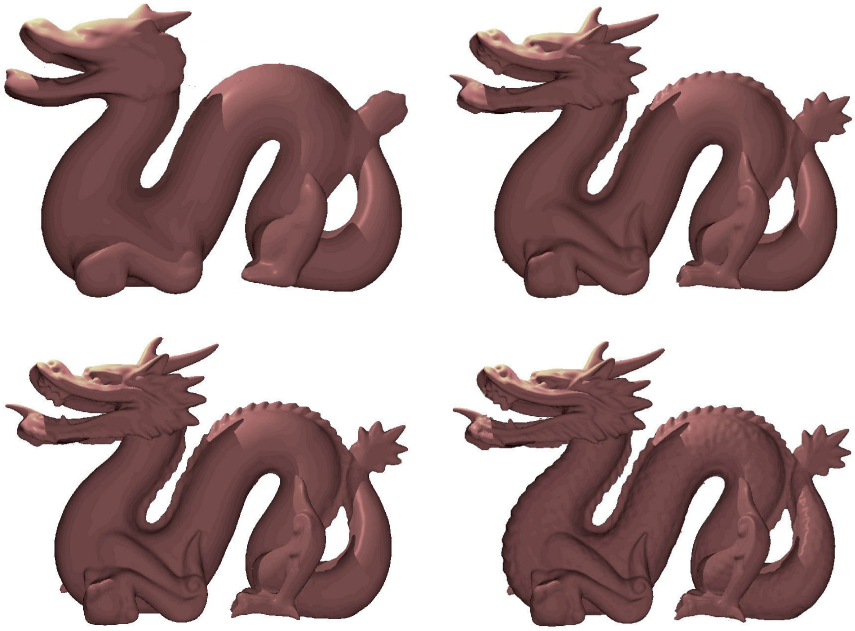


Figure 4.5: Various values of the regularisation parameters lead to various amounts of smoothing — here we set $\sigma_y = \sigma_n$ in (4.3) to a decreasing value from left to right of the figure.

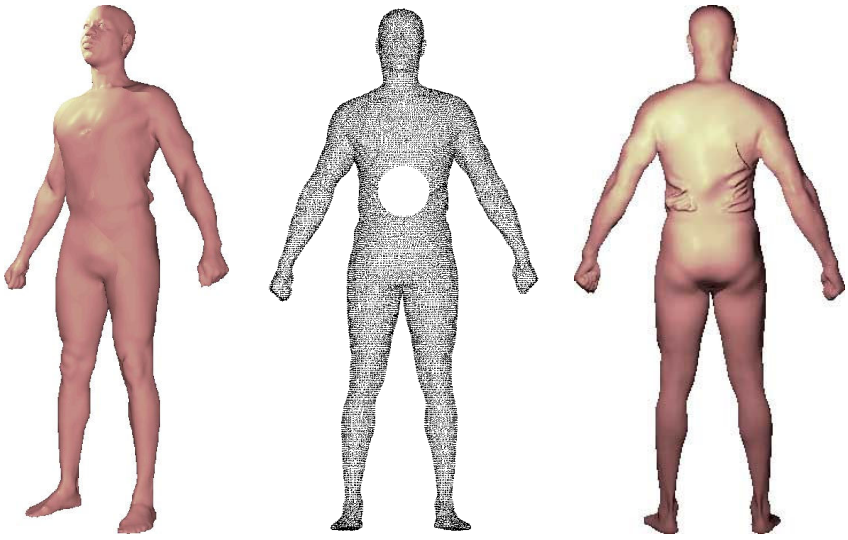


Figure 4.6: Missing data is handled by the interpolating properties of the thin-plate spline regulariser — the ray traced model was constructed from the incomplete data set depicted in the middle. Note that the model is wearing a loose t-shirt.

Figure 4.7: Ray traced three dimensional implicits, *Happy Buddha* (543K points with normals) and the *Thai Statue* (5 million points with normals).

Name	# Points	# Bases	Basis	K_{reg}	K_{xv}, $K_{zv\nabla}$	Multiply	Solve	Total
Bunny	34834	9283	0.4	2.4	3.7	12	20	39
Face	75970	7593	0.7	1.9	7.0	20	16	46
Armadillo	172974	45704	6.6	8.5	37	123.4	72	248
Dragon	437645	65288	14.4	16.3	71	322.8	1381	1806
Buddha	543197	105993	117.4	27.4	99	423.7	2909	3577
Asian Dragon	3609455	232197	441.6	61	608.3	1885.0	1010	4005
Thai Statue	4999996	530966	3742.0	198	1575.6	3121.2	2570	11206
Lucy	14027872	364982	1425.8	170.5	3484	9367.7	1341	15789

Table 4.1: Timing results with a 2.4GHz AMD Opteron 850 processor, for various 3D data sets. Column one is the number of points, each of which has an associated normal vector, and column two is the number of basis vectors (the p of subsection 3.1.1). The remaining columns are all in units of seconds: column three is the time taken to construct the function basis, columns four and five are the times required to construct the indicated matrices, column six is the time required to multiply the matrices as per (4.13), column seven is the time required to solve that same equation for $\boldsymbol{\pi}$ and the final column is the total fitting time.

4D Examples

It is also possible to construct higher dimensional implicit surfaces, particularly interesting being a 4D, or *3D plus time* representation of a continuously evolving 3D shape — one possible use for this is as means of constructing 3D animation sequences from a time series of 3D point cloud data. Optical scanning devices can produce 3D scans at a rate of the order of a hundred frames per second. In this case both spatial and temporal information can help to resolve noise or missing data problems within individual scans. We demonstrate this in the video accompanying [WSC06], which shows that 4D surfaces yield superior 3D animation results in comparison to constructing a separate 3D models for each instant. It is also interesting to note that the interpolating behaviour which we observed in three dimensions also occurs in four dimensions — in the video accompanying [WSC06] we effectively interpolate between two three dimensional shapes.

4.4 Reconstructing Surfaces without using Normals

So far we have seen two methods for implicit surface fitting, both of which require more than just a sampling of the surface to be estimated. In section 4.2 we assumed some additional points labeled as interior or exterior to the surface. In section 4.3 we improved on this by directly utilising surface normal vectors as targets for the gradient of the embedding function.

The present section, which expands on our work in [WCS05], takes a natural next step by dispensing with such additional knowledge entirely — here we assume nothing more than a noisy sampling of the target manifold (*cf.* Definition 4.1.1). The reasons for considering this setting are rather uncompelling within the domain of reconstructing physical surfaces sampled

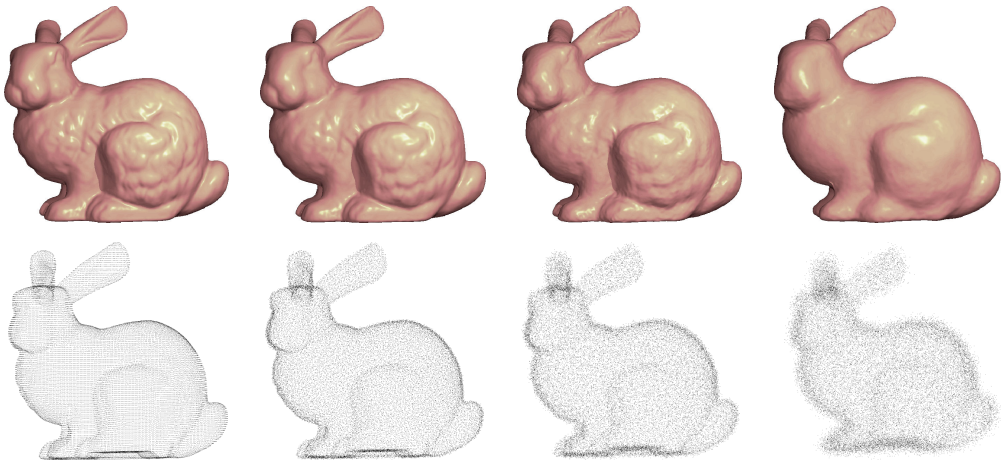


Figure 4.8: Implicit reconstruction of the Stanford bunny after the addition of white Gaussian noise, the variance of which increases from left (no noise) to right. The normal vectors were similarly corrupted. The parameters σ_y and σ_n were chosen automatically using five-fold cross validation.

Figure 4.9: Interpolation in the fourth dimension (call it time): a 4D implicit was constructed from 3D data of the face of a monkey (at time zero) and a man (at time one). Pictured are eight renderings of the implicit at evenly spaced times from zero to one. The data sets were acquired using an optical range finding system, producing noisy data, especially in the case of the monkey due to its hair. Note that we do not propose this as a general morphing algorithm (the interpolated frames are smoothed, which is inappropriate — preferable being a correspondence based approach such as [SSB05]), but rather wish to demonstrate the interpolation properties that are useful for reconstructing animated models from sequences of 3D data as in the video accompanying [WSC06].

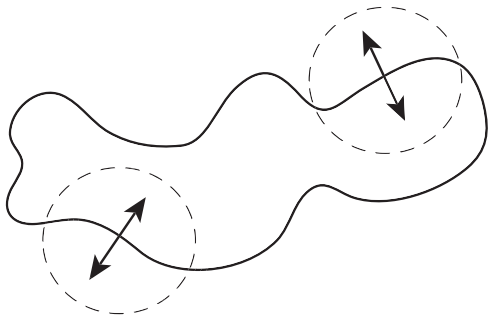


Figure 4.10: An illustration of one of the difficulties which arise in estimating a manifold based on a sampling only, *i.e.* without being given *e.g.* surface normal vectors. Within either dashed circle the surface normal vector can be determined only up to a sign change. The fact that this ambiguity can only be resolved based on the global picture — as well as the fact that f and $(-f)$ are equally good implicit surface embedding functions — indicates an inherent non convexity in the problem.

in $\mathbb{R}^3$. Indeed, as we argued already in subsection 4.1.2, normal vectors are usually available in such cases anyway.

In some other cases however it is not possible to derive these normal vectors reliably. In the case of sparse or high dimensional data for example, deriving the normal vectors (by some local processing) as an intermediate step to the final fitting process could introduce more difficulties than it avoids. To quote the documentation of the state of the art commercially available software product based on the work of [CBC+01]: "in general, without additional information determining normals [from points] is ambiguous".

In comparison to the previous sections of the present chapter, the present setting is considerably more challenging. Loosely speaking, this is due to the fact that since we aim for an implicit surface representation, the problem is non-convex by nature. Perhaps the simplest explanation for this fact is that all reasonable criteria of the goodness of fit (say *e.g.* $\mathcal{O}$) of an implicit surface embedding function to a given data set (at least all reasonable ones which we can think of) will be invariant to a change of sign of the embedding function — after all, the zero set of a function is itself invariant to such a sign change. But since the zero function is not in general equally good as other functions, there exist functions f such that $\mathcal{O}[f - f] > \mathcal{O}[f]$ but $\mathcal{O}[f] = \mathcal{O}[-f]$, implying the non-convexity of $\mathcal{O}$.

Thus, although the normals vectors may contain information which is highly redundant given a dense sampling of the surface, they make the surface estimation easier for technical reasons. Analogous is the semi-supervised learning problem. Semi supervised learning is essentially identical to supervised learning except that some training examples are not labelled as to their class membership. In this case, many authors have proposed augmenting standard learning algorithms such that the unlabelled examples are well

classified into any of the true classes, but not left near the decision boundary. To restate, this means that for a binary classification task the unlabelled points should lie either well on one or the other side of the decision boundary, but not close to it.

The difficulties of the present manifold estimation problem are similar to those of the semi-supervised learning problem. Here, the gradient of the embedding function should be orthogonal to the surface. Locally however, this only means that, loosely speaking, the gradient should point either into or out of the surface but not along it — the scenario is depicted in Figure 4.10. It is the *either-or* in previous sentence as well as the final sentence of the previous paragraph which seems to guarantee practical difficulties. As we have said, these difficulties manifest themselves in the non-convexity of the optimisation problems that we would most naturally like to solve. For example, the transductive s.v.m. [Vap95], that most natural generalisation of the s.v.m. to the semi-supervised setting, leads to a non-convex optimisation problem.

4.4.1 Related Work

Given the difficulties we have just discussed it should come as no surprise that few methods have been proposed for the implicit surface reconstruction problem which use only surface points. We will come to those shortly — first we note that there do exist other surface reconstruction methods which do not use normal vectors, indeed highly effective ones.

Perhaps the most important and representative of these methods are the so-called *point set surface* algorithms (see *e.g.* [ABCO$^+$01]), which are extremely simple. The important thing for us to note about them here, is that they construct only a projection operator from a point onto the surface. Since the surface thus defined is not oriented, *i.e.* the concept of interior/exterior is not represented, the problem does not suffer from the inherent non-convexity we described previously.

Another method that does not use normal vectors is the slab s.v.m. — a generalisation of the one-class s.v.m. for the purpose of implicit surface modelling [SGS05]. For the sake of clarity we shall take a simplified view of the slab s.v.m. — we replace the ϵ-insensitive loss with a simple one-norm loss, and assume that an radial basis function (r.b.f.) kernel is used. In this case the algorithm is similar to a one-class s.v.m., however the data incur a penalty proportional to $\sum_i |f(\boldsymbol{x}_i)|$, rather than the usual term $\sum \max(0, f(\boldsymbol{x}_i))$. It turns out that the optimisation problem it solves is

$$\arg\min_{f \in \mathcal{H}} ||f||^2_{\mathcal{H}} + C \sum_{i=1}^{m} |f(\boldsymbol{x}_i)| - \text{const}(f)$$

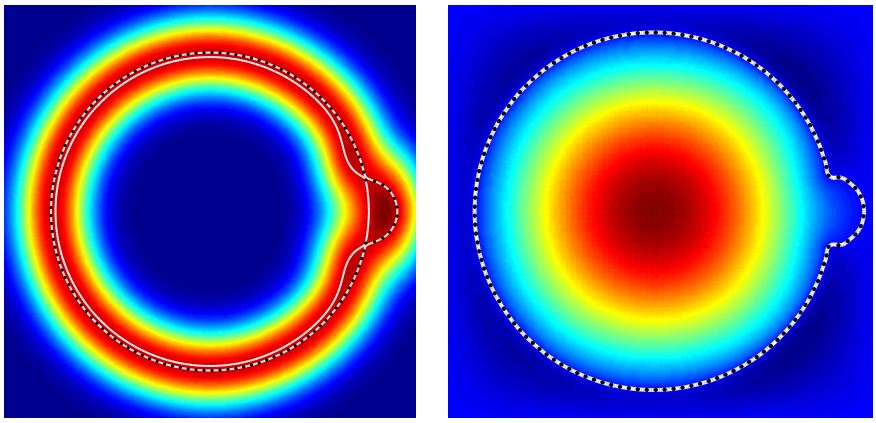


Figure 4.11: The slab s.v.m. embedding function (*left*) is reasonable for 3D rendering, however tends to a negative value both inside and outside the surface of interest, whereas the present method (*right*) indicates by way of the sign of the resultant embedding function whether a given point is interior to the surface. The dots are the 2D input data points, the colours represent function value, and the lines depict the zero level set.

where const(f) denotes the zero order component of the function f (usually denoted b or ρ in the s.v.m. literature), and $\mathcal{H}$ is an r.k.h.s. such that the term $||f||^2_{\mathcal{H}}$ in the above formulation acts as a regulariser. From this perspective we can view this as a variant of a regression problem that has target values of zero at each of the $\boldsymbol{x}_i$ — as mentioned in the Chapter introduction however, such a regression problem has the trivial solution $f = 0$, which the slab s.v.m. avoids by including the term const(f) in the objective function.

The above approach is advantageous insofar as it yields a convex optimisation problem, however the maximisation of const(f) has some undesirable properties — as a result of it, rather than being positive inside and negative outside the manifold of interest, if an r.b.f. kernel is used this property will often only hold within some local neighbourhood of the data points, as in Figure 4.11. This is an issue when rendering an implicit in three dimensions (since the extraneous zero set will always be obscured), but the property that the sign of the function indicates whether a given point is inside the shape no longer holds.

The goal of the present section 4.4 then, is to preserve the strengths of the slab SVM, especially the fact that it does not use normal vectors, while producing embedding functions with the behaviour depicted on the right side of Figure 4.11, as these are more suitable for inside/outside tests *etc.*

4.4.2 Algorithm

As we said at the very beginning of the Chapter, all of the different implicit surface fitting algorithms can be viewed as a variant of a regression problem

in which all the target values are zero. Essentially we aim for a means of avoiding the triviality of this problem which leads to a proper implicit surface embedding function (*cf.* the 4.1.3). In particular, given a noisy sampling $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_m \subset \mathbb{R}^d$ of the surface, the approach we propose is based on the following observations, generally true of such embedding functions:

1. **Data term.** The values $|f(\boldsymbol{x}_i)|$ are relatively small, as the $\boldsymbol{x}_i$ should lie near $f^{-1}(0)$.
2. **Energy term.** The value $|f(\boldsymbol{x})|$ is relatively large over most of the space $\mathcal{X}$, as f should be zero on the sampled manifold only.
3. **Gradient term.** The gradients $\|(\nabla f)(\boldsymbol{x}_i)\|_{\mathbb{R}^d}$ are relatively large, as the zero set would otherwise be unstable with respect to small changes in f.
4. **Regularisation term.** f is regular.

In the analogy with semi-supervised learning which we made previously, we relate the first item to the classification function targets on the labelled points, and the second item to the treatment of the unlabelled points.

As we have already made clear this tends to lead to difficult non-convex optimisation problems. Various ways of dealing with this are possible, and typically fall somewhere between the two extremes of either

1. Setting up an optimisation problem which is close to ideal but can only be approximately solved.
2. Opting instead for a problem that can be solved exactly, but that is further from the ideal.

Note that it would be tempting to write convex/non-convex instead of solvable/approximately solvable, but as we shall see there is a difference — indeed we shall ultimately opt for a non-convex problem which we can solve exactly. Both of the above approaches have their advantages and disadvantages, and although we will take the latter approach, before we proceed to the precise formulation let us suggest a possibility of the former. This will help us to understand the difficulties involved.

An Example of the First Approach (Not Precisely Solvable)

Naturally various formulations are plausible — sufficient for our illustrative purposes however, is *e.g.*

$$\arg\min_{f \in \mathcal{F}} \|f\|_{\mathcal{F}}^2 + \sigma_y^{-2} \sum_{i=1}^{m} |f(\boldsymbol{x}_i)| + \sigma_g^{-2} \sum_{i=1}^{m} \left(\|(\nabla f)(\boldsymbol{x}_i)\| - 1 \right)^2,$$

for some space of functions $\mathcal{F}$. Another interesting (and possibly easier to handle) alternative is to replace the third term with the following alternative means of obtaining a gradient at the surface points which has an approximately constant and non-zero norm, *i.e.*

$$\sum_{i=1}^{m} \left\| (\nabla f)(\boldsymbol{x}_i) \right\|^{-2}.$$

Although neither problem takes account of the energy term, both seem promising for the problem at hand. Note in particular that

- If $\left\| (\nabla f)(\boldsymbol{x}_i) \right\| \approx 1$, then $|f(\boldsymbol{x}_i)|$ is a first order approximation of the distance from $\boldsymbol{x}_i$ to the zero level set $f^{-1}(0)$.
- The method is in fact similar to that of section 4.3, except that the norm of the gradient is constrained rather than the norm and direction.
- Unfortunately, since the third term is non-convex in f the problem is rather difficult to solve.

An Example of the Second Approach (Precisely Solvable)

Rather than trying to solve approximately the problem formulation which we have just suggested, we opt instead for a similar problem which can be solved exactly. In particular we devise four quadratic penalties motivated by the four objectives which we stated previously, *i.e.*

$$f_{\text{eig}} \triangleq \arg\min_{f \in \mathcal{F}} \underbrace{\sum_{i=1}^{m} f(\boldsymbol{x}_i)^2}_{\text{data term}} + \underbrace{\lambda_\Omega < f, f >_{\mathcal{F}}^2}_{\text{regularisation term}} \underbrace{- \lambda_{\text{e}} \int_{\boldsymbol{u} \in \mathbb{R}^d} f(\boldsymbol{u})^2 d\mu(\boldsymbol{u})}_{\text{energy term}} \underbrace{- \lambda_\nabla \sum_{i=1}^{m} ||(\nabla f)(\boldsymbol{x}_i)||^2}_{\text{gradient term}} \quad (4.14)$$

The labels of the individual terms note the correspondence between the four terms of the objective function (balanced by the positive real-valued λ's) and the four properties we listed at the beginning of the present section. The terms themselves each appear in quadratic form in order to allow us to formulate the optimisation as an eigenvalue problem. To do this, we must also define the function class as a finite mixture of basis functions with unit norm coefficients — that is, we let

$$\mathcal{F} = \left\{ \sum_{k=1}^{p} \pi_k f_k : \|\boldsymbol{\pi}\|_{\mathbb{R}^p} = 1 \right\}. \quad (4.15)$$

Now, writing $f = \sum_{k=1}^{p} \pi_k f_k$ where

$$f_k(\cdot) = \phi_r(|| \cdot - \boldsymbol{v}_k||/s_k), \quad (4.16)$$

precisely as before in chapter 3 (*cf.* (3.2) and the surrounding comments), and maintaining our definition $[F_x]_{i,k} \triangleq f_k(\boldsymbol{x}_i)$ from that same chapter, the

data term can be written

$$\sum_{i=1}^{m} f(\boldsymbol{x}_i)^2 = \boldsymbol{\pi}^\top F_x^\top F_x \boldsymbol{\pi}$$

Next, due to the quadratic nature of the energy term as well as the linearity of integration, we can write the energy term as a vector-matrix-vector product in the following way. Let us define F_e as

$$[F_\mathrm{e}]_{i,j} = \int_{\boldsymbol{u}\in\mathbb{R}^d} \phi_r(||\boldsymbol{v}_j - \boldsymbol{u}||/s_j)\phi_r(||\boldsymbol{v}_k - \boldsymbol{u}||/s_k)\mathrm{d}\mu(\boldsymbol{u}),$$

then

$$\begin{aligned}\int_{\boldsymbol{u}\in\mathbb{R}^d} f(\boldsymbol{u})^2 d\mu(\boldsymbol{u}) &= \int_{\boldsymbol{u}\in\mathbb{R}^d} \left(\sum_j^n \pi_j \phi_r\left(||\boldsymbol{v}_j - \boldsymbol{u}||/s_j\right)\right)^2 d\mu(\boldsymbol{u}) \\ &= \sum_{j,k} \pi_j \pi_k [F_\mathrm{e}]_{j,k} \\ &= \boldsymbol{\pi}^\top F_\mathrm{e} \boldsymbol{\pi}\end{aligned}$$

Similarly we can rearrange the gradient term to get $\sum_{i=1}^m ||\nabla_{\boldsymbol{x}=\boldsymbol{x}_i} f(\boldsymbol{x})||^2 = \boldsymbol{\pi}^\top F_\nabla \boldsymbol{\pi}$, where we have defined

$$[F_\nabla]_{j,k} = \sum_{i=1}^{m} \langle \nabla_{\boldsymbol{x}=\boldsymbol{x}_i} \phi_r(||\boldsymbol{x} - \boldsymbol{v}_j||/s_j), \nabla_{\boldsymbol{x}=\boldsymbol{x}_i} \phi_r(||\boldsymbol{x} - \boldsymbol{v}_k||/s_k)\rangle$$

The only remaining term is the regulariser $||f||^2$, which we treat in the principled manner described in chapter 3.

Eigenvalue Problem

Since all of the terms are quadratic in $\boldsymbol{\pi}$, it is possible to take advantage of Rayleigh's principle [GV96], which states that the solution with respect to $\boldsymbol{x}$ of the problem

$$\arg \min_{\boldsymbol{x}^\top B \boldsymbol{x} = 1} \boldsymbol{x}^\top A \boldsymbol{x} = \arg \min_{\boldsymbol{x}} \frac{\boldsymbol{x}^\top A \boldsymbol{x}}{\boldsymbol{x}^\top B \boldsymbol{x}}, \tag{4.17}$$

is given by the eigenvector $\boldsymbol{x}^*$ with smallest eigenvalue λ^* of the generalised eigenvalue problem

$$A\boldsymbol{x}^* = B\boldsymbol{x}^*\lambda^*.$$

Thus, due to the constraint $\|\boldsymbol{\pi}\| = 1$ which we place on $\mathcal{F}$ (*cf.* (4.15)), (4.14) is solved by the eigenvector $\boldsymbol{\pi}^*$ with the most negative eigenvalue λ^* of the following non positive definite (p.d.) eigenvalue problem:

$$\left(K_{vx}K_{vx}^\top + \lambda_\Omega K_\Omega - \lambda_\mathrm{e} K_\mathrm{e} - \lambda_\nabla K_\nabla\right)\boldsymbol{\pi}^* = \lambda^* \boldsymbol{\pi}^*$$

which in our case is typically a large system with a high sparsity ratio. In practice we solved the eigenvalue problem using the JDQR algorithm [SV96]. Actually, we found that it was more numerically stable to add a multiple of the identity with magnitude

$$\lambda_{\mathrm{e}} \left\| K_{\mathrm{e}} \right\|_2 + \lambda_{\nabla} \left\| K_{\nabla} \right\|_2 ,$$

where $\left\| \cdot \right\|_2$ denotes the matrix spectral norm[2], in order to yield an equivalent but positive definite system, and then solve for the smallest magnitude eigenvalue rather than the most negative.

Some Comments on the Eigenvalue Formulation

We would like to make three comments on the formulation we have chosen.

First comment. Since we are minimising two terms and maximising another two, it may seem natural to minimise a ratio of sums of vector-matrix-vector products, as this leads to a positive definite generalised eigenvalue problem with more benign numerical properties. This is because we would then need to solve for the eigenvector with smallest magnitude rather than most negative eigenvalue, which tends to be easier from a numerical point of view [SV96]. To be more precise, the exact form of the eigenvalue problem would then be

$$\left(K_{vx} K_{vx}^{\top} + \lambda_{\Omega} K_{\Omega} \right) \boldsymbol{\pi}^* = \lambda^* \left(\lambda_{\mathrm{e}} K_{\mathrm{e}} + \lambda_{\nabla} K_{\nabla} \right) \boldsymbol{\pi}^* .$$

Unfortunately however, this costs us an important degree of freedom in balancing the individual terms. Let us illustrate this point — to begin with note that minimising a/b is the same as minimising $(\log(a) - \log(b))$. Now, if we balance the combination using a parameter c by minimising $(\log(a) - c \log(b))$ then it is equivalent to minimising a/b^c. In other words, many Rayleigh's quotient type problems probably ought to have an exponent in the denominator, but as this no longer corresponds to an equivalent eigenvalue problem, it will usually be considerably more difficult to solve. There are cases however in which both the numerator and denominator have equivalent units so that one may argue in favour of the ratio problem on the grounds that it measures a unitless ratio — popular among physicists, but this is not the case here.

Second comment. The constraint $\left\| \boldsymbol{\pi} \right\| = 1$ in (4.15) is not in itself justifiable, and will leave an imprint on the solution. Our only justification for the constraint is that it leads to a problem which we can solve exactly. Indeed, as we have explained already, such are the costs of restricting oneself to optimisation problems which can be solved exactly.

[2] The matrix spectral norm is equal to the largest magnitude of the eigenvalues of a matrix

Third comment. We now identify the main problem with the eigenvalue formulation which we have adopted — which we shall refer to as the *energy-focusing* problem. Due to the sum-of-squares nature of the objective function in (4.14), the optimal solution will tend to focus all of the energy of the function on the region of the space in which the data is most benign. By benign we mean amenable to achieving a low optimum value of the objective function (4.14). Consider by way of analogy the problem

$$\arg \min_{\boldsymbol{x} \in \left\{ \boldsymbol{x}' \in \mathbb{R}^d : \|\boldsymbol{x}'\|_1 = 1 \right\}} \|\boldsymbol{x}\|_2^2 .$$

The solution to the above problem is clearly not unique — the minimisers are all vectors with a single non zero element equal to ± 1. This is similar to the problem with the present eigenvalue formulation, though not identical. We explain the nature of the problem in the present case by considering the following pathological example.

Let's assume that the surface consists of two disconnected components of which the left one is more complex than the right, in the sense that if the two components were treated separately, we would achieve a lower minimum of the objective function in (4.14) by using data sampled from the right- rather than the left-hand side component. Now, if we solve for f_{eig} using data sampled from both components together, but we then increase the separating distance between the two components, then eventually the purely quadratic nature of the objective function will lead to a point at which the optimal solution has $f_{\text{eig}} = 0$ in the vicinity of the left component, and places all of the energy of the function on the right component. Alternatively, if both components were identical then there would be two solutions for f_{eig} — each of which places zero on either the left or the right component — and each solution would correspond to each of the first two eigenvectors of (4.17), which would have identical eigenvalues

Although such a situation will never arise in practice, it turns out that the energy focusing problem also manifests itself in more realistic examples. It is easiest to simply demonstrate this phenomenon, which brings us to the next sub-section.

4.4.3 Experiments and Discussion

Although implicit surfaces are most commonly used for problems in $\mathbb{R}^3$, it is useful to experiment in $\mathbb{R}^2$ since we can then visualise the embedding function as a two-dimensional colour intensity plot. Examples of this are given in figures 4.12 and 4.13. Note that while the energy-focusing problem is evident in Figure 4.13, it is not evident in figures 4.11 or 4.12. The reason for this seems to be that those examples feature data sets which have relatively uniform complexity over the space — there is no part of the data which is

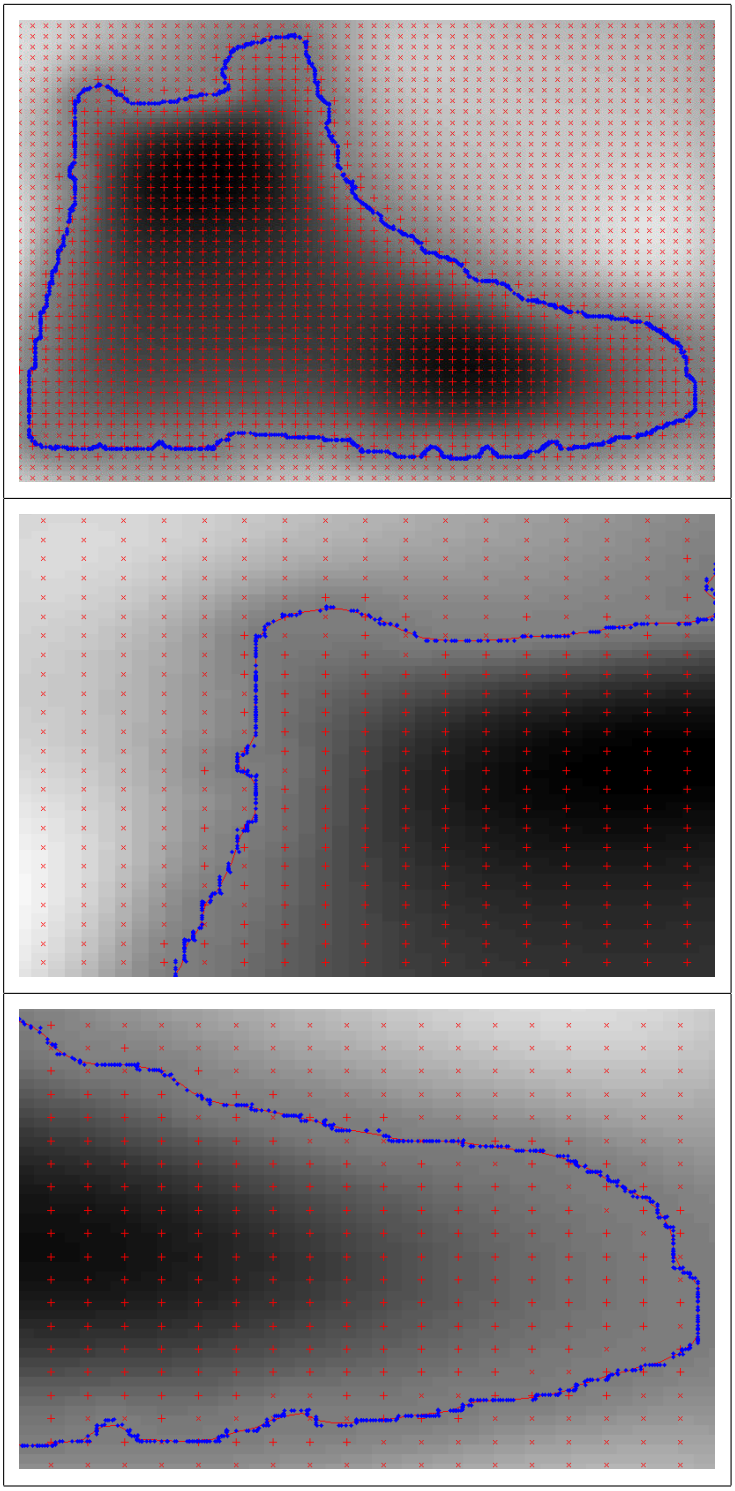


Figure 4.12: Here we used a simple function basis, the centres of which are denoted by pluses or crosses depending on whether the corresponding coefficient in $\boldsymbol{\pi}$ turned out to be positive or negative. For simplicity, we also used a Gaussian basis function and the regulariser (or function norm) naturally implied by it (*cf.* (2.27) and preceding comments). The background colour corresponds to the function's value and the thin red line its zero level. Happily, the red line is largely obscured by the blue dots which represent the given surface points.

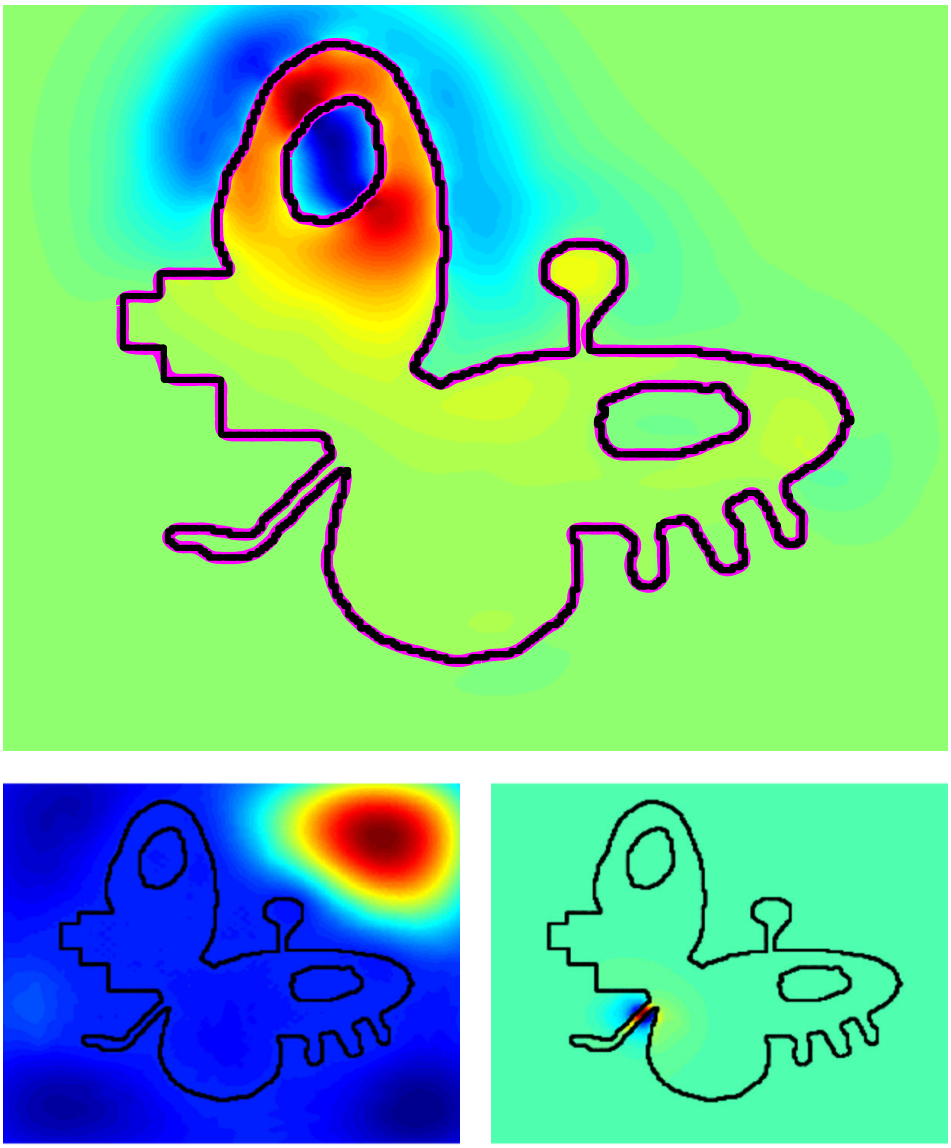


Figure 4.13: Investigating the role of the parameters in a two dimensional problem. **Top**: a parameter set chosen using our automatic criteria (see subsection 4.4.3) — visually there seems to be somewhat too large a derivative term, but the zero level rendered as a magenta line is reasonable — as it is largely obscured by the data one must zoom in on an electronic copy of the document to see it properly. **Bottom-left**: too large an energy term results in a large smooth bump in the function outside the shape. **Bottom-right**: too large a gradient term results in a concentrated region of high gradient at some subset of the data set, in this case in the bottom-left part of the shape. Note that the energy focusing problem is evident in all cases — the only variation being the part of the data set which (depending on the given parameter set) is most favourable for the solution to focus on.

sufficiently less complex in comparison to the rest for it to become the focus of the energy focusing problem. In general we found the energy focusing problem to be a serious one on complex data sets — the approach is suitable for relatively benign problems only.

As a consequence of the energy focusing problem, the algorithm is rather sensitive to the choice of parameters — of which there is a rather large number (*i.e.* the three λ's in (4.14)). Fortunately however the issue of parameter choice is somewhat less problematic than in *e.g.* a classification problem, since we can make some reasonable measures of the quality of the solution without requiring a holdout set or cross-validation process. In practice it was necessary to conduct exhaustive and time consuming parameter searches in order to determine an effective parameter set — a task which would likely have proven too difficult without the computer cluster which we had at our disposal.

Two measures which we found useful for these automatic parameter tuning exercises are defined as follows. Given an embedding function f that models our training data set $\mathcal{S}$, we take an approximately uniform sampling $\mathcal{R} \subseteq f^{-1}(0)$ of the zero set and compute the quantities

$$m_{\mathcal{RS}} = \max_{\boldsymbol{x}_1 \in \mathcal{R}} \min_{\boldsymbol{x}_2 \in \mathcal{S}} ||\boldsymbol{x}_1 - \boldsymbol{x}_2||,$$
$$\text{and} \quad m_{\mathcal{SR}} = \max_{\boldsymbol{x}_1 \in \mathcal{S}} \min_{\boldsymbol{x}_2 \in \mathcal{R}} ||\boldsymbol{x}_1 - \boldsymbol{x}_2||.$$

These are, roughly speaking, the largest distance from $\mathcal{R}$ to $\mathcal{S}$ and *vice versa*. Both of these measures are useful, and if either measure is large then the solution cannot be a good one.

Our experience has shown that both the energy and the gradient term are necessary for maximum topological stability of the solution. As evidenced by Figure 4.13 however, too great a value of λ_∇ leads to instability, while increasing λ_e reduces this effect, the net result being smoother but less precise solutions. We stress however that these are heuristic and imperfect measures of the quality of the solution — superior being a visual inspection. As expected, the regularisation parameter λ_∇ controls the trade off between smoothness and fidelity to the data.

We also ran our system on some three dimensional data sets, particularly focusing on two important applications of the algorithm which take advantage of the strengths of implicit surfaces.

The first is the filling or interpolation of holes in the data, as in Figure 4.14. Using a 3GHz Pentium III processor, we fit the bunny model in approximately forty minutes. The drawing took a similar amount of time, although this is due to the fact that we employed a rendering algorithm which naively evaluates the embedding function over the entire viewing region, rather than using a faster method that follows or ray-traces the surface. We avoided

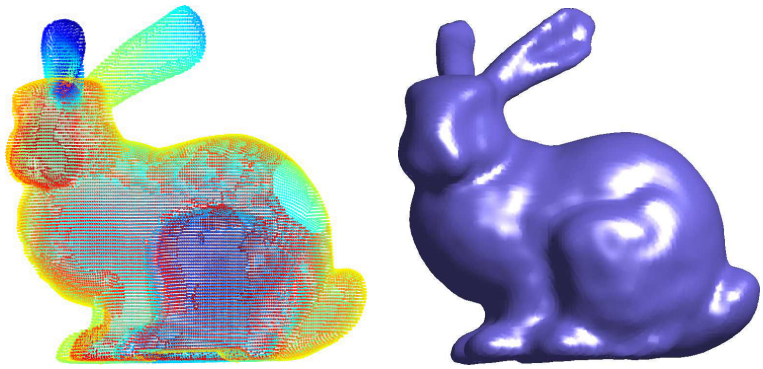


Figure 4.14: An example of smoothly interpolating missing data — we created a hole in the Stanford bunny data set (visible in the upper rear region of the bunny as shown on the left), resulting in a data set of 35K data points in three dimensions. The surface that we fit approximately recovers the original shape, as shown on the right.

such fast rendering schemes in order to be sure that there are no errant components of the zero set lying away from the data.

The second interesting property that we demonstrate is the ease with which set operations can be performed using implicit shape definitions. For example, given a pair of embedding functions f_1 and f_2 that are negative inside the shape and positive outside, the intersection of the shapes corresponds to the embedding function $f_\cap = \max(f_1, f_2)$. This is precisely how we generated Figure 4.15. Alternatively one can define an embedding function $|f_1| + |f_2|$, the zero level set of which is the intersection of the zero level sets of f_1 and f_2. The fitting of the knot model took roughly fifteen minutes. The plain knot model was drawn in about forty minutes, whereas the intersection picture took approximately a day using a higher resolution.

To define the union of the two implicit shapes we can use $f_\cup = \min(f_1, f_2)$. Similarly to the intersection case, we can alternatively use the product $f_1 f_2$, the zero level set of which is the union of those of f_1 and f_2, however this retains those components of the zero set interior to the resultant shape in spite of the fact that they will always be obscured from the observer.

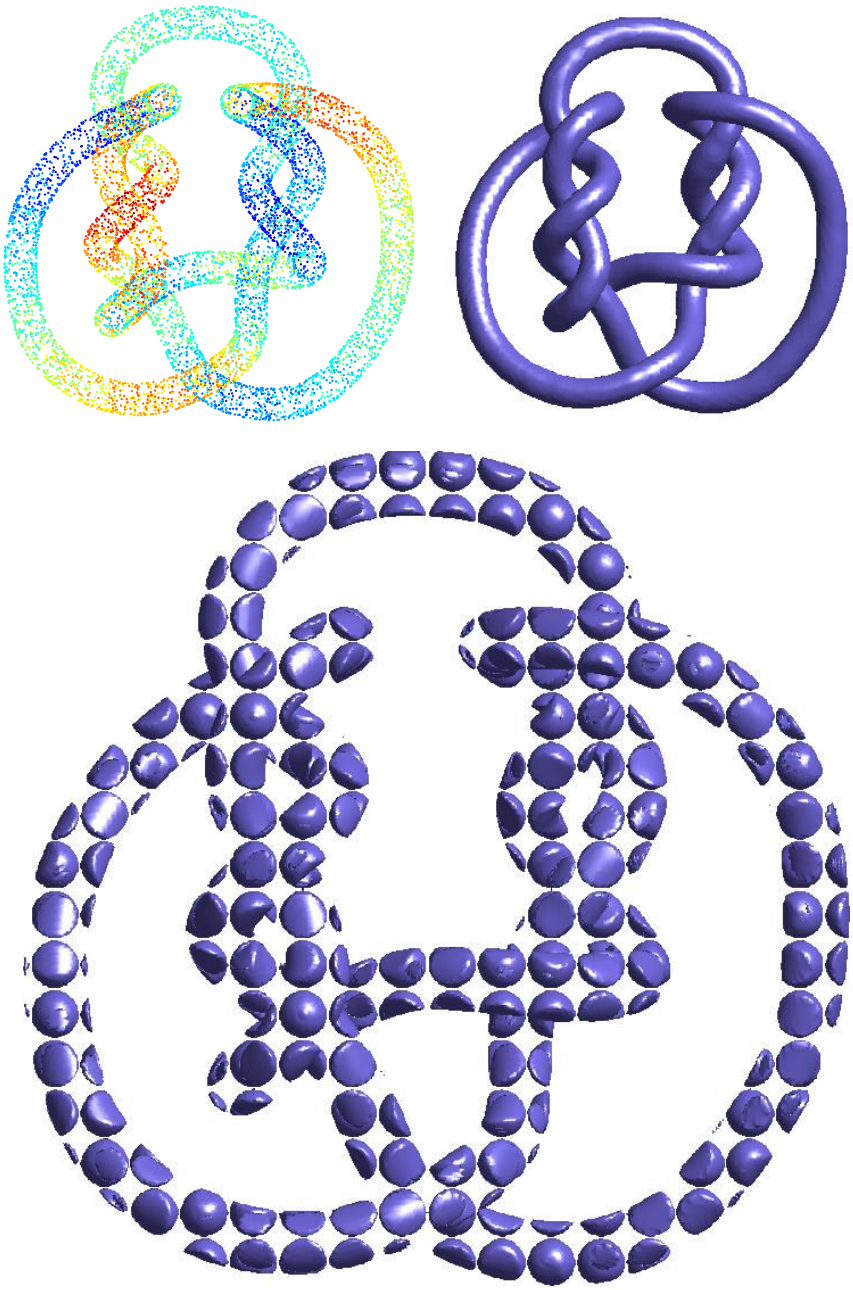


Figure 4.15: An example using the knot data set consisting of 10K data points in three dimensions, as shown on the top left (the colours have no special meaning). The implicit surface rendered on the top right was constructed using approximately 5K basis functions. The lower image demonstrates that set operations such as intersection are trivial with implicit shape definitions, regardless of the topological complexity that results — we intersected the knot shape with a grid of balls (also defined as an implicit).

Chapter 5

Kernels Invariant to Transformations

In section 4.3 we applied the thin-plate spline regulariser of (2.28) to the problem of implicit surface reconstruction. Interestingly, it is possible for the resulting solutions to capture fine details of the input data, while at the same time interpolating over relatively large unsampled regions. This type of behaviour is not possible with the Gaussian kernel for example. In the present chapter we shall investigate this phenomenon further from the point of view of *invariances* (we define what we mean by invariances shortly) — in the case just discussed, an important invariance is invariance to dilations (*i.e.* multiplicative scaling of the input data).

In particular, we investigate function norms which are invariant to such transformations as translation, rotation and dilation (*cf.* Figure 5.1), and provide necessary and sufficient conditions for a reproducing kernel (r.k.) to induce such a norm. Following this, we show that radial kernels which induce a dilation invariant norm can only be conditionally positive definite (c.p.d.), and analyse one particular such case, namely the thin-plate spline. Practically speaking, the thin-plate spline has the advantage that — as we show — varying the length-scale of the data has the same effect as varying *e.g.* the regularisation parameter in the support vector machine (s.v.m.)This allows one to build effective classifiers which have only one parameter.

Motivated by this finding, we consider the problem of using a c.p.d. kernel with an s.v.m., and propose a simple algorithm which makes this possible. This simple Newton method approach proceeds by repeatedly solving local second order approximations of the true problem. Furthermore, we provide a general procedure for deriving positive definite (p.d.) kernels from c.p.d. ones, giving in particular a p.d. analog of the thin-plate kernel, which can for example be employed in any kernel based algorithm. The experimental results which we conclude with demonstrate that the s.v.m. with thin-plate kernel, although possessing only one free parameter, performs as well on real

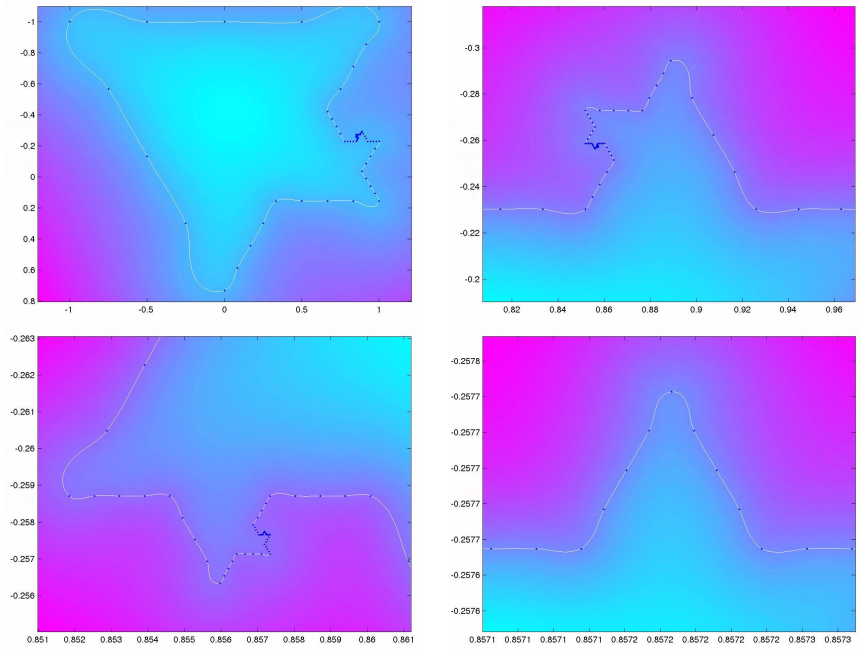


Figure 5.1: An implicit surface fit to a synthetic fractal-like data set using the method of section 4.3. The regulariser is the $m = 2$ thin-plate spline energy of (2.28) which, in the terminology of the present chapter, is rotation/translation invariant and dilation scaled. Such effective behaviour over so wide a range of scales is impossible with *e.g.* the Gaussian kernel.

data as the Gaussian kernel, which has two free parameters.

5.1 Transformation Scaled Spaces and Tikhonov Regularisation

Definition 5.1.1. Let $\mathcal{T}$ be a bijection on $\mathcal{X}$ and $\mathcal{F}$ a Hilbert space of functions on some non-empty set $\mathcal{X}$ such that $f \mapsto f \circ \mathcal{T}$ is a bijection on $\mathcal{F}$. Furthermore let $\mathcal{F}$ be endowed with an inner product. $\mathcal{F}$ is **$\mathcal{T}$-scaled** if

$$\langle f, g \rangle_{\mathcal{F}} = g_{\mathcal{T}}(\mathcal{F}) \langle f \circ \mathcal{T}, g \circ \mathcal{T} \rangle_{\mathcal{F}} \tag{5.1}$$

for all $f \in \mathcal{F}$, where $g_{\mathcal{T}}(\mathcal{F}) \in \mathbb{R}^+$ is the *norm scaling function* associated with the operation of $\mathcal{T}$ on $\mathcal{F}$. If $g_{\mathcal{T}}(\mathcal{F}) = 1$ we say that $\mathcal{F}$ is **$\mathcal{T}$-invariant**.

The following clarifies the behaviour of Tikhonov regularised solutions in such spaces.

Lemma 5.1.2. *For any $\Theta : \mathcal{F} \to \mathbb{R}$ and $\mathcal{T}$ such that $f \mapsto f \circ \mathcal{T}$ is a bijection of $\mathcal{F}$, if the left hand side is unique then*

$$\arg\min_{f \in \mathcal{F}} \Theta(f) = \left(\arg\min_{f_{\mathcal{T}} \in \mathcal{F}} \Theta(f_{\mathcal{T}} \circ \mathcal{T}) \right) \circ \mathcal{T}$$

Proof. Let $f^* = \arg\min_{f\in\mathcal{F}} \Theta(f)$ and By definition we have that $\forall g \in \mathcal{F}, \Theta(f^*_{\mathcal{T}} \circ \mathcal{T}) \leq \Theta(g \circ \mathcal{T})$. But since $f \mapsto f \circ \mathcal{T}$ is a bijection on $\mathcal{F}$, we also have $\forall g \in \mathcal{F}, \Theta(f^*_{\mathcal{T}} \circ \mathcal{T}) \leq \Theta(g)$. Hence, given the uniqueness, this implies $f^* = f^*_{\mathcal{T}} \circ \mathcal{T}$. □

The following Corollary follows immediately from lemma 5.1.2 and Definition 5.1.1.

Corollary 5.1.3. *Let L_i be any loss function. If $\mathcal{F}$ is $\mathcal{T}$-scaled and the left hand side is unique then*

$$\arg\min_{f\in\mathcal{F}} \left(\|f\|^2_{\mathcal{F}} + \sum_i L_i \left(f \left(\boldsymbol{x}_i \right) \right) \right) =$$
$$\left(\arg\min_{f\in\mathcal{F}} \left(\|f\|^2_{\mathcal{F}} / g_{\mathcal{T}} \left(\mathcal{F} \right) + \sum_i L_i \left(f \left(\mathcal{T}\boldsymbol{x}_i \right) \right) \right) \right) \circ \mathcal{T}.$$

Corollary 5.1.3 includes various learning algorithms for various choices of L_i — for example the s.v.m. with linear hinge loss for $L_i(t) = \max(0, 1 - y_i t)$, and kernel ridge regression for $L_i(t) = (y_i - t)^2$. Let us now introduce the specific transformations we will be considering.

Definition 5.1.4. Let W_s, $T_{\boldsymbol{a}}$ and O_A be the **dilation, translation and orthonormal transformations** $\mathbb{R}^d \to \mathbb{R}^d$ defined for $s \in \mathbb{R} \setminus \{0\}$, $\boldsymbol{a} \in \mathbb{R}^d$ and orthonormal $A : \mathbb{R}^d \to \mathbb{R}^d$ by $W_s\boldsymbol{x} = s\boldsymbol{x}$, $T_{\boldsymbol{a}}\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{a}$ and $O_A\boldsymbol{x} = A\boldsymbol{x}$ respectively.

Hence, for an reproducing kernel Hilbert space (r.k.h.s.) which is W_s-scaled for arbitrary $s \neq 0$, training an s.v.m. and dilating the resultant decision function by some amount is equivalent training the s.v.m. on similarly dilated input patterns but with a regularisation parameter adjusted according to Corollary 5.1.3.

While [FS03] demonstrated this phenomenon for the s.v.m. with a particular kernel, as we have just seen it is easy to demonstrate for the more general Tikhonov regularisation setting with any function norm satisfying our definition of transformation scaledness. In the next section we continue our investigation by deriving the necessary and sufficient conditions for an r.k. to induce a norm that is transformation scaled, before investigating some interesting consequences and special cases.

5.2 Transformation Scaled Reproducing Kernel Hilbert Spaces

We now derive the necessary and sufficient conditions for a r.k. to correspond to an r.k.h.s. which is $\mathcal{T}$-scaled. The relationship between $\mathcal{T}$-scaled r.k.h.s.'s

and their r.k.'s is easy to derive given the uniqueness of the r.k. [Wen04]. It is given by the following novel

Lemma 5.2.1 (Transformation scaled r.k.h.s.)**.** *The r.k.h.s. $\mathcal{H}$ with r.k. $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, i.e. with k satisfying*

$$\langle k(\cdot, \boldsymbol{x}), f(\cdot) \rangle_{\mathcal{H}} = f(\boldsymbol{x}), \tag{5.2}$$

is $\mathcal{T}$-scaled iff

$$k(\boldsymbol{x}, \boldsymbol{y}) = g_{\mathcal{T}}(\mathcal{H})\, k(\mathcal{T}\boldsymbol{x}, \mathcal{T}\boldsymbol{y}). \tag{5.3}$$

Which we prove in section B.2. It is now easy to see that, for example, the homogeneous polynomial kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle^p$ corresponds to a W_s-scaled r.k.h.s. $\mathcal{H}$ with $g_{W_s}(\mathcal{H}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle^p / \langle s\boldsymbol{x}, s\boldsymbol{y} \rangle^p = s^{-2p}$. Hence when the homogeneous polynomial kernel is used with the *hard-margin* s.v.m. algorithm, the result is invariant to multiplicative scaling of the training and test data. If the soft-margin s.v.m. is used however, then the invariance holds only under appropriate scaling (as per Corollary 5.1.3) of the margin softness parameter (*i.e.* λ of the later equation (5.11)).

We can now show that there exist no non-trivial r.k.h.s.'s with radial kernels that are also W_s-scaled for all $s \neq 0$. First however we need the following standard result on homogeneous functions:

Lemma 5.2.2. *If $\phi : [0, \infty) \to \mathbb{R}$ and $g : (0, \infty) \to \mathbb{R}$ satisfy $\phi(r) = g(s)\phi(rs)$ for all $r \geq 0$ and $s > 0$ then $\phi(r) = a\delta(r) + br^p$ and $g(s) = s^{-p}$, where $a, b, p \in \mathbb{R}$, $p \neq 0$, and δ is Dirac's function.*

Which we prove in section B.2. Now, suppose that $\mathcal{H}$ is an r.k.h.s. with r.k. k on $\mathbb{R}^d \times \mathbb{R}^d$. If $\mathcal{H}$ is $T_{\boldsymbol{a}}$-invariant for all $\boldsymbol{a} \in \mathbb{R}^d$ then

$$k(\boldsymbol{x}, \boldsymbol{y}) = k(T_{-\boldsymbol{y}}\boldsymbol{x}, T_{-\boldsymbol{y}}\boldsymbol{y}) = k(\boldsymbol{x} - \boldsymbol{y}, \boldsymbol{0}) \triangleq \phi_T(\boldsymbol{x} - \boldsymbol{y}).$$

If in addition to this $\mathcal{H}$ is O_A-invariant for all orthogonal A, then by choosing A such that $A(\boldsymbol{x} - \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|\, \hat{e}$ where $\hat{e}$ is an arbitrary unit vector in $\mathbb{R}^d$ we have

$$k(\boldsymbol{x}, \boldsymbol{y}) = k(O_A\boldsymbol{x}, O_A\boldsymbol{y}) = \phi_T(O_A(\boldsymbol{x} - \boldsymbol{y})) = \phi_T(\|\boldsymbol{x} - \boldsymbol{y}\|\, \hat{e}) \triangleq \phi_{OT}(\|\boldsymbol{x} - \boldsymbol{y}\|)$$

i.e. k is radial. All of this is straightforward, and a similar analysis can be found in [Wen04]. Indeed the widely used Gaussian kernel satisfies both of the above invariances. But if we now also assume that $\mathcal{H}$ is W_s-scaled for all $s \neq 0$ — this time with arbitrary $g_{W_s}(\mathcal{H})$ — then

$$k(\boldsymbol{x}, \boldsymbol{y}) = g_{W_s}(\mathcal{H}) k(W_s\boldsymbol{x}, W_s\boldsymbol{y}) = g_{W_{|s|}}(\mathcal{H}) \phi_{OT}(|s|\, \|\boldsymbol{x} - \boldsymbol{y}\|)$$

so that letting $r = \|\boldsymbol{x} - \boldsymbol{y}\|$ we have that $\phi_{OT}(r) = g_{W_{|s|}}(\mathcal{H})\phi_{OT}(|s|\, r)$ and hence by lemma 5.2.2 that $\phi_{OT}(r) = a\delta(r) + br^p$ where $a, b, p \in \mathbb{R}$. This

is positive semi-definite for the trivial case $p = 0$, but there are various ways of showing this cannot be non-trivially positive semi-definite for $p \neq 0$. One simple way is to consider two arbitrary vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ such that $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| = d > 0$. For the corresponding Gram matrix

$$K \triangleq \begin{pmatrix} a & bd^p \\ bd^p & a \end{pmatrix},$$

to be positive semi definite we require $0 \leq \det(K) = a^2 - b^2 d^{2p}$, but for arbitrary $d > 0$ and $a < \infty$, this implies $b = 0$. This may seem disappointing, but fortunately there do exist c.p.d. kernel functions with the stated properties, such as the thin-plate kernel.

5.3 Thin-Plate Kernel

Definition 5.3.1. The m-th order **thin-plate kernel** $\phi_m : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is given by

$$\phi_m(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} (-1)^{m-(d-2)/2} \|\boldsymbol{x} - \boldsymbol{y}\|^{2m-d} \log(\|\boldsymbol{x} - \boldsymbol{y}\|) & \text{if } d \in 2\mathbb{N}, \\ (-1)^{m-(d-1)/2} \|\boldsymbol{x} - \boldsymbol{y}\|^{2m-d} & \text{if } d \in (2\mathbb{N} - 1), \end{cases} \tag{5.4}$$

for $\boldsymbol{x} \neq \boldsymbol{y}$, and zero otherwise. ϕ_m is c.p.d. with respect to $\pi_{m-1}(\mathbb{R}^d)$, the set of d-variate polynomials of degree at most $m - 1$. The kernel induces the following norm on the space $F_{\phi_m}\left(\mathbb{R}^d\right)$ of Definition 2.2.10 (this is not obvious — see *e.g.* [Wen04, Wah90])

$$\begin{aligned} &\langle f, g \rangle_{F_{\phi_m}\left(\mathbb{R}^d\right)} \triangleq \langle \psi f, \psi g \rangle_{L_2(\mathbb{R}^d)} \\ &= \sum_{i_1=1}^{d} \cdots \sum_{i_m=1}^{d} \int_{x_1=-\infty}^{\infty} \cdots \int_{x_d=-\infty}^{\infty} \left(\frac{\partial}{\partial x_{i_1}} \cdots \frac{\partial}{\partial x_{i_m}} f \right) \left(\frac{\partial}{\partial x_{i_1}} \cdots \frac{\partial}{\partial x_{i_m}} g \right) \mathrm{d}x_1 \cdots \mathrm{d}x_d \end{aligned}$$

where $\psi : F_{\phi_m}\left(\mathbb{R}^d\right) \to L_2(\mathbb{R}^d)$ is a *regularisation operator*, implicitly defined above.

Clearly $g_{O_A}(F_{\phi_m}\left(\mathbb{R}^d\right)) = g_{T_{\boldsymbol{a}}}(F_{\phi_m}\left(\mathbb{R}^d\right)) = 1$. Moreover, from the chain rule we have

$$\frac{\partial}{\partial x_{i_1}} \cdots \frac{\partial}{\partial x_{i_m}} (f \circ W_s) = s^m \left(\frac{\partial}{\partial x_{i_1}} \cdots \frac{\partial}{\partial x_{i_m}} f \right) \circ W_s \tag{5.5}$$

and therefore since $\langle f, g \rangle_{L_2(\mathbb{R}^d)} = s^d \langle f \circ W_s, g \circ W_s \rangle_{L_2(\mathbb{R}^d)}$,we can immediately write

$$\begin{aligned} \langle \psi (f \circ W_s), \psi (g \circ W_s) \rangle_{L_2(\mathbb{R}^d)} &= s^{2m} \langle (\psi f) \circ W_s, (\psi g) \circ W_s \rangle_{L_2(\mathbb{R}^d)} \\ &= s^{2m-d} \langle \psi f, \psi g \rangle_{L_2(\mathbb{R}^d)} \end{aligned}$$

so that

$$g_{W_s}(F_{\phi_m}\left(\mathbb{R}^d\right)) = s^{-(2m-d)}. \tag{5.6}$$

Note that although it may appear that this can be shown more easily using (5.4) and an argument similar to lemma 5.2.1, the process is actually more involved due to the log factor in the first case of (5.4), and it is necessary to use the fact that the kernel is c.p.d. with respect to (w.r.t.) $\pi_{m-1}(\mathbb{R}^d)$. Since this is redundant and not central to the paper we omit the details.

Understanding the thin-plate invariances from the kernel

We have just seen that the invariances of the thin-plate spline are easy to see from the definition of the regulariser. It turns out however that our characterisation of the kernel function of transformation scaled r.k.h.s.'s in lemma 5.2.1 does not hold for c.p.d. kernels. For completeness we now use only the corresponding kernel function (introduced shortly) to demonstrate that the thin-plate spline is indeed translation and rotation invariant as well as dilation scaled.

The precise form of ϕ_m was derived by Duchon [Duc77], and is different in even and odd space dimensions. As we mentioned in subsection 2.2.4 ϕ_m is only c.p.d. with respect to the null space of the semi-norm, namely $\pi_{m-1}(\mathbb{R}^d)$. Now, lemma 5.2.1 is not valid for c.p.d. kernels as the function $\phi(\cdot, \boldsymbol{x})$ is not the representer of evaluation at $\boldsymbol{x} \in \mathcal{X}$ for $F_\phi(\mathcal{X})$ — indeed it is generally not even in $F_\phi(\mathcal{X})$.

This explains why the condition

$$s^{2m-d}\phi_m(\boldsymbol{x}, \boldsymbol{y}) = \phi_m(s\boldsymbol{x}, s\boldsymbol{y})$$

is not satisfied in general (as lemma 5.2.1 would require, if the kernel were p.d.). We now proceed to demonstrate that the form of ϕ_m does indeed imply the invariances which we mentioned in Definition 5.3.1. For this purpose it is sufficient to note that if ϕ is a c.p.d. kernel on $\mathcal{X} \times \mathcal{X}$ with respect to $\mathcal{P}$ then if

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j\phi(\boldsymbol{x}_i, \boldsymbol{x}_j) = g_{\mathcal{T}}(F_\phi(\mathcal{X}))\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j\phi(\mathcal{T}\boldsymbol{x}_i, \mathcal{T}\boldsymbol{x}_j) \text{ for all } \boldsymbol{\alpha} \in \mathcal{P}^\perp, \tag{5.7}$$

then $F_\phi(\mathcal{X})$ is obviously $\mathcal{T}$-scaled. Of course, for this to be so we additionally require that

$$\sum_{j=1}^{m}\alpha_j\phi(\cdot, \mathcal{T}\boldsymbol{x}_i)$$

is actually in $F_\phi(\mathcal{X})$. In other words we require that

$$\mathcal{P}^\perp(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) = \mathcal{P}^\perp(\mathcal{T}\boldsymbol{x}_1, \ldots, \mathcal{T}\boldsymbol{x}_m),$$

but as the reader may verify this is true for the case we are interested in, namely the space $F_\phi(\mathcal{X}) = F_{\phi_m}\left(\mathbb{R}^d\right)$ and $\mathcal{T} \in \{T_a, O_A, W_s\}$.

We now proceed to show that ϕ_m implies the translation, dilation and orthogonal transformation scaledness which, as we saw in Definition 5.3.1, are possessed by $F_{\phi_m}\left(\mathbb{R}^d\right)$.

We can immediately see that (5.7) holds for the thin-plate for $\mathcal{T} \in \{T_a, O_A\}$, as well as the case $\mathcal{T} = W_s$ with odd d, since in all of these cases we have

$$\phi_m(\boldsymbol{x}, \boldsymbol{y}) = g_{\mathcal{T}}(F_{\phi_m}\left(\mathbb{R}^d\right))\phi(\mathcal{T}\boldsymbol{x}, \mathcal{T}\boldsymbol{y}),$$

e.g. for the case $\mathcal{T} = W_s$ when d is odd we have

$$g_{W_s}(F_{\phi_m}\left(\mathbb{R}^d\right)) = \phi_m(\boldsymbol{x}, \boldsymbol{y})/\phi_m(s\boldsymbol{x}, s\boldsymbol{y}) = s^{-(2m-d)},$$

in agreement with (5.6). The only remaining case is $\mathcal{T} = W_s$ when d is even, which is less obvious due to the log factor in ϕ_m. To show this final case we will use the following

Lemma 5.3.2. *For all sets* $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\} \subseteq \mathbb{R}^d$, *all* $\boldsymbol{\alpha} \in \mathbb{R}^N$ *satisfying* $\sum_{i=1}^N \alpha_i p(\boldsymbol{x}_i) = 0$ *for all* $p \in \pi_{m-1}(\mathbb{R}^d)$, *and all polynomials* q *of degree less than* $2m$

$$\sum_{j,k=1}^{N} \alpha_j \alpha_k q(\boldsymbol{x}_j - \boldsymbol{x}_k) = 0. \tag{5.8}$$

For a proof we refer the reader to, *e.g.* Section 8.1 of [Wen04]. Now, if we define

$$n(s) = \sum_{i,j} \alpha_i \alpha_j \phi_{F_{\phi_m}\left(\mathbb{R}^d\right)}(s\boldsymbol{x}_i, s\boldsymbol{x}_j)$$

it is sufficient to show that

$$\sum_j \alpha_j p(\boldsymbol{x}_j) = 0 \text{ for all } p \in \pi_{m-1}(\mathbb{R}^d) \tag{5.9}$$

implies that $n(s)$ grows like $g_{W_s}(F_{\phi_m}\left(\mathbb{R}^d\right))^{-1}$ in s, *i.e.*

$$\frac{\partial}{\partial s}\left(n(s) g_{W_s}\left(F_{\phi_m}\left(\mathbb{R}^d\right)\right)\right) = 0. \tag{5.10}$$

Putting in (5.4) we have

$$\begin{aligned}\frac{\partial}{\partial s}\left(n(s) g_{W_s}\left(F_{\phi_m}\left(\mathbb{R}^d\right)\right)\right) &= \sum_{i,j} \alpha_i \alpha_j \frac{\partial}{\partial s} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^{2m-d} \log(s\|\boldsymbol{x}_i - \boldsymbol{x}_j\|) \\ &= \frac{1}{s} \sum_{i,j} \alpha_i \alpha_j \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^{2m-d},\end{aligned}$$

which is identically zero due to (5.9) and lemma 5.3.2, concluding a simple way of showing that the thin-plate kernel implies all of the invariances which we observed in the norm that defines it.

5.4 Thin-Plate Spline s.v.m.

In the section 5.2 we showed that non-trivial kernels which are both radial and dilation scaled cannot be p.d. but rather only c.p.d. It is therefore somewhat surprising that the s.v.m. — one of the most widely used kernel algorithms — has been applied only with p.d. kernels, or kernels which are c.p.d. respect only to $\mathcal{P} = \{1\}$ (see *e.g.* [BTB05]). After all, it seems interesting to construct a classifier independent not only of the absolute positions of the input data, but also of their absolute multiplicative scale.

Hence we propose using the thin-plate kernel with the s.v.m. by minimising the s.v.m. objective over the space $F_\phi(\mathcal{X}) \oplus \mathcal{P}$ (or in some cases just over $F_\phi(\mathcal{X})$, as we shall see in subsection 5.4.1). For this we require somewhat non-standard s.v.m. optimisation software. The method we propose seems simpler and more robust than previously mentioned solutions. For example, [SSM98] mentions the numerical instabilities which may arise with the direct application of standard solvers.

5.4.1 Optimising an s.v.m. with c.p.d. Kernel

It is simple to implement an s.v.m. with a kernel ϕ which is c.p.d. w.r.t. an arbitrary finite dimensional space of functions $\mathcal{P}$ by extending the primal optimisation approach of [Cha07] to the c.p.d. case. The quadratic loss s.v.m. solution can be formulated as $\arg\min_{f \in F_\phi(\mathcal{X}) \oplus \mathcal{P}}$ of

$$\lambda \left\| P_\phi(\mathcal{P}) f \right\|^2_{F_\phi(\mathcal{X})} + \sum_{i=1}^{n} \max(0, 1 - y_i f(\boldsymbol{x}_i))^2, \tag{5.11}$$

Note that for the second order thin-plate case we have $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{P} = \pi_1(\mathbb{R}^d)$ (the space of constant and first order polynomials). Hence $\dim(\mathcal{P}) = d + 1$ and we can take the basis to be $p_j(\boldsymbol{x}) = [\boldsymbol{x}]_j$ for $j = 1 \ldots d$ along with $p_{d+1} = 1$.

It follows immediately from theorem 2.2.15 that, letting $p_1, p_2, \ldots p_{\dim(\mathcal{P})}$ span $\mathcal{P}$, the solution to (5.11) is given by $f_{\text{svm}}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i, \boldsymbol{x}) + \sum_{j=1}^{\dim(\mathcal{P})} \beta_j p_j(\boldsymbol{x})$. Now, if we consider only the margin violators — those vectors which (at a given step of the optimisation process) satisfy $y_i f(\boldsymbol{x}_i) < 1$, we can replace the $\max(0, \cdot)$ in (5.11) with $(\cdot)$. This is equivalent to making a local second order approximation. Hence by repeatedly solving in this way while updating the set of margin violators, we will have implemented a so-called Newton optimisation. Now, since

$$\left\| P_\phi(\mathcal{P}) f_{\text{svm}} \right\|^2_{F_\phi(\mathcal{X})} = \sum_{i,j=1}^{n} \alpha_i \alpha_j \phi(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{5.12}$$

the local approximation of the problem is, in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$

$$\text{minimise } \lambda\boldsymbol{\alpha}^\top\Phi\boldsymbol{\alpha} + \|\Phi\boldsymbol{\alpha} + P\boldsymbol{\beta} - \boldsymbol{y}\|^2, \text{ subject to } P^\top\boldsymbol{\alpha} = \mathbf{0}, \tag{5.13}$$

where $[\Phi]_{i,j} = \phi(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $[P]_{j,k} = p_k(\boldsymbol{x}_j)$, and we assumed for simplicity that all vectors violate the margin. The solution in this case is given by [Wah90]

$$\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} = \begin{pmatrix}\lambda I + \Phi & P^\top\\ P & \mathbf{0}\end{pmatrix}^{-1}\begin{pmatrix}\boldsymbol{y}\\ \mathbf{0}\end{pmatrix}. \tag{5.14}$$

In practice it is essential that one makes a change of variable for $\boldsymbol{\beta}$ in order to avoid the numerical problems which arise when P is rank deficient or numerically close to it. In particular we make the QR factorisation [GV96] $P^\top = QR$, where $Q^\top Q = I$ and R is square. We then solve for $\boldsymbol{\alpha}$ and $\overline{\boldsymbol{\beta}} = R\boldsymbol{\beta}$. As a final step at the end of the optimisation process, we take the minimum norm solution of the system $\overline{\boldsymbol{\beta}} = R\boldsymbol{\beta}$, $\boldsymbol{\beta} = R^{\#}\overline{\boldsymbol{\beta}}$ where $R^{\#}$ is the pseudo inverse of R. Note that although (5.14) is standard for squared loss regression models with c.p.d. kernels, our use of it in optimising the s.v.m. is new. The precise algorithm is given in section C.2, where we also detail two efficient factorisation techniques, specific to the new s.v.m. setting. Moreover, the method we present in subsection 5.4.2 deviates considerably further from the existing literature. An example solution obtained this way is depicted in Figure 5.2.

5.4.2 Constraining the solution

Previously, if the data can be separated with only the $\mathcal{P}$ part of the function space — *i.e.* with $\boldsymbol{\alpha} = \mathbf{0}$ — then the algorithm will always do so regardless of λ. This is correct in that, since $\mathcal{P}$ lies in the null space of the regulariser $\|P_\phi(\mathcal{P})\cdot\|^2_{F_\phi(\mathcal{X})}$, such solutions minimise (5.11), but may be undesirable for various reasons. Firstly, the regularisation cannot be controlled via λ. Secondly, for the thin-plate, $\mathcal{P} = \pi_1(\mathbb{R}^d)$ and the solutions are simple linear separating hyperplanes. Finally, there may exist infinitely many solutions to (5.11). It is unclear how to deal with this problem — after all it implies that the regulariser is simply inappropriate for the problem at hand. Nonetheless we still wish to apply a (non-linear) algorithm with the previously discussed invariances of the thin-plate.

To achieve this, we minimise (5.11) as before, but over the space $F_\phi(\mathcal{X})$ rather than $F_\phi(\mathcal{X}) \oplus \mathcal{P}$. It is important to note that by doing so we can no longer invoke theorem 2.2.15, the representer theorem for the c.p.d. case. This is because the solvability argument of lemma 2.2.11 no longer holds. Hence we do not know the optimal basis for the function, which may involve infinitely many $\phi(\cdot, \boldsymbol{x})$ terms. The way we deal with this is simple — instead of minimising over $F_\phi(\mathcal{X})$ we consider only the finite dimensional subspace

given by

$$\left\{ \sum_{j=1}^{n} \alpha_j \phi(\cdot, \boldsymbol{x}_j) : \boldsymbol{\alpha} \in \mathcal{P}^{\perp}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \right\},$$

where $\boldsymbol{x}_1, \ldots \boldsymbol{x}_n$ are those of the original problem (5.11). The required update equation can be acquired in a similar manner as before. The closed form solution to the constrained quadratic programme is in this case given by (see section C.2)

$$\boldsymbol{\alpha} = -P_{\perp} \left(P_{\perp}^{\top} \left(\lambda \Phi + \Phi_{sx}^{\top} \Phi_{sx} \right) P_{\perp} \right)^{-1} P_{\perp}^{\top} \Phi_{sx}^{\top} \boldsymbol{y}_s \tag{5.15}$$

where $\Phi_{sx} = [\Phi]_{s,:}$, s is the current set of margin violators and $P_{\perp}$ the null space of P satisfying $PP_{\perp} = \mathbf{0}$. The precise algorithm we use to optimise in this manner is given in section C.2, where we also detail efficient factorisation techniques. An example solution obtained in this manner is depicted in Figure 5.2.

5.4.3 p.d. Kernels from c.p.d. Kernels

The method we now propose involves modifying c.p.d. kernels such that they are p.d., but hopefully retain some of the properties of the regulariser to which the original c.p.d. kernel corresponds. As a starting point consider the following result [BCR84]:

Theorem 5.4.1. *Let $\mathcal{X}$ be a non-empty set, $\boldsymbol{x}_0 \in \mathcal{X}$ and $k_{cpd} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric kernel and put*

$$k(\boldsymbol{x}, \boldsymbol{y}) = k_{cpd}(\boldsymbol{x}, \boldsymbol{y}) - k_{cpd}(\boldsymbol{x}, \boldsymbol{x}_0) - k_{cpd}(\boldsymbol{y}, \boldsymbol{x}_0) + k_{cpd}(\boldsymbol{x}_0, \boldsymbol{x}_0).$$

k is p.d. if and only if k_{cpd} is c.p.d. with respect to $\{1\}$, the constant function.

Proof. If $\sum_{i=1}^{m} \alpha_i = 0$ then

$$\sum_{j,k=1}^{m} \alpha_j \alpha_k k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{j,k=1}^{m} \alpha_j \alpha_k k_{\mathrm{cpd}}(\boldsymbol{x}_i, \boldsymbol{x}_j),$$

so if k is p.d. then k_{cpd} is c.p.d. Now if we choose $\alpha_0 = -\sum_{i=1}^{m} \alpha_i$ then $\sum_{i=0}^{m} \alpha_i = 0$ so that if k_{cpd} is c.p.d. we have

$$\begin{aligned}
0 &< \sum_{j,k=0}^{m} \alpha_j \alpha_k k_{\mathrm{cpd}}(\boldsymbol{x}_j, \boldsymbol{x}_k) \\
&= \sum_{j,k=1}^{m} \alpha_j \alpha_k k_{\mathrm{cpd}}(\boldsymbol{x}_j, \boldsymbol{x}_k) + \sum_{j=1}^{m} \alpha_j \alpha_0 k_{\mathrm{cpd}}(\boldsymbol{x}_j, \boldsymbol{x}_0) + \sum_{k=1}^{m} \alpha_0 \alpha_k k_{\mathrm{cpd}}(\boldsymbol{x}_0, \boldsymbol{x}_k) + \alpha_0 \alpha_0 k_{\mathrm{cpd}}(\boldsymbol{x}_0, \boldsymbol{x}_0) \\
&= \sum_{j,k=1}^{m} \alpha_j \alpha_k \left(k_{\mathrm{cpd}}(\boldsymbol{x}_j, \boldsymbol{x}_k) - k_{\mathrm{cpd}}(\boldsymbol{x}_j, \boldsymbol{x}_0) - k_{\mathrm{cpd}}(\boldsymbol{x}_0, \boldsymbol{x}_k) + k_{\mathrm{cpd}}(\boldsymbol{x}_0, \boldsymbol{x}_0) \right) \\
&= \sum_{j,k=1}^{m} \alpha_j \alpha_k k(\boldsymbol{x}_j, \boldsymbol{x}_k).
\end{aligned}$$

□

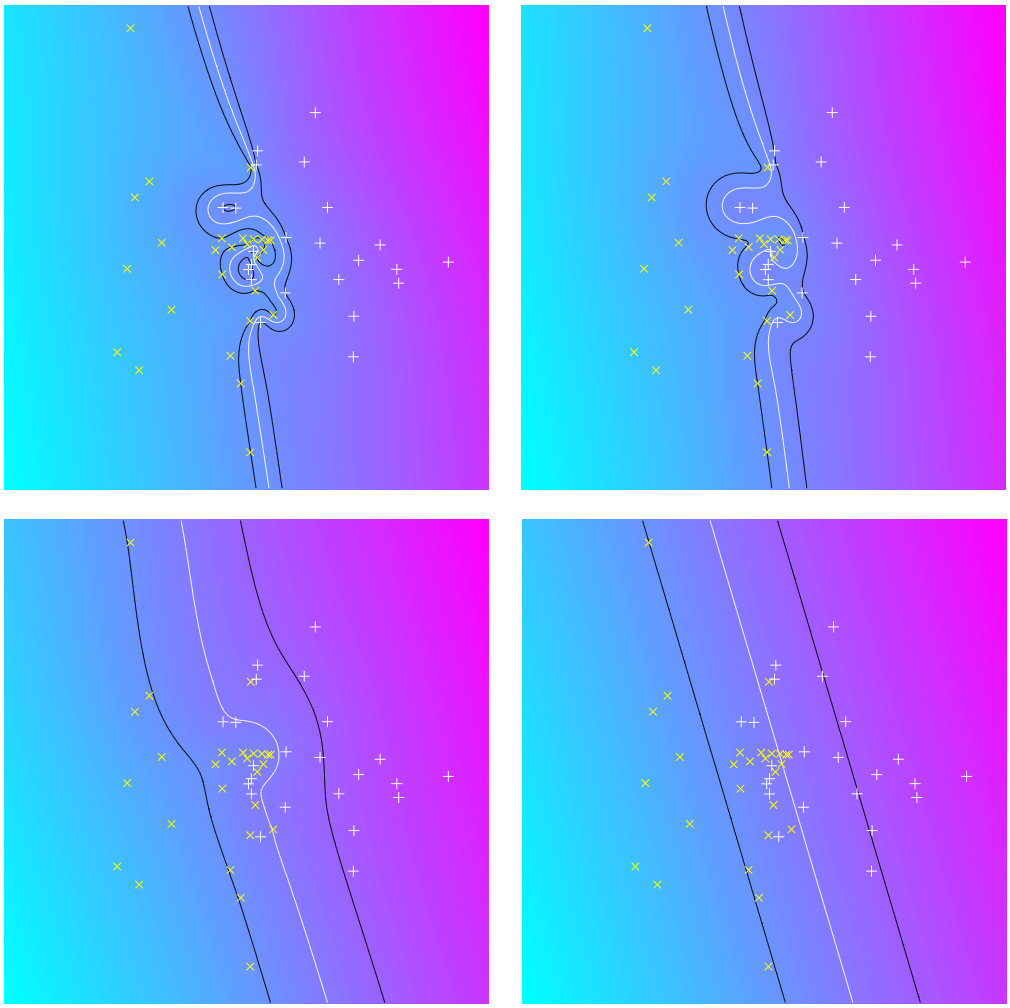


Figure 5.2: An s.v.m. separating white pluses from yellow crosses, with quadratic loss (*i.e.* $p = 2$ in (5.11)) and the $m = 2$ thin-plate spline kernel as per Definition 5.3.1. The algorithm is invariant to rotation and translation, and dilating the data has the same effect as varying the regularisation parameter. The solutions are exact ones resulting from the procedure described in the first part of subsection 5.4.1. The white line represents the zero level, the black line the ± 1 levels of the decision function f_{svm}. The regularisation parameter λ (or equivalently, to restate, the absolute scale of the data) increases from ≈ 0 on the top-left to $\approx \infty$ on the bottom-right.

We now derive an analogous case for arbitrary c.p.d. kernels. The basic idea is already clear from the previous theorem — there, an extra point $\boldsymbol{x}_0$ was chosen along with its coefficient α_0 such that the constraint $\sum_i \alpha_i = 0$ was satisfied. Now we need to choose an extra $\dim(\mathcal{P})$ points along with the corresponding coefficients such that the resulting coefficients lie in $\mathcal{P}^\perp$. The following Lemma does exactly this. The form of the Lemma as we derived it was inspired by by the above result. We have since become aware however that a similar statement is made in *e.g.* [Wah90] and the *Native Spaces* Chapter of [Wen04], although with a completely different proof idea.

Theorem 5.4.2. *Let $\mathcal{X}$ be a non-empty set, $\boldsymbol{z}_1, \ldots \boldsymbol{z}_{dim(\mathcal{P})}$ be a $\mathcal{P}$-unisolvent subset of $\mathcal{X}$, $p_1, \ldots p_{dim(\mathcal{P})}$ be a Lagrange basis with respect to $\mathcal{P}$, $k_{cpd} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric kernel and put $k(\boldsymbol{x}, \boldsymbol{y})$ equal to*

$$\begin{aligned} k_{cpd}(\boldsymbol{x}, \boldsymbol{y}) - \sum_{k=1}^{dim(\mathcal{P})} p_k(\boldsymbol{x}) k_{cpd}(\boldsymbol{z}_k, \boldsymbol{y}) - \sum_{k=1}^{dim(\mathcal{P})} p_k(\boldsymbol{y}) k_{cpd}(\boldsymbol{x}, \boldsymbol{z}_k) \\ + \sum_{k,l=1}^{dim(\mathcal{P})} p_k(\boldsymbol{x}) p_l(\boldsymbol{y}) k_{cpd}(\boldsymbol{z}_k, \boldsymbol{z}_l) \end{aligned} \tag{5.16}$$

k is p.d. if and only if k_{cpd} is c.p.d. with respect to $\mathcal{P}$.

Proof. If $\sum_{j=1}^m \alpha_j p_k(\boldsymbol{x}_j) = 0$ for all $k = 1 \ldots \dim(\mathcal{P})$ then

$$\sum_{j,k=1}^m \alpha_j \alpha_k k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{j,k=1}^m \alpha_j \alpha_k k_{\mathrm{cpd}}(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{5.17}$$

so if k is p.d. then k_{cpd} is c.p.d. Now, due to the statements about the $\mathcal{P}$-unisolvence and the Lagrange basis we have

$$p_k(\boldsymbol{z}_l) = \delta_{kl},$$

so that if $m > \dim(\mathcal{P})$ and we choose for $k = 1 \ldots \dim(\mathcal{P})$

$$\boldsymbol{x}_k = \boldsymbol{z}_k$$

and

$$\alpha_k = - \sum_{j=\dim(\mathcal{P})+1}^m \alpha_j p_k(\boldsymbol{x}_j),$$

then for all $k = 1 \ldots \dim(\mathcal{P})$ we have

$$\sum_{i=1}^m \alpha_i p_k(\boldsymbol{x}_i) = 0,$$

and if k_{cpd} is c.p.d. it is simple but tedious to see that

$$
\begin{aligned}
0 &< \sum_{i,j=1}^{m} \alpha_i \alpha_j k_{\text{cpd}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
&= \sum_{i=\dim(\mathcal{P})+1}^{m} \sum_{j=\dim(\mathcal{P})+1}^{m} \alpha_i \alpha_j k_{\text{cpd}}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{k=1}^{\dim(\mathcal{P})} \sum_{j=\dim(\mathcal{P})+1}^{m} \alpha_k \alpha_j k_{\text{cpd}}(\boldsymbol{x}_k, \boldsymbol{x}_j) \\
&+ \sum_{i=\dim(\mathcal{P})+1}^{m} \sum_{k=1}^{\dim(\mathcal{P})} \alpha_i \alpha_k k_{\text{cpd}}(\boldsymbol{x}_i, \boldsymbol{x}_k) + \sum_{k=1}^{\dim(\mathcal{P})} \sum_{l=1}^{\dim(\mathcal{P})} \alpha_k \alpha_l k_{\text{cpd}}(\boldsymbol{x}_k, \boldsymbol{x}_l) \\
&= \sum_{i=\dim(\mathcal{P})+1}^{m} \sum_{j=\dim(\mathcal{P})+1}^{m} \alpha_i \alpha_j k_{\text{cpd}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
&- \sum_{j=\dim(\mathcal{P})+1}^{m} \sum_{i=\dim(\mathcal{P})+1}^{m} \sum_{k=1}^{\dim(\mathcal{P})} \alpha_i p_k(\boldsymbol{x}_i) \alpha_j k_{\text{cpd}}(\boldsymbol{x}_k, \boldsymbol{x}_j) \\
&- \sum_{i=\dim(\mathcal{P})+1}^{m} \sum_{j=\dim(\mathcal{P})+1}^{m} \sum_{k=1}^{\dim(\mathcal{P})} \alpha_j p_k(\boldsymbol{x}_j) \alpha_i k_{\text{cpd}}(\boldsymbol{x}_i, \boldsymbol{x}_k) \\
&+ \sum_{i=\dim(\mathcal{P})+1}^{m} \sum_{j=\dim(\mathcal{P})+1}^{m} \sum_{k=1}^{\dim(\mathcal{P})} \sum_{l=1}^{\dim(\mathcal{P})} \alpha_j p_k(\boldsymbol{x}_j) \alpha_i p_l(\boldsymbol{x}_i) k_{\text{cpd}}(\boldsymbol{x}_k, \boldsymbol{x}_l) \\
&= \sum_{i,j=\dim(\mathcal{P})+1}^{m} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j). \qquad \square
\end{aligned}
$$

It is not clear that a p.d. kernel so constructed has similar properties to the original c.p.d. kernel — but as shown in Figure 5.3, there does appear to be a qualitative similarity at least with data in $\mathbb{R}^2$. Note that the Lagrange basis can be constructed using numerical means. Assume that we have a function

$$
\begin{aligned}
q : \mathcal{X} &\to \mathbb{R}^{\dim(\mathcal{P})} \\
\boldsymbol{x} &\to \left(q_1(\boldsymbol{x}), \ldots, q_{\dim(\mathcal{P})}(\boldsymbol{x})\right)^\top,
\end{aligned}
$$

where the q_k span $\mathcal{P}$, as well as a $\mathcal{P}$-unisolvent basis $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{\dim(\mathcal{P})}$ (*i.e.* a basis leading to a non-singular Q_z below). If we now define

$$
Q_z = \left(q(\boldsymbol{z}_1) \cdots q(\boldsymbol{z}_{\dim(\mathcal{P})})\right),
$$

then can construct a Lagrange basis $p_1, \ldots, p_{\dim(\mathcal{P})}$ of the form

$$
\left(p_1(\boldsymbol{x}), \ldots, p_{\dim(\mathcal{P})}(\boldsymbol{x})\right)^\top \triangleq Q_z^{-1} q(\boldsymbol{x}). \tag{5.18}
$$

This may not be the most efficient approach, but it does allow one to conveniently experiment with p.d. analogs of arbitrary c.p.d. kernels. Next, we take a more elegant approach to the specific thin-plate spline case in which we are interested.

Example 5.4.3
One practical problem with (5.16) *is the double sum, which leads to a computational time complexity of* $O(dim\,(\mathcal{P})^2)$*. In some cases this can be avoided however — consider the kernel* k_{tp} *on* $\mathbb{R}^d \times \mathbb{R}^d$ *defined by*

$$k_{tp}(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|^2 \log(\|\boldsymbol{x} - \boldsymbol{y}\|),$$

which is c.p.d. with respect to $\pi_1(\mathbb{R}^d)$ *as we can be proven using Michelli's Theorem [Wen04]. Since this expression vanishes when* $\|\boldsymbol{x} - \boldsymbol{y}\| = 1$*, if we choose for the* $\pi_1(\mathbb{R}^d)$*-unisolvant basis (i.e. the* $\boldsymbol{z}_1, \ldots \boldsymbol{z}_{dim(\mathcal{P})}$ *of theorem 5.4.2) the* $dim(\pi_1(\mathbb{R}^d)) = d + 1$ *vectors*

$$\mathbf{0}, \boldsymbol{e}_1/\sqrt{2}, \boldsymbol{e}_2/\sqrt{2}, \ldots, \boldsymbol{e}_d/\sqrt{2} \;\subset\; \mathbb{R}^d$$

then the problem with the double sum is avoided. It is easy to verify that a Lagrange basis for $\pi_1(\mathbb{R}^d)$ *with respect to the above basis is, for* $k = 1 \ldots d$

$$p_k(\boldsymbol{x}) = \sqrt{2}\,[\boldsymbol{x}]_k\,,$$

along with $p_{d+1} = 1 - \sqrt{2}\boldsymbol{e}^\top \boldsymbol{x}$*. Putting this into* (5.16) *and making a few simplifications leads to*

$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{y}) = k_{tp}(\boldsymbol{x}, \boldsymbol{y}) + \sqrt{2}\,(\breve{x} k_{tp}(\mathbf{0}, \boldsymbol{y}) + \breve{y} k_{tp}(\mathbf{0}, \boldsymbol{x})) + \frac{\log(2)}{2}\,(\breve{x}\overline{y} + \breve{y}\overline{x}) \\ - \sqrt{2} \sum_{k=1}^{d} \left(k_{tp}(\boldsymbol{x}, \boldsymbol{e}_k/\sqrt{2})\,[\boldsymbol{y}]_k + k_{tp}(\boldsymbol{y}, \boldsymbol{e}_k/\sqrt{2})\,[\boldsymbol{x}]_k \right), \end{aligned} \tag{5.19}$$

as our $O(d)$ *time complexity p.d. analog of* k_{tp}*, where* $\overline{x} = \boldsymbol{e}^\top \boldsymbol{x}$ *and* $\breve{x} = \overline{x} - 1/\sqrt{2}$*. We provide classification performance results using the above kernel in Table 5.1. Note that the kernel tends to behave better when the input points are not too close to the set* $\mathbf{0}, \boldsymbol{e}_1/\sqrt{2}, \boldsymbol{e}_2/\sqrt{2}, \ldots, \boldsymbol{e}_d/\sqrt{2}$*. Accordingly, for our classification tests we first added a multiple of* $\boldsymbol{e}$ *to all of the input vectors.*

5.4.4 Experiments

We now investigate the behaviour arising from the algorithms which we have just discussed. To summarise, the three types of thin-plate spline based s.v.m. which we employ are

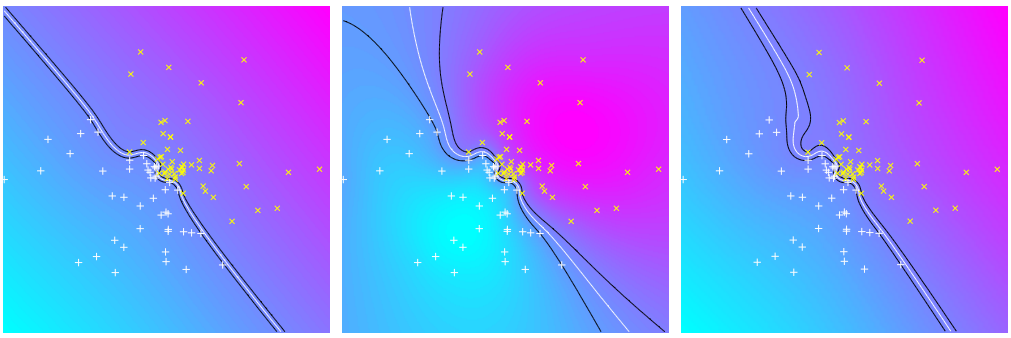


Figure 5.3: An s.v.m. separating white pluses from yellow crosses, with quadratic loss (*i.e.* $p = 2$ in (5.11)). The white line represents the zero level, the black line the ± 1 levels of the decision function f_{svm}. **Left:** the exact solution with the $m = 2$ thin-plate spline kernel as per Definition 5.3.1, solved by the algorithm of subsection 5.4.1. **Middle:** similar but with the $\boldsymbol{\beta} = \mathbf{0}$ constraint discussed in the latter part of subsection 5.4.1. **Right:** the solution arising from the p.d. analog (5.19) of the thin-plate spline. Note that the three solutions are rather similar to one another in the central part of the figure, but diverge as one moves outwards. Also interesting is the fact that in the left figure, points are allowed to be closer to the zero level as this choice yields a more linear classification function, that is one which relies more on the $\mathcal{P}$ part of the function space. All three solutions were obtained via Algorithm C.1.

1. The optimisation over $F_\phi(\mathcal{X}) \oplus \mathcal{P}$ as per subsection 5.4.1.
2. The approximate optimisation over $F_\phi(\mathcal{X})$ as per the latter part of subsection 5.4.1.
3. The optimisation using the p.d. analog (5.19) of the thin-plate spline.

As demonstrated in Figure 5.3, the above approaches do seem to yield results which bear qualitative similarities, in particular stability on data which has features on various scales.

Using the Gaussian kernel (2.26) as a baseline, we also compared the algorithms on real world classification problems. For the true thin-plate spline kernel case corresponding to the first two points above, we used the first method if the data was not linearly separable, otherwise we used the second.

The results, given in Table 5.1, indicate that the thin-plate methods are competitive with the Gaussian kernel method. Importantly however, the dilation scaled nature of the thin-plate regulariser means that the thin-plate approach is free of a length scale parameter. Indeed, in the experiments summarised in the table, for the thin-plate cases we had to perform cross validation in order to choose only the regularisation parameter λ, whereas for the Gaussian we had to choose both λ and the scale parameter σ. The discovery of an equally effective algorithm which has only one parameter is interesting, since the Gaussian is probably the most popular and effective kernel used with the s.v.m. [HCL03].

Data-set	Gaussian	p.d. thin-plate	thin-plate	dim	n
UCI banana	**10.567 (0.547)**	10.667 (0.586)	10.667 (0.586)	2	3000*
UCI breast cancer	**26.574 (2.259)**	28.767 (3.136)	28.026 (2.900)	9	263
UCI diabetis	23.578 (0.989)	23.966 (1.152)	**23.452 (1.215)**	8	768
UCI flare solar	36.143 (0.969)	**35.429 (0.620)**	38.190 (2.317)	9	144
UCI german	**24.700 (1.453)**	**24.700 (1.342)**	24.800 (1.373)	20	1000
UCI heart	17.407 (2.142)	**16.667 (1.932)**	17.037 (2.290)	13	270
UCI image	3.210 (0.504)	1.915 (0.342)	**1.867 (0.338)**	18	2086
UCI ringnorm	**1.533 (0.229)**	1.767 (0.193)	1.833 (0.200)	20	3000*
UCI splice	8.931 (0.640)	9.106 (0.683)	**8.651 (0.433)**	60	2844
UCI thyroid	4.199 (1.087)	3.701 (1.160)	**3.247 (1.211)**	5	215
UCI twonorm	1.833 (0.194)	**1.800 (0.200)**	1.867 (0.254)	20	3000*
UCI waveform	8.333 (0.378)	**7.933 (0.441)**	8.233 (0.484)	21	3000
SSL 1: Digit 1	**2.068 (0.418)**	3.666 (0.347)	2.269* (0.540)	241	1500
SSL 2: USPS	**2.867 (0.445)**	5.133 (0.703)	2.933* (0.424)	241	1500
SSL 3: COIL_2	**0.467 (0.142)**	1.333 (0.385)	0.800* (0.259)	241	1500
SSL 4: BCI	**15.750 (1.789)**	19.500 (2.000)	16.500* (2.014)	117	400
SSL 5: g241c	11.067 (0.896)	**10.933 (0.796)**	16.133 (1.299)	241	1500
SSL 7: g241n	**8.597 (0.916)**	11.994 (1.040)	15.067 (1.178)	241	1500
USPS 0	0.600 (0.191)	0.767 (0.228)	**0.500* (0.181)**	256	3000*
USPS 1	0.433 (0.118)	0.542 (0.155)	**0.325* (0.147)**	256	2769*
USPS 2	1.729 (0.273)	1.894 (0.348)	**1.071* (0.196)**	256	2429*
USPS 3	**1.548 (0.302)**	2.237 (0.286)	1.550* (0.206)	256	2324*
USPS 4	**1.404 (0.354)**	2.296 (0.371)	2.254* (0.805)	256	2352*
USPS 5	**1.625 (0.235)**	1.987 (0.280)	1.715* (0.268)	256	2216*
USPS 6	0.728 (0.231)	1.286 (0.286)	**0.685* (0.194)**	256	2334*
USPS 7	1.047 (0.186)	1.527 (0.198)	**0.959* (0.192)**	256	2292*
USPS 8	**1.404 (0.172)**	2.174 (0.189)	1.540* (0.245)	256	2208*
USPS 9	1.077 (0.116)	1.594 (0.204)	**1.034* (0.115)**	256	2321*

Table 5.1: Classification performance of the s.v.m. with quadratic loss (*i.e.* $p = 2$ in (2.20)), using either the Gaussian kernel or the p.d. analog of the thin-plate kernel as per (5.19). To compute each error measure, we used five splits of the data — testing on each split after training on the remainder, and report the results in the form of "mean % classification error (standard error)". For parameter selection, we performed five-fold cross validation on the four-fifths of the data available for training each split, over an exhaustive search of the algorithm parameter(s) (σ and λ for the Gaussian and λ for the other two), taking the parameter(s) with lowest mean error rate and retraining on the entire four-fifths. In each case we made sure the chosen parameters were well within the searched range by visually inspecting the cross validation error as a function of the parameters. *dim* is the input dimension and n the total number of training examples (*i.e.* which we randomly split into five folds). A star in the n column means that more examples were available but we discarded them in order to reduce the computational burden. The UCI and USPS data sets are standard ones (as used in *e.g.* [MRW+03]), and *e.g.* "USPS 3" means learning to discriminate the digit three from all other digits. The SSL data-sets are described in [CSZ06]. A star in the *thin-plate* column indicates that the data was linearly separable and so we used the $\boldsymbol{\beta} = \mathbf{0}$ constrained version of the optimisation as described in the latter part of subsection 5.4.1.

Appendix A

Formulae and Notation

A.1 Notation and Abbreviations

Abbreviations

r.k.h.s.	Reproducing Kernel Hilbert Space
m.a.p.	Maximum *a posteriori*
k.r.r.	Kernel Ridge Regression
g.p.	Gaussian Process
s.v.m.	Support Vector Machine
r.b.f.	Radial Basis Function
p.d.	Positive Definite
p.s.d.	Positive Semi-Definite
c.p.d.	Conditionally Positive Definite
p.d.f.	Probability Density Function
i.i.d.	Independent and Identically Distributed
f.m.m.	Fast Multipole Method
w.r.t.	With Respect To

Notation

$\triangleq$	Equals by definition
$\imath$	The unit imaginary number, *i.e.*$\sqrt{-1}$
$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \ldots$	Column vectors (bold lowercase)
$\wedge, \vee$	Logical *and*, logical *or*
$A, B, C, \ldots$	Matrices (uppercase)
$[\boldsymbol{a}]_i$	The i-th element of the vector $\boldsymbol{a}$
$[A]_{i,j}$	The i, j-th element of the Matrix A
$[A]_{:,j}$	A column vector of the j-th column of A
$[A]_{i,:}$	A row vector of the i-th row of A
$[A]_{1:i,j}$	Submatrix of rows 1 to i and column j of A
$[B\ C]$	B and C are partitions of the matrix $[B\ C]$
$\mathbf{0}$	Column vector of zeros
$\boldsymbol{e}$	Column vector of ones
$\boldsymbol{e}_i$	Column vector satisfying $[\boldsymbol{e}_i]_j = \delta_{ij}$
I	The identity matrix
∇	The gradient operator
$\nabla^2 f$	The Laplacian operator
∂_x	Partial derivative operator, *i.e.*$\frac{\partial}{\partial x}$
$g', g'', g^{(n)}$	1-st, 2-nd, and n-th derivative of g
$\mathrm{diag}(\boldsymbol{a})$	A diagonal matrix with $[\mathrm{diag}(\boldsymbol{a})]_{ii} = [\boldsymbol{a}]_i$
$C(\mathcal{X}), C^m(\mathcal{X})$	Continuous (cont. in m-th derivative) fns on $\mathcal{X}$
$\mathrm{diag}(A)$	A column vector with $[\mathrm{diag}(A)]_i = [A]_{ii}$
$a\|b$	Conditional random variable
$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$	Multivariate normal distribution (see A.4)
$\mathcal{F}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\right](\boldsymbol{\omega})$	Fourier transform (see A.3)
$\mathcal{F}^{-1}_{\boldsymbol{\omega}}\left[F(\boldsymbol{\omega})\right](\boldsymbol{x})$	Inverse Fourier transform (see A.3)
$\delta(\cdot)$	Dirac delta distribution
δ_{ij}	Kronecker delta
$\mathrm{supp}(f)$	Support of a function, *i.e.*$\{\boldsymbol{x} : f(\boldsymbol{x}) \neq 0\}$
$(f \otimes g)(\boldsymbol{y})$	Vector Convolution, *i.e.*$\langle f(\cdot), g(\cdot - \boldsymbol{y})\rangle_{L_2}$
$\pi_m(\mathbb{R}^d)$	Space of d-variate polynomials with degree $\leq m$
$\arg\min_{x\in\mathcal{X}} f(x)$	Minimiser(s) of f, $\{x : \forall x' \in \mathcal{X}, f(x) \leq f(x')\}$
$\arg\max_{x\in\mathcal{X}} f(x)$	Maximiser(s) of f, $\{x : \forall x' \in \mathcal{X}, f(x) \geq f(x')\}$
$O(\cdot)$	Big O notation (asymptotic complexity)

A.2 Useful Algebra

Matrix Inversion Lemma

The Matrix Inversion Lemma or Sherman-Morris-Woodbury formula states that

$$\left(A + CBC^\top\right)^{-1} = A^{-1} - A^{-1}C\left(B^{-1} + C^\top A^{-1}C\right)^{-1}C^\top A^{-1}, \qquad \text{(A.1)}$$

provided the inverses exist.[1] In particular if P and R are positive definite then

$$\left(P^{-1} + B^\top R^{-1}B\right)^{-1}B^\top R^{-1} = PB^\top\left(BPB^\top + R\right)^{-1}.$$

Partitioned Matrix Inverse

If A is a symmetric matrix given by

$$A = \begin{bmatrix} P & Q \\ Q^\top & S \end{bmatrix} = \begin{bmatrix} \tilde{P} & \tilde{Q} \\ \tilde{Q}^\top & \tilde{S} \end{bmatrix}^{-1},$$

where P (resp. S) is square and the same size as $\tilde{P}$ ($\tilde{S}$), then

$$\begin{aligned} \tilde{P} &= P^{-1} + P^{-1}QM^{-1}Q^\top P^{-1}, \\ \tilde{Q} &= -P^{-1}QM^{-1}, \\ \tilde{S} &= M^{-1}, \end{aligned}$$

where $M = S - Q^\top P^{-1}Q$.

Multinomial Theorem

$$\left(\sum_{i=1}^{m} x_i\right)^p = \sum_{\substack{p_1,p_2,\ldots,p_m \geq 0 \\ p_1+p_2+\ldots+p_m=p}} \frac{p!}{\prod_{j=1}^{m} p_j!} \prod_{i=1}^{m} x_i^{p_i}$$

A.3 Fourier Transforms

Definition

We use the following pedantic notation for the Fourier transform and its inverse:

$$\begin{aligned} \mathcal{F}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\right](\boldsymbol{\omega}) &= (2\pi)^{-d/2}\int_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x})\exp(-\boldsymbol{x}^\top\boldsymbol{\omega}\imath)\mathrm{d}\boldsymbol{x} \\ \mathcal{F}^{-1}_{\boldsymbol{\omega}}\left[F(\boldsymbol{\omega})\right](\boldsymbol{x}) &= (2\pi)^{-d/2}\int_{\boldsymbol{\omega}\in\mathbb{R}^d} F(\boldsymbol{\omega})\exp(\boldsymbol{x}^\top\boldsymbol{\omega}\imath)\mathrm{d}\boldsymbol{\omega}. \end{aligned}$$

[1] The proof is analogous to (2.19) *etc.*in the text.

Fourier Transform of Radial Functions

For radially symmetric functions in $\mathbb{R}^d$ the Fourier transform is its own inverse (as is easily seen from the definitions above), and can be computed by the single integral

$$\mathcal{F}_{\boldsymbol{x}}\left[g_r(\|\boldsymbol{x}\|)\right](\|\boldsymbol{\omega}\|) = \frac{(2\pi)^{\frac{d}{2}}}{||\boldsymbol{\omega}||^{\frac{d-2}{2}}} \int_0^\infty r^{\frac{d}{2}} g_r(r) J_{\frac{d-2}{2}}(||\boldsymbol{\omega}||r)\mathrm{d}r, \tag{A.2}$$

where $J_\nu(r)$is the ν-th order Bessel function of the first kind,

$$J_\nu(r) = \sum_{m=0}^{\infty} \frac{(-1)^m}{\Gamma(m+\nu+1)m!} \left(\frac{r}{2}\right)^{2m+\nu}.$$

Fourier Transform Identities

$$\mathcal{F}_{\boldsymbol{x}}\left[(f \otimes g)(\boldsymbol{x})\right](\cdot) = \mathcal{F}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\right](\cdot)\mathcal{F}_{\boldsymbol{x}}\left[g(\boldsymbol{x})\right](\cdot) \tag{A.3}$$

$$\mathcal{F}_{\boldsymbol{x}}\left[\frac{\partial}{\partial\left[\boldsymbol{x}\right]_j} f(\boldsymbol{x})\right](\boldsymbol{\omega}) = \left(2\pi\left[\boldsymbol{\omega}\right]_j \imath\right) \mathcal{F}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\right](\boldsymbol{\omega}) \tag{A.4}$$

$$\mathcal{F}_{\boldsymbol{x}}\left[f(\boldsymbol{x}-\boldsymbol{a})\right](\boldsymbol{\omega}) = \exp(-\boldsymbol{a}^\top\boldsymbol{\omega}\imath)\mathcal{F}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\right](\boldsymbol{\omega}) \tag{A.5}$$

$$\mathcal{F}_{\boldsymbol{x}}\left[f(s\boldsymbol{x})\right](\boldsymbol{\omega}) = \frac{1}{|s|^d}\mathcal{F}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\right](\boldsymbol{\omega}/s) \tag{A.6}$$

A.4 Gaussian Random Variable

The multivariate Gaussian probability density function (p.d.f.) on $\boldsymbol{x} \in \mathbb{R}^d$ is given by

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-d/2}\left|\boldsymbol{\Sigma}\right|^{-1/2} \exp\left(-\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)^\top \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)\right).$$

Now if $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ then

$$(A\boldsymbol{x}+\boldsymbol{b}) \sim \mathcal{N}(A\boldsymbol{\mu}+\boldsymbol{b}, A\boldsymbol{\Sigma}A^\top).$$

If $\boldsymbol{x}$ and $\boldsymbol{y}$ be distributed according to

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}, \begin{bmatrix} A & C \\ C^\top & B \end{bmatrix}\right),$$

then the conditional distributions are

$$\begin{aligned} \boldsymbol{x}|\boldsymbol{y} &\sim \mathcal{N}\left(a + CB^{-1}(\boldsymbol{y}-\boldsymbol{b}), A - CB^{-1}C^\top\right), \\ \boldsymbol{y}|\boldsymbol{x} &\sim \mathcal{N}\left(b + C^\top A^{-1}(\boldsymbol{x}-\boldsymbol{a}), B - C^\top A^{-1}C\right). \end{aligned} \tag{A.7}$$

Multiplication or convolution of Gaussians leads to an unnormalised Gaussian:

$$\begin{aligned}\mathcal{N}(\boldsymbol{a}, A)\mathcal{N}(\boldsymbol{b}, B) &\propto \mathcal{N}\left(CA^{-1}\boldsymbol{a} + CB^{-1}\boldsymbol{b}, \left(A^{-1} + B^{-1}\right)^{-1}\right), \\ \mathcal{N}(\boldsymbol{a}, A) \otimes \mathcal{N}(\boldsymbol{b}, B) &\propto \mathcal{N}\left(a + b, A + B\right).\end{aligned}$$

The Fourier transform of the Gaussian p.d.f. is an unnormalised Gaussian:

$$\mathcal{F}_{\boldsymbol{x}}\left[\exp\left(-\left\|\boldsymbol{x}\right\|_{\mathbb{R}^d}^2/(2\sigma^2)\right)\right](\boldsymbol{\omega}) = \sigma^d \exp\left(-\left\|\boldsymbol{\omega}\right\|_{\mathbb{R}^d}^2 \sigma^2/2.\right) \tag{A.8}$$

Finally we have the following integral in closed form:

$$\int_{\mathbb{R}^d} (\boldsymbol{x} - \boldsymbol{\mu})\, \boldsymbol{\Sigma}^{-1}\, (\boldsymbol{x} - \boldsymbol{\mu})\, \mathcal{N}(\boldsymbol{m}, S)\mathrm{d}\boldsymbol{x} = (\boldsymbol{\mu} - \boldsymbol{m})^\top\, \boldsymbol{\Sigma}^{-1}\, (\boldsymbol{\mu} - \boldsymbol{m}) + \mathrm{Tr}\left[\boldsymbol{\Sigma}^{-1} S\right].$$

Appendix B

Mathematical Addenda

B.1 Proof of Lemma 3.1.2

We need to a) prove which (dilated) Gaussian functions lie in the reproducing kernel Hilbert space (r.k.h.s.) of a Gaussian, and b) compute the norm of those which do. The inner product then follows from the parallelogram identity. Note that the result corresponding to a) is mentioned in [BJ01] (in the proof of their Theorem 2), based on a characterisation of the r.k.h.s. in terms of Fourier transforms — for a sketch of this simple approach see subsection B.1.2.

B.1.1 Eigendecomposition Based Approach

Presently we prove both a) and b) (described above) in a direct manner using the following

Theorem B.1.1 (Aronzajn [Aro50])**.** *The function f belongs to the r.k.h.s. $\mathcal{H}$ with reproducing kernel (r.k.) k if and only if there exists an $\epsilon > 0$ such that*

$$R_\epsilon(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y}) - \epsilon f(\boldsymbol{x}) f(\boldsymbol{y}),$$

is positive definite (p.d.), in which case

$$\|f\|_{\mathcal{H}}^2 = \inf \{1/\epsilon : R_\epsilon \text{ is p.d.}\}.$$

Let

$$p(x) = \exp(-2ax^2)$$

and

$$k(x, y) = \exp(-b(x - y)^2).$$

It turns out that we can write ([RW06], *Chapter 4.3, Eigenfunction Analysis of Kernels*)

$$k(x,y) = \sum_{m=0}^{\infty} \lambda_m \phi_m(x)\phi_m(y),$$

where

$$\lambda_m = \sqrt{\frac{2c}{A}}\frac{B^m}{2^m m!} \quad \neq \quad \sqrt{\frac{2a}{A}}\,B^m,$$
$$\phi_m(x) = \exp(-(c-a)x^2)H_m(\sqrt{2c}\,x),$$

and

$$c = \sqrt{a^2+2ab},\ A = a+b+c,\ B = b/A.$$

Note that what follows the $\neq$ for λ_m was given in [RW06], but is incorrect. The functions $\{\phi_i\}$ are orthogonal with respect to the Lebesgue measure, with

$$\begin{aligned}\int_{\mathbb{R}} \phi_m(x)\phi_n(x)\mathrm{d}p(x) &= \int_{\mathbb{R}} \exp(-2cx^2)H_m(\sqrt{2c}\,x)H_n(\sqrt{2c}\,x)\mathrm{d}x \\ &= 1/\sqrt{2c}\int_{\mathbb{R}} \exp(-x^2)H_m(x)H_n(x)\mathrm{d}x \\ &= \sqrt{\frac{\pi}{2c}}\,2^m m!\,\delta_{m,n}.\end{aligned}$$

Here we made a change of variables $\int_{\mathbb{R}} f(x)\mathrm{d}x = a\int_{\mathbb{R}} f(ax)\mathrm{d}x$, and used (7.374.1) from Gradshteyn and Ryzhik [GR80], namely

$$\int_{\mathbb{R}} \exp(-x^2)H_m(x)H_n(x)\mathrm{d}x = \sqrt{\pi}\,2^m m!\,\delta_{ij}.$$

Since the basis functions are not normalised, we have the following relation

$$s = \sum_{m=0}^{\infty} \zeta_m\phi_m \Rightarrow \langle s|p|\phi_n\rangle = r_n^2\zeta_n.$$

Hence can write $f(x) = \exp(-x^2) = \sum_{i=0}^{\infty}\gamma_i\phi_i(x)$ where

$$\begin{aligned}r_{2m}^2\gamma_{2m} &= \int_{\mathbb{R}} f(x)\phi_{2m}(x)\mathrm{d}p(x) \\ &= \int_{\mathbb{R}} \exp(-x^2)\exp(-(c-a)x^2)H_{2m}(\sqrt{2c}\,x)\exp(-2ax^2)\mathrm{d}x \\ &= 1/\sqrt{F}\int_{\mathbb{R}} \exp(-x^2)H_{2m}\left(\sqrt{2c/F}\,x\right)\mathrm{d}x \\ &= \sqrt{\frac{\pi}{F}}\,\frac{(2m)!}{m!}\,(2c/F-1)^m\,.\end{aligned}$$

where $F = c+a+1$. To evaluate this integral we used (7.373.2) from [GR80], namely

$$\int_{\mathbb{R}} \exp(-x^2) H_{2m}(xy)\mathrm{d}x = \sqrt{\pi}\,\frac{(2m)!}{m!}(y^2-1)^m,$$

from which we also have $\gamma_{2m+1} = 0, \forall m \in \mathbb{N}_0$. This leads to

$$\gamma_{2m} = \sqrt{\frac{\pi}{F}}\,\frac{(2m)!}{m!}\,(2c/F-1)^m \,/\left(\sqrt{\frac{\pi}{2c}}\,2^{2m}(2m)!\right) = \sqrt{\frac{2c}{F}}\,\frac{(2c/F-1)^m}{2^{2m}m!},$$

and so we have

$$\begin{aligned} R_\epsilon(x,y) &= k(x,y) - \epsilon f(x) f(y) \\ &= \sum_{i=0}^{\infty} \lambda_i \phi_i(x)\phi_i(y) - \epsilon \sum_{j=0}^{\infty} \gamma_j \phi_j(x) \sum_{k=0}^{\infty} \gamma_k \phi_k(y). \end{aligned}$$

Letting $g = \sum_{p=0}^{\infty} \xi_p \phi_p$ be arbitrary, to satisfy Mercer's condition we require that $\mathcal{I} \geq 0$ where

$$\begin{aligned} \mathcal{I} &\triangleq \int\int g(x) g(y) R_\epsilon(x,y)\mathrm{d}p(x)\mathrm{d}p(y) \\ &= \int\int \sum_{p=0}^{\infty} \xi_p \phi_p(x) \sum_{q=0}^{\infty} \xi_q \phi_q(y) \\ &\quad \cdot \left(\sum_{i=0}^{\infty} \lambda_i \phi_i(x)\phi_i(y) - \epsilon \sum_{j=0}^{\infty} \gamma_j \phi_j(x) \sum_{k=0}^{\infty} \gamma_k \phi_k(y) \right) \mathrm{d}p(x)\mathrm{d}p(y) \\ &= \sum_{p=0}^{\infty} \xi_p^2 r_p^4 \lambda_p - \epsilon \sum_{p=0}^{\infty} \xi_p r_p^2 \gamma_p \sum_{q=0}^{\infty} \xi_q r_q^2 \gamma_q. \end{aligned}$$

This can be written $\boldsymbol{\xi}^\top M \boldsymbol{\xi}$ where $M = N - \epsilon \boldsymbol{p}^\top \boldsymbol{p}$ is an infinite dimensional diagonal matrix minus a rank one matrix. Since $\lambda_k > 0$ for all k, we can use the fact that (*Observation 1, [Pan90]*)

$$\left(N - \epsilon \boldsymbol{p}\boldsymbol{p}^\top \succeq 0\right) \Leftrightarrow \left(\boldsymbol{p}^\top N^{-1} \boldsymbol{p} \leq 1/\epsilon\right),$$

and hence we equivalently require

$$1/\epsilon \geq \sum_{k=0}^{\infty} \gamma_k^2/\lambda_k = \frac{b}{\sqrt{2b-1}}.$$

The above equality can be shown using (in addition to various tedious algebraic manipulations) formula (0.241.3) from [GR80], namely

$$\sum_{m=1}^{\infty} p^m (2m)!/(m!)^2 = 1/\sqrt{1-4p}.$$

Note that the final sentence in Lemma 3.1.2 follows from theorem B.1.1 and the fact that for $b \leq 1/2$ there is no $\epsilon > 0$ satisfying the above inequality.

It also follows from theorem B.1.1 that $\|f\|^2_{\mathcal{H}(k)} = b/\sqrt{2b-1}$, where we denote by $\mathcal{H}(k)$ the r.k.h.s. with r.k. k. To compare this to Lemma 3.1.2, note that

$$f = \sqrt{\pi}\, g(0, \cdot, 1/2), \tag{B.1}$$

and

$$k(x, y) = \sqrt{\frac{\pi}{b}}\, g(x, y, 1/(2b)). \tag{B.2}$$

Using Lemma 3.1.2 and $\langle f, g\rangle_{\mathcal{H}(k)} = c\,\langle f, g\rangle_{\mathcal{H}(ck)}$, one can verify with a little algebra that

$$\left\langle \sqrt{\pi}\, g(0, \cdot, 1/2), \sqrt{\pi}\, g(0, \cdot, 1/2) \right\rangle_{\mathcal{H}(\sqrt{\frac{\pi}{b}}\, g(x,y,1/(2b)))} = \frac{b}{\sqrt{2b-1}}.$$

Finally, it is straightforward to generalise the result to the Gaussian kernel on $\mathbb{R}^d$, since this is merely the product of d univariate Gaussian kernels.

B.1.2 Sketch of the Fourier Transform Based Approach

Assuming $k : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, by letting

$$k(x, y) = \phi(x - y) = \int_{\mathbb{R}} \Phi(w) \exp(-\imath(x - y)w) \mathrm{d}w,$$

we can verify that

$$\left\langle \int_{\mathbb{R}} F(w) \exp(-\imath x w) \mathrm{d}w, \int_{\mathbb{R}} G(w) \exp(-\imath x w) \mathrm{d}w \right\rangle_{\mathcal{H}(k)} = \int_{\mathbb{R}} \frac{F(w)G(w)}{\Phi(w)} \mathrm{d}w,$$

because the reproducing property holds, *i.e.*

$$\begin{aligned}
&\langle f(x), k(x, y)\rangle_{\mathcal{H}(k)} \\
&\quad = \left\langle \int_{\mathbb{R}} F(w) \exp(-\imath x w) \mathrm{d}w, \int_{\mathbb{R}} \Phi(w) \exp(-\imath x w) \exp(\imath y w) \mathrm{d}w \right\rangle_{\mathcal{H}(k)} \\
&\quad = \int_{\mathbb{R}} \frac{F(w)\Phi(w) \exp(\imath y w)}{\Phi(w)} \mathrm{d}w \\
&\quad = f(y).
\end{aligned}$$

The result follows by substituting the known Fourier transforms for the Gaussian, and requiring a finite norm.

B.2 Additional Proofs for Chapter 5

Lemma 5.2.1

Proof. Using (5.1) and (5.2) we have

$$f(\boldsymbol{x}) = \left(f \circ \mathcal{T}^{-1}\right)(\mathcal{T}\boldsymbol{x}) = \left\langle f \circ \mathcal{T}^{-1}, k(\cdot, \mathcal{T}\boldsymbol{x})\right\rangle_{\mathcal{H}} = g_{\mathcal{T}}(\mathcal{H})\left\langle f, k(\mathcal{T}\cdot, \mathcal{T}\boldsymbol{x})\right\rangle_{\mathcal{H}},$$

which along with the uniqueness of r.k.'s implies (5.3) as required for the *only if*. Since $\{k(\cdot, \boldsymbol{x}) : \boldsymbol{x} \in \mathcal{X}\}$ spans $\mathcal{H}$, it is sufficient for the *if* to show that (5.2) and (5.3) imply (5.1) by showing they imply

$$g_{\mathcal{T}}(\mathcal{H})\left\langle k(\mathcal{T}\cdot, \boldsymbol{x}), f \circ \mathcal{T}\right\rangle_{\mathcal{H}} = \left\langle k(\cdot, \boldsymbol{x}), f\right\rangle_{\mathcal{H}}.$$

Now (5.2) implies that the r.h.s. equals $f(\boldsymbol{x})$, and (5.2) and (5.3) imply that the l.h.s. equals

$$\begin{aligned} g_{\mathcal{T}}(\mathcal{H})\left\langle k(\mathcal{T}\cdot, \boldsymbol{x}), f \circ \mathcal{T}\right\rangle_{\mathcal{H}} &= g_{\mathcal{T}}(\mathcal{H})\left\langle k(\cdot, \mathcal{T}^{-1}\boldsymbol{x})/g_{\mathcal{T}}(\mathcal{H}), f \circ \mathcal{T}\right\rangle_{\mathcal{H}} \\ &= (f \circ \mathcal{T})(\mathcal{T}^{-1}\boldsymbol{x}) \\ &= f(\boldsymbol{x}). \end{aligned}$$

□

Lemma 5.2.2

Note that there are standard results on homogeneous functions which are similar to Lemma 5.2.2. Homogeneous functions are those satisfying (B.3) but where $g(s)$ is fixed to s^{-p} *a priori*. Here we begin with general g and show that it must take the stated form.

Proof. We will show that

$$\phi(r) = g(s)\phi(rs) \tag{B.3}$$

implies that ϕ is affine in the log-log domain. To do this we let $\tilde{\phi}(\cdot) = \log(\phi(\exp(\cdot)))$, $\tilde{g}(\cdot) = \log(g(\exp(\cdot)))$, $\tilde{r} = \log(r)$ and $\tilde{s} = \log(s)$, so that if we assume that $r > 0$ (in addition to the assumption of the lemma that $s > 0$), then we have from (B.3) that $\tilde{\phi}(\tilde{r}) = \tilde{g}(\tilde{s}) + \tilde{\phi}(\tilde{r} + \tilde{s})$. Hence $\tilde{\phi}(\tilde{r}) - \tilde{\phi}(\tilde{q}) = \tilde{\phi}(\tilde{r} + \tilde{s}) - \tilde{\phi}(\tilde{q} + \tilde{s})$, and we see that $\tilde{\phi}$ is affine, because it satisfies

$$\tilde{\phi}(\tilde{r}) - \tilde{\phi}(\tilde{r} + \tilde{s}) = \tilde{\phi}(\tilde{q}) - \tilde{\phi}(\tilde{q} + \tilde{s}).$$

Hence we can write it as $\log(\phi(\exp(\log(r)))) = p\log(r) + \log(b)$, which implies that $\phi(r) = br^p, \forall r > 0$. But since any value of $\phi(0)$ satisfies (B.3), we have $\phi(r) = a\delta(r) + br^p, \forall r \geq 0$, and hence $g(s) = s^{-p}, \forall s > 0$. □

Lemma 2.2.11

Taken from Section 8.6 of [Wen04]:

Proof. Letting $[\Phi]_{i,j} = \phi(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $[P]_{j,k} = p_k(\boldsymbol{x}_j)$, we need to prove the solvability of

$$\begin{pmatrix} \Phi & P \\ P^\top & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{y} \\ \mathbf{0} \end{pmatrix}. \tag{B.4}$$

Let $V \triangleq P(\mathbb{R}^r) \subset \mathbb{R}^m$. Then the orthogonal complement $V^\perp$ of V is the null space of $P^\top$. The system (B.4) is solvable for every $\boldsymbol{y}$ if $\Phi V^\perp + V = \mathbb{R}^m$. But this is a direct sum because $\boldsymbol{x} \in \Phi V^\perp \cap V$ means that $\boldsymbol{x} = \Phi\boldsymbol{\alpha} = P\boldsymbol{\beta}$ with a certain $\boldsymbol{\alpha} \in V^\perp$ and a $\boldsymbol{\beta} \in \mathbb{R}^r$, which implies that $\boldsymbol{\alpha}^\top \Phi \boldsymbol{\alpha} = (P^\top \boldsymbol{\alpha})^\top \boldsymbol{\beta} = \mathbf{0}$ and hence $\boldsymbol{\alpha} = 0$ and $\Phi\boldsymbol{\alpha} = 0$. But the fact that the intersection of $\Phi V^\perp$ and V contains only the zero vector gives

$$\dim(\Phi V^\perp + V) = \dim(\Phi V^\perp) + \dim(V) = \dim(V^\perp) + \dim(V) = m,$$

because the mapping $V^\perp \to \Phi V^\perp$ is bijective. Hence the system (B.4) is solvable. □

Appendix C

Implementation Details

C.1 Thin-plate regularisation of the B_3-spline

The thin plate spline inner product of order $m = 2$, between two B_3-splines in three dimensions, separated by a distance r (measured between their maxima), and dilated such that they have radii of support s_1 and s_2 can be computed by the C language code of Figure C.1. The following function evaluates the B_3-spline (dilated such that it has radius of support *sup*) at a distance r from its maxima. Note that the implementation could be more computationally efficient.

```
void b3spline(double *rval, double *r, double *sup) {
  double rr = *r / (*sup), rr2 = rr*rr, rr3 = rr2 * rr;
  *rval = 1.0 + 6.0 * rr3 - 6.0 * rr2;
  if (rr > 0.5) {
    *rval -= 8.0 * rr3 - 12.0 * rr2 + 6.0 * rr - 1.0;
    if (rr > 1.0) {
       *rval += 2.0 * rr3 - 6.0 * rr2 + 6.0 * rr - 2;
    }
  }
}
```

C.2 Support Vector Machine with c.p.d. Kernel

The precise algorithm we use for optimising the support vector machine (s.v.m.) with a conditionally positive definite (c.p.d.) kernel function is analogous to that used for the p.d. case in [Cha07], which provides a more detailed discussion of the basic ideas used in both cases. Here we focus mainly on the aspects particular to the c.p.d. case. The main logical flow is given as Algorithm C.1.

To derive (5.15), we first use the standard second order update rule $\alpha = -H_\alpha^{-1} v_\alpha$ where

$$H_\alpha \triangleq \lambda\Phi + \Phi_{sx}^\top \Phi_{sx}$$

```
#define sign(x) ((x<0) ? -1 : 1)
double b3spline_reg(double r, double s1, double s2) {
  return
    (32*pow(3.14159265358979323846 ,8)*(6*s1*(-48*pow(r,5) +
    240*pow(r,3)*pow(s1,2) + 360*pow(r,2)*pow(s1,3) + 210*r*pow(s1,4)
    + 45*pow(s1,5) + 80*pow(r,4)*s2 - 240*pow(r,2)*pow(s1,2)*s2 -
    240*r*pow(s1,3)*s2 - 70*pow(s1,4)*s2 - 80*pow(r,2)*pow(s2,3) +
    40*pow(s1,2)*pow(s2,3) - 60*r*pow(s2,4) - 14*pow(s2,5) -
    96*pow(fabs(r - s1),5) + 6*pow(fabs(-2*r + s1),5) + pow(fabs(-2*r
    + s1 - 2*s2),5) - pow(fabs(2*r + s1 - 2*s2),5) + 16*pow(fabs(r +
    s1 - s2),5) + 4*pow(fabs(2*r + s1 - s2),5) - 2*pow(fabs(-2*r -
    2*s1 + s2),5) + 2*pow(fabs(2*r - 2*s1 + s2),5) - 16*pow(fabs(r -
    s1 + s2),5) - 4*pow(fabs(2*r - s1 + s2),5) - 4*pow(fabs(-2*r + s1
    + s2),5) - 16*pow(fabs(-r + s1 + s2),5) + 2*pow(fabs(-2*r + 2*s1 +
    s2),5) + pow(fabs(-2*r + s1 + 2*s2),5)) + 6*s2*(-48*pow(r,5) +
    80*pow(r,4)*s1 - 80*pow(r,2)*pow(s1,3) - 60*r*pow(s1,4) -
    14*pow(s1,5) + 240*pow(r,3)*pow(s2,2) - 240*pow(r,2)*s1*pow(s2,2)
    + 40*pow(s1,3)*pow(s2,2) + 360*pow(r,2)*pow(s2,3) -
    240*r*s1*pow(s2,3) + 210*r*pow(s2,4) - 70*s1*pow(s2,4) +
    45*pow(s2,5) - 2*pow(fabs(-2*r + s1 - 2*s2),5) + 2*pow(fabs(2*r +
    s1 - 2*s2),5) - 96*pow(fabs(r - s2),5) - 16*pow(fabs(r + s1 -
    s2),5) - 4*pow(fabs(2*r + s1 - s2),5) + 6*pow(fabs(-2*r + s2),5) +
    pow(fabs(-2*r - 2*s1 + s2),5) - pow(fabs(2*r - 2*s1 + s2),5) +
    16*pow(fabs(r - s1 + s2),5) + 4*pow(fabs(2*r - s1 + s2),5) -
    4*pow(fabs(-2*r + s1 + s2),5) - 16*pow(fabs(-r + s1 + s2),5) +
    pow(fabs(-2*r + 2*s1 + s2),5) + 2*pow(fabs(-2*r + s1 + 2*s2),5)) +
    4*(-576*pow(r,6) - 96*pow(r + s1,6) + 6*pow(2*r + s1,6) - 96*pow(r
    + s2,6) + 6*pow(2*r + s2,6) - 16*pow(r + s1 + s2,6) - 4*pow(2*r +
    s1 + s2,6) + pow(2*r + 2*s1 + s2,6) + pow(2*r + s1 + 2*s2,6) -
    96*pow(r - s1,6)*sign(r - s1) + 6*pow(-2*r + s1,6)*sign(r - s1/2.)
    - 96*pow(r - s2,6)*sign(r - s2) - 16*pow(-r + s1 + s2,6)*sign(r -
    s1 - s2) + pow(-2*r + s1 + 2*s2,6)*sign(r - s1/2. - s2) + pow(2*r
    + s1 - 2*s2,6)*sign(r + s1/2. - s2) - 16*pow(r + s1 - s2,6)*sign(r
    + s1 - s2) - 4*pow(2*r + s1 - s2,6)*sign(2*r + s1 - s2) +
    6*pow(-2*r + s2,6)*sign(r - s2/2.) + pow(-2*r + 2*s1 +
    s2,6)*sign(r - s1 - s2/2.) - 4*pow(-2*r + s1 + s2,6)*sign(r -
    s1/2. - s2/2.) + pow(-2*r - 2*s1 + s2,6)*sign(r + s1 - s2/2.) +
    pow(2*r - 2*s1 + s2,6)*sign(r - s1 + s2/2.) - 16*pow(r - s1 +
    s2,6)*sign(r - s1 + s2) - 4*pow(2*r - s1 + s2,6)*sign(2*r - s1 +
    s2) + pow(-2*r + s1 - 2*s2,6)*sign(r - s1/2. + s2)) +
    15*s1*s2*(-8*pow(r + s1 + s2,4) - 2*pow(2*r + s1 + s2,4) + pow(2*r
    + 2*s1 + s2,4) + pow(2*r + s1 + 2*s2,4) - 8*pow(-r + s1 +
    s2,4)*sign(r - s1 - s2) + pow(-2*r + s1 + 2*s2,4)*sign(r - s1/2. -
    s2) - pow(2*r + s1 - 2*s2,4)*sign(r + s1/2. - s2) + 8*pow(r + s1 -
    s2,4)*sign(r + s1 - s2) + 2*pow(2*r + s1 - s2,4)*sign(2*r + s1 -
    s2) + pow(-2*r + 2*s1 + s2,4)*sign(r - s1 - s2/2.) - 2*pow(-2*r +
    s1 + s2,4)*sign(r - s1/2. - s2/2.) - 16*pow(r + s1 -
    s2/2.,4)*sign(r + s1 - s2/2.) - pow(2*r - 2*s1 + s2,4)*sign(r - s1
    + s2/2.) + 8*pow(r - s1 + s2,4)*sign(r - s1 + s2) + 2*pow(2*r - s1
    + s2,4)*sign(2*r - s1 + s2) - 16*pow(r - s1/2. + s2,4)*sign(r -
    s1/2. + s2)))) /(5.*r*pow(s1,5)*pow(s2,5));
}
```

Figure C.1: An implementation in the C language of the inner product calculation described in section C.1 and Example 3.1.4 in subsection 2.2.3. Note that the implementation is rather innefficient, *e.g.*the *pow* function is very slow for integer exponents and ought not to be used.

is the Hessian and

$$v_\alpha \triangleq \Phi_{sx}^\top \boldsymbol{y}_s$$

is the gradient of the objective function. We then make a change of variables $\boldsymbol{\alpha} = P_\perp \boldsymbol{\gamma}$ where $P_\perp$ is the null space of P satisfying $PP_\perp = \mathbf{0}$. Hence the Hessian and gradient in this new parametrisation are $H = P_\perp^\top H_\alpha P_\perp$ and $v = P_\perp^\top v_\alpha$.

The following matrix factorisation techniques can be used to improve the constant factor of the run time for the $\boldsymbol{\beta} = 0$ case of the algorithm corresponding to subsection 5.4.2 and the parameter setting b_β = true in Algorithm C.1.

Rank One Cholesky Updates

What follows is a standard idea. Instead of solving (5.15) directly, it is more efficient and computationally stable to maintain a Cholesky decomposition of the Hessian matrix. That is, we maintain the factorisation $H = L^\top L$ where L is upper triangular, and solve (5.15) using two back substitutions. Importantly, it is possible to efficiently recompute L as the set s changes, in spite of the $P_\perp$ terms. This is because the new Hessian H_{new} after, say, the inclusion of one new margin violator with index w, is related to the old one by

$$H_{\text{new}} = H + \left([\Phi]_{w,:} P_\perp \right)^\top \left([\Phi]_{w,:} P_\perp \right),$$

which is a standard rank one Cholesky update and can be computed in $O(n^2)$ time where $H \in \mathbb{R}^{n,n}$ [GV96]. Note that the removal of a margin violator can be handled similarly using a so called Cholesky down date.

Cholesky Expansions

What now follows is somewhat less obvious, and appears to be new at least in the context of the s.v.m. as it cannot be readily applied to the normal p.d. s.v.m. algorithm of *e.g.* [Cha07]. Algorithm C.1 employs a recursive divide and conquer strategy in order to find the margin violators efficiently by working on an increasing subset of the data. It turns out that we can also efficiently recompute the Cholesky factorisation after adding elements to the working set. To do so is fairly straightforward — first we have to make sure that $P_\perp$ is constructed[1] in such a way that

$$[P]_{(1,2,\ldots,n)(1,2,\ldots n')} [P_\perp]_{(1,2,\ldots,n')(1,2,\ldots n)} = 0$$

for all $n \in 1, 2, \ldots m$ where $n' = r - n + 1$ and r is the number of rows of $P_\perp$. Such a $P_\perp$ can easily be derived from the row reduced echelon form of P [GV96][2]. The point is that if we now increase the working set size by

[1] Note that the null space of a matrix is not unique.

[2] For instance by calling `null(P,'r')` in Matlab 7.

adding elements in counting order (as is the case in Algorithm C.1), then the corresponding $P_\perp$ is modified by merely adding rows and columns to the previous one. As can be easily verified, the update to the Hessian H_α in $\boldsymbol{\alpha}$ also consists of merely adding rows and columns, and so the new Hessian in $\boldsymbol{\gamma}$ after bringing a single new element into the working set is given by

$$H' = {P'_\perp}^\top H'_{\boldsymbol{\alpha}} P'_\perp$$

where

$$P'_\perp = \begin{pmatrix} P_\perp & p_b \\ p_a & p_c \end{pmatrix}$$

and

$$H'_\alpha = \begin{pmatrix} H_\alpha & h \\ h^\top & h^* \end{pmatrix}.$$

As one may verify, we can now write $H'_{\boldsymbol{\alpha}} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ where

$$A = P_\perp^\top (H_\alpha P_\perp + h p_a) + p_c^\top (h^\top P_\perp + h^* p_a),$$

and the forms of B, C and D are easy to derive. Now recomputing the Cholesky factorisation after adding rows and columns to the matrix is a straightforward matter — in fact most algorithms for computing Cholesky factorisation proceed iteratively in this manner anyway. Hence the only remaining trick is to efficiently compute the Cholesky factorisation of A. But we can now take advantage of the way $P_\perp$ is constructed using the row reduced echelon form of P, which can be sketched as

$$P_\perp = \begin{pmatrix} \tilde{P}_\perp \\ I \end{pmatrix}.$$

In particular, for working set sizes greater than the number of columns of P — which is dependent only on the dimension of the space $\mathcal{P}$ with respect to which the kernel is c.p.d., and in particular independent of the number of training points — we have $p_a = \mathbf{0}$. Hence A reduces to the simple form

$$\begin{aligned} A &= P_\perp^\top H_\alpha P_\perp + p_c^\top h^\top P \\ &= H + \frac{1}{2}\left((p_c + P^\top h)^\top (p_c + P^\top h) - p_c^\top p_c - (P^\top h)^\top (P^\top h)\right), \end{aligned}$$

from which it is clear that, given the Cholesky factorisation of H, that of A can be computed using one update and two down dates. Remarkably, this means that the Cholesky factorisation of H' after increasing the working set size by one can be computed in quadratic time with respect to the current working set size. Indeed, as a proof of concept we have developed a simple implementation based on the above ideas.

Algorithm C.1:
$(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \text{CPDPrimalSVM}\,(\Phi, Q, R, y, \lambda, b_\beta)$
input:
$[\Phi]_{j,k} = \phi(\boldsymbol{x}_j, \boldsymbol{x}_k)$ where ϕ is c.p.d. w.r.t. $\{p_1, \ldots, p_r\}$
Q, R such that $P^\top = QR$, $Q^\top Q = I$ & $[P]_{k,j} = p_k(\boldsymbol{x}_j)$
class labels $\boldsymbol{y} \in \{1, -1\}^n$
regularisation parameter λ
$b_\beta \in \{\text{true}, \text{false}\}$ = "enforce $\boldsymbol{\beta} = \mathbf{0}$"
output:
$\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ such that $f_{\text{svm}} = \sum_{j=1}^n \alpha_j \phi(\cdot, \boldsymbol{s}_j) + \sum_{k=1}^r \beta_k p_k(\cdot)$

$n \leftarrow \text{length}(\boldsymbol{y})$
if $(n > 500) \ \wedge \ \text{not}(b_\beta)$ **then**
  $n_2 \leftarrow n/2$
  $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \leftarrow \text{CPDPrimalSVM}([\Phi]_{1:n_2,1:n_2}, [Q]_{1:n_2,:}, R, [\boldsymbol{y}]_{1:n_2}, \lambda, b_\beta)$
  $s \leftarrow$ indices of non-zero components of $\boldsymbol{\alpha}$
else
  $s \leftarrow \{1, \ldots, n\}$
end if
$o_{\text{old}} \leftarrow \infty$
$\boldsymbol{\alpha}_{\text{old}} \leftarrow \mathbf{0}$
$\overline{\boldsymbol{\beta}}_{\text{old}} \leftarrow \mathbf{0}$
repeat
  if b_β **then**
    $\overline{\boldsymbol{\beta}} \leftarrow \mathbf{0}$
    $\boldsymbol{\alpha} \leftarrow$ r.h.s. of (5.15)
  else
    $$\begin{pmatrix} [\boldsymbol{\alpha}]_s \\ \overline{\boldsymbol{\beta}} \end{pmatrix} = \begin{pmatrix} \lambda I + [\Phi]_{s,s} & [Q]_{s,:} \\ [Q]_{s,:}^\top & \mathbf{0} \end{pmatrix}^{-1} \begin{pmatrix} [\boldsymbol{y}]_s \\ \mathbf{0} \end{pmatrix}$$
    $[\boldsymbol{\alpha}]_{\{1,\ldots,n\}\setminus s} \leftarrow 0$
  end if
  $s_{\text{old}} \leftarrow s$
  $t = 0$
  repeat
    $\boldsymbol{\alpha}_{\text{new}} \leftarrow 2^{-t}\boldsymbol{\alpha} + (1 - 2^{-t})\boldsymbol{\alpha}_{\text{old}}$
    $\overline{\boldsymbol{\beta}}_{\text{new}} \leftarrow 2^{-t}\overline{\boldsymbol{\beta}} + (1 - 2^{-t})\overline{\boldsymbol{\beta}}_{\text{old}}$
    $s \leftarrow \left\{i : y_i \left[\Phi\boldsymbol{\alpha}_{\text{new}} + Q\overline{\boldsymbol{\beta}}_{\text{new}}\right]_i < 1\right\}$
    $o_{\text{new}} \leftarrow \lambda\boldsymbol{\alpha}_{\text{new}}^\top\Phi\boldsymbol{\alpha}_{\text{new}} + \left\|[\Phi]_{s,:}\,\boldsymbol{\alpha}_{\text{new}} + [Q]_{s,:}\,\overline{\boldsymbol{\beta}}_{\text{new}} - [\boldsymbol{y}]_s\right\|^2_{\mathbb{R}^{|s|}}$
    $t \leftarrow t + 1$
  until $o_{\text{new}} \leq o_{\text{old}}$
  $\boldsymbol{\alpha}_{\text{old}} \leftarrow \boldsymbol{\alpha}_{\text{new}}$
  $\overline{\boldsymbol{\beta}}_{\text{old}} \leftarrow \overline{\boldsymbol{\beta}}_{\text{new}}$
  $o_{\text{old}} \leftarrow o_{\text{new}}$
until $s = s_{\text{old}}$
$\boldsymbol{\beta} \leftarrow \left(R^\top R\right)^{-1} R^\top \overline{\boldsymbol{\beta}}_{\text{new}}$

Bibliography

[ABCO+01] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *IEEE Visualization 2001*, pages 21–28. IEEE Computer Society, October 2001. 53, 73

[Aro50] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68, 1950. 34, 107

[BBX95] Chandrajit L. Bajaj, Fausto Bernardini, and Guoliang Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. *Computer Graphics*, 29(Annual Conference Series):109–118, 1995. 53

[BCR84] C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag, New York, 1984. 94

[BG97] Rick Beatson and Leslie Greengard. A short course on fast multipole methods. In *Wavelets, Multilevel Methods and Elliptic PDEs*, pages 1–37, 1997. 43, 47

[BJ01] Francis R. Bach and Michael I. Jordan. Kernel independent component analysis. Technical Report UCB/CSD-01-1166, EECS Department, University of California, Berkeley, Nov 2001. 107

[BPT06] R. K. Beatson, M. J. D. Powell, and A. M. Tan. Fast evaluation of polyharmonic splines in 3-dimensions. Technical Report NA2006/03, Numerical Analysis Group, University of Cambridge, 2006. 47, 48

[BTB05] S. Boughorbel, J.-P. Tarel, and N. Boujemaa. Conditionally positive definite kernels for svm based image recognition. In *Proc. of IEEE ICME'05*, Amsterdam, 2005. 92

[CBC+01] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *ACM*

SIGGRAPH 2001, pages 67–76. ACM Press, 2001. 43, 47, 55, 56, 58, 66, 72

[Cha07] Olivier Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007. 14, 59, 92, 113, 115

[CSZ06] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, 2006. 100

[DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, New York, 2000. 2nd Edition. 18

[Duc77] J. Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. *Constructive Theory of Functions of Several Variables*, pages 85–100, 1977. 23, 90

[FS03] F. Fleuret and H. Sahbi. Scale-invariance of support vector machines based on the triangular kernel. *Proc. of ICCV SCTV Workshop*, 2003. 87

[FWML06] Nando De Freitas, Yang Wang, Maryam Mahdaviani, and Dustin Lang. Fast krylov methods for n-body learning. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 251–258. MIT Press, Cambridge, MA, 2006. 49

[GJP93] Federico Girosi, Michael Jones, and Tomaso Poggio. Priors stabilizers and basis functions: From regularization to radial, tensor and additive splines. Technical Report AI 1430, Massachusetts Institute of Technology, 1993. 20

[GR80] Izrail S. Gradshtein and Iosif Ryzhik. *Table of Integrals, Series, and Products*. Academic Press, New York, corrected and enlarged edition, 2nd printing edition, 1965, 1980. 108, 109

[GR97] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comp. Phys.*, pages 280–292, 1997. 56

[GV96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore MD, 2nd edition, 1996. 32, 43, 77, 93, 115

[HB04] Matthias Hein and Olivier Bousquet. Kernels, associated structures and generalizations. Technical report, Max Planck Institute for Biological Cybernetics, Department of Empirical Inference, Tbingen, Germany, July 2004. 11

[HCL03] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, National Taiwan University, 2003. 22, 99

[Hör60] L. Hörmander. Estimates for translation invariant operators in l^p spaces. *Acta Mathematica*, 104:93–140, 1960. 21

[KD05] Sathiya Keerthi and Dennis DeCoste. A modified finite newton method for fast solution of large scale linear svms. *J. Mach. Learn. Res*, 6:341–361, 2005. 59

[KW71] G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971. 13

[LGM06] Dongryeol Lee, Alexander Gray, and Andrew Moore. Dual-tree fast gauss transforms. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 747–754. MIT Press, Cambridge, MA, 2006. 47

[LMGY04] Ting Liu, Andrew Moore, Alexander Gray, and Ke Yang. An investigation of practical approximate nearest neighbor algorithms. In *Advances in Neural Information Processing Systems 14*, 12 2004. 46

[Mac98] D. J. C. MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, NATO ASI Series, pages 133–166. Kluwer Academic Press, 1998. 17

[MBC99] Cameron Mouat, Rick Beatson, and Jon Cherrie. Fast fitting of radial basis functions: Methods based on preconditioned gmres iteration. In *Surface Approximation and Visualisation*, 1999. 48

[MPL00] C. Merkwirth, U. Parlitz, and W. Lauterborn. Fast nearest neighbor searching for nonlinear signal processing. *Phys. Rev. E*, 62(2):2089–2097, 2000. 32

[MRW+03] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A.J. Smola, and K.-R. Müller. Constructing descriptive and discriminative nonlinear features: Rayleigh coefficients in feature spaces. *IEEE PAMI*, 25(5):623–628, 2003. 100

[MYC+01] Bryan S. Morse, Terry S. Yoo, David T. Chen, Penny Rheingans, and K. R. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *SMI '01: Proc. Intl. Conf. on Shape Modeling & Applications*, Washington, 2001. IEEE Computer Society. 56

[OBA+03] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics*, 22(3):463–470, July 2003. 56

[OBS03] Y. Ohtake, A. Belyaev, and Hans-Peter Seidel. A multi-scale approach to 3d scattered data interpolation with compactly supported basis functions. In *Proc. Intl. Conf. Shape Modeling*, Washington, 2003. IEEE Computer Society. 31, 56

[Pan90] C. T. Pan. A modification to the linpack downdating algorithm. *BIT Numerical Mathematics*, 30(4):707–722, 1990. 109

[PGK02] Mark Pauly, Markus Gross, and Leif P. Kobbelt. Efficient simplification of point-sampled surfaces. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 163–170, Washington, 2002. IEEE Computer Society. 41, 42

[Pop68] K.R. Popper. *The Logic of Scientific Discovery*. Hutchinson, 1968. 6

[QCR05] J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:1935–1959, 12 2005. 40

[RG01] C. E. Rasmussen and Z. Ghahramani. Occam's razor. In Thomas G. Dietterich Todd Leen and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 294–300. MIT Press, 2001. 6

[Roa70] G. F. Roach. *Green's Functions*. Cambridge University Press, Cambridge, UK, 1970. 19, 20

[RS80] M. Reed and B. Simon. *Methods of modern mathematical physics. Vol. 1: Functional Analysis*. Academic Press, Address San Diego, 1980. 14

[RW06] C. E. Rasmussen and C. K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Massachusetts, 01 2006. 17, 108

[Set98] J. Sethian. Level set methods and fast marching methods: Evolving interfaces in computational geometry, 1998. 53

[SGS05] Bernhard Schölkopf, Joachim Giesen, and Simon Spalinger. Kernel methods for implicit surface modeling. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005. 73

[SHS01] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *Proc. of the 14th Annual Conf. on Computational Learning Theory*, pages 416–426, London, UK, 2001. Springer-Verlag. 13

[Sol00] P. Sollich. Probabilistic methods for support vector machines. In S. A. Solla, T. K. Leen, and K.R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 349–355. MIT Press, 2000. 18

[SOS04] Chen Shen, James F. O'Brien, and Jonathan R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *ACM SIGGRAPH 2004*. ACM Press, August 2004. 56

[SS02] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, 2002. 10, 11, 14, 18, 20, 21, 24

[SS03] Valeria Simoncini and Daniel B. Szyld. Theory of inexact krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.*, 25(2):454–477, 2003. 49

[SSB05] Florian Steinke, Bernhard Schölkopf, and Volker Blanz. Support vector machines for 3d shape processing. *Computer Graphics Forum (Proc. EUROGRAPHICS)*, 24(3), 2005. 58, 71

[SSM98] A.J. Smola, B. Schölkopf, and K.-R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11:637–649, 1998. 24, 92

[SV96] Gerard L. G. Sleijpen and Henk A. Van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 17(2):401–425, 1996. 78

[TA77] Andrey Tikhonov and Vasiliy Arsenin. *Solutions of Ill-posed Problems*. V.H. Winston and Sons, Washington, 1977. 6

[TO99] Greg Turk and James F. O'Brien. Shape transformation using variational implicit functions. In *Proceedings of ACM SIGGRAPH 1999*, pages 335–342, August 1999. 56

[Vap95] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995. 18, 73

[Wah90] G. Wahba. *Spline Models for Observational Data*. Series in Applied Math., Vol. 59, SIAM, Philadelphia, 1990. 12, 13, 20, 23, 24, 89, 93, 96

[WB98] Ross T. Whitaker and David E. Breen. Level-set models for the deformation of solid objects. In Dietmar Saupe Jules Bloomenthal, editor, *Implicit Surfaces 98 Proceedings*, Eurographics/ACm Workshop, pages 19–35, 1998. 53

[WCS05] Christian Walder, Olivier Chapelle, and Bernhard Schölkopf. Implicit surface modelling as an eigenvalue problem. In *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, 2005. 57, 70

[WCS+06] J. Weston, R. Collobert, F. Sinz, L. Bottou, and V. Vapnik. Inference with the universum. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006. 60

[Wen04] Holger Wendland. *Scattered Data Approximation*. Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004. 11, 12, 24, 88, 89, 91, 96, 98, 112

[WLK03] Christian Walder, Brian Lovell, and Peter Kootsookos. Algebraic curve fitting support vector machines. In *DICTA 2003*, pages 693–702, Sydney, 2003. IEEE Computer Society. 57, 58, 59, 60

[WSC06] Christian Walder, Bernhard Schölkopf, and Olivier Chapelle. Implicit surface modelling with a globally regularised basis of compact support. *Proceedings of Eurographics*, 25(3):635–644, 2006. 13, 57, 66, 70, 71

[YG89] A.L. Yuille and N.M. Grzywacz. A mathematical analysis of the motion coherence theory. *International Journal of Computer Vision*, 3(2):155–175, June 1989. 22