Generalised Kernel Representations with Applications to Data Efficient Machine Learning

ANTHONY TOMPKINS

Supervisor: Professor Fabio Ramos

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

School of Computer Science
Faculty of Engineering
The University of Sydney
Australia

13 October 2023

Certificate of Original Authorship

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Name: *Anthony Tompkins*

Signature:

Date: 26th of September 2022

Abstract

The universe of mathematical modelling from observational data is a vast space. It consists a cacophony of differing paths, with doors to worlds with seemingly diametrically opposed perspectives that all attempt to conjure a crystal ball of both intuitive understanding and predictive capability. Among these many worlds is an approach that is broadly called *kernel methods*, which, while complex in detail, when viewed from afar ultimately reduces to a rather simple question: how close is something to something else? What does it mean to be close? Specifically, how can we *quantify* closeness in some reasonable and principled way?

This thesis presents four approaches that address generalised kernel learning. Firstly, we introduce a probabilistic framework that allows joint learning of model and kernel parameters in order to capture nonstationary spatial phenomena. Secondly, we introduce a theoretical framework based on optimal transport that enables online kernel parameter transfer. Such parameter transfer involves the ability to re-use previously learned parameters, without re-optimisation, on newly observed data. This extends the first contribution which was unable operate in real-time due to the necessity of re-optimising parameters to new observations. Thirdly, we introduce a learnable Fourier based kernel embeddings that exploits generalised quantile representations for stationary kernels. Finally, a method for input warped Fourier kernel embeddings is proposed that allows nonstationary data embeddings using simple stationary kernels.

By introducing theoretically cohesive and algorithmically intuitive methods this thesis opens new doors to removing traditional assumptions that have hindered adoption of the kernel perspective. We hope that the ideas presented will demonstrate a curious and inspiring view to the potential of learnable kernel embeddings.

Acknowledgements

I would like to thank everyone with whom I have journeyed with over this PhD adventure.

Thank you to my supervisor Fabio for your guidance and the freedom to let me travel an untrodden research path. This thesis would not have been possible without your support throughout.

Thank you to all my wonderful labmates: Ransalu, Philippe, Rafael, Rafael, Kelvin, Lionel, Charika, Sheila, Thushan, Tom, Gilad, Harrison, Will, Tin, Nick, Louis, and Lucas! I have enjoyed so much the countless coffee breaks and impromptu discussions on research and life. I would especially like to thank Ransalu for your continued academic guidance and collaborative fellowship, Philippe for the cheerful banter, Rafael for your kindness. I will be forever grateful for the time spent in shared research rabbit holes and paper writing sleepless overnighters. Thank you Lionel for your technical talents, Kelvin for your excitement and open ears!

Thank you Marlon and Gio your help! And thank you to all the friendly souls I have bumped into at conferences for a chat.

My warmest gratitude to my love Mahnaz who has been with me through the upbeat ups and challenging downs - you have been there and supported me, patiently, the entire way. Last but not least I would like to thank my parents, and friends for their support throughout.

Anthony Tompkins
The University of Sydney
2022

List of Publications

The contributions of this thesis have been published in the following conferences and constitute the main chapters of this thesis. The author's attribution includes, but is not limited to, the motivation, conceptualization, formalization, derivation, theorization, experimentation, and communication of the papers.

Chapter 3 - Automorphing Kernels

R. Senanayake*, **A. Tompkins***, F. Ramos (2018). 'Automorphing Kernels for Nonstationarity in Mapping Unstructured Environments'. *Conference on Robot Learning (CoRL)*

Chapter 4 - Parameter Optimal Transport for Online Kernel Domain Adaptation

A. Tompkins*, R. Senanayake*, F. Ramos (2020). 'Online Domain Adaptation for Occupancy Mapping'. *Robotics: Science and Systems (RSS)*

Chapter 5 - Quantile Representations for Approximate Kernel Learning

A. Tompkins*, R. Senanayake*, P. Morere*, F. Ramos (2019). 'Black Box Quantiles for Kernel Learning'. *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Chapter 6 - Warped Input Measures for Nonstationary Kernel Learning

A. Tompkins*, R. Oliveira*, F. Ramos (2020). 'Sparse Spectrum Warped Input Measures for Nonstationary Kernel Learning'. *Advances in Neural Information Processing Systems (NeurIPS)*

*Equal contribution

Attribution Statement

In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author. I am incredibly grateful to all my wonderful collaborators in the papers included in this thesis. My supervisor Fabio was providing a broad support for all the previously published papers. Paper writing in all papers was shared amongst all authors.

In *Automorphing Kernels for Nonstationarity in Mapping Unstructured Environments*, Ransalu introduced me to fundamental problems in probabilistic mapping and together we worked on the solution and experimental design. In *Online Domain Adaptation for Occupancy Mapping*, I conceptualized the parameter transport idea and Ransalu helped with formalization, data curation, experimentation and software development. In *Black Box Quantiles for Kernel Learning* I conceptualized the generalised Fourier space based kernel while Ransalu, Philippe and myself jointly developed the software and experimental framework. For *Sparse Spectrum Warped Input Measures for Nonstationary Kernel Learning*, I conceptualized the idea of the nested warping kernel alongside Rafael who helped with formalization and software.

Name: *Anthony Tompkins*
Signature:
Date: 16th of April 2023

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Name: *Professor Fabio Ramos*
Signature:
Date:

Contents

Nomenclature

Abbreviations

ABHM	Automorphing Bayesian Hilbert Map
ACC	Accuracy
ARD	Automatic Relevance Determination
AUC	Area Under the Curve
BBQ	Black Box Quantile
BHM	Bayesian Hilbert Map
BLR	Bayesian Linear Regression
DA	Domain Adaptation
DGP	Deep Gaussian Process
DKL	Deep Kernel Learning
DOGM	Dynamic Occupancy Grid Maps
DSDGP	Doubly Stochastic Deep Gaussian Process
ELBO	Evidence Lower Bound
GAM	Generalised Additive Model
GAN	Generative Adversarial Network
GPOM	Gaussian Process Occupancy Map
GP	Gaussian Process
HM	Hilbert Map
ICP	Iterative Closest Point
KL	Kullback-Leibler Divergence
MAP	Maximum a Posteriori
MCMC	Markov Chain Monte Carlo
MC	Monte Carlo
MLE	Maximum Likelihood Estimate

MNLP	Mean Negative Log Probability
NLL	Negative Log Likelihood
NLML	Negative Log Marginal Likelihood
OGM	Occupancy Grid Map
OT	Optimal Transport
PCHIP	Piecewise Cubic Hermite Interpolation
POT	Parameter Optimal Transport
QMC	Quasi-Monte Carlo
RBF	Radial Basis Function
RePOT	Refined Parameter Optimal Transport
RFF	Random Fourier Feature
RKHS	Reproducing Kernel Hilbert Space
RMSE	Root Mean Square Error
SE	Squared Exponential
SGD	Stochastic Gradient Descent
SGPR	Sparse Gaussian Process Regression
SLAM	Simultaneous Localisation and Mapping
SM	Spectral Mixture
SSGP	Sparse Spectrum Gaussian Process
SSWIM	Sparse Spectrum Warped Input Measures
SVGP	Sparse Variational Gaussian Process
SVM	Support Vector Machine
VI	Variational Inference
VSDGPOM	Variational Sparse Dynamical Gaussian Process Occupancy Maps
WGP	Warped Gaussian Process

Symbols

$[\cdot, \cdot]$	vector or matrix concatenation
$[]$	vector or matrix
$\lvert\cdot\rvert$	absolute value
$\lvert\cdot\rvert$	cardinality of a set

argmax	argument that maximises
argmin	argument that minimises
$\circ$	function composition
$\operatorname{Cov}[\cdot, \cdot]$	covariance
ϵ	noise
$\mathbb{E}[\cdot]$	expected value
$\forall$	for all
$\gg$	much greater than
$\boldsymbol{I}$	identity
$\langle \cdot, \cdot \rangle$	dot product
$\ll$	much less than
$\mathbb{C}$	complex numbers
$\mathbb{KL}$	Kullback-Leibler divergence
$\mathbb{P}$	probability measure
$\mathbb{R}$	real numbers
$\mathcal{D}$	dataset
$\mathcal{GP}$	Gaussian process
$\mathcal{H}$	Hilbert space
$\mathcal{N}$	normal distribution
$\mathcal{O}$	computational complexity
$\max$	maximum
$\min$	minimum
$\|\cdot\|$	norm
$\odot$	Hadamard (element-wise) product
$\boldsymbol{\Phi}$	feature matrix
$\boldsymbol{\phi}$	feature vector
θ	parameters
$\mathbb{V}[\cdot]$	variance
$\mathbf{w}$	weight vector
$\boldsymbol{X}$	matrix

$\boldsymbol{x}$	vector
$\boldsymbol{X}^\top$	matrix transpose
$\boldsymbol{x}^\top$	vector transpose
$\{\}$	set
$f(\cdot)$	generic function
i, j	indexing
k	kernel function
M	number of features
N	number of data points
$p(\cdot)$	probability density function
p	probability
Q	quantile function
x	scalar

List of Figures

List of Tables

CHAPTER 1

Introduction

The overarching theme of this thesis is to showcase the possibilities of moving the theory and application of modelling with kernels into a more general, data-centric archetype. We aim to show that improving the capabilities of kernel methods is both algorithmically intuitive and desirable from a practitioner's perspective.

1.1 Motivation

Kernel methods are a long studied topic within the space of machine learning. Despite enjoying earlier successes within various learning algorithms, such as the well known support-vector machine (SVM), the utility of kernel methods has been marred by the effectiveness and study of recently ubiquitous *neural networks*. Even though equivalences between various statistical modelling approaches have been shown (such as the convergence of certain infinitely wide or deep neural networks to equivalent kernels), comparatively little effort has been focused on asking: why don't kernel methods scale as effectively; or how could you can scale them? Indeed it seems more reasonable to see *all* learning methods as similar faces of the same multi-faceted coin. Using this universal perspective as a starting point we take this as inspiration for a fresh take on the kernel learning perspective.

Kernels are, loosely speaking, functions that represent similarity between objects. More precisely, they encode a mathematical quantification of how one object relates, according to some interpretation, to another object. The critical notion here is that of *interpretation*. In other words, while there are a vast number of kernel functions that have been discovered, each kernel, precisely through their mathematical construction, encodes its own unique meaning

of similarity. To date the large majority of kernel functions in the literature are well defined, convenient, analytic functions with well studied properties. We wish to show that these *alone* are insufficient when one attempts to embed and model real-world data, which is usually full of diversity.

We are ultimately guided by a desire to relax the assumptions that are held by the conventional library of well known and well defined kernels. We wish to explore relaxations of not just the kernels themselves but the kernel along with its corresponding modelling algorithm (for a kernel is just a tool that needs a host) in a holistic manner.

In summary this thesis addresses the following key problems:

(1) How to reduce the reliance on the restrictive assumptions of canonical kernels.
(2) What are effective ways of representing and modelling with newly proposed generalised kernels.
(3) How to efficiently estimate, given data, the parameters of a new family of generalised kernels.
(4) How to show that practice follows theory in a meaningful context within the domain of real world data.

1.2 Challenges

In order to accomplish our goal of exploring the relatively unwalked path of free-form kernel learning, there are various essential theoretical and applied stepping stones that must be crossed. In particular, this thesis will focus on the following challenges:

- **Breaking Classical Kernel Assumptions:** What different mathematical parameterisations allow us to escape from the safe and well defined kernels that are used?
- **Kernel Learning Models:** What mathematical models allow us to seamlessly integrate a data-driven kernel design?
- **Hyperparameter Learning:** How do we actually learn to adjust our newly minted kernels and learning models given the data?

- **Interpretability:** Can we describe our newfound flexible kernel representations in an intuitive way that doesn't diminish our understanding and can even showcase *why* it's so important for us to discover flexible data representations?
- **Performance:** Can we move beyond the often slow optimisation procedures that are typically necessary to learn nuanced kernel and model representations?

1.3 Contributions

The following outlines our contributions:

A probabilistic framework for joint learning of model and kernel parameters in a unified model that can capture nonstationarity in observations in an automatic manner. We propose a theoretical framework that works well in practice to learn all parameters of a classification model that uses explicitly placed kernels in the data domain. We showcase the method on the contemporary problem of probabilistic occupancy mapping in robotics. Experimental results validate the method with significant improvements in representational performance for robotic mapping in highly unstructured and nonstationary environments.

A theoretical framework for online parameter transfer using the theory of optimal transport. We introduce a way to adapt previously trained kernel models, on demand and in real-time, using a parallelised decomposition of the learning problem. We demonstrate the process on the domain adaptation paradigms of *intra* and *inter*-domain transfer in a robotic mapping problem.

A new method that learns quantile representations of generalised stationary kernels in the Fourier domain. We introduce a relaxation of an approximate Fourier representation of a kernel that adapts to the observed data. We additionally show that the quantile representation naturally allows for improved approximation efficiency through quasi-Monte Carlo integral sampling. We validate the method on various datasets and problems that exhibit complex phenomena like pseudo-periodicity that cannot be modelled with conventional stationary kernels within limited data.

An approach to learning nonstationary approximate Gaussian process models via input warping. We introduce an intuitive measure-value input-dependent warping function alongside its multi-level warping extension by propagating moments. Comprehensive experiments shows competitive performance with many deep learning and alternative approximate Gaussian process methods.

1.4 Thesis Overview

This thesis begins its journey by introducing a Bayesian approximation to a general learning problem that uses explicit *placements of kernels* within the observed data space. While invariant to the actual problem, we demonstrate our proposed method on an important problem domain within the field of robotics: probabilistic mapping of environments. We demonstrate how relaxing assumptions on the form of the kernels that are placed within the environment, and taking advantage of probabilistic programming and variational inference, that it is possible to let the data define the (hyper-)parameters of the kernels and model in an end-to-end fashion. We follow this thread of kernel learning into an online estimation to address computational limitations of the previous end-to-end learning paradigm. Taking a short detour through the mathematical theory of optimal transport, we will discover that it is possible to frame the online learning situation under the light of domain adaptation: treating the learned distributions that represent the kernel parameters as points on some manifold.

Continuing the desire to replace canonical kernels with their relaxed counterparts, our story will take a turn to *spectral representations* of kernels and accommodating model representations. Specifically, we will consider kernels in the Fourier domain allowing us to view them in a different light that allows us to expand our arsenal of mathematical intervention by viewing the kernel as itself a distribution. We show that it is both possible and even desirable to formulate the kernel's distributional representation as a parameterised quantile function. By providing an end-to-end learning pipeline we can let the observed data inform the shape that the kernel should take rather than assuming that canonical kernels are *a priori* the right choice (which is almost never the case). The thesis concludes its journey by relaxing further

assumptions that our data generating processes are stationary, and we provide a simple but effective representation for embedding our data into a kernel space for effective inference.

1.5 Thesis Structure

Following this introductory chapter, the thesis presents essential background knowledge in Chapter 2 underlying the critical components of the main contributions. We will bring the reader up to speed with a summary of Bayesian learning and a canonical review of both standard and Bayesian linear regression. The background will then introduce the fundamental concept of a *kernel* (as a way of evaluating *closeness between data*) and its critical role in modelling. After this we broadly introduce some theory and intuition regarding *optimal transport*. We close the background with a brief summary of robotic mapping and how kernel methods have attempted to tackle this challenging real world problem.

Our first major contribution is presented in Chapter 3. This is our first foray into what one could term learning a *locally explicit* nonstationary kernel. We first describe related work and preliminaries on probabilistic mapping in robotics, which is our sandbox to demonstrate the technique. Then we introduce our fundamental idea of relaxing the conventional (stationary and local) kernel assumptions to produce a model that adapts to the observed data. This chapter also introduces the fundamentals for the following chapter. We report a variety of experiments comparing nonstationary probabilistic mapping with conventional, non-adaptive probabilistic mapping methods.

Chapter 4 extends the previous work by addressing a serious performance constraint in the flexible learning process. By leveraging the theory and application of *optimal transport*, we transform the time consuming end-to-end learning process into a real-time problem under the perspective of domain adaptation. We explain how the problem can be formulated as an efficient online optimisation problem under the paradigm of optimal transport theory. We then report results on a number of experiments on the domain of probabilistic mapping shown previously.

In Chapter 5 we shift gears from the previous two chapters on *locally explicit* kernel representation and learning to *globally implicit* kernel approximations. In other words, we cease from placing explicit kernel functions around the data domain and adopt a spectral representation in the Fourier domain. We proceed to introduce our fundamental contribution that relaxes the analytic representation of familiar, conventional kernel functions. This discourse will involve a Fourier theory perspective on kernel functions and their connection to probability theory. Thus we make the connection with quantiles (a.k.a. the inverse cumulative distribution function) and how we can represent arbitrarily flexible kernels *implicitly* without ever knowing their analytic form. We demonstrate how we can learn this flexible kernel given data and present some experiments that validate the contributions.

Chapter 6 is our final exploration of the *globally implicit* perspective on kernel learning. It is a natural successor to Chapter 5 by further relaxing some limiting assumptions. While the quantile formulation of the kernel presented previously is arbitrarily flexible, it is only flexible in a *stationary* sense. That is to say, it does not allow one to capture data-dependent kernel variation with respect to our observations. Thus we introduce a way to capture nonstationarity within the kernel approximation paradigm. We provide extensive experimental validation of the proposed methodology on toy and real world datasets.

Finally, Chapter 7 concludes the thesis, with a brief retrospective on the contributions and possibilities for future work.

CHAPTER 2

Background

This chapter introduces the foundational methods to our proposed modelling and kernel approximation methods. We first introduce Bayesian learning in the context of linear regression and the Bayesian settings correspond to an equivalent kernel. Next we describe kernel methods, in particular the Gaussian process (GP) and related methods explaining kernel approximations and optimisation strategies. We will then provide an introductory discourse on combining *a priori* known kernels with kernel compositions. Next follows an introduction of some basic optimal transport (OT) theory which will be necessary for understanding the domain adaptation concepts introduced in more detail later. Finally we introduce the application domain of probabilistic robotic mapping for unstructured environments. This will be our applied domain of focus for demonstrating the effectiveness of some of the proposed models.

2.1 Learning Model Representations

As the systems we create and use increase in complexity, a stronger need arises to help us make sense of both the increasing complexity of data and the increasing complexity of the systems we use to digest such information. Indeed it could be argued that the very reason we create and use such systems is to drive more *informed* decision making that can account for associated risks our models bring to the real world. Robotics, medical treatment, exploration, financial and political systems that govern our lives, space exploration, material and drug discovery are but a few examples of the countless domains that inevitably require us to create and use methods that can learn and improve themselves. More pressingly, these systems will

need to operate with larger deluges of information that often exceed the capacity of human capability.

The general domain of *probabilistic* modelling affords us a way of thinking that allows us to both make predictions that extrapolate *and* reason about the uncertainties involved with such predictions. This section will provide a simple introduction to some common terminology and methodologies that we will dive into greater detail later on. We begin with an extremely broad categorization of modelling methods: that of *parametric* and *non-parametric* modelling.

Parametric methods can loosely be understood as a modelling paradigm in which one defines some, usually fixed, set of parameters, denoted θ, that govern the operation of a model. This modelling approach necessarily imbues some assumptions on the underlying distribution of the data. One then attempts to *learn* some ideal values for these parameters given the model structure and observed data. Perhaps the most ubiquitous example is the simple linear regression model in which the predictions correspond to a vector-matrix dot product. The parameters here correspond to the vector and the data to the matrix. Parametric methods can effectively represent complex phenomena accurately assuming one has made accurate assumptions on the underlying data generating process. Unfortunately this is often not the case.

In a seeming contrast, **non-parametric** (Wasserman 2006) methods are methods that assume the data distribution *cannot* be defined by a finite set of parameters. Notable examples include Gaussian processes (Rasmussen and Williams 2006), infinite Hidden Markov Models, infinite latent factor models, and Dirichlet process mixtures. One can understand inference complexity using these methods to typically 'grow' in complexity in proportion to the number of observations. While these methods can be more expressive, due to their usually explicit dependency on the observations, their drawback is notably evident in the context of dataset with very large numbers of observations.

As a short aside and before moving onto further details, one should note that although the terms *parametric* and *non-parametric*, when taken at face value, are simply common parlance in the literature. However, the terms bring with them a somewhat misleading, but historically

perpetuated, dichotomy between modelling approaches: non-parametric models still have parameters themselves! It is more reasonable to think of this dichotomy as a divergence in the *strictness* of a model's dependency on observed data when it comes to making any future inferences with the chose model.

2.1.1 Linear Regression

As hinted above, perhaps the canonical parametric method is *linear regression*. In many problems one wishes to find some functional mapping $f(\cdot)$ which takes some D-dimensional query input vector $\boldsymbol{x}_* \in \mathbb{R}^D$ by applying $f(\boldsymbol{x}_*)$ for some training data $\mathcal{D} = \{\mathbf{X}_n, \mathbf{y}_n\}_{n=1}^N$ where $\mathbf{X} \in \mathbb{R}^D$ are inputs, $\mathbf{y} \in \mathbb{R}$ are associated target values, and $n = 1, 2, ..., N$. Linear regression then regards the mapping between input and output as a *linearly weighted* combination of input variables:

$$\begin{aligned} f(\boldsymbol{x}) &= \mathrm{w}_0 + \mathrm{w}_1 x_1 + ... + \mathrm{w}_D x_D \\ &= \boldsymbol{x}^\top \mathbf{w}. \end{aligned} \tag{2.1}$$

The critical thing to note here is that the output is linear in both the parameters ('weights') *and* the input variables x_i. A trivial but important extension of the linear model is that we can generalise it to a *nonlinear* map from the input to the output. In this case, we transform the input variables using some *basis* transformation $\phi(\boldsymbol{x})$:

$$\begin{aligned} f(\boldsymbol{x}) &= \sum_{j=0}^{M-1} \phi_j(\boldsymbol{x})^\top \mathrm{w}_j \\ &= \boldsymbol{\phi}(\boldsymbol{x})^\top \mathbf{w}, \end{aligned} \tag{2.2}$$

where we have introduced a dummy variable $\phi_0(\boldsymbol{x}) = 1$ that allows the parameter w_0 to act as a *bias* variable. This basis function transforms an input D-dimensional data vector into some M dimensional vector. There is no necessary restriction on the size of M. In this

non-linear transformation, we *still have a linear model*, except the output is no longer linear in the original data, but *linear in the transformation of the original data*.

This transformed data is often called the *design matrix* and we denote it as $\mathbf{\Phi}$. It has elements defined as $\Phi_{n,j} = \phi_j(\mathbf{x}_n)$ giving

$$\mathbf{\Phi}(\boldsymbol{x}) = \begin{bmatrix} \phi_0(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_1) & \dots & \phi_{M-1}(\boldsymbol{x}_1) \\ \phi_0(\boldsymbol{x}_2) & \phi_1(\boldsymbol{x}_2) & \dots & \phi_{M-1}(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\boldsymbol{x}_N) & \phi_1(\boldsymbol{x}_N) & \dots & \phi_{M-1}(\boldsymbol{x}_N) \end{bmatrix}. \tag{2.3}$$

The linear model, as it is, assumes a completely deterministic relation between the output and the inputs (assuming our weights $\mathbf{w}$ are deterministic). We can now extend the model to allow for an independent additive noise parameter ϵ giving the noise-augmented model y:

$$y = f(\boldsymbol{x}) + \epsilon, \tag{2.4}$$

where ϵ is some random variable, typically Gaussian with zero mean and variance σ_n^2; i.e. $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, however this is not necessary and one could replace this term with an arbitrary distribution or even an input dependent distribution.

There are two well known approaches to solving for the weights of the linear model. These are *Maximum Likelihood* (MLE) and *Maximum a Posteriori* (MAP). MLE solves the linear model by taking the partial derivative of the log likelihood with respect to the weights and noise model. MAP is similar to MLE but additionally assumes another *prior distribution over the parameters* $\mathbf{w}$. Even though MLE and MAP have probabilistic motivations and derivations, both methods only produce *point* estimates of the parameters and consequently all predictive uncertainty information is lost. More details can be found in (Bishop 2007).

In the next section we will introduce the Bayesian Linear Regression (BLR) model, also called *empirical Bayes* or *type-2 MLE*.

2.1.2 Bayesian Linear Regression

Instead of obtaining point estimates of model parameters, we can estimate the entire distribution over the parameters. In other words, we marginalise over (integrate out) all possible parameter values and weight the result using the posterior. Sometimes, but incorrectly, called the 'fully Bayesian approach', one assumes a *hyperprior* (simply another distribution) over the parameters, of the distributions, that represent the 'bottom level' parameters. To put it another way, type-2 MLE is a method by which we compute point estimates of parameters that parameterise the weight priors. Under this interpretation we can see the appropriateness of the term type-2 MLE. In theory one could regress this over uncountably many hierarchical hyperpriors but this is computationally implausible. In any case, what we are usually interested in under this paradigm, is that we can obtain a full *predictive distribution* for making predictions f_* for new values $\boldsymbol{x}_*$:

$$p(f_*|\boldsymbol{x}_*, \boldsymbol{x}, \mathbf{y}, \alpha, \beta) = \int p(f_*|\boldsymbol{x}_*, \mathbf{w}, \beta)p(\mathbf{w}|\boldsymbol{x}, \mathbf{y}, \alpha, \beta)d\mathbf{w}, \tag{2.5}$$

where α is termed the inverse weight variance or *precision*, and β is the precision of the noise. If we assume (for simplicity) that we can use univariate Gaussian distributions for the distribution parameterisations, we can obtain an analytic predictive distribution following from the analytic posterior distribution over the weights:

$$p(f_*|\boldsymbol{x}_*, \boldsymbol{x}, \mathbf{y}, \alpha, \beta) = \mathcal{N}(f_*|\boldsymbol{\mu}^\top \phi(\boldsymbol{x}_*), \sigma_*^2(\boldsymbol{x}_*)), \tag{2.6}$$

where

$$\boldsymbol{\mu} = \beta \boldsymbol{\Sigma} \boldsymbol{\Phi}^\top \mathbf{y}, \tag{2.7}$$

$$\sigma_*^2(\boldsymbol{x}_*) = \frac{1}{\beta} + \phi(\boldsymbol{x}_*)^\top \boldsymbol{\Sigma} \phi(\boldsymbol{x}_*), \tag{2.8}$$

$$\boldsymbol{\Sigma}^{-1} = \alpha \mathbf{I} + \beta \boldsymbol{\Phi}^\top \boldsymbol{\Phi}. \tag{2.9}$$

2.1.3 Variational Parameter Estimation

As we have seen above, certain Bayesian interpretations under specific distributional choices, e.g. Gaussian, allow estimation methods such as MLE and MAP to produce closed form posterior solutions. It is often the case that such simplified distributional assumption are not justified and one wishes to pick from more flexible or appropriate distributions. Variational Inference (VI) (Wainwright, Jordan et al. 2008) is a method for determining approximate probabilistic posterior inference using tractable optimisation. At a high level it consists of two steps: First, we assume some approximating distribution family $q(\mathbf{z};\lambda)$ over the latent variables $\mathbf{z}$ we wish to estimate, and second, optimise over the variational parameters λ to reduce some measure of distributional divergence between the variational distribution $q(\mathbf{z};\lambda)$ and the true, but unknown, posterior $p(\mathbf{z}|\mathbf{x})$ where $\mathbf{x}$ is the set of all observed variables. This optimization procedure can generally be expressed as follows:

$$\lambda^* = \underset{\lambda}{\operatorname{argmin}}\ D(p(\mathbf{z}|\mathbf{x}), q(\mathbf{z};\lambda)), \tag{2.10}$$

where D is the divergence measure.

In most real world situations, the posterior $p(\mathbf{z}|\mathbf{x})$is intractable and thus it is necessary to represent and solve for an approximate generating model instead. One of the most common ways to minimise the divergence is by using the Kullback-Leibler (KL) divergence from $q(\mathbf{z};\lambda)$ to $p(\mathbf{z} \mid \mathbf{x})$,

$$\lambda^* = \underset{\lambda}{\operatorname{argmin}}\ \mathrm{KL}(q(\mathbf{z};\lambda)\|p(\mathbf{z} \mid \mathbf{x})) \tag{2.11}$$

$$= \underset{\lambda}{\operatorname{argmin}}\ \mathbb{E}_{q(\mathbf{z};\lambda)}\big[\log q(\mathbf{z};\lambda) - \log p(\mathbf{z} \mid \mathbf{x})\big]. \tag{2.12}$$

Unfortunately the problem defined in (2.12) is intractable since it depends on the posterior, however it is possible to take advantage of the property

$$\log p(\mathbf{x}) = \mathrm{KL}(q(\mathbf{z};\lambda)\|p(\mathbf{z} \mid \mathbf{x})) + \mathbb{E}_{q(\mathbf{z};\lambda)}\big[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\lambda)\big]. \tag{2.13}$$

The left hand side is the logarithm of the marginal likelihood where $p(\mathbf{x}) = \int p(\mathbf{x},\mathbf{z})\mathrm{d}\mathbf{z}$ is the model evidence. Since the evidence is constant relative to the variational parameters λ,

we can minimise $\mathrm{KL}(q|p)$ by maximising the probability of observing the data. This can be seen as a lower bound on the evidence and is called the *Evidence Lower Bound* (ELBO),

$$\mathrm{ELBO}(\lambda) = \mathbb{E}_{q(\mathbf{z};\lambda)}\big[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\lambda)\big]. \tag{2.14}$$

Both $p(\mathbf{x},\mathbf{z})$ and $q(\mathbf{z};\lambda)$ are tractable within the ELBO and thus we have an optimisable objective:

$$\lambda^* = \operatorname*{argmax}_{\lambda} \mathrm{ELBO}(\lambda). \tag{2.15}$$

Although there are various ways to perform this optimisation, we will follow the approach in (Kingma and Welling 2014) that allows us to reparameterise our distributions in such a way that allows the use of automatic differentiation over the variational distributions' gradients. The key insight to this is that some variational distributions $q(\mathbf{z};\lambda)$ allow us to define some random variable $\epsilon \sim q(\epsilon)$ that *doesn't depend on the variational parameters* λ, and a deterministic function $\mathbf{z} := \mathbf{z}(\epsilon;\lambda)$, which contains our variational parameters. As a consequence we can obtain unbiased Monte Carlo estimates of the gradient:

$$\nabla_\lambda \mathrm{ELBO}(\lambda) = \mathbb{E}_{q(\epsilon)}\big[\nabla_\lambda\big(\log p(\mathbf{x},\mathbf{z}(\epsilon;\lambda)) - \log q(\mathbf{z}(\epsilon;\lambda);\lambda)\big)\big]. \tag{2.16}$$

In other words, the gradient of the ELBO is an expectation over some distribution $q(\epsilon)$. Consequently, this variational formulation allows us to tractably optimise weights and hyperparameters in a principled manner. Later in the thesis we will show how we use this methodology to optimise a previously intractable kernel learning model.

2.2 Kernel Methods

2.2.1 Equivalent Kernel

Returning to our Bayesian Linear Regression solution in the previous section, first observe the posterior mean given by equation (2.7). If we take (2.7) and substitute it into (2.2) we

arrive at the predictive mean in the following form:

$$\begin{aligned} f(\boldsymbol{x}_*, \boldsymbol{\mu}) &= \boldsymbol{\mu}^\top \boldsymbol{\phi}(\boldsymbol{x}_*) \\ &= \beta \boldsymbol{\phi}(\boldsymbol{x}_*)^\top \boldsymbol{\Sigma} \boldsymbol{\Phi}^\top \mathbf{y} \\ &= \sum_{n=1}^{N} \beta \boldsymbol{\phi}(\boldsymbol{x}_*)^\top \boldsymbol{\Sigma} \boldsymbol{\phi}(\boldsymbol{x}_n) \mathrm{y}_n. \end{aligned} \tag{2.17}$$

This implies that the mean value of the predictive distribution at some test point $\boldsymbol{x}_*$ is some linear combination of training targets y_n. This observation motivates the following interpretation of the predictive distribution:

$$f(\boldsymbol{x}_*) = \sum_{n=1}^{N} k(\boldsymbol{x}_*, \boldsymbol{x}_n) \mathrm{y}_n, \tag{2.18}$$

where we have the function

$$k(\boldsymbol{x}, \boldsymbol{x}') = \beta \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\Sigma} \boldsymbol{\phi}(\boldsymbol{x}'), \tag{2.19}$$

which we may call the *equivalent kernel*.

By considering the covariance between $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ we can see:

$$\begin{aligned} \mathrm{Cov}\left(f(\boldsymbol{x}), f(\boldsymbol{x})'\right) &= \mathrm{Cov}\left(\boldsymbol{\phi}(\boldsymbol{x})^\top \mathbf{w}, \mathbf{w}^\top \boldsymbol{\phi}(\boldsymbol{x}')\right) \\ &= \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\Sigma} \boldsymbol{\phi}(\boldsymbol{x}') \\ &= \beta^{-1} k(\boldsymbol{x}, \boldsymbol{x}'). \end{aligned} \tag{2.20}$$

This equivalent kernel given in (2.19) fulfills a key property of kernels in general: an inner product with respect to some vector $\psi(\boldsymbol{x})$ such that:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \psi(\boldsymbol{x})^\top \psi(\boldsymbol{x}'), \tag{2.21}$$

with $\psi(\boldsymbol{x}) = \beta^{1/2} \boldsymbol{\Sigma}^{1/2} \phi(\boldsymbol{x})$.

This short segue from linear models, to a particular Bayesian solution to the type-2 MLE brings us to kernel methods (Schölkopf and Smola 2002). Perhaps one of the most widespread examples of non-parametric modelling methods in which $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is some kernel on an input domain $\mathcal{X} \subset \mathbb{R}^D$. One may interpret the kernel k as some embedding in a high-dimensional Hilbert space $\mathcal{H}$ through a feature map $\Phi : \mathcal{X} \to \mathcal{H}$ as an inner product between points from the feature map with $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$.

This continuing section from our discourse on Bayesian Linear Regression naturally leads to the Gaussian process (GP) – an extremely popular and powerful method of nonparametric modelling. Following this we expand upon what it means to be a *covariance function*, or simply *kernel* and then demonstrate how it is possible to *compose* kernels to produce more complicated covariances that can represent information that individual, canonical, kernels cannot. We then introduce *kernel approximation* methods in terms of the general form of *function* approximation and in the context of the key methods we will expand upon in the subsequent chapters.

2.2.2 Gaussian Processes

Gaussian process (GP) regression (Rasmussen and Williams 2006) is a method of learning some probability distribution over functions $f(\boldsymbol{x})$ given inputs $\boldsymbol{x} \in \mathbb{R}^D$ given training data $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$ where $n = 1, 2, ..., N$.

The model may be defined for some set of parameters $\boldsymbol{\theta}$ as a Gaussian random process prior:

$$f \sim \mathcal{GP}(\mathbf{m}, k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})), \tag{2.22}$$

where

$$\mathbf{m} = \mathbb{E}[f(\boldsymbol{x})]. \tag{2.23}$$

Our GP has the predictive distribution

$$p(f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \mathcal{D}) = \mathcal{N}(\mathbb{E}[f(\boldsymbol{x}_*)], \mathbb{V}[f(\boldsymbol{x}_*)]). \tag{2.24}$$

This has a closed form solution for the predictive mean:

$$\mathbb{E}[f(\boldsymbol{x}_*)] = k_*^T (K + \sigma^2 I)^{-1} y, \tag{2.25}$$

and variance:

$$\mathbb{V}[f(\boldsymbol{x}_*)] = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k_*^T (K + \sigma^2 I)^{-1} k_*, \tag{2.26}$$

where $\mathbf{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the Gram matrix of our kernel function k, k_* is an N-dimensional vector with the i^{th} entry being $k(\boldsymbol{x}_*, \boldsymbol{x}_i)$ and y a vector of the N observations. To solve the GP to obtain a prediction, one typically needs to perform an $N \times N$ matrix inversion of the gram matrix K which significantly hinders scalability to very large datasets since this involves an $O(N^3)$ complexity. To try and alleviate this complexity, there have been many recent attempts to approximate the 'full GP' problem.

As can be seen above, the result of the GP relies critically on the choice of kernel k. The next section will introduce a small library of various canonical kernels and trivial ways that one may combine such kernels to obtain more flexible representations.

2.2.3 Kernel Functions

The kernel (Schölkopf and Smola 2002) function is a general mathematical notion that lends itself to integration into many machine learning algorithms such as the well known GP model above and well popularised Support Vector Machine (SVM), amongst many others. In essence, the kernel encodes some *prior belief* (as imposed by its mathematical definition) on the function one is trying to model whether or not it is with a GP or some other model that integrates kernels. Broadly speaking, the kernel is a measure of *similarity* between two objects. We show a small set of some common kernels realised over a linear span of scalars in Figure 2.1. To show what's possible with increasing the expressiveness of kernels we show realisations of *compositions* of kernels in Figure 2.2.

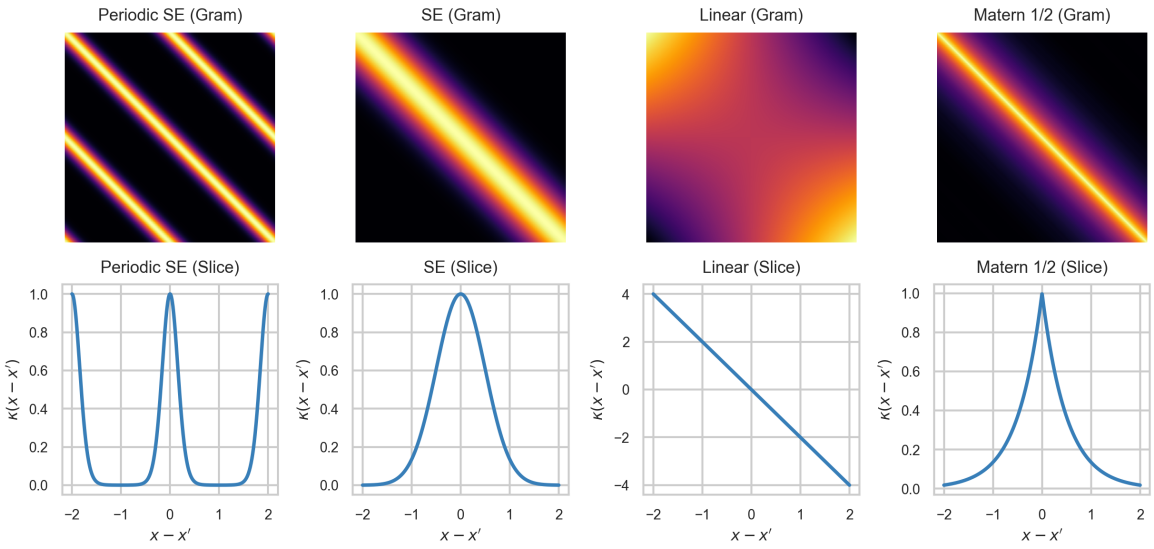


Figure 2.1. A small kernel zoo. Various canonical kernels with visualisation of their respective gram matrices (top row) and arbitrary slices taken from their corresponding gram (bottom row). (Tompkins 2018)

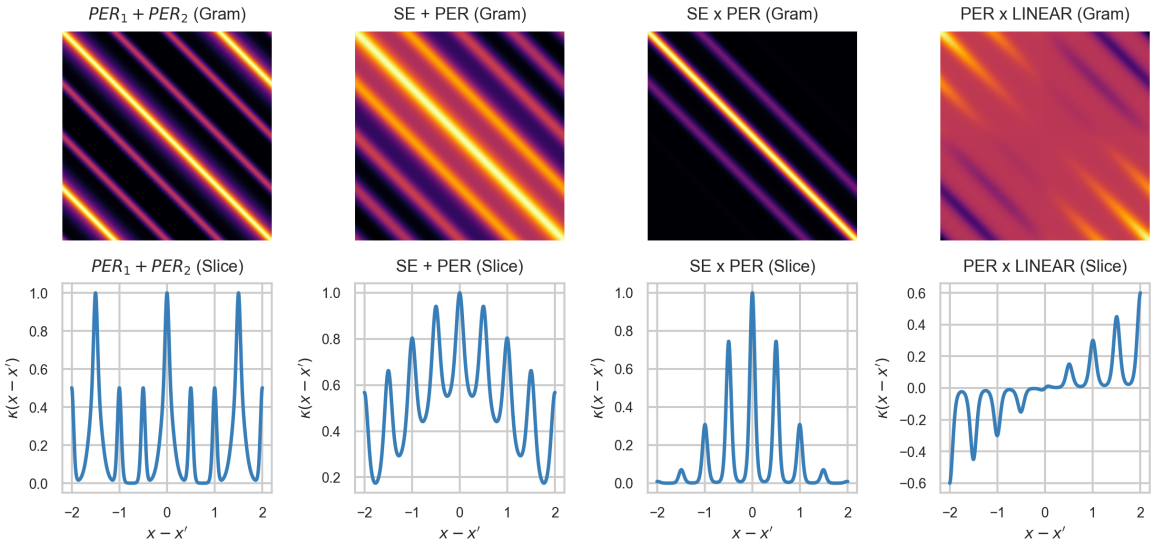


Figure 2.2. A small *compositional* kernel zoo. Various compositions of kernels with visualisation of their respective gram matrices (top row) and arbitrary slices taken from their corresponding gram (bottom row). (Tompkins 2018)

DEFINITION 1. *(Positive definite kernel) For some nonempty set $\mathcal{X}$, a symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite (p.d.) kernel on $\mathcal{X}$ if and only if,*

$$\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0 \quad \forall \quad \boldsymbol{x}_i \in \mathbb{R}^D \text{ and } a_i, a_j \in \mathbb{R}. \tag{2.27}$$

More formally, a covariance function is a special kind of function of two input data points $\boldsymbol{x}$, $\boldsymbol{x}'$, where $\boldsymbol{x}$ and $\boldsymbol{x}'$ are typically vectors in some D dimensional space. A typical requirement for kernels is that for all possible input points on the kernel's operating space (observations, query points), and hyperparameters, is that they produce *positive semi-definite* covariance matrices as per Definition 1.

Some key terminology one may encounter regarding covariance functions include: *stationary kernels*, *non-stationary kernels*, and *hyperparameters* or *kernel parameters*.

Stationary kernels: Often due to their analytic simplicity and broad effectiveness, the most commonly used kernels are *stationary*. A stationary kernel is a kernel which is *a function of the difference between two points* $\boldsymbol{x}$ and $\boldsymbol{x}'$: $\boldsymbol{\tau} = |\boldsymbol{x} - \boldsymbol{x}'|$. In upcoming chapters, we will show how stationary kernels have a natural interpretation through the lens of harmonic analysis: kernels of *stationary* stochastic processes have a corresponding Fourier transform of some positive finite measure.

Non-stationary kernels: Non-stationary kernels refers to kernels that evaluate differently, *conditional*, on where the data lies – in contrast the stationary kernels which just cares about the difference between. Additional details can be found in Chapter 4 of (Rasmussen and Williams 2006) and (Paciorek and Schervish 2004).

Kernel parameters: Most kernels will typically have their own adjustable parameters that affect its output. These parameters are typically called *hyperparameters*. For example, the *lengthscale* is a hyperparameter of the squared exponential kernel, and the *period* and *lengthscale* are hyperparameters of the periodic squared exponential kernel.

2.3 Slightly Expressive Kernels

We introduce some well known kernels.

Squared exponential: This is perhaps the most explored and applied kernel. It is infinitely differentiable and consequently is appropriate for modelling smooth processes. It is conventionally equipped with two simple hyperparameters: 1) the lengthscale $\boldsymbol{\Sigma}$. This determines how much the input variables affect the output; and 2) the signal variance h^2 which determines the overall magnitude. It is formally defined as follows:

$$k_{\text{SE}}(\boldsymbol{x}, \boldsymbol{x}') = h^2 \exp\Big(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^\top \boldsymbol{\Sigma} (\boldsymbol{x} - \boldsymbol{x}')\Big). \tag{2.28}$$

For additional flexibility, a more general form of the symmetric lengthscale matrix $\boldsymbol{\Sigma}$ allows for anisotropic modelling in the sense of treating each dimension independently. Various choices may be made for $\boldsymbol{\Sigma}$ such as $\boldsymbol{\Sigma} = l^{-2}\boldsymbol{I}$, $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{l})^{-2}$, $\boldsymbol{\Sigma} = \Lambda\Lambda^\top + \text{diag}(\boldsymbol{l})^{-2}$ where $\boldsymbol{l}$ is a vector of positive values and Λ is a $D \times k$ matrix where $k < D$. Interestingly, the spectral density of the SE kernel has the same exponential form as its time domain form: $S(s) = (2\pi\boldsymbol{\Sigma})^{D/2} \exp(-2\pi^2\boldsymbol{\Sigma}^2 s^2)$. One potential downside, while the SE kernel has many useful properties it is often regarded as encoding *too* much smoothness which is often inappropriate for modelling real world processes (Stein 1999).

Matérn class: In contrast to the SE kernel, The Matérn class of covariance functions is an example of a kernel that is not infinitely differentiable. The kernel's construction manifests itself in a very general way with the option to specify the varying amounts of differentiabilty. Its most general form is defined as follows:

$$k_{\text{matern}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \Big(\frac{\sqrt{2\nu}\,|\boldsymbol{x} - \boldsymbol{x}'|}{l} \Big)^\nu K_\nu \Big(\frac{\sqrt{2\nu}\,|\boldsymbol{x} - \boldsymbol{x}'|}{l} \Big), \tag{2.29}$$

where $K_\nu(z)$ is a modified Bessel function of the second kind of order $\nu > 0$ and argument z, Γ is the gamma function, and $l > 0$ is the lengthscale parameter.

Perhaps the two most popular concrete realisations of the Matérn, which are respectively one and two times differentiable, are when $\nu = \frac{3}{2}$ and $\nu = \frac{5}{2}$ in which case we obtain the simplified expressions:

$$k_{\nu=3/2}(\boldsymbol{x}, \boldsymbol{x}') = \Big(1 + \frac{\sqrt{3}\,|\boldsymbol{x} - \boldsymbol{x}'|}{l}\Big) \exp\Big(-\frac{\sqrt{3}\,|\boldsymbol{x} - \boldsymbol{x}'|}{l}\Big), \tag{2.30}$$

$$k_{\nu=5/2}(\boldsymbol{x}, \boldsymbol{x}') = \Big(1 + \frac{\sqrt{5}\,|\boldsymbol{x} - \boldsymbol{x}'|}{l} + \frac{\sqrt{3}\,|\boldsymbol{x} - \boldsymbol{x}'|^2}{l^2}\Big) \exp\Big(-\frac{\sqrt{5}\,|\boldsymbol{x} - \boldsymbol{x}'|}{l}\Big). \tag{2.31}$$

An interesting note about the Matérn is that one can show that as $\nu \to \infty$ the kernel becomes the squared exponential, and when $\nu = \frac{1}{2}$ one recovers what is often called the *exponential* kernel which is continuous but not differentiable. Indeed the exponential kernel correspond to the covariance function of the Ornstein-Uhlenbeck process which is the mathematical model one obtains from a particle following Brownian Motion (Uhlenbeck and Ornstein 1930).

2.4 Slightly More Expressive Kernels

At this point you may be wondering could we combine some of these individual kernels to get some kind of trade-off between them? The answer to this is yes! It is possible to trivially compose standard p.d. kernels using addition and product operations. One can adopt composition operators on the full covariance (Schölkopf and Smola 2002) as well as the in the feature space view (Shawe-Taylor and Cristianini 2004). This can be summarised with the following operator notation for *sum* and *product* operations on kernels.

The addition of two covariance matrices generated by two kernels k_1 and k_2:

$$(k_1 + k_2)(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}') + k_2(\boldsymbol{x}, \boldsymbol{x}'), \tag{2.32}$$

and the multiplication of two covariance matrices generated by two kernels k_1 and k_2:

$$(k_1 \times k_2)(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}') \times k_2(\boldsymbol{x}, \boldsymbol{x}'). \tag{2.33}$$

2.4.1 Sums of Kernels

One way to interpret a sum of kernels is through a logical *or* operation over the summed kernels. One instance where we can see the benefit of assuming additivity can be seen when

used in the context of GP modelling. Essentially, if our true data generating process is made up of an additive superposition of multiple processes each satisfying the assumptions of each individual kernel, then we would be able to efficiently model the data. Some visual examples of kernel sums can be seen in Figure 2.3.

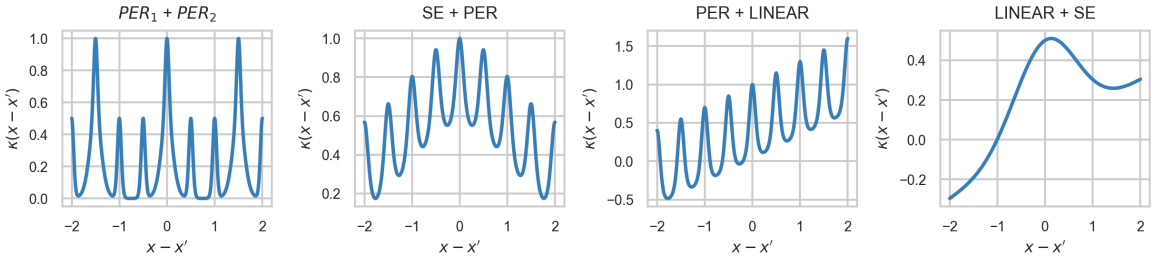


Figure 2.3. Examples of kernel compositions using **addition**. (Tompkins 2018)

2.4.2 Products of Kernels

In the same way we can consider additive kernels as logical *or* operations, the product of two kernels can be interpreted as a logical *and* operation between the participating kernels. Various interesting phenomena can be naturally expressed with products of kernels. Just a few examples can include: *functions of growing amplitude* by multiplying a stationary lengthscale (e.g. SE) kernel with a linear kernel, M^{th} *degree polynomial priors* by multiplying M linear kernels together (indeed this corresponds precisely to well known polynomial features), and *locally periodic structure* by multiplying periodic kernels with some stationary kernel like the SE or Matérn. Some visual examples of kernel products can be seen in Figure 2.4.

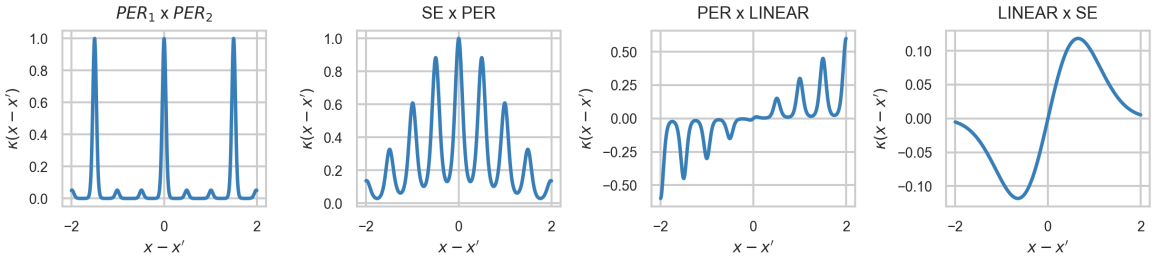


Figure 2.4. Examples of kernel compositions using **multiplication**. (Tompkins 2018)

2.4.3 Multi-dimensional Compositions

One more variation we could achieve with kernel compositions is by applying specific kernels to corresponding subsets of our full input dimension. Let $\mathcal{D} \in \mathbb{Z}^+$ be an ordered set of integers representing a subset of dimensions. An example of such a composition can be written as follows:

$$k_{composition} = k_1^{\mathcal{D}=(1,2,5)} + k_2^{\mathcal{D}=(3,4)}, \tag{2.34}$$

where $k_1^{\mathcal{D}=(1,2,5)}$ is one kernel applied to dimensions $1, 2, 5$ and $k_2^{\mathcal{D}=(3,4)}$ is a second kernel applied to dimensions $3, 4$.

2.4.4 Other Kernel Operations

There are countless more, usually 'discrete' auxiliary variations related to compositional kernel learning. A few notable examples include: *changepoint kernels* for detecting regime changes in data, *manifold learning* by warping data in a non-linear fashion, *categorical kernels* operating on categorical variables, *symmetry* respecting kernels such as those that model symmetries around some origin that represent certain invariant behaviours, and kernels for *multiple-output* GPs. The interested reader may find more details in (MacKay 1998), (Rasmussen and Williams 2006), and Chapter 2 of (Duvenaud 2014).

2.4.5 Caveats of Kernel Compositions

While the aforementioned kernel compositions are well formulated and very interpretable (in the sense of the individual kernels which make up the aggregated kernel), they are still extremely limited. They beg the question: why should we restrict ourselves to *this* particular stationary kernel or and addition between *this* and *that* kernel? The real answer is there is no reasonable reason other than 'we think our data generating process has these properties'. This is somewhat disappointing! If we knew the properties of our data generating process then why would we be going to all this trouble optimising and finding appropriate kernels in the first place. This is ultimately the biggest caveat of both using canonical kernels and compositions

of canonical kernels – their base assumptions (such as on smoothness) are ultimately arbitrary or unjustified in the regime of discrete observations. We will see later on, throughout the main body of this thesis, how we can try to address this problem by taking a more fundamental and holistic perspective of what it means to be a kernel.

2.4.6 Spectral Kernel Representations

All the kernels we have encountered above are what could be termed the *primal* form of the kernel – analytically exact functional definitions that act directly on the data and explicitly construct a Gram or kernel covariance matrix. This matrix is precisely quadratic in size, proportional to the number of observations used to construct it. Consequently, it is obvious to see that this results in significant drawbacks when dealing with larger amounts of data. In fact this is a systemic issue with non-parametric methods that often require the entire dataset or a subset of the full dataset to be stored in order to perform the desired inference. It is therefore reasonable, as the sample size of our datasets become larger, to resort to approximations. These usually manifest themselves as either data-dependent sampling methods, or data-independent projections that produce low-rank covariance structures.

In the regime of data sampling based methods, the Nyström method is perhaps the most prominent. Details can be found in (Williams and Seeger 2001) for the original version and (Gittens and Mahoney 2013) for a more elaborate discourse. To summarise the method, one selects a subset of observations from the original dataset and then computes the columns of the full Gram that correspond to these sampled points. This incomplete Gram matrix is then used as a surrogate to the full Gram. One may consider this as a *data-dependent* approximation method.

Contrasting the data-dependent approach is that of *data-independent* approximation. We will explore this perspective in greater detail throughout this thesis. Perhaps the most significant literature regarding approximate kernel representation is from (Rahimi and Recht 2007b) with a method termed the *Random Fourier Feature* (RFF). The key result hearkens back to an old result in harmonic analysis called Bochner's theorem:

THEOREM 1. *(Bochner 1933) A complex-valued function $g : \mathbb{R}^D \to \mathbb{C}$ is positive definite if and only if it is the Fourier Transform of a finite non-negative Borel measure μ on $\mathbb{R}^D$:*

$$g(\boldsymbol{\tau}) = \hat{\mu}(\boldsymbol{\tau}) = \int_{\mathbb{R}^D} e^{-i\boldsymbol{\tau}^T\boldsymbol{\omega}} d\mu(\boldsymbol{\omega}), \quad \forall \boldsymbol{\tau} \in \mathbb{R}^D. \tag{2.35}$$

In essence, the theorem provides a direct connection between shift-invariant kernels and embeddings of distributions. The kernel is given an integral representation and consequently this implies it can be approximated via a sampling of integral. Some of the main results shown in this thesis will show how we can leverage such a perspective to construct *arbitrary* but theoretically valid kernels.

To get a better understanding of function integration let us consider an integral I of the following form:

$$I_D[f] = \int_{[0,1]^D} f(\boldsymbol{x}) d\boldsymbol{x}. \tag{2.36}$$

If $\boldsymbol{x}$ is some random vector uniformly distributed over $[0, 1]^D$ then we have the expectation $I_D[f] = \mathbb{E}[f(\boldsymbol{x})]$. We can estimate the expected value by taking discrete samples using independent samples from $[0, 1]^D$ with a random set $S = \{\mathbf{w}_1, ..., \mathbf{w}_s\}$. This empirical method of estimation is commonly known as the *Monte Carlo* (MC) method (Robert 2004):

$$I_S[f] = \frac{1}{s} \sum_{\mathbf{w} \in S} f(\mathbf{w}). \tag{2.37}$$

It is possible to specify an integration error w.r.t. the set S as,

$$\epsilon_S[f] = |I_D(f) - I_S(f)|. \tag{2.38}$$

According to the Central Limit Theorem, if S is randomly drawn and $s = |S|$ tends toward infinity, then we have that $\epsilon_S[f] \approx \sigma[f] s^{-1/2} \nu$ where ν is a standard normal r.v. and $\sigma[f]$ is

the square root of the variance of f. This represents the root mean square error (RMSE) of the MC method:

$$\sigma^2[f] = \int_{[0,1]^D} (f(\boldsymbol{x}) - I_D(f))^2 d\boldsymbol{x}. \tag{2.39}$$

This yields in expectation:

$$(\mathbb{E}_S[\epsilon_S[f]^2])^{1/2} \approx \sigma[f] s^{-1/2}, \tag{2.40}$$

which has a rate of convergence of $O(s^{-1/2})$.

This Monte Carlo approximation will turn out to be at the heart of the novel approximate kernel representations later in this thesis.

2.4.7 Kernel Learning

For a non-empty set $\mathcal{X} \subset \mathbb{R}^D$, let us denote the kernel function as $k : \mathcal{X} \times \mathcal{X} \to \mathbb{C}$. By adding the inner product structure $\langle \cdot, \cdot \rangle$, a kernel can be represented as $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$ for $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, where $\phi : \mathcal{X} \mapsto \mathcal{H}$ is a mapping from the low dimensional input space $\mathcal{X}$ into a possibly infinite-dimensional Hilbert space $\mathcal{H}$. Intuitively, these inner product kernels quantify the similarity between two input points. The simplest kernel is the linear kernel given by $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^* \mathbf{x}'$. However, arguably, the most popular kernel is the squared-exponential kernel $k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp(-2\gamma^2 \|\mathbf{x} - \mathbf{x}'\|_2^2)$ with parameters σ^2 and γ^2. If less smooth fittings are intended, then the user would choose kernels such as the neural network or Matérn $\frac{3}{2}$ (Rasmussen 2004). Similarly, if there is a seasonal pattern it is desirable to use a periodic kernel $k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp(-2\gamma^2 \sin^2(\frac{\rho\pi}{2}|\mathbf{x} - \mathbf{x}'|))$ with the periodicity parameter ρ.

Because of the inner product, the kernel function is, i) positive-definite, ii) conjugate symmetric $k(\mathbf{x}, \mathbf{x}') = k^*(\mathbf{x}', \mathbf{x})$, and iii) satisfies linearity in the first argument of the kernel. Due to these properties, functional composition, and convolution of kernels result in another valid kernel. Even though kernels constructed with these operations are capable of computing

complex patterns, the choice of kernels and their combinations often depend on the domain knowledge and physical observations of the data analyst.

COROLLARY 1. *If the measure μ in Theorem 1 is a probability measure with $\hat{\mu}(0) = 1$ and has a probability density function (pdf) f_Ω on the random variable Ω with its realization $\boldsymbol{\omega} \in \mathbb{R}^D$, then $\hat{\mu}(\mathbf{x} - \mathbf{x}') =: k(\mathbf{x}, \mathbf{x}')$ is a continuous, stationary, and positive-definite covariance function that satisfies,*

$$k(\mathbf{x}, \mathbf{x}') = \int_{\mathbb{R}^D} e^{-i(\mathbf{x}-\mathbf{x}')^\top \boldsymbol{\omega}} f_\Omega(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}. \tag{2.41}$$

In a series of pioneering work (Wilson and Adams 2013; Wilson et al. 2013), the covariance function of a Gaussian process prior is modeled by making use of a result similar to Corollary 1 as a spectral representation (Rasmussen 2004). Taking advantage of mixture representations and sampling from a pdf has been explored in (Lázaro-Gredilla et al. 2010; Oliva et al. 2016).

2.5 Optimal Transport (OT)

The final concept we will explore is that of *optimal transport*. We will utilise this later on to reframe conventional long winded sequential learning problems into multiple short parallel optimisation problems.

2.5.1 Discrete Optimal Transport

At a high level, optimal transport seeks to answer the question: what is the best way to move information from one location to another location with the the least effort; indeed a classical example of this problem is a toy example of moving 'dirt' from one configuration to another with the least effort. The Monge-Kantorovich Theorem is a convex relaxation of the optimal transport problem proposed by Monge (Monge 1781; Villani 2008). Unlike in Monge's formulation, Monge-Kantorovich guarantees the existence of a transport map $P : \Omega^{(\mathcal{S})} \to \Omega^{(\mathcal{T})}$.

THEOREM 2. *(Monge-Kantorovich) (Villani 2008) Let $\Omega^{(\mathcal{S})}$ and $\Omega^{(\mathcal{T})}$ be two separable metric spaces such that probability measures $\boldsymbol{\mu}^{(\mathcal{S})}$ and $\boldsymbol{\mu}^{(\mathcal{T})}$ on $\Omega^{(\mathcal{S})}$ and $\Omega^{(\mathcal{T})}$, respectively, are*

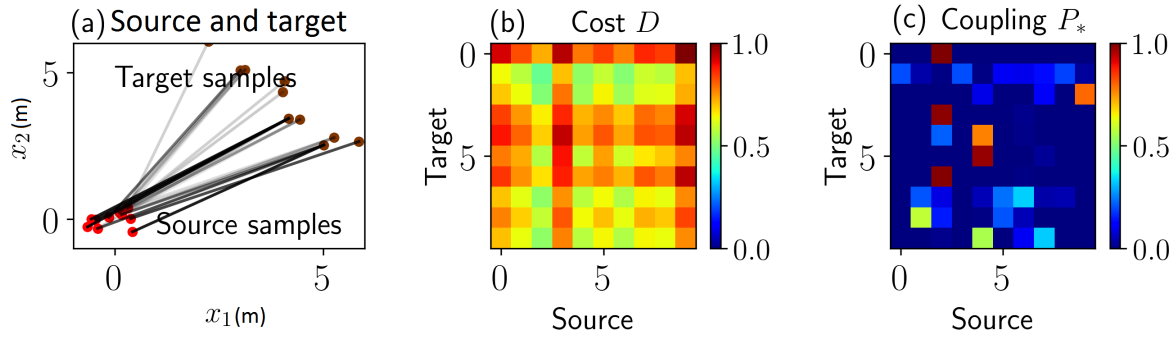


Figure 2.5. (a) 10 red and 10 brown dots indicate samples in $\mathbb{R}^2$ from a *source* and *target* distributions, respectively. The higher the transparency of gray lines, the lower the probability of couplings (matches) obtained after solving Equation (2.43). (b) 10×10 pairwise cost matrix D between the positions of samples. (c) 10×10 coupling matrix P_* indicates the optimal coupling probability of source points and all other target points. Determining this matrix (and gray lines in (a)) is one of the goals of optimal transport.

Radon measures. The optimal coupling,

$$P_* = \underset{P \in \Gamma(\boldsymbol{\mu}^{(\mathcal{S})}, \boldsymbol{\mu}^{(\mathcal{T})})}{\operatorname{arginf}} \int_{\Omega^{(\mathcal{S})} \times \Omega^{(\mathcal{T})}} D(\boldsymbol{\mu}^{(\mathcal{S})}, \boldsymbol{\mu}^{(\mathcal{T})}) \mathrm{d}P(\boldsymbol{\mu}^{(\mathcal{S})}, \boldsymbol{\mu}^{(\mathcal{T})}), \tag{2.42}$$

always exists for a distance function $D : \Omega^{(\mathcal{S})} \times \Omega^{(\mathcal{T})} \to [0, \infty)$*, where* Γ *is the set of all couplings (probability measures) on* $\Omega^{(\mathcal{S})}$ *and* $\Omega^{(\mathcal{T})}$ *with marginals* $\boldsymbol{\mu}^{(\mathcal{S})}$ *and* $\boldsymbol{\mu}^{(\mathcal{T})}$*, respectively.*

As stated earlier, and as illustrated in Figure 2.5, the OT problem attempts to determine the ideal way to transfer some distribution from one location to another. If $\boldsymbol{\mu}^{(\mathcal{S})}$ and $\boldsymbol{\mu}^{(\mathcal{T})}$ constitute two datasets of size $N^{(\mathcal{S})}$ and $N^{(\mathcal{T})}$, respectively, there always exists an optimal probabilistic coupling $P_* \in \mathbb{R}^{N^{(\mathcal{S})} \times N^{(\mathcal{T})}}$ between the two datasets (Courty et al. 2017).

As a concrete example, observe Figure 2.5 where we can see colour coded samples coming from two different distributions, on the left and right, in red and brown respectively. The goal is to find some transport plan (allocation) and this is demonstrated in the coupling matrix P_*. This is a doubly stochastic matrix, where each row and column sums to one, each cell indicating the probability of a sample in the source distribution aligning with a sample in the target distribution.

More formally, with some observed source data, for a new target dataset, we attempt to obtain the optimal coupling,

$$P_* = \underset{P \in \Gamma(\mathbf{x}^{(\mathcal{S})}, \mathbf{x}^{(\mathcal{T})})}{\operatorname{argmin}} \sum_{ij} P_{ij} D_{ij} - \lambda^{-1} r(P), \tag{2.43}$$

for a given $D \in \mathbb{R}^{N^{(\mathcal{S})} \times N^{(\mathcal{T})}}$ distance matrix (for example, one could use the squared Euclidean distance between source-target pairs) with the information entropy of P,

$$r(P) = -\sum_{ij} P_{ij} \log P_{ij}. \tag{2.44}$$

An approximation of the original problem, this entropic regularisation, commonly known as the Sinkhorn distance (Cuturi 2013; Genevay, Peyré and Cuturi 2018), allows one to replace an otherwise difficult integer programming problem with a significantly more efficient iterative algorithm (Sinkhorn and Knopp 1967). A single hyperparamter, λ, is introduced to control the regularisation strength. The larger one sets this the closer we approach the original optimisation problem.

2.5.2 Wasserstein metric

There is a special case of Theorem 2 that measures the distance between two probability distributions. Known as the *2-Wasserstein distance* it is calculate between Dirac measures $\boldsymbol{\mu}^{(\mathcal{S})}$ and $\boldsymbol{\mu}^{(\mathcal{T})}$ introduced in Section 2.5.1 is defined as,

$$\mathcal{W}_2(\boldsymbol{\mu}^{(\mathcal{S})}, \boldsymbol{\mu}^{(\mathcal{T})}) = \left\{ \min_{P \in \mathcal{P}_*} \sum_{ij} P_{ij} D_{ij} \right\}^{\frac{1}{2}} \tag{2.45}$$

where $D_{ij} = \|\mathbf{x}_i^{(\mathcal{S})} - \mathbf{x}_j^{(\mathcal{T})}\|_2$ (i.e. squared Euclidean distance).

We will see later how to take advantage of this metric to perform domain adaptation from pre-trained models to construct new, data conditioned models, in real-time at test time.

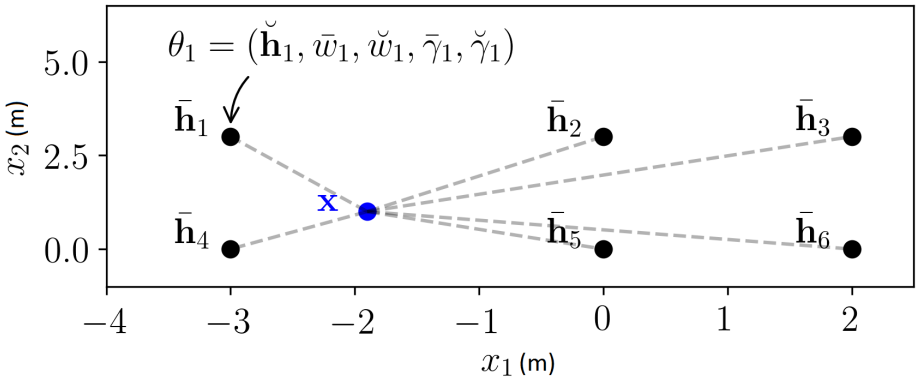


Figure 2.6. Kernel positioning. Kernels are placed in different locations $\bar{\mathbf{h}}$. For instance, here, the distance between each data point $\mathbf{x}$ and $\{\bar{\mathbf{h}}_m\}_{m=1}^{M=6}$ has to be evaluated as in Equation 4.1.

2.6 Probabilistic Mapping in Robotics

In essence, robotics is the study of robots and their agency within some, often unstructured, environment. In order to make the most of its capabilities, a robot should have some kind of representation of *the world out there* – that is to say, the environment that the robot resides within. Without the ability to represent its surrounding environment in some meaningful way, a robot would have no means by which to operate effectively. There are countless downstream tasks in robotics that depend on useful world representations, i.e. mapping, path planning, and control. In this thesis we will use the problem of *occupancy mapping* as our playground for exploring more flexible kernel learning.

2.6.1 Representing Occupancy

An occupancy model is typically represented as a parameterised function that models the occupancy probability of each location in the environment. The objective is to learn the model parameters θ given a set of observations from what is usually a kind of light based range finder – commonly a LIDAR system. After parameter estimation is complete, it is possible to query $y_* = p(\text{occupied}|\mathbf{x}_*, \theta) \in [0, 1]$ anywhere in the 2D space. For now we will focus our discussion and experimentation to domains in 2D space $\mathbf{x}_* \in \mathbb{R}^2 := (x_1, x_2)$, while the theory extends naturally to higher dimensions. We label locations where the LIDAR

has reflected upon an object as $y := 1 := \text{occupied}$, and randomly sampled points, in the scan's ray of travel defined as the space between each LIDAR hit and the LIDAR sensor, as $y := 0 := \text{free}$. From this we can construct a dataset $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$. Here, $\mathbf{x}_n \in \mathbb{R}^2$ are the corresponding spatial locations of $y_n \in \{0, 1\}$.

There have been multiple models proposed in the literature to represent the occupancy model. One of the seminal works, that operates on a discrete formulation, is the Occupancy Grid Map (OGM) (Elfes 1989). An important attempt to transition from an discrete cell based construction from OGMs is the Hilbert map (HM) (Ramos and Ott 2015) which was later extended with Gaussian process occupancy maps (GPOMs) (O'Callaghan and Ramos 2012; Wang and Englot 2016). Both the HM and GPOM ideas use kernel functions as critical components that can represent spatial correlations in an effective and continuous manner. Indeed the kernel methods used in GPOMs come with the extended flexibility of incorporating other useful attributes such as dynamics into occupancy mapping (Senanayake, O'Callaghan and Ramos 2017; Senanayake et al. 2016b). The HM is based on a parametric (linear regression with kernel features) model while GPOMs are operating on a non-parametric Gaussian process based model which additional allows effective characterisation of uncertainty. Despite their attractive theoretical properties, GPOMs are impractical for real-world and real-time usage because of the $\mathcal{O}(N^3)$ run-time and memory complexity of the naive Gaussian process implementation they use. An extension of Hilbert maps is the recently proposed Bayesian Hilbert maps (BHMs) (Senanayake and Ramos 2017) which comprises most of the positive attributes of GPOMs with a computational complexity cost of $\mathcal{O}(M^3)$ where $M \ll N$ and M is the number of features. That is to say the computational complexity depends on the features used rather than the number of observations – in contrast the GPOMs which has complexity growth coupled to the number of observations. To summarise BHM, it can be seen as performing Bayesian logistic regression in a high-dimensional feature space $\mathbb{R}^M$ using *k*ernels embeddings of observations (LIDAR hits and misses) (Hofmann, Schölkopf and Smola 2008; Senanayake and Ramos 2018). Although extremely effective, a major drawback of BHMs is that it firstly has no automatic ability to learn the kernel hyperparameters, and secondly the *same* kernel must be used for the entire spatial extent of the observed map – i.e.

it cannot accommodate nonstationarity in the observed environment space. This is one of the key problem areas we address in this thesis.

2.6.2 Kernel Methods in Occupancy Mapping

Kernel methods have been applied extensively in robotics and especially so when it is either necessary or useful to model nonlinear patterns with a small and large amounts of data (Mukadam, Yan and Boots 2016; Deisenroth and Rasmussen 2011; Kingravi, Maske and Chowdhary 2016; Kingravi et al. 2012; Guizilini and Ramos 2017b). Although only SE kernels with fixed lengthscales are typically used in robotic mapping (Ramos and Ott 2015; Doherty, Wang and Englot 2016; Senanayake and Ramos 2017), different kernel learning techniques have also been developed and explored in machine learning; and extensively so in the GP literature. It can be observed that the choice of kernels for a particular application is typically done through expert human knowledge (Duvenaud et al. 2013), a model selection criteria such as Bayesian information criteria (Duvenaud 2014), or compute intensive optimisation procedures (Bach, Lanckriet and Jordan 2004). As we have seen earlier, one can also compose a discrete set of existing kernels (Duvenaud et al. 2013), or represent them as a

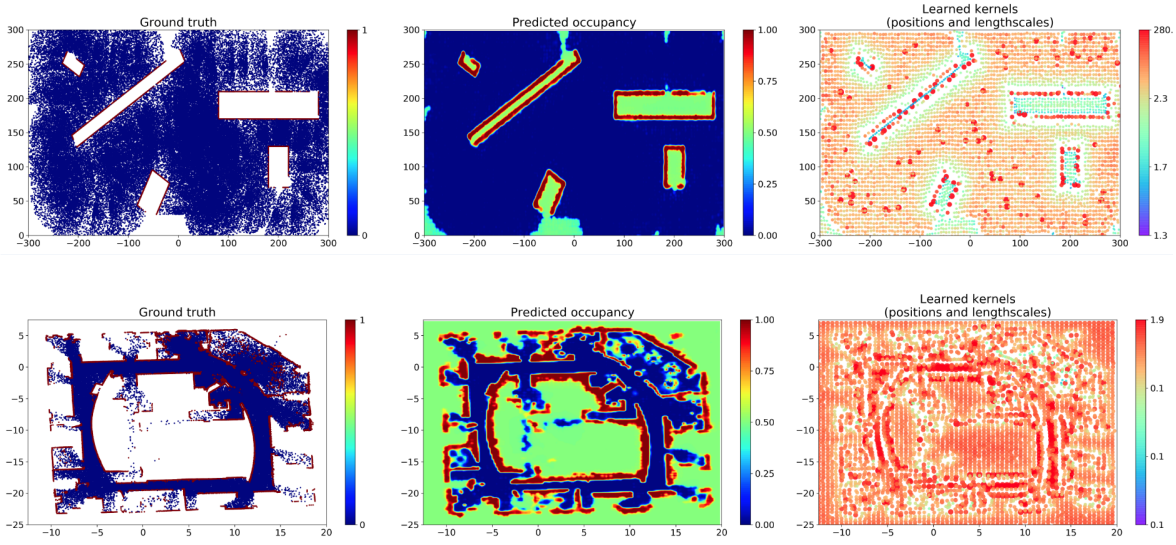


Figure 2.7. Examples of some learned Hilbert maps.
Left: laser scans, middle: predicted occupancy, right: learned kernel positions and lengthscales

spectral mixture (Wilson et al. 2013). However, unlike in Gaussian process learning where optimising the hyperparameters is a well-studied problem and solutions are easily represented with the log marginal likelihood, kernel parameter optimisation for Hilbert mapping has not been so straightforward. Some examples of learned probabilistic maps from LIDAR scans can be seen in Figure 2.7

As technology improved and became more accessible, depth sensors such as LIDAR and sonar became more natural ways to provide active sensing. Indeed it came to be that occupancy grid maps developed in the 1980s (Elfes 1989) became an exceedingly popular way to provide a computable representation for sensing the world. At a high level, OGMs splits the world into discretised grid cells with a well define grid resolution. For each cell, a Bayes filter is applied to each cell independently and this sequentially updates each cell as new depth sensor information is acquired. The problem with OGMs is that OGMs maintains an independence assumption on the sensor data acquired within each cell – each cell has no statistical connection to neighboring cells. This is problematic because in reality sensor information can be correlated by the active sensing apparatus. (O'Callaghan, Ramos and Durrant-Whyte 2009) was proposed a method called Gaussian process occupancy maps (GPOMs) which enabled the ability to capture spatial correlation with a kernel function embedding. As mentioned previously, the classic Gaussian process based methods have $\mathcal{O}(N^3)$ runtime complexity for N data points for both learning and prediction. The problem with this modelling approach that, for sensor modalities which have the ability to sample at very high rates, they are not able to scale with continually growing streams of data which is the case with active real world systems. To mitigate this problem, another kernel based method, called Hilbert maps (HMs) was introduced (Ramos and Ott 2015). HMs utilise a parametric model which, unlike the non-parametric Gaussian process in GPOMs, is able to scale linearly to larger and larger datasets.

Unlike the cell structure in OGMs, HMs learn the occupancy map in a reproducing kernel Hilbert space (RKHS) where spatial relationships are modelled using the designated kernel function. In HMs, a kernel $k(\mathbf{x}, \tilde{\mathbf{x}}) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a function that measures the similarity between two multidimensional inputs $\mathbf{x}$, $\tilde{\mathbf{x}} \in \mathcal{X} \subset \mathbb{R}^2$. Concretely, in two dimensional

HMs, we compute pairwise similarities between the elements of the two sets of points $\{\mathbf{x}_n \in \mathbb{R}^2\}_{n=1}^N$ and $\{\tilde{\mathbf{x}}_m \in \mathbb{R}^2\}_{m=1}^M$. Specifically, $\mathbf{x}$ represent longitude and latitude positions of either empty or occupied $y \in \{0, 1\} = \{\text{free}, \text{occupied}\}$ data points corresponding to samples from LIDAR scans and $\tilde{\mathbf{x}}$ are points placed, or *hinged*, at pre-defined locations of the space. A squared-exponential (SE) kernel $k(\mathbf{x}_n, \tilde{\mathbf{x}}_m; l) = \exp\big(-\|\mathbf{x}_n - \tilde{\mathbf{x}}_m\|_2^2/2l^2\big)$ with a heuristically chosen lengthscale l is used to compute the the feature vector $\phi(\mathbf{x}_n; l) = (k(\mathbf{x}_n, \tilde{\mathbf{x}}_1; l), k(\mathbf{x}_n, \tilde{\mathbf{x}}_2; l), ..., k(\mathbf{x}_n, \tilde{\mathbf{x}}_M; l)) \in \mathbb{R}^{M\times 1}$ for all data points $\{\mathbf{x}_n\}_{n=1}^N$. We can then say, $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ is the dataset and $\big\{l, \{\tilde{\mathbf{x}}_m\}_{m=1}^M\big\}$ is the pre-defined parameter set. Although random Fourier features or Nyström features can also be used for occupancy mapping, hinged kernel features are both highly intuitive and have shown improved performance for occupancy mapping (Ramos and Ott 2015) and related problems (Kingravi, Maske and Chowdhary 2016; Whitman and Chowdhary 2017).

Once the kernel based feature vector is calculated, we simply pass it through a sigmoidal function to predict an estimate of the *probability of occupancy* $\hat{y} = p(y|\mathbf{x}_*, \mathbf{w}) = 1/(1 + \exp(\mathbf{w}^\top \phi(\mathbf{x}_*; l)))$ of a query point in the space $\mathbf{x}_*$, given the weights $\mathbf{w} \in \mathbb{R}^{M\times 1}$. Since the query point is an arbitrary longitude-latitude coordinate in the real domain, HMs are able to produce maps with *arbitrary* resolution at prediction time – in contrast to OGMs which are limited to their chosen cell resolution. To learn the model weights, the loss function $\sum_{i=1}^N \log(1 + \exp(y_n \mathbf{w}^\top \Phi(\mathbf{x}_n; l, \tilde{\mathbf{x}}))) + \lambda_1 \|\mathbf{w}\|_2^2 + \lambda_2 \|\mathbf{w}\|_1$ with heuristically chosen penalty terms λ_1 and λ_2 is minimised. The problem with this setup is that it is requires a human expert is to choose the parameters l, $\tilde{\mathbf{x}}$, λ_1, and λ_2. One attempt to fix this (Senanayake and Ramos 2017) attempted to minimise this complexity by removing λ_1 and λ_2 which are responsible for over-fitting, the lengthscales of the kernel l and where to place them $\tilde{\mathbf{x}}$ were still heuristically chosen. While the model in (Senanayake and Ramos 2017) works effectively if l and $\tilde{\mathbf{x}}$ are chosen appropriately, the model is based on a fixed approximate lower bound which does not allow further modifications that would allow automatic hyperparameter discovery.

2.7 Summary

In this background chapter we have introduced the basics of learning predictive models with a brief tour through parametric model representations and learning procedures, using both analytic and approximate Bayesian estimation. We then proceeded to introduce the idea of a kernel, and the classical Gaussian process model which has kernels at its heart. We then shifted gears to the theory of optimal transport which will prove to be crucial for later exposition. Finally we gave a short introduction to probabilistic mapping in robotics which we will treat as a running applied sandbox for validating some of our contributions.

CHAPTER 3

Automorphing Kernels

The capacity of a robot to adapt on their own to changes in their environment is a key factor to the effectiveness of their deployment. An example of this is in mobile robotics where a robot is typically required to learn a map of its environment with minimal human assistance. In this chapter, we leverage the latest developments in automatic machine learning, also known as AutoML, and probabilistic programming, under the Hilbert mapping paradigm. The probabilistic representations that Hilbert mapping gives allows us to represent the occupancy of the environment as a continuous function of real world spatial location, and we construct an end-to-end Bayesian framework that allows us to learn all critical parameters of the map model. That is to say, the robot, by itself, is able to learn dataset-optimal location and shape hyperparameters of the the Hilbert map's kernel embeddings. In this way, we can lift the problem of picking kernel hyperparameters to a more general problem of specifying broader prior probability distributions over plausible hyperparameters without needing expert human domain knowledge. The methodology we introduce employs stochastic variational inference and is able to learn tens of thousands of parameters within minutes with both a large and small numbers observations. We additionally validate the proposed method on various real-world and simulated datasets in static and dynamic environments showing significant performance improvements over existing stationary occupancy mapping techniques. Perhaps the fundamental takeaway of this chapter is that we showcase the importance of learning data-dependent position-shape hyperparameters of kernels with respect to the observed data.

3.1 Introduction

In order to facilitate more effective and safer decision making for essential tasks like path planning, it is absolutely critical for a robot to have a faithful and informative model of its environment. A robot using sensors such as LIDAR or sonar must be able to discern occupied areas (where obstacles exist) from unoccupied areas (where obstacles are absent). The realisation of such occupancy states is a highly nonlinear and spatially correlated problem that cannot be expressed with a simple linear classification model. Thus, it seems appropriate that we approach this problem from a machine learning perspective where deep learning models and kernel-based models can be regarded as excellent candidates for occupancy mapping – for they are known to perform well in nonlinear classification settings. However, since it is often necessary to represent the sensed environment's occupancy with variably sparse-to-dense sensor measurements in a reasonable time, kernel methods have emerged as a natural choice in recent occupancy mapping methods (Ramos and Ott 2015; Doherty, Wang and Englot 2016). Multiple related theoretical and experimental works have corroborated their promising applications in 2D, 3D, and spatiotemporal mapping (Ramos and Ott 2015; Doherty, Wang and Englot 2016; Senanayake et al. 2016a; Guizilini and Ramos 2017a) – although often these rely on some expert-assisted heuristic parameter choices.

It can be said that perhaps the *essential challenge* in employing kernel methods in occupancy mapping is the requirement of accurately determining the kernel hyperparameters in conjunction with the model parameters (Senanayake and Ramos 2017). To move towards the ideal of robots achieving full autonomy in potentially unknown environments, with or without human or other-robot interaction, it would be ideal that the robot is able to automatically determine their own model parameters from observational data. In the real world, it is usually only the most simple environments that maintain spatially *homogeneous* features. Unfortunately, homogeneous environments are not the typical case in real-world environments. For example, walls and furniture may contribute to sharp features while open spaces and large hills may contribute to spatially smooth features. In our probabilistic mapping paradigm, to better understand the significance of representing nonstationarity in terms of kernels, first consider the squared-exponential kernel. This is a kernel which is parameterised with lengthscale

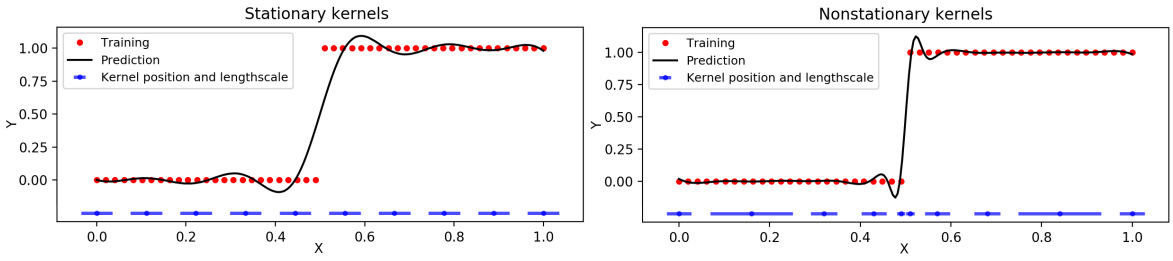


Figure 3.1. Comparison of stationary and nonstationary squared-exponential kernels, $\exp(-\|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2/2l^2)$ with bivariate Gaussian distributions $\tilde{\mathbf{x}}$ hinged on the environment with lengthscales l, and their ability to represent sharp spatial changes. Note that both examples have the same number of kernels, however in the nonstationary case the kernels have different positions and lengthscales to account for abrupt changes in the training data.

and position hyperparameters. As seen in Figure 3.1 larger lengthscales allow the capture of smoother changes across the space, while smaller lengthscales allow the capture of relatively sharper changes in the space. Hyperparameter optimisation is critical for almost all machine learning methods and the best values are almost always dependent on the dataset. Often, a single best lengthscale is chosen that performs, on average, the best for the entire dataset. There is also a computational constraint of the modelling method that restricts one from placing a large number of kernels around the space being mapped – this introduces the problem of where one should place kernels. In our contribution, we address both problems – where to place kernels and what lengthscales they should have. We maintain that these autodidactic and model adaptation paradigms are vital for a robot to achieve full autonomy.

Another critical consideration for real-world robot deployment is the acknowledgement that there is uncertainty inherent to all parts of system operation, from sensor and actuator imperfections to model misspecifications. Indeed the real-world is, to the best of our knowledge, analogue and noisy! Therefore it is subject to such observational imperfections. To accommodate for possibly encountered uncertainty, probabilistic formulations (Thrun 2000; Thrun, Burgard and Fox 2005) are widely adopted in robotic applications such as in simultaneous localisation and mapping (SLAM) (Dissanayake et al. 2001), occupancy grid mapping (Elfes 1987), Bayesian occupancy mapping (Kim, Kim et al. 2014; Kim and Kim 2013; Duong, Yip and Atanasov 2022), and human-robot interaction (Campbell and Amor 2017; Unhelkar et al. 2018; Rana et al. 2017). While these models largely exploit Bayes theorem to determine

inverse probabilities assuming the conjugacy of prior-posterior probability distribution pairs (Bishop 2007), they usually do not leverage a more general Bayesian treatment of introducing probability distributions over all parameters. This is, in essence, a cleaner way of dealing with potentially diverse sources of uncertainty. Bayesian models are particularly suitable for small data settings (Gelman et al. 2013), largely due to the assumptions often encoded in choice of prior distribution and their natural regularising effect. It follows naturally from this that their benefit in robotics should apparent – they provide a principled way to accommodate prior knowledge, which is usually derived from expert human knowledge. Although these upsides seem attractive, there are still significant drawbacks to Bayesian methods which are often hindered by i) the necessity for often tedious and problem-dependent mathematical derivations, and ii) such models are often unable to scale or adapt to real-world robotics applications. All of this said, recent developments in AutoML methods in machine learning can alleviate some of these issues which makes them attractive for use in a variety of applied robotics problems such as kernelised continuous occupancy mapping.

In the method proposed in this chapter, we show how the the reparameterisation trick (Kingma and Welling 2013) along with stochastic gradient descent is a perfect candidate to help solve the challenging learning problem of determining parameters for Hilbert maps in probabilistic robotic mapping. Specifically, we show how it can solve the previously intractable problem of automatically learning kernel parameters that plagued previous Hilbert mapping techniques that have required human chosen heuristic values for the kernel parameters.

3.2 Contributions

This chapter focuses on the problem of scalable and automatic hyperparamter learning for nonstationary kernel representations. We specifically tackle kernel learning in the application domain of probabilistic robotic mapping for which prior work has only been capable of automatically learning model parameters but not kernel parameters in a scalable way. To address this problem, we present the following contributions:

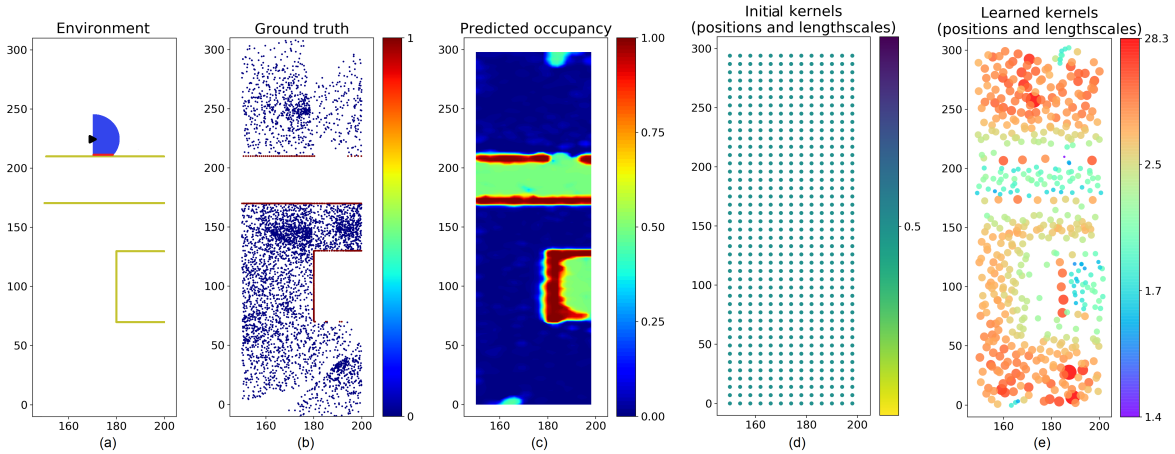


Figure 3.2. (a) A 50×300 m section of a simulated environment with obstacles in yellow. A robot shown as a black arrow has a LIDAR with beams shown in blue and the laser hit points in red. (b) The robot moves around and collects data. Red points are laser hit points and blue points are samples taken from LIDAR beams between the robot and laser hit points (c) The occupancy probability map. Red indicates occupied space, blue indicates free space, and colors in-between indicates the uncertainty of occupancy. (d) The map is built based on a set of squared-exponential kernels. The mean of the initial bivariate Gaussians is shown here—Gaussians are in a grid. (e) The proposed algorithm can learn both kernel parameters l and positions $\tilde{\mathbf{x}}$ alongside other model parameters. Both the color and the size of the marker indicates the size of the learned lengthscales. For instance, larger lengthscales are shown in a bigger marker size and in red.

(1) Proposing a theoretical framework that works well in practice to learn all parameters in Hilbert maps in both static and dynamic environments,

(2) Learning kernels to account for nonstationary and nonlinear patterns,

(3) Proposing the use of low-discrepancy sampling in robotic mapping,

(4) Demonstrating the importance of using complex Bayesian formulations for uncertainty representation in robotics and learning thousands of parameters in both small and bigdata settings without laborious mathematical derivations, and

(5) A thorough analysis of known critical factors that affect Hilbert mapping.

3.3 Related Work

Kernel methods have been used extensively in robotics. Predominantly we see their application when the objective requires representation of nonlinear patterns with small amounts of data (Mukadam, Yan and Boots 2016; Deisenroth and Rasmussen 2011; Kingravi, Maske and Chowdhary 2016; Kingravi et al. 2012). In robotic mapping, one typically sees only Squared Exponential kernels that have fixed (but hand tuned) lengthscales (Ramos and Ott 2015; Doherty, Wang and Englot 2016; Senanayake and Ramos 2017).

A variety of kernel learning (see Section 2.4.7) techniques have been previously discussed in the machine learning literature, with arguably the largest represented proportion existing in the Gaussian process literature. The selection of kernels is typically done through expert human knowledge (Duvenaud et al. 2013), a model selection criteria such as Bayesian information criteria (Duvenaud 2014), or expensive optimization procedures (Bach, Lanckriet and Jordan 2004).

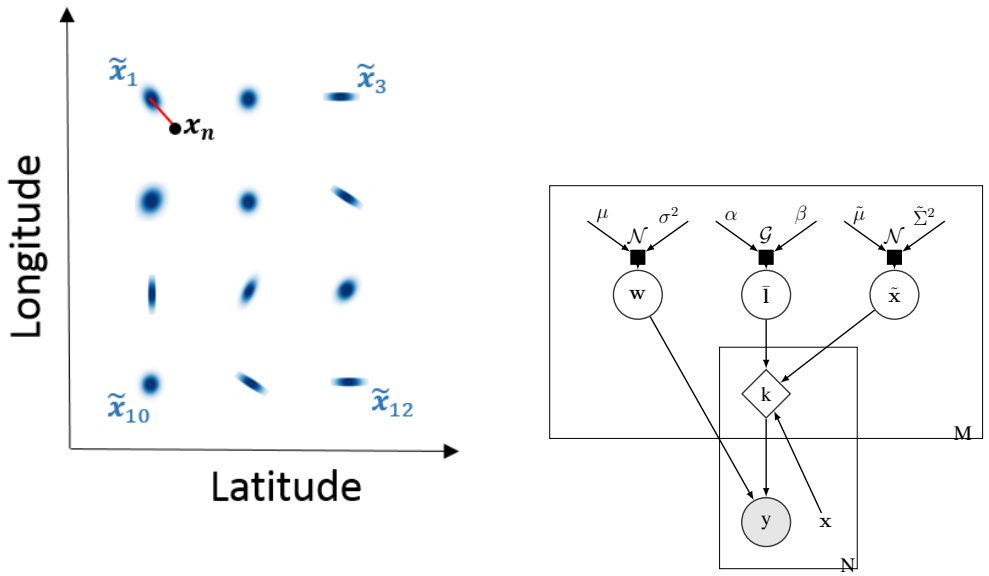
It is also possible to compose kernels as a sum or a product of previously defined kernels (Duvenaud et al. 2013), or as representing them as a spectral mixture in the frequency domain (Wilson et al. 2013). However, unlike in Gaussian process where optimizing the hyperparameters is well-studied and readily available through the log marginal likelihood, directly learning parameters online in a classification setting is not straightforward in HMs.

3.4 Automorphing Kernels for Nonstationary Hilbert Mapping

It is known that kernel methods are well suited for occupancy mapping, if parameters are appropriately set (Ramos and Ott 2015; Doherty, Wang and Englot 2016). In this section, we propose novel techniques for mapping unstructured environments without a human explicitly

*The gamma distribution is defined as $\mathcal{G}(x;\alpha,\beta) := \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1}e^{-\beta x}$ where $\Gamma(\alpha) := \int_0^\infty z^{\alpha-1}e^{-z}\mathrm{d}z$ is the gamma function. $\alpha > 0$ is the shape parameter and $\beta > 0$ is the rate parameter. In literature, the scale parameter is sometimes defined as the inverse of the rate parameter instead of the rate parameter.

*The log-normal distribution $\mathcal{LN}$ is obtained by transforming a standard normal variable $z' = \exp(\mu + \sigma z)$.



(a) Feature vector computation. $\{\tilde{\mathbf{x}}\}_{m=1}^{M=12}$ are hinge distributions and $\mathbf{x}_n$ is the n^{th} data point.

(b) The graphical model. k represents the kernel which is evaluated $N \times M$ times.

Figure 3.3. Description of model parameters. Assuming independence, individual distributions are associated with all hinge points $m = 1...M$

Var.	Distributions
Prior distributions:	
w_m	$\mathcal{N}(\mu_m, \sigma_m^2)$
$\bar{l}_m$	$\mathcal{G}(\alpha_m, \beta_m)$ *
$\tilde{\mathbf{x}}_m$	$\mathcal{N}(\tilde{\mu}, \tilde{\Sigma}^2)$
Variational distributions:	
qw_m	$\mathcal{N}(\mu_m^{(q)}, \sigma_m^{2(q)})$
$q\bar{l}_m$	$\mathcal{LN}(\alpha_m^{(q)}, \beta_m^{(q)})$ *
$q\tilde{\mathbf{x}}_m$	$\mathcal{N}(\tilde{\mu}^{(q)}, \tilde{\sigma}^{2(q)})$

providing hyper-parameters. Firstly, we propose different positioning techniques for hinging kernels in Section 3.4.3. Then, we discuss the importance of nonstationary learning (Paciorek and Schervish 2004) in occupancy mapping. That is, rather than having a single lengthscale for all kernels as in (Ramos and Ott 2015; Senanayake and Ramos 2017), kernels should have

different parameters, depending on where they are placed in the space. For instance, with regards to SE kernels, kernels should have smaller lengthscales close to walls in order to capture sharp transitions from occupied to unoccupied. Similarly, bigger lengthscales are expected in largely unoccupied or unobserved areas. With the motivation from Section 3.4.3, we propose a technique to simultaneously learn the position and kernel parameters in Section 3.4.

Although the aggregation method proposed in Section 3.4.3 partially accounts for nonstationarity, it requires a set of predefined parameters. As with other continuous mapping techniques, the method cannot learn where to place kernels. However, for a robot to adapt to changes in unstructured environments, it is crucial to take the human out of the loop of parameter tuning. In the following sections, without loss of generality to other kernels, we explain using SE kernels to solve these issues.

3.4.1 Model Specification

In this section, as the main contribution of this chapter, we propose a principled approach to provide two traits of adaptation to kernels in Hilbert maps: plasticity and mobility. That is, both the shape l of the kernel and its locations $\tilde{\mathbf{x}}$ can be learned alongside feature weights $\mathbf{w}$ under the proposed framework. Further, as shown in Figure 3.2e, rather than considering a single lengthscale as in previous work (Senanayake and Ramos 2017) or a small set of lengthscales as in Section 3.4.3, the new technique can not only learn any lengthscale in $\mathbb{R}$, but also the kernels associated with each hinge location has its own local lengthscale. These individual lengthscales $\{l_m\}_{m=1}^{M}$ essentially model the nonstationary behavior and can easily acclimatise to local changes in the environment.

Since observed occupancy values are always binary and they are independent of each other, we assume the likelihood follows a Bernoulli distribution $p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \mathbf{l}, \tilde{\mathbf{x}})$ where $\log(\theta/(1-\theta))) = \mathbf{w}^\top \Phi(\mathbf{x}; \mathbf{l}, \tilde{\mathbf{x}})$. Note that this is equivalent to logistic regression (Bishop 2007). The prior distributions over weights are defined as Gaussian distributions. Since the hinge locations can be anywhere in the space, they are also defined as Normal distributions. As shown in Section 3.4, kernel functions are now implicitly evaluated between datapoints and hinge

distributions, naturally accounting for uncertainty. A meaningful lengthscale can only be positive and hence the prior distribution over inverse squared-lengthscales $\bar{l} = 1/2l^2$ is defined as a gamma distribution. The first three rows of Figure 3.3 provides a summary of all variables. Here, for computational efficiency we assume all variables are independent. As these prior distributions were empirically sufficient for modelling occupancy, we do not complicate the model with hyper-prior distributions.

Our objective is to learn the posterior distribution: parameters conditioned on data. However, because of the Bernoulli likelihood, the posterior is intractable and hence is approximated using another distribution q. Indicating longitude and latitude with lon and lat, respectively, the basic formulation with mean-field variational approximation is given in Section 3.4 and the following equation,

$$\underbrace{\prod_{m=1}^{M} q(w_m)q(l_m^{\text{lon}})q(l_m^{\text{lat}})q(\tilde{\mathbf{x}}_m)}_{\text{factorised variational distribution}} = \underbrace{q(\mathbf{w},\mathbf{l},\tilde{\mathbf{x}})}_{\substack{\text{variational}\\ \text{distribution}}} \approx \underbrace{p(\mathbf{w},\mathbf{l},\tilde{\mathbf{x}}|\mathbf{x},\mathbf{y})}_{\text{posterior}} \propto \underbrace{p(\mathbf{w})p(\mathbf{l})p(\tilde{\mathbf{x}})}_{\text{priors}} \underbrace{p(\mathbf{y}|\mathbf{x},\mathbf{w},\mathbf{l},\tilde{\mathbf{x}})}_{\text{likelihood}}.$$

3.4.2 Model Learning

The posterior distribution defined in Section 3.4.1 is intractable due to the Bernoulli likelihood. Since distributions over lengthscales and positions are introduced in addition to distributions over weights, obtaining a maximum a posteriori (MAP) estimation is not feasible even with the lower bound derived in (Jaakkola and Jordan 1997). As occupancy mapping is a very high dimensional problem, obtaining the posterior using Markov chain Monte Carlo (McMC) techniques is costly (Bishop 2007). As an alternative, we use variational inference (VI) with the reparameterisation trick (Kingma and Welling 2013). Although there are other alternatives such as VI with stochastic search (Paisley, Blei and Jordan 2012) and Hamiltonian Monte-Carlo (Neal 1993), we used the well-established method (Kingma and Welling 2013) as it can easily perform stochastic gradient descent (SGD) with minibatches. Rather than deriving the lower-bound for the specific case, as an AutoML technique, we made use of probabilistic programming (Tran et al. 2017) to minimise the Kullback-Leibler (KL) divergence between the variational distribution and posterior $\mathbb{KL}(q\|p)$. For this reason, the proposed model is

easily amenable for extensions. The choice of distributions is detailed in Figure 3.3. To keep variances of variational distributions non-negative a softplus transformation was applied.

3.4.3 Various Initialisations for Hinged Kernels

Rather than initialising hinge points randomly in the space, it is possible to place them based on quasi-random locations in order to guarantee a more uniform spread of kernels in the space. This is done by using a low-discrepancy sequence (Dick, Kuo and Sloan 2013). Some of the commonly used Quasi-Monte Carlo (QMC) sampling techniques include Halton, generalised Halton, and Sobol sequences (Dick, Kuo and Sloan 2013; Yang et al. 2014).

SE kernels with a fixed lengthscale have been the *de facto* choice in Hilbert mapping (Ramos and Ott 2015; Doherty, Wang and Englot 2016; Vallicrosa and Ridao 2018; Senanayake and Ramos 2017). We propose to hinge multiple kernels with varying properties on every hinge location $\tilde{\mathbf{x}}_m$ rather than hinging a single kernel. For instance, it is possible to hinge a set of SE kernels $\{k(\cdot, \tilde{\mathbf{x}}_m; l_r)\}_{r=1}^{R}$ with R different lengthscales l_r. More broadly, these can also be a set of R different kernel types $\{k_r(\cdot, \tilde{\mathbf{x}}_m; \theta_r)\}_{r=1}^{R}$ with θ_r parameters corresponding to the the r^{th} kernel. In addition to the hackneyed SE kernels, this pool of kernels can consist of Matérn kernels, rational quadratic kernels, etc. A spectral mixture of kernels (Wilson and Adams 2013) is also another choice.

Adding hinged kernels increases the dimensionality of the feature vector by R times. For Φ_r feature vectors individually computed on M hinge points as in Section 2.6.2, define the aggregated feature vector $\Phi_\Sigma = \|_{r=1}^{R} \Phi_r \in \mathbb{R}^{1\times RM}$ with $\|$ indicating vector concatenation. The classification model is $\hat{y} = p(y|\mathbf{x}_*, \mathbf{w}_\Sigma) = 1/(1+\exp(-\mathbf{w}_\Sigma^\top \Phi_\Sigma))$ with $\mathbf{w}_\Sigma \in \mathbb{R}^{RM\times 1}$. This is equivalent to joining R individual sets of model weights $\mathbf{w}_r \in \mathbb{R}^{M\times 1}$ as $\sum_{r=1}^{R} \mathbf{w}_r^\top \Phi_r$ before the sigmoidal transformation.

Table 3.1. Description of the datasets.

Dataset	Real	Dynamic	Description
1	✗	✗	A $600 \times 300\text{m}^2$ area (Senanayake and Ramos 2017). This is a simple but large environment.
2	✓	✗	Intel lab dataset: a complex indoor environment.
3	✗	✓	Vehicles move in two directions with robot in the middle (Senanayake and Ramos 2017).
4	✓	✓	LIDAR dataset in a busy intersection (Senanayake and Ramos 2017).

3.5 Experiments

We conducted a series of experiments on four different datasets given in Table 3.1. These datasets contain both static and dynamic environments. As with (Senanayake and Ramos 2017; Senanayake, O'Callaghan and Ramos 2017), our model will estimate the average long-term occupancy which is different to mapping short-term occupancy (Senanayake et al. 2016a) or removing dynamics to build a static occupancy map (Meyer-Delius, Beinhofer and Burgard 2012; Stachniss and Burgard 2005). When demonstrating basic concepts and observations, a portion of dataset 1 is used as it is simple and easy to visualise. We used TensorFlow with the Edward library (Tran et al. 2016) to program. Demonstrations can be found at https://github.com/MushroomHunting/automorphing-kernels.

3.5.1 Experiment 1: Effect of Kernel Aggregation

As the first experiment, to verify that capturing nonstationarity is important, we hinged three SE kernels with $l = 5, 1, 0.1$ on fixed locations. We observe that by changing the lengthscale of all kernels we can observe differently learned weights – this indicates the requirement of learning kernel parameters depending on where they are in the environment.

3.5.2 Experiment 2: Effect of Hinging Techniques

In order to demonstrate the effect of positioning kernels, we learn the map for a fixed lengthscale chosen from five-fold cross-validation that minimises AUC. We consider positioning kernels on a regular grid, MC samples, and three QMC sampling techniques—Halton, generalised Halton, and Sobol. The number of hinge kernels was set to 222. As shown in the

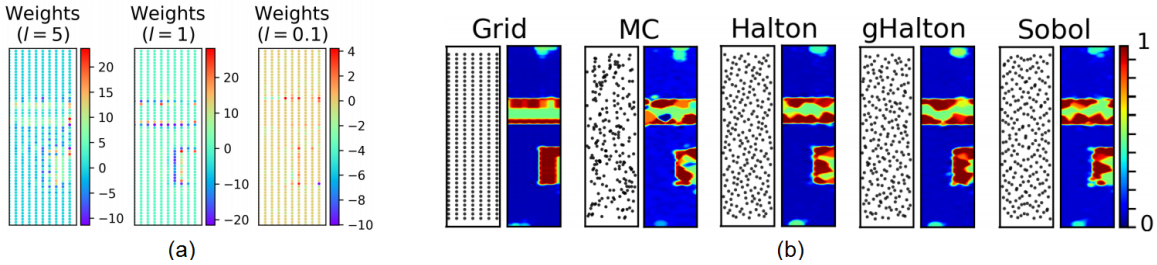


Figure 3.4. These images are based on a simulated dataset. (a) Kernel aggregation in experiment 1 (b) Positioning kernels in HMs with fixed position and lengthscales for different hinging schemes.

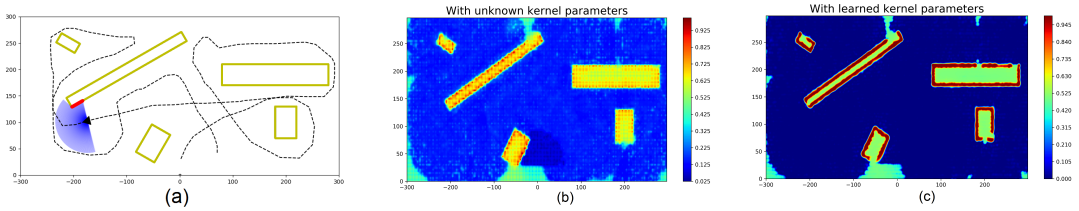


Figure 3.5. (a) Environment and entire dataset 1 (b) The averaged occupancy map of BHM with a random set of lengthscales (c) Predicted occupancy map using ABHM.

Figure 3.4, unlike the QMC techniques, MC sampling tends to make clusters causing the Hilbert maps to be less accurate. Therefore, if kernel positions are not learned, they should be placed either quasi-randomly or on a regular grid.

For the intel dataset, as shown in Figure 3.8, we do further analysis to see the effect of the number of samples trade off. Interestingly, the time increases almost linearly with the number of features while the accuracy does not significantly improve after a certain number of features.

3.5.3 Experiment 3: Effect of Learning Parameters

This experiment was designed to validate the main contribution of the method—learning lengthscales and hinge locations. The learned environments for different datasets are shown in Figure 3.5 and Figure 3.6. To understand the full effect of the proposed model it is not enough to look at the predicted occupancy map—we must consider the underlying distributions. Figure 3.7 provides a visual map of the means and variances of a learned model's variational

posteriors. Accounting for a large part of the upper and lower parts of the map, the position variance in Figure 3.7b shows that in areas of dense laser scans where no walls exist, a larger but uniform variance for each spatial dimension is learned. For the areas where the laser scanner has detected walls one observes a stark contrast exhibited by the smaller spatial variances. In the walled area spanning the middle of the map the learned variances in the latitudinal direction are stretched out further relative to the longitudinal direction reflecting the narrow corridor-like shape of the wall. Concerning now the lengthscale mean and variance

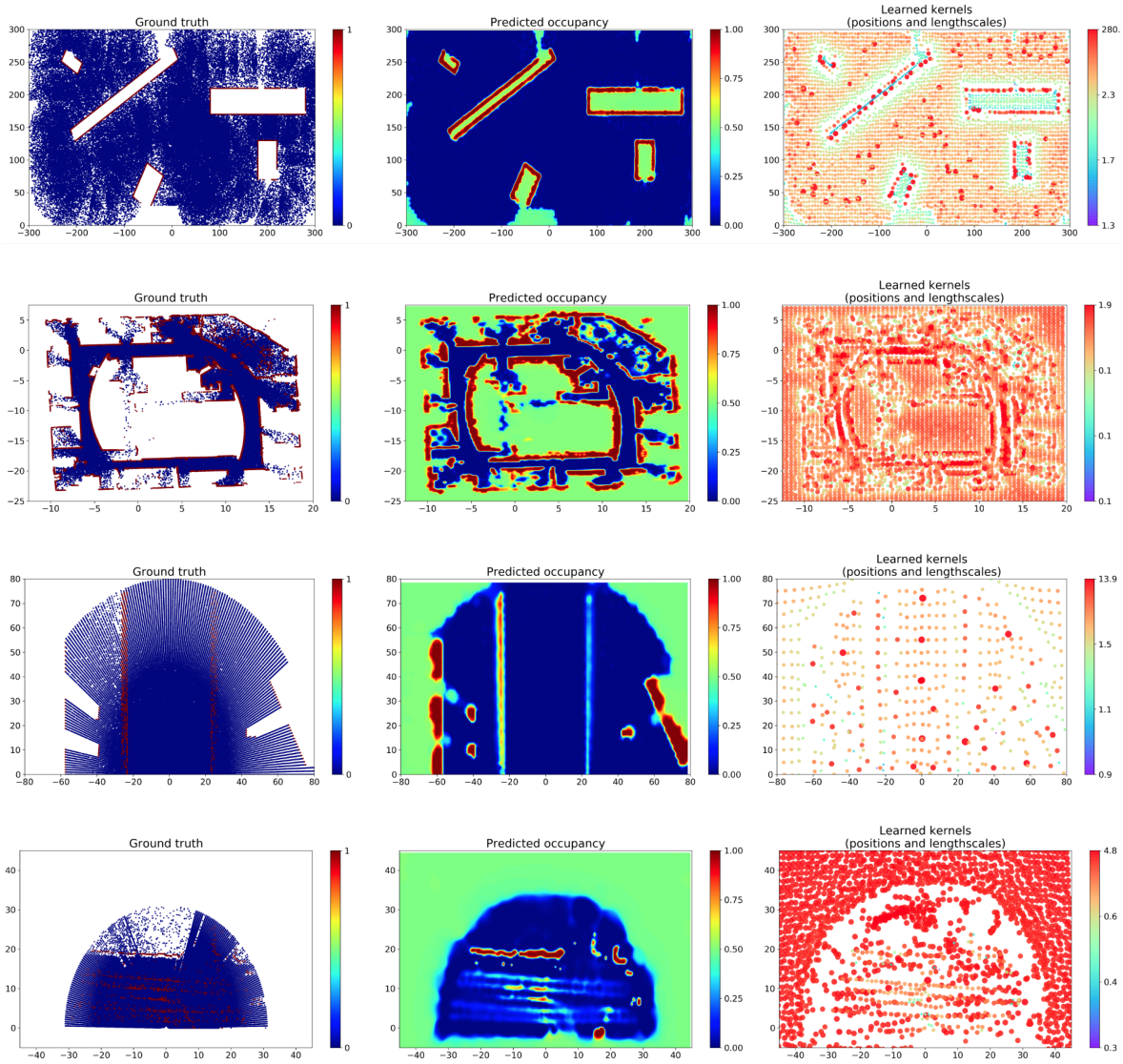


Figure 3.6. Left: laser scans, middle: predicted occupancy, right: learned kernel positions and lengthscales

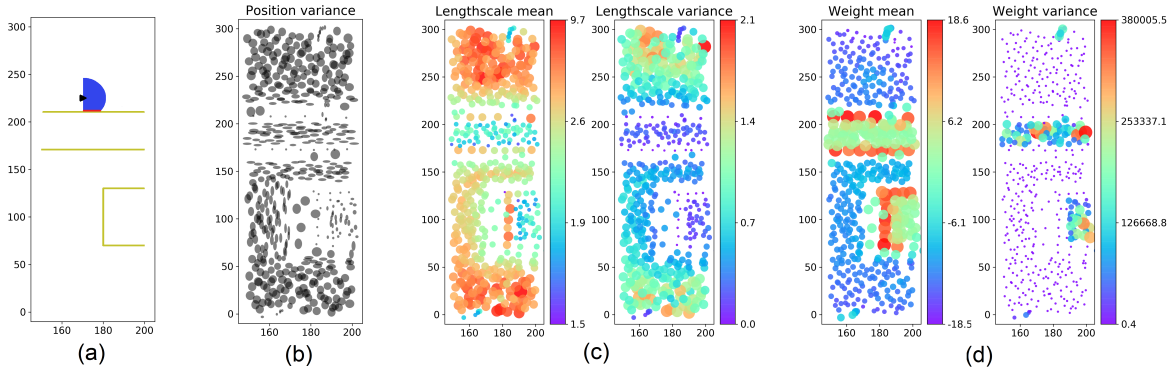


Figure 3.7. Uncertainty plots (a) A portion of the environment (b) Positions of hinge kernels $\tilde{\mathbf{x}}$ (c) Lengthscales (d) Weights

in Figure 3.7c we can observe the most significant effect in terms of the learned posteriors. At the top and the bottom open areas the largest lengthscales are observed signifying a minimal complexity of occupancy. Paralleling the learned position variances, the learned lengthscale means are clustered around either areas of detail or areas of uncertain occupancy. This effect is repeated in the lengthscale variance.

The kernel weights means and variances are depicted in Figure 3.7d where one can see the highest weights appear around areas associated with the smallest position and lengthscale variances. Contrastingly, the most negative weights appear in regions of highly confident predicted empty occupancy. The weights closest to zero occur in areas of the map the robot has no visual perception and these constitute the insides of walls. The effect of the weight means is reflected in the weight variance where areas of high observability, which include open spaces and walls, have a low uncertainty in their estimates. Areas of low observability, i.e. inner parts of walls, have extremely high variances. This underlying analysis of the learned posterior distributions not only substantiates the motivation for spatially adaptive kernel learning, but also gives an explainable and intuitive understanding of what the model has learned which is often critically important for robotic tasks that interact with real-world environments.

Using all four datasets, the area under curve (AUC) and mean negative log loss (MNLL) were calculated. As reported in Table 3.2, these metrics were also calculated for occupancy grid maps with dynamic updates (DOGM), variational sparse dynamic Gaussian process occupancy

Table 3.2. Experiment 3: Losses on all real datasets. The higher the area under curve (AUC) or the lower the mean negative log loss (MNLL), the better the model is.

Method	Dataset 1		Dataset 2		Dataset 3		Dataset 4	
	AUC	MNLL	AUC	MNLL	AUC	MNLL	AUC	MNLL
ABHM	**0.999**	**0.015**	**0.994**	**0.093**	**0.993**	**0.175**	**0.889**	**0.477**
BHM	**1.000**	0.176	0.921	0.362	0.990	0.280	0.825	0.570
HM	0.992	0.226	0.938	0.666	0.920	0.903	0.778	0.677
VSDGPOM	0.801	0.372	0.794	0.530	0.990	0.233	0.788	0.886
DOGM	0.792	0.593	0.901	0.744	0.980	0.495	0.779	3.449

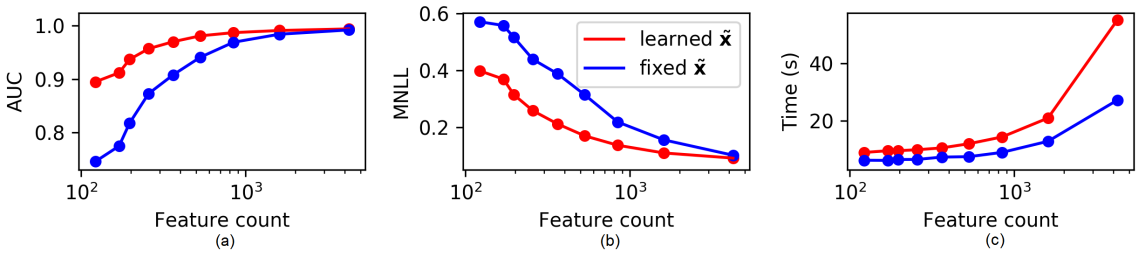


Figure 3.8. Performance vs. number of features for dataset 2. The blue lines show performance for fixed hinge positions while the red lines show the full ABHM model.

maps (VSDGPOM) (Senanayake, O'Callaghan and Ramos 2017), HMs, and Bayesian Hilbert Maps with sequential updates (BHM). The best lengthscales for previous Hilbert mapping techniques were determined using five-fold cross validation. Even when compared with hand-crafted features, ABHM outperforms. This is because it models nonstationarity and can capture subtle changes. For dataset 1 which has straight boundaries, the AUC value of both BHM and ABHM are comparable. However, ABHM outperforms in complex datasets such as in dataset 2 and dynamic environments such as in datasets 3 and 4. This is because only ABHMs can adjust the position and shape of kernels to locally adapt to environments.

To further understand the relationship between the performance and number of hinge points, we analysed the speed time and accuracy for dataset 2. We did this by, 1) learning both lengthscales and position, and 2) learning only the lengthscale keeping the kernels hinged on a grid. As shown in Figure 3.8, to achieve the same level of accuracy, only a smaller number of features is required when learning both the lengthscale and position.

Runtime: All experiments were conducted on a computer with a GTX1080 Ti 11 GB. For datasets 1 and 2, on average it takes around 10 minutes to learn all parameters. Note that this is to learn upwards of 57,600 parameters (8 parameters per hinge with more than 7200 hinges) and 300,000 data points. In contrast, (Senanayake and Ramos 2017) has an inevitable computational complexity $\mathcal{O}(M^3)$ while the proposed method uses stochastic gradient descent (SGD). Although analysing the theoretical asymptotic complexity is not straightforward, it linearly increases with M and N empirically. In ABHM, we take the advantage of SGD to scale effectively to significantly larger datasets.

3.6 Summary

In this chapter we have presented a new methodology for representing and learning probabilistic representations of kernel and model parameters in the domain of probabilistic robot mapping. The formulation takes advantage of mean-field variational inference under a probabilistic programming framework in order to learn essential parameters using gradient descent. We showed that the method produces more accurate representations of the target distribution and removed the necessity for hand-tuning kernel parameters that was required by previous methods. Furthermore, by exploiting the latest AutoML techniques we show it is possible to learn complex models without relying on explicit mathematical derivations.

We demonstrated the effectiveness of the ABHM formulation on a comprehensive set of experiments including both simulated and challenging real-world data. The results demonstrate that since the proposed method performs exceptionally well by taking advantage of data-specific nuances that the learned kernels were able to discover. While the predictive performance is exceptional, the *overall* procedure is prohibitive to operation in real-time which is a necessary condition for integration into real-world systems. This is the problem to which we will focus on, and provide a solution to, in the next chapter.

CHAPTER 4

Parameter Optimal Transport for Online Kernel Domain Adaptation

---

In this chapter we build on work from Chapter 3, aiming to extend the capabilities that end-to-end learning brought to the nonstationary kernel learning problem. Specifically, we solve a critical but problematic side-effect that the flexibility of probabilistically learning kernels brought: performance. In doing so, we take a short detour through the theory of *optimal transport* which is concerned with moving *mass* something from one location to another in an energy preserving manner. The consequence of framing the problem of kernel learning through the lens of optimal transport allows us to perform nonstationary kernel *transport* in real time. That is to say, instead of performing lengthy end-to-end learning, we can transport prior learned models in an online fashion circumventing a lengthy sequential optimisation problem into multiple short and parallelised optimisations problems.

4.1 Introduction

As we have seen earlier, accurate spatial representations that can accommodate observation and model uncertainty are a fundamental concern for autonomous robots to safely navigate in unstructured environments. We have also seen, in the previous chapter, that while LIDAR based mapping techniques can learn excellent maps, the required compute of these models discourages their real-world applications in tasks, such as autonomous driving, where they would be most valuable. In this work, we make the observation that real-world structures exhibit similar fundamental geometric features across a variety of urban environments and exploit this to accelerate learning. We show that it is unnecessary to re-learn all geometry dependent parameters from scratch. To this end, this chapter introduces a theoretical framework that

builds upon the theory of *optimal transport* in which we propose the *observation-conditioned adaptation of model parameters* which accounts for the observed environment geometry. Experimentally, we demonstrate the new parameter adaptation paradigm on 2D and 3D occupancy maps derived from both high-fidelity driving simulators and real-world datasets. We additionally consider various domain adaptation paradigms that include inter-domain feature transfer and simulation-to-real feature transfer. Our experimental validation verifies the possibility of adapting parameters with comparatively negligible compute and memory cost.

As discussed previously, the ability for a robot to adapt to continuously changing environments is a hallmark of its real-world utility. If our model is represented as a parameterised statistical model, it is reasonable that its parameters should be able to adjust to new observations. In a sense this ability to adapt to its domain is essential to performing effectively in real-world settings where the robot is often required to interact with humans, other robots, or simply with the changing environment.

One crucial issue that hinders adaptability is if the robot's adaptation process is slow in the sense of being unable to perform its required tasks. One cause of this is if the learning or adaptation process is computationally expensive, then adapting parameters in real-time becomes a challenge. We can observe this in all kinds of models such as in deep learning and

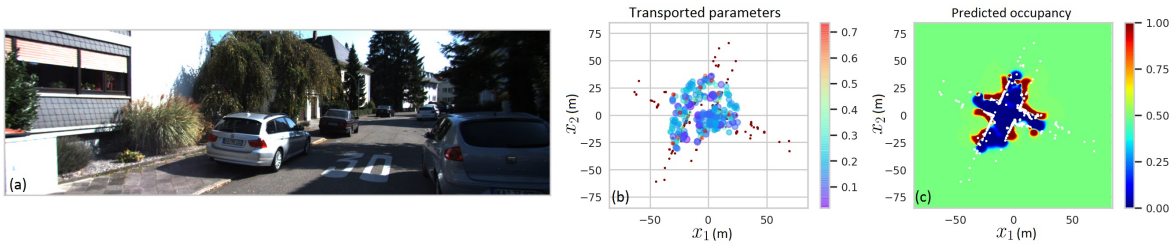


Figure 4.1. (a) Forward camera-view from a car that has just passed an urban intersection (KITTI dataset). (b) A set of occupancy model parameters estimated using the proposed Parameter Optimal Transport (POT) method. Values of these parameters depend on the geometry of the environment. Note that these parameters were *transferred online* from a simulated environment and were *never learned from scratch*. (c) Mean occupancy map obtained from the transferred parameters.

even Bayesian techniques. Many attempts have been made to enable adaptation in neural-network models (Ammar et al. 2015; Sadeghi et al. 2017; Finn, Abbeel and Levine 2017; Fang et al. 2018; Wulfmeier, Bewley and Posner 2018), yet this is a relatively under-explored area in Bayesian modelling (Meier, Hennig and Schaal 2014) – despite their wide applicability in robotics (Deisenroth and Rasmussen 2011; Wüthrich et al. 2016; Campbell and Amor 2017; Burchfiel and Konidaris 2017; Unhelkar and Shah 2018). Since uncertainty is a natural byproduct of probabilistic representations in Bayesian models, it stands to reason that we must therefore adapt *distributions* themselves to new domains. And so we can ask: how could we solve the problem of efficient distributional adaptation without being burdened by requiring retraining models from scratch? In this chapter, we focus on answering this question in the kernel learning domain by transferring kernel and model parameters, online, and in a zero-shot fashion (Isele, Rostami and Eaton 2016). The practical consequence we explore is how we can adapt kernel based occupancy models to unknown environments without having to train models from scratch.

The fundamental techniques developed in this chapter are broad enough to transfer to other applications that require timely domain adaptation capabilities. Most directly, any kernel modelling technique in which observed data is coupled in a true geometric sense will be able to benefit – e.g. data-efficient robot perception and planning, or even spatio-temporal modelling. Our focus for the time being will remain with continuous mapping methods for arbitrarily large environments with an emphasis on online operations. The proposed formulation will be applied to the kernel mapping method introduced in the previous chapter: automorphing Bayesian Hilbert maps (ABHMs). While ABHM is a theoretically rich method, it is impractical to apply in real-time due to the large compute cost required to train the model. ABHM produces probabilistic (uncertainty aware) estimates of its sensed environments making it applicable to domains that must recognise possible environmental uncertainties (Lasota, Fong, Shah et al. 2017) like autonomous driving and motion planning that requires one to be risk-aware in the decision making process (Akametalu et al. 2014; Vallicrosa and Ridao 2018).

Perhaps the core reason that prohibits most learning methods, concretely exhibited in the previous chapter, is the run-time cost of having to learn from what could be described as a uniform prior. We observed a similar situation with ABHM which required a black-box variational setup that required a lengthy optimisation procedure to obtain idealised parameter estimates. In ABHM we could observe that the parameters are collocated with the geometrical features of the sensed objects in the natural environment (LIDAR hits and misses). Consequently, ABHM required a large-batch assumption and learning spatially dependent parameters for *every* spatial location. This could be problematic for more complex tasks that involve dynamic environments in which parameters would need to adapt in real-time to accommodate changing situations.

We therefore propose to transfer *geometry-dependent spatial features* from a pre-existing set of models to new unobserved test scene, while 'geometrically conditioning' on the spatial observations. This method fundamentally contrasts to having to relearn parameters from scratch by using novel re-interpretation of the problem formulation using the theory of optimal transport (Villani 2008; Arjovsky, Chintala and Bottou 2017; Solomon et al. 2015). The proposed approach completely skips the need to explicitly learn parameters of the statistical model which are conventionally learned through a differentiable log-likelihood based optimisation. As we show in Figure 4.1, the algorithm *transports* location and geometry-dependent parameters of the model from one place to another place by aligning geometric similarity between new observations and prior observations. In the context of kernel-based occupancy mapping, this *parameter transport* procedure is shown to successfully construct geometry-dependent kernels with less computational cost than required by existing methods.

4.2 Contributions

This chapter focuses on the problem of online hyperparameter estimation for nonstationary kernel models in the context of probabilistic robotic mapping. Prior methods have approached the learning problem from global perspective which requires singular, lengthy optimization procedures to infer optimal hyperparameters. While such methods can learn near-optimal

models, they are severely handicapped by the time for optimization to achieve convergence. To address this fundamental problem we view the problem from the lens of domain adaptation and in doing so present the following contributions:

(1) a theoretical framework for parameter transfer in robotics using the mathematical framework of optimal transport;
(2) intra-domain transfer: sequentially building a map based on features learned in previous time frames;
(3) inter-domain transfer: mapping an environment with features learned from another environment. This includes parameter transfer from one town to another, static to dynamic environments, and simulation to real-world; and
(4) online and efficient mapping of large-scale 2D and 3D environments.

4.3 Preliminaries

Notation given in Table 4.1 will be used throughout this chapter.

Table 4.1. Table of notations and terminology

Notation	Description
$^{-}$and$^{\smile}$	Mean and variance of Gaussian; shape and scale of Gamma
$\mathbf{x}$ and y	LIDAR data positions and labels
N and M	Number of data points and number of parameters
$\bar{\mathbf{h}}$	Kernel positions
θ	Parameter set except $\bar{\mathbf{h}}$
$^{(\mathcal{S})}$ and $^{(\mathcal{T})}$	Source and target
P	Coupling matrix
$a \rightarrow b$	transport = transfer = domain adaptation = transform = map = convert (from a to b)

4.3.1 Uncertainty of Occupancy

An occupancy model is typically represented as a parameterised function that models the occupancy probability of each location in the environment. The objective is to learn the

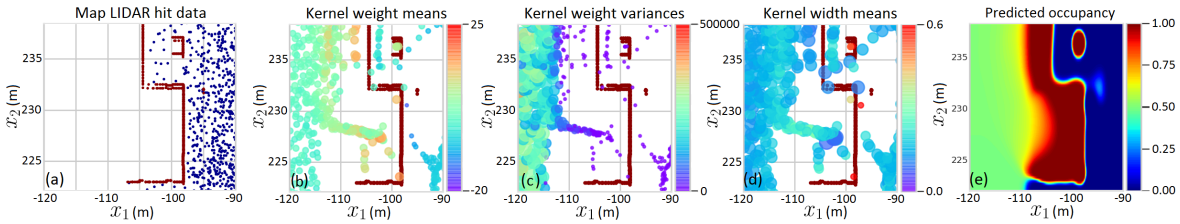


Figure 4.2. Spatial correlation among obstacles in the environment and some ABHM parameters. (a) LIDAR data: $y = 1$ (hits) in red and $y = 0$ in blue. (b)-(d) kernel weight means $\bar{w}_m$, weight variances $\breve{w}_m$, and width means $\bar{\gamma}_m$. Each point is a kernel placed in the shown location $\bar{\mathbf{h}}$ in the x_1-x_2 space. Refer Observation 1 for further interpretation. (e) Predicted occupancy.

model parameters θ given a set of observations from LIDAR beams. Once the parameters are estimated, it is possible to query $y_* = p(\text{occupied}|\mathbf{x}_*, \theta) \in [0, 1]$ anywhere in the 2D space* $\mathbf{x}_* \in \mathbb{R}^2 := (x_1, x_2)$. Labeling LIDAR hits as $y = 1 = \text{occupied}$ and randomly sampled points between each LIDAR hit and the LIDAR sensor as $y = 0 = \text{free}$, a dataset $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ can be generated. Here, $\mathbf{x}_n \in \mathbb{R}^2$ are the corresponding spatial locations of $y_n \in \{0, 1\}$.

Various models have been proposed for the occupancy function. Gaussian process occupancy maps (GPOMs) (O'Callaghan and Ramos 2012; Wang and Englot 2016) have been presented as an alternative to improve occupancy grid mapping (OGM) (Elfes 1989; Arbuckle, Howard and Mataric 2002) and Hilbert maps (Ramos and Ott 2015). In addition to considering neighborhood information for accurate occupancy predictions, kernel methods used in GPOMs come with the flexibility of incorporating other aspects such as dynamics into occupancy mapping (Senanayake, O'Callaghan and Ramos 2017; Senanayake et al. 2016b). On the other hand, GPOMs account for uncertainty as they are based on a Bayesian nonparametric model. Regardless of their attractive theoretical properties, GPOMs are impractical for real-world usage because of the $\mathcal{O}(N^3)$ run-time and memory complexity. Recently proposed Bayesian Hilbert maps (BHMs) (Senanayake and Ramos 2017), on the other hand, encompass all positive traits of GPOMs but at a cost of $\mathcal{O}(M^3)$ where $M \ll N$ is the number of features that correlates with the accuracy. Since ABHM considers the full Bayesian treatment over

*We limit our discussion to 2D for simplicity. All theory are readily extensible to 3D.

parameters of (Senanayake and Ramos 2017) to account for local spatial changes in the environment, it achieves a significantly higher accuracy.

BHM can be summarised as performing Bayesian logistic regression in a high-dimensional feature space $\mathbb{R}^M$ using *k*ernels (Hofmann, Schölkopf and Smola 2008; Senanayake and Ramos 2018). BHM uses the same kernel for the entire map. ABHM is an extension to BHM to learn all location-dependent nonstationary kernel parameters. While BHM can be run in near real-time in an online fashion, ABHM is computationally expensive as it requires learning thousands of parameters offline. In ABHM, the occupancy probability of a point $\mathbf{x}_*$ is given by,

$$p(y_* = 1|\mathbf{x}_*) = \text{sigmoid}\Bigg(\sum_{m=1}^{M} w_m \underbrace{\exp\big(-\gamma_m \|\mathbf{x}_* - \mathbf{h}_m\|_2^2 \big)}_{m^{\text{th}} \text{ SE kernel}} \Bigg), \tag{4.1}$$

where $w, \mathbf{h}$, and γ are parameters learned from data $\mathcal{D}$. The inner part of the equation is a w weighted sum of M kernels placed in 2D spatial locations $\mathbf{h}$. In areas where there are more LIDAR hits in the locality of a kernel, then its associated weight w_m will be higher, and vice versa. This is because, as illustrated in Figure 4.2, here, M squared-exponential (SE) kernels positioned at mean locations $(\bar{\mathbf{h}}_1, \bar{\mathbf{h}}_2, \ldots, \bar{\mathbf{h}}_M)$ are used to project 2D data into an M dimensional vector such that each kernel has more effect from data in its locality. γ are positive parameters that control the width of each kernel. Probability distributions $w_m \sim \mathcal{N}(\bar{w}_m, \breve{w}_m)$, $\mathbf{h}_m \sim \mathcal{N}(\bar{\mathbf{h}}_m, \breve{\mathbf{h}}_m)$, and $\gamma_m \sim \text{Gamma}(\bar{\gamma}_m, \breve{\gamma}_m)$ are induced on the parameters to naturally encode uncertainty. Here, slightly abusing standard notations, $\bar{}$ and $\breve{}$ symbols are used to represent the mean and dispersion parameters, respectively (Table 4.1).

The parameters of the model are learned using variational inference (Senanayake, Tompkins and Ramos 2018). See Figure 4.2 for some of the estimated parameters. Since there are 8 parameters ($\bar{w}_m, \breve{w}_m, \bar{\gamma}_m, \breve{\gamma}_m \in \mathbb{R}$ and $\bar{\mathbf{h}}_m, \breve{\mathbf{h}}_m \in \mathbb{R}^2$) associated with each kernel, it is required to learn $8M$ parameters. In order to achieve a practically satisfactory accuracy to cover a 100 m^2 area, it is necessary to have over 10000 kernels which would take around 10 minutes on a GPU. On the other hand, although ABHM provides high-quality maps, it is

required to first collect the entire dataset as it does not support sequential training, making it practically unsuitable for mobile robotics applications.

4.3.2 Optimal Transport (OT)

Intuitively, in its simplest form, Optimal Transport aims at determining the optimal way to change one probability distribution to another probability distribution with the least effort. One of the classical toy examples is moving "dirt" from one configuration to another with the least effort. Monge-Kantorovich Theorem is a convex relaxation to the more general optimal transport problem proposed by Monge (Monge 1781; Villani 2008). Unlike in the Monge's general formulation, Monge-Kantorovich guarantees the existence of a transport map $P : \Omega^{(\mathcal{S})} \rightarrow \Omega^{(\mathcal{T})}$.

From Monge-Kantorovich Theorem 2, our objective is to determine the optimal coupling matrix (plan) between measures $\boldsymbol{\mu}^{(\mathcal{S})}$ and $\boldsymbol{\mu}^{(\mathcal{T})}$. We represent source LIDAR locations $\mathbf{x}_1^{(\mathcal{S})}, \mathbf{x}_2^{(\mathcal{S})}, \ldots, \mathbf{x}_{N^{(\mathcal{S})}}^{(\mathcal{S})}$ and target LIDAR locations as $\mathbf{x}_1^{(\mathcal{T})}, \mathbf{x}_2^{(\mathcal{T})}, \ldots, \mathbf{x}_{N^{(\mathcal{T})}}^{(\mathcal{T})}$ corresponding to both $y = 0$ and $y = 1$ as discrete measures,

$$\boldsymbol{\mu}^{(\mathcal{S})} = \sum_{i=1}^{N^{(\mathcal{S})}} a_i^{(\mathcal{S})} \delta_{\mathbf{x}_i^{(\mathcal{S})}} \quad \text{and} \quad \boldsymbol{\mu}^{(\mathcal{T})} = \sum_{j=1}^{N^{(\mathcal{T})}} a_j^{(\mathcal{T})} \delta_{\mathbf{x}_j^{(\mathcal{T})}} \tag{4.2}$$

where $\delta_{\mathbf{x}}$ is the Dirac at $\mathbf{x}$, $\sum_{i=1}^{N^{(\mathcal{S})}} a_i^{(\mathcal{S})} = 1 = \sum_{j=1}^{N^{(\mathcal{T})}} a_j^{(\mathcal{T})}$, and $a_i^{(\mathcal{S})}, a_j^{(\mathcal{T})} \geq 0$. The existence of transport maps for discrete measures is identified by the Minkowski–Carathéodory Theorem (Villani 2003).

4.3.3 Automorphing Bayesian Hilbert maps (ABHM)

The conventional approach for occupancy mapping is discretizing the environment with a predefined cell resolution and update cells individually (Elfes 1989; Arbuckle, Howard and Mataric 2002). Instead if discretizing the environment, continuous occupancy mapping techniques such as Automorphing Bayesian Hilbert Maps (ABHM) (Senanayake, Tompkins and Ramos 2018) learn the environment as a continuous function. Unlike occupancy grid maps

(OGM) (Elfes 1989), such mapping techniques take into account neighborhood information and epistemic uncertainty to improve the map quality (Senanayake, Tompkins and Ramos 2018). The speciality about this mapping technique is that it fully estimates all parameters of the model accounting for all geometric nuances in the environment. However, this model is extremely slow to run in real-time. All the necessary details for ABHM are given in previous section chapter 3.

4.3.4 Kernel Methods in Robotics

Loosely speaking, Hilbert space can be thought of as a generalization of the Euclidean space. Reproducing kernel Hilbert space (RKHS) is a Hilbert space of functions with special propertiesthat enable easy computations. Intuitively, kernel methods project data into a high dimensional linear space and perform computations there. Since most of the robotics tasks are nonlinear, kernel methods can be effectively used in robotics. Especially, they are useful in settings where alternative nonlinear function learning techniques such as neural networks cannot be used. For instance, when the dataset is small or sparse, as in occupancy mapping, learning in RKHS is efficient. Kernel methods have applications in both perception and control. For instance, Gaussian processes with various kernels have been used for mapping (O'Callaghan and Ramos 2014; Norouzii, Miro and Dissanayake 2016), and modelling for grasping (Bohg 2011), adaptive control (Chowdhary et al. 2014), data-efficient reinforcement learning (Deisenroth and Rasmussen 2011), learning inverse dynamics (Nguyen-Tuong, Seeger and Peters 2009), and simultaneous localization and mapping (Norouzi, Miro and Dissanayake 2017).

4.3.5 Domain Adaptation

The learned model parameters for a sample environment can be visualised in Figure 4.2.

OBSERVATION 1. *Once the full ABHM model is learned, the following can be observed:*

(1) As shown in Figure 4.2 (b), the mean values of weights $\bar{w}$ are higher in areas where there are LIDAR hits, and vice versa. In areas where there are no observations

at all ($x_1 \lessapprox -105$ in Figure 4.2 (a)), the variance values $\breve{w}$ are high as shown in Figure 4.2 (c).

(2) The mean widths $\bar{\gamma}$, as can be observed in Figure 4.2 (d), are higher close to the obstacles, indicating sharp edges.

(3) The mean positions of kernels $\bar{\mathbf{h}}$ align according to the geometry of the obstacles (Figure 4.2 (b)-(d)).

PREMISE 1. *Based on Observation 1, there is geometric correspondence between parameter values and obstacles observed by the LIDAR. Therefore, we argue that spatially dependent parameters for a new environment, defined as the target domain, can be estimated by discovering correspondence between the target (new) LIDAR data and source (known) LIDAR data with associated parameters. Here, the source is an environment whose parameters are known or pre-estimated using a method such ABHM in a simple environment, and the target is a complex and large environment whose parameters are not known and challenging to estimate. This requires transferring features from source to target domains.*

Transferring knowledge obtained from one domain to the other has been widely discussed in the machine learning literature (Ammar et al. 2015; Pan and Yang 2009). The broader class of transferring from one type of domain to the other, e.g. images to text, is known as transfer learning. If the type of source and target domains are the same, as in occupancy mapping, the transfer process is called domain adaptation (DA). Applications in robotics include transferring control policies from simulation to real-world (Sadeghi et al. 2017; Bousmalis et al. 2018), and making image processing tasks invariant to lighting and other changes (Wulfmeier, Bewley and Posner 2018; Hoffman et al. 2016).

Variations of generative adversarial networks (GANs) such as DTN (Ghifary et al. 2016), CycleGAN (Zhu et al. 2017), DiscoGAN (Kim et al. 2017), UNIT (Liu, Breuel and Kautz 2017), DART (Fang et al. 2018) have been widely used for domain adaptation of RGB images. However, not only do these methods require a large amount of data but also it is not immediately clear how to use these techniques with sparse LIDAR data nor transferring probability distributions. In the next section, we consider an alternative domain adaptation

method based on optimal transport (OT) (Villani 2008) to transfer parameters of the Bayesian occupancy model using sparse LIDAR data.

4.4 Optimal Parameter Transport

In this section, we present the proposed algorithm. Transferring parameters is a two-step procedure: creating a source dataset offline (Section 4.4.1) and transferring them to a target domain online (Section 4.4.2). Section 4.4.3 is a generalization and is the actual algorithm used in experiments. Section 4.4.4 is an extension to further improve the map quality.

4.4.1 Preparing the Source Dictionary of Atoms

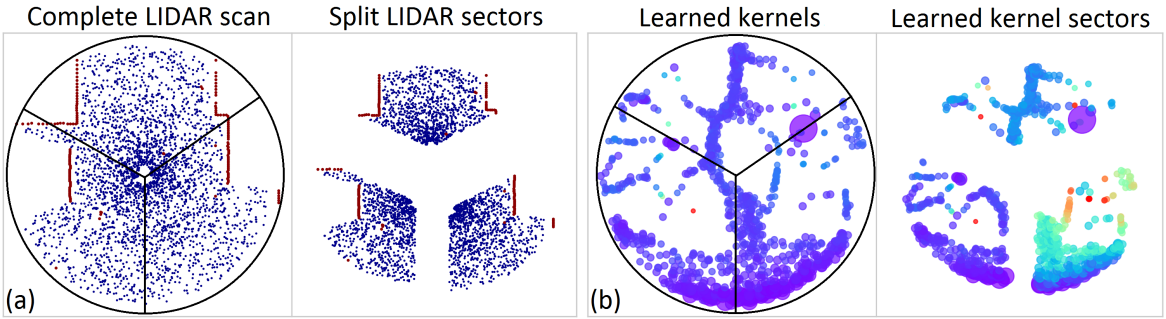


Figure 4.3. Extracting source data. (a) Splitting source LIDAR scans into 3 sectors. (b) Corresponding kernels parameters are also split the same way. Only kernel position means and weight means are shown here.

In order to take advantage of domain adaptation we must have accurately pre-trained maps from which we can extract spatially relevant features. In the context of our problem, we must extract LIDAR scans (hits and free) with their corresponding model parameters including kernel weights, positions, and widths. To provide high-quality training data we extract learned model parameters from ABHM maps. Since ABHM can only be used on small areas due to the high computational cost, we learn separate ABHM maps for different areas and construct a dictionary of source atoms which we call a *dictionary of atoms*.

To construct the dictionary, as illustrated in Figure 4.3, we split each LIDAR scan into circular sectors with radii equal to the specified maximum LIDAR distance. Rather than

using the entire LIDAR scan as the source dataset, this split not only results in a diverse set of geometric primitives but also provides simpler sources for the transfer procedure presented in the following section. The corresponding learned model parameters for each sector are considered as source parameters that we wish to transfer to the target domain. For each sector, we have $M^{(\mathcal{S})}$ parameters $\{\theta_m^{(\mathcal{S})}\}_{m=1}^{M^{(\mathcal{S})}}$ associated with $N^{(\mathcal{S})}$ LIDAR hits or free points $\{(\mathbf{x}_n^{(\mathcal{S})}, y_n^{(\mathcal{S})})\}_{n=1}^{N^{(\mathcal{S})}}$. The collection of these different LIDAR sectors constitutes the dictionary of source atoms $\mathcal{X}^{(\mathcal{S})}$.

4.4.2 Source to Target Parameter Transport

Until we present the general transfer procedure that we used in experiments in Section 4.4.3, for the sake of simplicity of the following discussion, let us assume that the dictionary of atoms contains only one LIDAR sector and associated parameters.

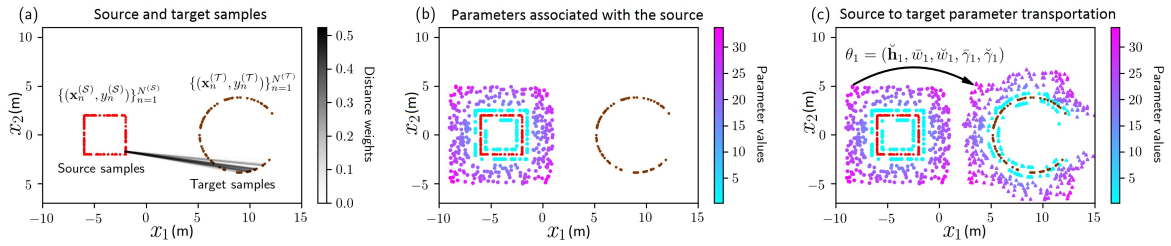


Figure 4.4. Optimal transport from a square to an arc. (a) If there are $N^{(\mathcal{S})}$ and $N^{(\mathcal{T})}$ number of data points in the source (red) and target (brown) datasets, the coupling matrix γ is size $N^{(\mathcal{S})} \times N^{(\mathcal{T})}$ where any column or any row sums to 1. A given row in γ indicates the probabilities of the sample associated with that row could be coupled to all samples in the target dataset. Probabilities associated with one such source point to target matches are shown in white-black color scale. Note that only the 10 highest matches are shown for clarity. (b) For a given set of LIDAR hits (red) spatial parameters can be learned using ABHM. Here we see kernel parameters spread across the environment. However, for another set of LIDAR hits (brown) we would prefer not re-learning parameters because it is expensive. (c) Based on the coupling matrix between the source and the target, we transport (move from the target area to the source area) the parameters around each point. Note that how the small lengthscales (cyan) stays close to the LIDAR hits and larger lengthscales (magenta) move away from the LIDAR.

$$\text{learn } P_* \text{ for } \mathbf{x}^{(\mathcal{S})} \xrightarrow[\text{transport}]{\text{explicit}} \mathbf{x}^{(\mathcal{T})}$$

$$\text{predict} \quad \begin{matrix} \bar{\mathbf{h}}^{(\mathcal{S})} \\ \vdots \\ \theta^{(\mathcal{S})} \end{matrix} \begin{matrix} \xrightarrow[\text{transport}]{\text{explicit}} \\ \\ \xrightarrow[\text{transport}]{\text{implicit}} \end{matrix} \begin{matrix} \bar{\mathbf{h}}^{(\mathcal{T})} \\ \vdots \\ \theta^{(\mathcal{T})} \end{matrix} \text{ using } P_*$$

Figure 4.5. Parameter optimal transport. Known and unknown quantities are in blue and red, respectively. We learn an optimal coupling matrix P_* using source and target LIDAR. Then we use this coupling matrix to predict target kernel positions corresponding to the source kernel position. By doing this, the other parameters associated with each kernel are also implicitly transported by treating them as labels.

Objective: Having determined source LIDAR data $\{(\mathbf{x}_n^{(\mathcal{S})}, y_n^{(\mathcal{S})})\}_{n=1}^{N^{(\mathcal{S})}}$ and corresponding parameters $\{\theta_m^{(\mathcal{S})}\}_{m=1}^{M^{(\mathcal{S})}}$, our objective is to determine the new set of parameters $\{\theta^{(\mathcal{T})}\}_{m=1}^{M^{(\mathcal{T})}}$ for a new LIDAR dataset $\{(\mathbf{x}_n^{(\mathcal{T})}, y_n^{(\mathcal{T})})\}_{n=1}^{N^{(\mathcal{T})}}$. This problem is illustrated in Figure 4.4 (a) and (b). In other words, we are looking for a nonlinear mapping technique to convert a source $(\mathcal{S})$ to a target $(\mathcal{T})$. We recognise this as an *optimal transport (OT) problem* given in Theorem 2.

Having obtained the optimal coupling between source and target LIDAR, as illustrated in Figure 4.4 (b)-(c) and Figure 4.5, now it is possible to transport source parameters $\theta^{(\mathcal{S})}$ to the target domain. This is done by associating the source parameter positions $\bar{\mathbf{h}}^{(\mathcal{S})}$ with source samples $\mathbf{x}^{(\mathcal{S})}$ as a linear map (Perrot et al. 2016), and transporting them to the target domain $\bar{\mathbf{h}}^{(\mathcal{S})} \rightarrow \bar{\mathbf{h}}^{(\mathcal{T})}$ according to the coupling matrix P_* learned from LIDAR matching. All other $\theta^{(\mathcal{S})}$ parameters associated with the kernels positioned at $\bar{\mathbf{h}}^{(\mathcal{S})}$ will also be transported to the target domain. This implicit transfer process is depicted in Figure 4.5.

4.4.3 Transport from a Dictionary of Atoms

Although we created a dictionary of atoms consisting of diverse geometric primitives in Section 4.4.1, the transfer procedure introduced in Section 4.4.2 was limited to a single LIDAR sector. In order to effectively make use of the entire dictionary, it is required to find the optimal coupling matrix over all elements in the dictionary $\mathbf{x}^{(\mathcal{S})} \in \mathcal{X}^{(\mathcal{S})}$.

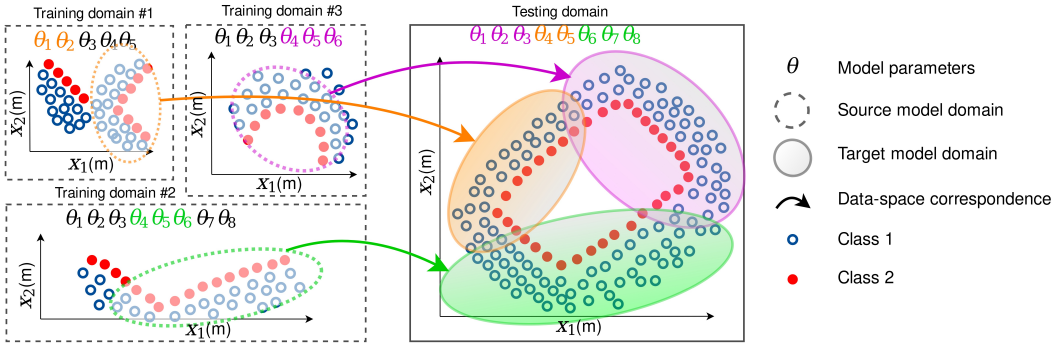


Figure 4.6. A high-level overview of the proposed method: Parameter Optimal Transport (POT). *Training domains* correspond to potentially independent, data-intensive, expensive, yet small-scale pre-learned models. After storing in a dictionary of atoms, representative data-space and model-parameter tuples from the pre-learned set of models, we find data-space correspondences using optimal transport. These correspondences are then used to transport pre-learned parameters to out-of-sample *test domains*.

As another fact, although Equation (2.43) can be used to obtain a translation and scale invariant solution, it is not robust enough against large rotation variations. However, we can rotate data about the centroid of each atom using the rotation matrix,

$$R(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}, \tag{4.3}$$

for a discrete set of rotations $\alpha \in \mathcal{A}$.

Overall, we obtain a candidate optimal coupling set of size $|\mathcal{X}^{(\mathcal{S})}| \times |\mathcal{A}|$ by minimizing Equation (2.43) over all rotations and atoms,

$$\mathcal{P}_* = \left\{ \underset{P \in \Gamma(\mathbf{x}^{(\mathcal{S})}, R(\alpha)\mathbf{x}^{(\mathcal{T})})}{\operatorname{argmin}} \sum_{ij} P_{ij} D_{ij} - \lambda^{-1} r(P) \right\}_{\substack{\mathbf{x}^{(\mathcal{S})} \in \mathcal{X}^{(\mathcal{S})} \\ \alpha \in \mathcal{A}}}. \tag{4.4}$$

Ultimately, we select the overall best coupling matrix from the candidate set $\mathcal{P}_*$ as the candidate that has the minimum 2-Wasserstein distance Equation (2.45) to the target,

$$P_* = \underset{P \in \mathcal{P}_*}{\operatorname{argmin}} \sum_{ij} P_{ij} D_{ij}. \tag{4.5}$$

This P_* can now be used to transfer parameters using the same method explained in Figure 4.5. As a result of the computation procedure introduced in this section, as depicted in Figure 4.6, atoms from various domains will be transferred to the target. Because atoms only consist of a few hundred LIDAR points, this transfer can be performed in real-time. Unlike in BHM or ABHM, we can now introduce thousands of kernels. The increasing number of pre-learned kernels as well as the nonstationarity help to improve the accuracy. The entire Parameter Optimal Transport (POT) algorithm is summarised in Algorithm 1.

P_* is computed for each candidate source prepared in Section 4.4.1. At runtime, it is necessary to determine the best source atom to transport to the test data using the Wasserstein-2 distance between each atom and test data. Since the computation is rotation invariant, each source dataset is rotated about its centroid using quaternion geometry before computing the optimal coupling P_* for each such configuration. The source that minimises the overall distance for a finite set of rotation matrices $\mathcal{R}$, given in (Equation (4.4)), is used as the best-rotated source atom to transfer.

4.4.4 POT Maps and Refined POT Maps

Transporting parameters can be performed in two different ways. It is possible to transport parameters for each LIDAR scan separately, and immediately build the occupancy map. This results in an *instantaneous map* which is useful for understanding the occupancy of the surrounding at present. Such maps can be used for safe decision-making and control in the locality of the robot. On the other hand, it is also possible to build the *overall map* by sequentially aggregating the transported parameters as the robot moves. The overall map model completely discards training LIDAR data after transporting the parameters. This enables mapping large areas at a constant cost.

Once the parameters are transported with the intention of building an instantaneous or overall map, an occupancy map can be generated by plugging in the transported parameters to eq. (1) and querying occupancy probabilities. It will not only provide the mean occupancy map, but also the uncertainty as the variance estimate. Since only the parameters of the continuous

Algorithm 1: Transferring parameters to a new domain

Input: New LIDAR scans, Source dictionary of atoms
while *new scan in new domain* **do**
  $\mathcal{P}_* = \{\}$;
  for *each atom in* $\mathcal{X}^{(\mathcal{S})}$ **do**
    for *each rotation in* $\mathcal{A}$ **do**
      $\mathcal{P}_*.\mathrm{insert}(\text{Compute the coupling matrix})$ (Equation (4.4));
    end
  end
  $P_* \leftarrow$ Determine the best coupling matrix (Equation (4.5));
  $\theta^{(\mathcal{T})} \leftarrow$ Transfer the source parameters to the target domain using P_* (Figure 4.5);
end
Output: Parameters $\theta^{(\mathcal{T})}$

mapping function Equation (4.1) are stored, the occupancy map can later be queried *at any time at any resolution*.

Learning kernel parameters γ and $\mathbf{h}$ in real-time is not feasible with ABHM. However, learning weights w, assuming other parameters are given, we have a fast approximation given by Bayesian Hilbert maps (BHMs) (Senanayake and Ramos 2017). As an additional step to further improve the map quality, we propose to *use transported parameters as prior distributions* of the BHM and simply update the weights w by using (Senanayake and Ramos 2017). We call this improved map, the refined POT (RePOT) map.

4.5 Experiments

Both simulated and real-world datasets were used to assess the quality of POT. To generate simulated data, Carla v.0.9.2 simulator (Dosovitskiy et al. 2017) was used as it closely resembles real-world towns. As a real-world dataset, we used the KITTI benchmark dataset (Geiger et al. 2013). All datasets are listed in Table 4.2 and each of these environments is considered as a domain. As evaluation metrics, we used accuracy (ACC), area under the receiver operating characteristic (ROC) curve (AUC) and negative log-likelihood (NLL) (Bishop 2006) which is also know as log loss or cross entropy loss. Unlike ACC and AUC,

Table 4.2. Description of domains

Domains (Datasets)	Description
Carla Town 1	a 2D dataset in town 1 in Carla (3.7 km).
Carla Town 2	a 2D dataset in town 1 in Carla (1.5 km).
Carla Town 3	a 2D dataset in town 1 in Carla (8.6 km).
Carla Town 1 3D	a 3D dataset in town 1 in Carla.
Carla Town 1 Dyna	Carla Town 1 with 120 vehicles running around.
KITTI Dyna	a 2D dataset (the middle LIDAR channel).

NLL takes into account uncertainty of predictions. NLL is defined as follows

$$\text{NLL} = -y \log(y_*) + (1 - y) \log(1 - y_*), \tag{4.6}$$

where $y \in \{0, 1\}$ is the true label and $y_* \in [0, 1]$.

Accuracy is defined as follows

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4.7}$$

where TP = True positive, FP = False positive, TN = True negative, and FN = False negative predictions.

The ROC curve is a curve constructed by varying some discrimination threshold $threshold \in [0, 1]$ plotted over the fraction from the true positive rate (TPR) over the false positive rate (FPR). TPR is the number of true positive (TP) out of all positives, and the FPR is the number of false positives out of all negatives.

The higher the AUC or lower the NLL, the better the model is.

4.5.1 Dataset Details

To generate simulation datasets, a car equipped with 20 m - 360^o LIDAR was driven 3.7 km, 1.5 km, 8.6 km with varying speeds in Town 1, Town 2, and Town 3, respectively in the Carla simulator (Dosovitskiy et al. 2017). A 32 beams LIDAR with the z-axis view of -30^o- 10^o

was used to generate the 3D dataset in Town 1. The KITTI dataset was used as the real dataset (Geiger et al. 2013). It has a car driving in an urban environment carrying a 120 m -360^o LIDAR. As in other occupancy mapping techniques we assumed that the localization is given from an another sensor.

4.5.2 OGM Hyperparameters

We considered the rotations $\mathcal{A} = \{0^o, 90^o, 180^o, 270^o\}$. For Sinkhorn, λ was set to 0.005 with maximum 1000 iterations. OGM was obtained by with the best of cell resolutions of 0.1, 0.5, 1, 5, and 10 m.

4.5.3 Snapshot of Domain Adaptation Pipeline

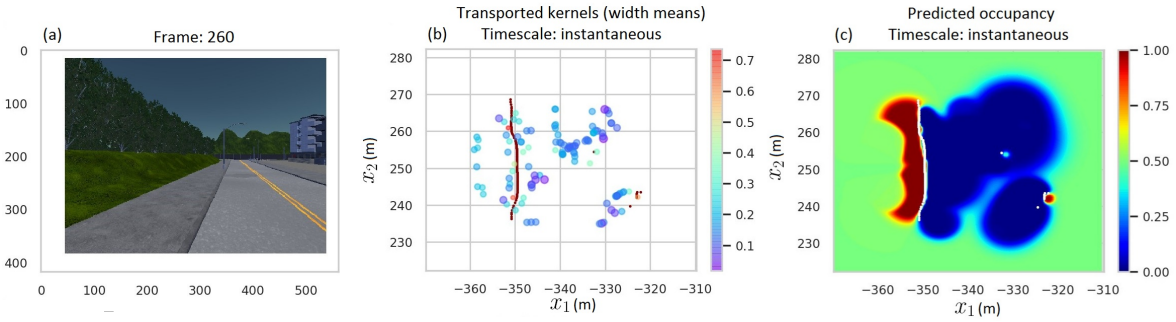


Figure 4.7. An example of domain-to-domain parameter transfer. (a) Camera-view of the target environment. (b) Transported model parameters (kernels) for a given test LIDAR scan. (c) Target occupancy map at current time step.

Figure 4.7 shows an instantaneous map for the Carla Town 1 environment. Visualization of all datasets and corresponding kernel position means and width means are shown in Figure 4.9. Figure 4.8 demonstrates how POT can be used for 3D LIDAR data.

4.5.4 Intra-domain and Inter-domain Adaptation

In this experiment, we consider two paradigms: intra-domain and inter-domain transfer. In intra-domain transfer, the source atoms are generated from the first 10 frames of a particular dataset and parameters are transferred to the rest of the same dataset while they are transferred

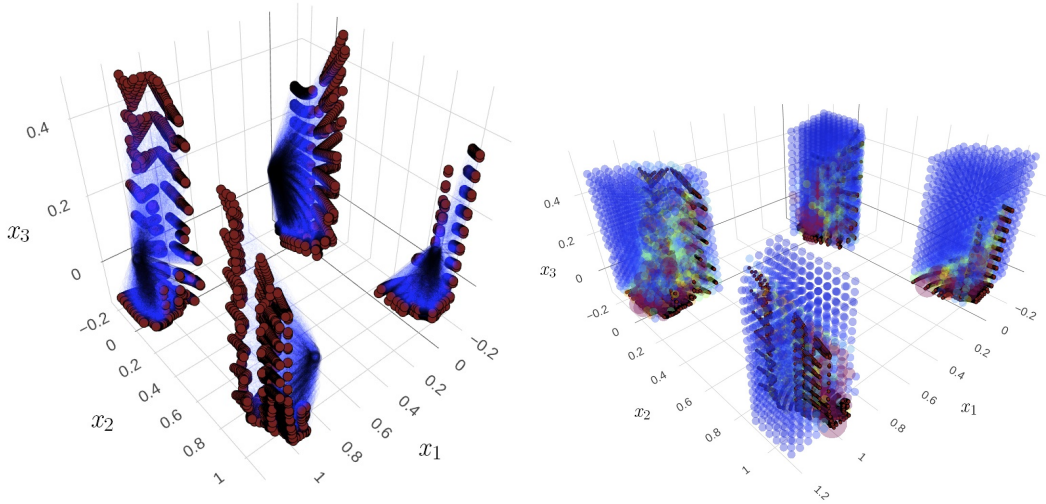


Figure 4.8. Training atoms for Town 1 in 3D. The left figure shows 3D LIDAR scans while the right figure shows the learned ABHM kernels for the corresponding LIDAR scans. Instead of extracting 2D atoms, we simply extract radial 3D atoms and use their previously learned model parameters for transfer.

to a completely different domain in inter-domain transfer. Based on results reported in Table 4.3 with 20% randomly sampled test LIDAR beams from each town, it is possible to accurately transfer parameters using POT. This enables mapping large scale towns in real-time. All parameters are aggregated over time to build occupancy maps of the entire environments as visualised in Figure 4.9. Using the Town 1 3D dataset, we demonstrate the possibility of extending POT to 3D environments. In this case, source atoms described in Section 4.4.1, were circular cylindrical sectors (i.e. pie slice shaped). The post-hoc refinement procedure, RePOT, introduced in Section 4.4.4, further improved the map significantly. A visualization of RePOT is shown in Figure 4.10 and performance improvement, in direct comparison with results in Table 4.3, is reported in Table 4.4.

4.5.5 Building Instantaneous Maps

This experiment was designed to demonstrate how parameters can be instantaneously transported to build the instantaneous map of the surrounding. For this purpose, we used the two dynamic environments: Town 1 Dyna and KITTI Dyna. The source dictionary of atoms was prepared similar to the intra/inter-domain adaptation experiment. Such a map is shown in

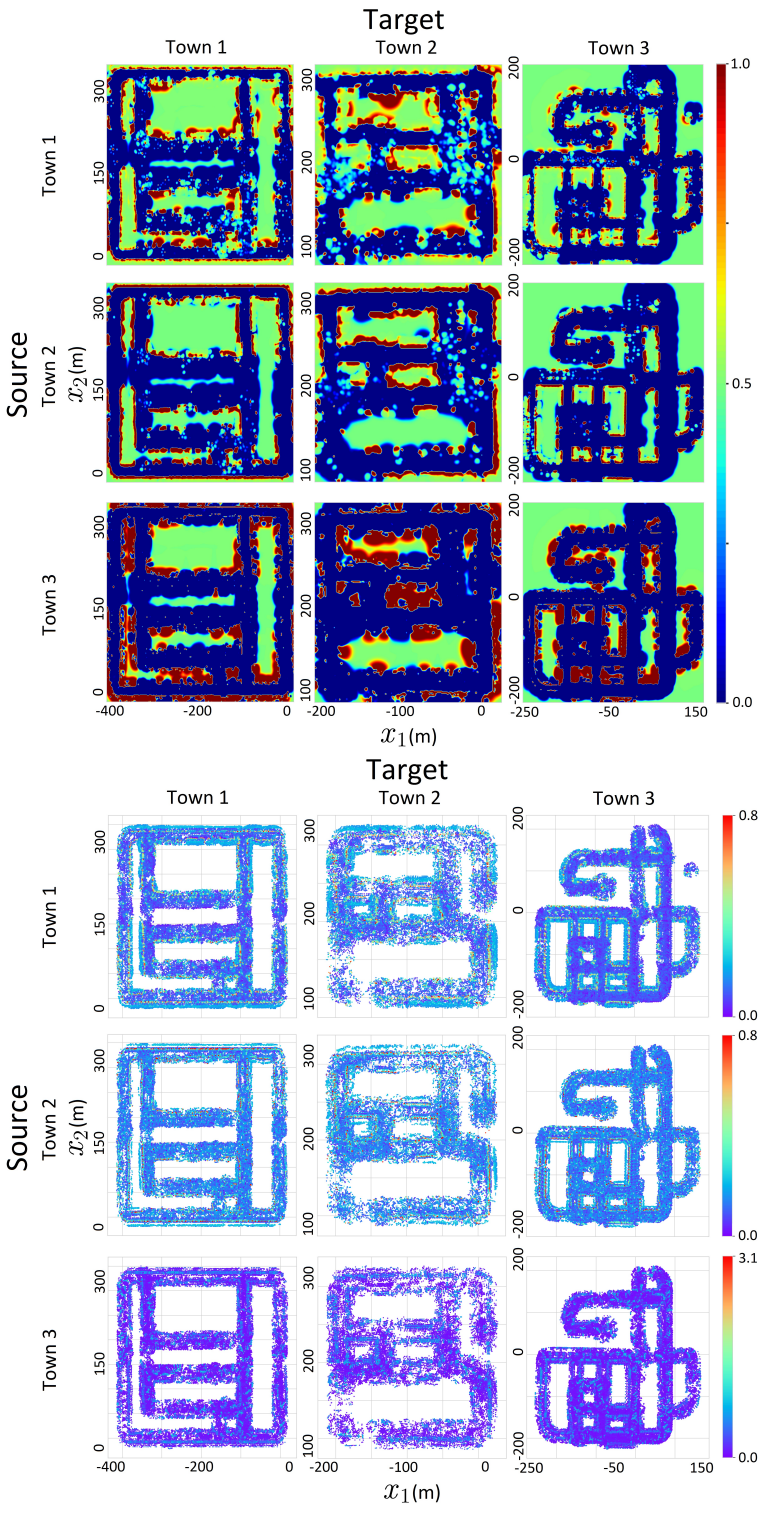


Figure 4.9. Transported kernels (Bottom) and their occupancy maps (Top) for the inter and intra domain town datasets. From top to bottom and left to right are towns 1, 2, and 3.

Table 4.3. Performance of intra-domain (diagonal entries of the table) and inter-domain (off-diagonal entries of the table) transfer.

Source			Target		
			Town1	Town2	Town3
Source	ACC	Town1	0.79	0.82	0.76
		Town2	0.70	0.72	0.58
		Town3	0.85	0.83	0.84
	AUC	Town1	0.88	0.88	0.90
		Town2	0.85	0.83	0.83
		Town3	0.92	0.92	0.93
	NLL	Town1	1.14	0.97	1.40
		Town2	3.30	3.23	5.98
		Town3	1.64	1.69	1.79

Table 4.4. Performance of Refined POT (RePOT) across intra- and inter-domain transfers.

Source			Target		
			Town1	Town2	Town3
Source	ACC	Town1	0.95	0.93	0.95
		Town2	0.91	0.91	0.92
		Town3	0.95	0.92	0.93
	AUC	Town1	0.99	0.98	0.98
		Town2	0.98	0.98	0.98
		Town3	0.99	0.97	0.97
	NLL	Town1	0.71	1.4	1.12
		Town2	1.40	1.74	1.85
		Town3	0.96	1.62	1.44

Table 4.5. Instantaneous maps in dynamic environments: Experiments for sim2sim and sim2real with mean and SD.

Source			Target	
			Town 1 Dyna	KITTI Dyna
Source	ACC	Town 1	0.74 ± 0.10	0.69 ± 0.06
		Town 2	0.70 ± 0.10	0.58 ± 0.06
		Town 3	0.74 ± 0.11	0.71 ± 0.07
	AUC	Town 1	0.81 ± 0.11	0.77 ± 0.06
		Town 2	0.77 ± 0.12	0.73 ± 0.06
		Town 3	0.78 ± 0.15	0.73 ± 0.09
	NLL	Town 1	1.06 ± 0.56	1.42 ± 0.38
		Town 2	1.90 ± 0.79	3.63 ± 1.04
		Town 3	1.89 ± 1.30	2.30 ± 0.83

Figure 4.1. The performance of the model was evaluated on 20% of data that were not used for optimal transport. Table 4.5 shows the performance of transferring features extracted from each town to the dynamic datasets.

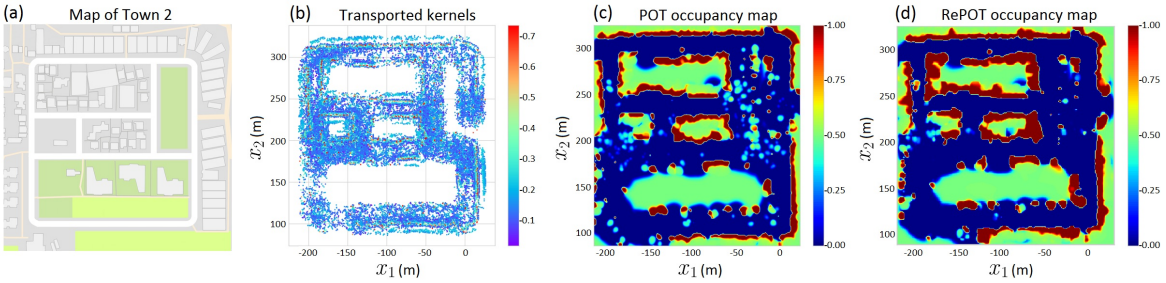


Figure 4.10. Large-scale map building with POT and RePOT. (a) Carla Town 2 plan. (b) Transferred kernel mean width and position parameters. (c) Occupancy prediction with POT. (d) Occupancy prediction with RePOT.

Table 4.6. Performance per time unit for RePOT, POT, ABHM, and BHM. Though OGM results are reported for reference purposes, unlike other methods, OGM cannot be computed for per time unit basis.

		Target		
	Method	Town1	Town2	Town3
ACC	RePOT	0.95	0.93	0.95
	POT	0.85	0.83	0.84
	ABHM	0.77	0.59	0.86
	BHM	0.66	0.61	0.71
	OGM	0.78	0.78	0.77
AUC	RePOT	0.99	0.98	0.98
	POT	0.92	0.92	0.93
	ABHM	0.95	0.96	0.96
	BHM	0.94	0.92	0.91
	OGM	0.89	0.91	0.90
NLL	RePOT	0.71	1.41	1.12
	POT	1.64	1.69	1.79
	ABHM	0.58	0.71	0.41
	BHM	0.63	0.69	0.61
	OGM	2.00	1.34	1.13

4.5.6 Performance Comparison

In this experiment, we compared various occupancy mapping algorithms in terms of accuracy and speed. Since these algorithms cannot be trained or queried in a similar fashion, we measured the per time unit performance. For instance, ABHM can only be trained in small environments although our datasets consist of large towns. Firstly, we measure the time for running POT per LIDAR scan. Then we decide the number of kernels to match the same runtime for BHM and ABHM. Results are reported in Table 4.6. Though OGM cannot be

computed per time basis, we report the results for reference. GPOM cannot be executed for datasets this large. As expected, ABHM outperforms BHM in all metrics because ABHM is a nonstationary model that takes into account local geometry. Theoretically, in the infinite memory and computation time limit, ABHM should outperform all methods. Nonetheless, practically, POT has a higher ACC and AUC compared to ABHM as POT can transfer kernels online to accommodate the complexity of the environment. However, the increase in NLL in POT compared to ABHM, indicates the inherent uncertainties of the transfer procedure. Once the weights were refined using RePOT, NLL has dropped as the weight distributions can be optimised to reduce the uncertainty giving better predictions.

Runtime: With a laptop with 4 cores and 8 GB RAM, on average, POT, programmed in Python, takes around 1 s update time. This is without parallelizing any part of the code. Note that eq. (Equation (4.4)) is highly parallelizable making the algorithm $|\mathcal{X}^{(\mathcal{S})}| \times |\mathcal{A}|$ faster (approx. 25 times). This is a significant improvement to algorithms such as BHM and ABHM which would take several hours to build a large-scale map as they rely on complicated variational inference procedures. POT run-time increases with increasing λ in the Sinkhorn algorithm we used in POT. As $\lambda \to \infty$ the convergence is guaranteed.

4.6 Summary

In this chapter we highlighted a major issue with learning data-dependent Bayesian models using learnable kernels: they have no closed form solution and the optimisation problem is compute intensive such that the methodology is prohibitive to real-time use. We thus proposed a novel method using optimal transport that allows us to transform the long sequential optimisation problem into a domain-adaption procedure that involves much smaller parallel optimisations.

In optimal transport, we consider the problem of transforming one probability measure to another. This also loosely relates to the point cloud registration problem typically addressed by the iterative closest point (ICP) algorithm (Besl and McKay 1992). However, unlike ICP which only has a single set of translation and rotation parameters, in optimal transport,

each data point in the source dataset has a highly nonlinear relationship with every other point in target datapoints through the optimal coupling matrix P_*. Another reason why we cannot resort to a popular algorithm such as ICP is because it only works for slight changes in translation and rotation. When a robot moves in dynamic environments, it is essential to adapt for sudden, potentially large, nonlinear changes in geometry.

One remarkable aspect of being able to transport distributions is that it endows us the ability to adapt Bayesian models in the sense of an *informed prior* (Gronau et al. 2017) enabling expedited parameter tuning. We have demonstrated such a use case in this chapter with in RePOT that brings significant improvements in overall predictive quality.

In conclusion, this chapter introduced parameter optimal transport (POT), an efficient framework for geometric domain adaptation. By combining the formalism of automorphing Bayesian Hilbert maps with optimal transport theory, patterns from one environment can be seamlessly transferred to another in a fraction of a second. We show that this framework can be effectively used to map large urban environments, transferring learned patterns between two cities, between simulated and real environments, and between static and dynamic environments.

CHAPTER 5

Quantile Representations for Approximate Kernel Learning

This chapter presents a major shift in perspective, with regard to kernel representation, compared to the previous chapters. Instead of explicitly located kernel embeddings in our observation space, we will exploit the Fourier domain representation of stationary kernels. The Fourier perspective contrasts with the kernel representation and learning methodologies presented previously in Chapter 3 and Chapter 4 which used explicit *placements* of known analytic kernels, in precisely the same domain that the raw data resides, to construct learnable embeddings. By adopting the well known Fourier representation of kernels and consequent parametric learning paradigm, we introduce a way to fundamentally generalise the structure of the kernel into a learnable representation founded on the quantile representation of a distribution. By doing so, we create an *implicit* kernel that indirectly represents some arbitrary, non-analytic kernel that is free to adapt its representation to observations. We demonstrate the flexibility of the new construction on a variety of interesting problems ranging from non, quasi and periodic phenomena.

5.1 Introduction

Historically, as with typically black box approaches using deep neural networks, kernel methods gained popularity due to their ability to discover nonlinear patterns (Schölkopf, Smola and Müller 1998; Cristianini and Shawe-Taylor 2000). Traditionally applied in data scarce regimes, kernel methods have been shown to be similarly effective in big data settings (Cortes and Vapnik 1995; Dai et al. 2014; Hensman, Fusi and Lawrence 2013a). In the literature, there have been multiple endeavours for fully automated inference pipelines in

machine learning (Ghahramani 2014; Tran et al. 2017; Tran et al. 2016; Feurer et al. 2015; Inc. 2018). Among these, we could broadly classify them under black box (Valera and Ghahramani 2017; Nguyen and Bonilla 2014) and grey box (Galliani et al. 2017) approaches where some informative amounts of prior knowledge is injected into the inference pipeline. Kernel methods seem to have had comparatively less effort invested under such generalist learning paradigms.

It has been seen many times that kernel methods are highly effective in wringing out the essential nonlinear structures in data from a variety of domains and tasks. As we have seen earlier, one of the major problems with kernel methods is that there is almost always the necessity for some kind of heuristic when it comes to choosing the kernel. In a sense this means the choice of kernel function is handcrafted due to the fundamental differences between the way various kernels encode data. We argue that this very human constraint is *undesirable*. While certain kernels have very interesting theoretical and well studied properties, it is usually the case that *a priori* we *simply do not know the structure of the very data we are trying to model*! So it therefore stands to reason that we should not be so confident (indeed, helpless) in choosing *any* well known analytic kernel. From this, one could reason that we could simply *try them all*; this is in fact reasonable and even attempted. Many practitioners will try an SE kernel, a Matérn kernel, a variety of lengthscales for each, perhaps a reasonably complex but finite weighted composition as seen in Section 2.4, and then call it a day. However, this is really like an answer to the wrong question. We can do better. A better question to ask would be: *What is the most fundamental, necessarily sufficient, and computable form of a kernel?* In this chapter we propose an attempt to answer this question.

As hinted earlier, it can be observed that most kernel learning techniques have focused on choosing a mixture of popular kernels and learning their weights (Duvenaud 2014; Wilson and Adams 2013). In contrast, in the work presented in this chapter, we demonstrate how to learn kernels in a more general and black box way that makes no prior assumption on any prior analytic kernel. In order to address our desire for greater generality, this chapter will present a novel technique to learn kernels that best fit the data. To do this, we will adopt the measure-theoretic view of a shift-invariant kernel given by Bochner's theorem Theorem 1. We

will additionally propose to learn the measure in terms of a *parameterised quantile* (inverse cdf) function in light of this new distributional perspective. With this learnable *black box* quantile, which is amenable to both Monte Carlo (MC) and Quasi-Monte Carlo (QMC) sampling, we show that it is possible to construct quasi-random Fourier feature maps that can approximate arbitrary kernels. This process generates novel shift-invariant kernels that are guaranteed to be positive-definite. The kernels learned in the proposed method can be used in any kernelised inference algorithm such as kernelised regression or classification by optimizing the parameters of the black box quantile (BBQ) alongside the parameters of the inference algorithm. It can also be noted that interpreting the learned quantile as a black box does not hinder the interpretability of the inference algorithm since the reconstructed kernel itself can still be understood as a similarity function. To validate the methodology, we demonstrate the kernel's capabilities on a variety of datasets, showing how it is able to automatically able to extract complex patterns in the underlying data with minimal necessity for human knowledge.

5.2 Contributions

This chapter focuses on the problem of learning expressive and scalable approximate kernel representations of stationary kernels. By approaching kernel embeddings from the Fourier domain we are able to relax the definition of kernel functions from rigid analytic functions to arbitrary distributions. Specifically, we propose to represent kernels implicitly by using the *quantile* form of a distribution which we can parameterise to enable learnable implicit kernel approximations. Specifically, we present the following contributions:

(1) We propose a new method to learn quantile representations of stationary kernels in the Fourier domain. We term the method *Black Box Quantiles* (BBQ);
(2) We demonstrate how to integrate the kernel learning in an end-to-end fashion; and
(3) Validate the method on various datasets and problems exhibiting complex phenomena like periodicity that is not straightforward to model with conventional stationary kernels.

5.3 Related Work

As described in Section 2.4.7 kernel learning is a fundamental area of research since it clear, by their corresponding definitions, that kernel methods completely and utterly depend on their realisation of hyperparameters.

Previous works have looked at augmenting kernels with flexible learning methods like neural networks (Rasmussen 2004) while others attempt to find optimal combinations of existing kernels (Duvenaud et al. 2013). Nonetheless, because the manual construction of a sophisticated kernel is not straightforward, similar to a typical model selection problem in statistics, researchers have attempted to learn kernels as multi-task learning and through expensive optimization techniques (Duvenaud 2014; Lanckriet et al. 2004; Bach, Lanckriet and Jordan 2004). These approaches to finding sophisticated kernels is known as kernel learning. In this work, we revisit kernel learning with novel techniques developed in recent years to approximate kernels by a dot product of basis functions. In this regard, the spectral domain representation of the kernel defined in Theorem 1 and Corollary 1 has gained popularity (Rahimi and Recht 2007a).

In a series of works (Wilson and Adams 2013; Wilson et al. 2013), using an interpretation similar to Corollary 1 motivates the learning of a general covariance function of a Gaussian process model. Taking advantage of mixture representations and sampling from a pdf has also been explored in (Lázaro-Gredilla et al. 2010; Oliva et al. 2016). In the next section, we discuss the generic sampling based approximation for kernels with a fixed structure (i.e. kernels are not learned).

As previously detailed in Section 2.4.6, (Rahimi and Recht 2007a) proposed to pick a known probability density function $f_\Omega(\omega)$ in such a way that the required kernel can be reconstructed using Corollary 1. For instance, samples drawn from a standard normal distribution can reconstruct a squared exponential kernel. More generally, representing the known pdf using a finite number of Monte-Carlo (MC) samples $\{\omega_m \overset{\text{iid}}{\sim} f_\Omega(\omega)\}_{m=1}^{M}$ creates a finite dimensional approximation of the feature map $\hat{\phi}(\mathbf{x}) \in \mathbb{C}^M$ (Sriperumbudur and Szabó 2015; Rudi and

Rosasco 2017; Bach 2017). That is,

$$k(\mathbf{x}, \mathbf{x}') \approx \frac{1}{M} \sum_{m=1}^{M} e^{-i(\mathbf{x}-\mathbf{x}')^\top \omega_m} = \langle \hat{\phi}(\mathbf{x}), \hat{\phi}(\mathbf{x}') \rangle_{\mathbb{C}}. \tag{5.1}$$

The approximate feature map can be decomposed into, $\hat{\phi}(\mathbf{x}) = \frac{1}{\sqrt{M}}[e^{-i\mathbf{x}^\top \omega_1}, \ldots, e^{-i\mathbf{x}^\top \omega_M}] \in \mathbb{C}^M$. For real valued kernels, (Equation (5.1)) can be further reduced into cosine and sine terms (Rahimi and Recht 2007a). This work has gained attention in recent years because of the simplicity, solid theoretical basis (Sutherland and Schneider 2015; Choromanski et al. 2018), and superiority in various applications (Rajeswaran et al. 2017; Kulis and Grauman 2009). Further advantages and outstanding performance of randomization based algorithms have been discussed in (Pastur 1973; Rahimi and Recht 2008; Lopez-Paz, Muandet and Recht 2015; Mania, Guy and Recht 2018).

5.4 Black Box Quantile Kernels

In this section, we propose black box quantiles (BBQ) as an alternative parameterisation of the probability measure to capture complex patterns in data. The method is general and can be used with other kernel composition techniques. Without loss of generality, BBQ features are introduced for univariate data x in Section 5.4.2 to Section 5.4.4 and a summary of the algorithm is given in Figure 5.1 and Algorithm 2. The treatment for multidimensional data is discussed in Section 5.5.

5.4.1 Why are Quantile Representations Suitable?

Quantiles are flexible in terms of both representation as well as representable for easy optimization. Learning complicated pdfs using a mixture of Gaussians or other parameterised distributions is challenging, while it is straightforward using our proposed method. By representation as a quantile, i) it is immediately possible to take advantage of low-discrepancy QMC samples without performing the "inverse transform sampling," ii) there are natural extensions connecting with copulas to enable cross-dimensional correlations between quantiles, iii) we

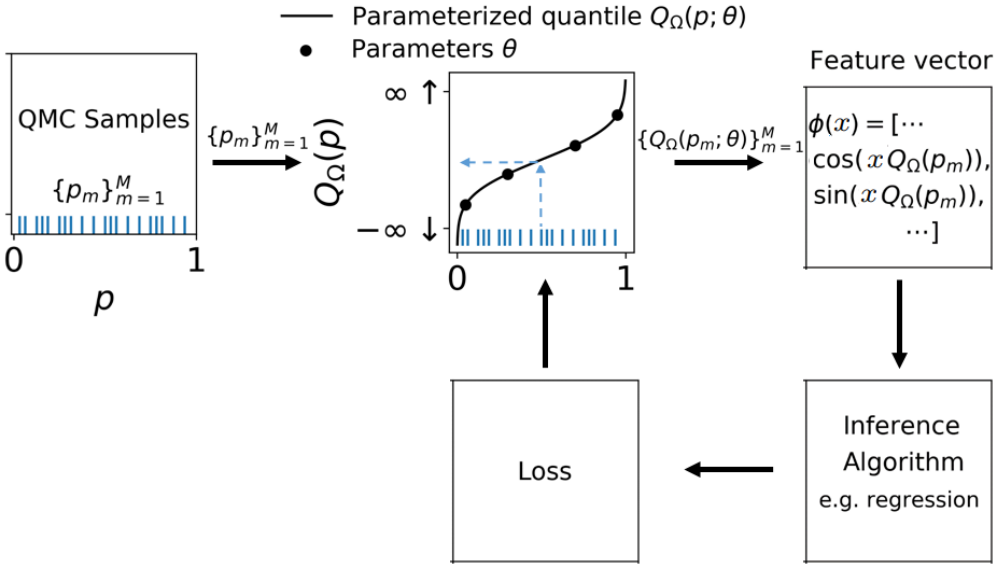


Figure 5.1. A summary of the BBQ algorithm.

have a closed form, in the sense of an explicit parameterisation of a distribution, giving an explainable "view" of the generalised kernel we are representing in the Fourier domain.

5.4.2 Parameterised Quantiles

As discussed in Section 2.4.6, known pdfs are used in (Rahimi and Recht 2007a) to construct popular kernels using Corollary 1. Unfortunately, to approximate complicated kernels, the method requires the specification of generic pdfs which can be notoriously difficult as pdfs need to satisfy $\int_{-\infty}^{\infty} f_\Omega(\omega)\mathrm{d}\omega = 1$. Observing the definition of Bochner's theorem (Theorem 1) given in a measure-theoretic view, we can alternatively prescribe the probability distribution in terms of a parameterised quantile function.

If the measure in Theorem 1 is defined by the cumulative distribution function (cdf), $p = P(\Omega \leq \omega) =: F_\Omega(\omega) : \mathbb{R} \to [0,1]$, then its quantile function is $\inf\{\omega \in \mathbb{R}; p \leq F_\Omega(\omega)\} =: Q_\Omega : [0,1] \to \mathbb{R}$. Intuitively, with strict monotonicity and continuity assumptions, the quantile can be regarded as the inverse of the cdf. Quantile functions of known distributions are widely used in many application domains in statistics (Steinbrecher and Shaw 2008; Sankaran, Nair and Midhu 2016). In Corollary 2, we obtain the kernel by a parameterised quantile function.

Algorithm 2: BBQ algorithm

Data: $x_{train}, y_{train}, x_{test}, y_{test}$
Result: Optimal kernel parameters θ_*
Optimal inference algorithm parameters w_*
$p = \{p_m\}_{m=1}^{M}$ low discrepancy sequence
Initialise θ
while *inference loss not converged* **do**
  Construct quantile $Q(\cdot;\theta)$ from θ
  $\hat{\phi}\big(Q\big) \leftarrow$ Compute features as in (Equation (5.4))
  Train inference model using $\hat{\phi}\big(Q\big)$
  Compute inference loss as in Section 5.4.4
  $\theta \leftarrow$ Next parameter from optimiser
end
return Best θ, corresponding w_*

COROLLARY 2. *Following Theorem 1, if $Q_\Omega(p;\theta)$ is a quantile function associated with a random variable Ω and parameterised by θ, the associated continuous, positive-definite, and shift invariant kernel, $k(x, x';\theta)$, is given by,*

$$k(x, x';\theta) = \int_{[0,1]} e^{-i(x-x')Q_\Omega(p;\theta)} \mathrm{d}p. \tag{5.2}$$

Our objective is to implicitly learn a sophisticated kernel by explicitly learning the parameters θ that define a quantile function. To this end, the properties that need to be satisfied are given in Theorem 3.

THEOREM 3. *The function Q_Ω is a quantile function iff,*

(1) $\lim_{p\to 0} Q_\Omega(p) = -\infty$ *and* $\lim_{p\to 1} Q_\Omega(p) = \infty$,
(2) $Q_\Omega(p)$ *is a non-decreasing function of* p,
(3) $\lim_{p\uparrow p_0} Q_\Omega(p) = Q_\Omega(p_0), \forall p_0 \in [0, 1]$, *i.e.* $Q_\Omega(p)$ *is left-continuous.*

Proof sketch: Writing Q_Ω using probability functions verifies the necessary conditions. For sufficiency, the existence of a sample space $\mathcal{S}$, a probability function P on $\mathcal{S}$, and a random variable Ω defined on $\mathcal{S}$ such that Q_Ω is a quantile function should be verified. A more detailed axiomatic description of probability and quantiles can be found in (Casella and Berger 2002; Parzen 1980; Chambers 2009).

Since building kernels by direct integration in Corollary 2 is intractable, we adopt a Monte Carlo approximation similar to random Fourier features discussed in Section 2.4.6, such that $k(\mathbf{x}, \mathbf{x}'; \theta) \approx \langle \hat{\phi}(\mathbf{x}; \theta), \hat{\phi}(\mathbf{x}'; \theta) \rangle_{\mathbb{C}}$. However, in this setting, the parameterised quantile is evaluated on a low-discrepancy sequence of quasi-Monte Carlo (QMC) samples. QMC approximations are known to be superior over MC approximations irrespective of the dimensionality (Dick and Pillichshammer 2010; Dick, Kuo and Sloan 2013; Yang et al. 2014) and with known kernels (Yang et al. 2014). With $\{p_m\}_{m=1}^{M}$ QMC samples, the approximated feature map is given by,

$$\hat{\phi}(x; \theta) = \frac{1}{\sqrt{M}} [e^{-ixQ_\Omega(p_1;\theta)}, \ldots, e^{-ixQ_\Omega(p_M;\theta)}] \in \mathbb{C}^M. \tag{5.3}$$

For real valued kernels, this can be further simplified as,

$$\begin{aligned} \hat{\phi}(x; \theta) = \sqrt{\frac{2}{M}} \Big[& \\ & \cos\big(xQ_\Omega(p_1; \theta)\big), \ldots, \cos\big(xQ_\Omega(p_M; \theta)\big), \\ & \sin\big(xQ_\Omega(p_1; \theta)\big), \ldots, \sin\big(xQ_\Omega(p_M; \theta)\big) \Big], \end{aligned} \tag{5.4}$$

where $\hat{\phi}(\mathbf{x}; \theta) \in \mathbb{R}^{2M}$. The objective is to learn $Q_\Omega(p; \theta)$ by adjusting θ.

5.4.3 BBQ Parameterisation

In this section we demonstrate how to parameterise an arbitrary quantile such that it can be evaluated for $p \in [0, 1]$. We restrict ourselves to fully continuous functions and therefore condition 3 in Theorem 3 is not of our interest. Intuitively, we are interested in seeking a parameterisation technique to represent a non-decreasing continuous function with vertical asymptotes at 0 and 1. In order to satisfy properties listed in Theorem 3, there are numerous ways to parameterise a valid quantile function such as Bernstein polynomials (Han et al. 2016). While such complex quantile formulations can surely be advantageous, the aim of this chapter is to lay the foundation to quantile induced kernels and demonstrate their importance through various applications. Therefore, in the following sections we restrict ourselves to

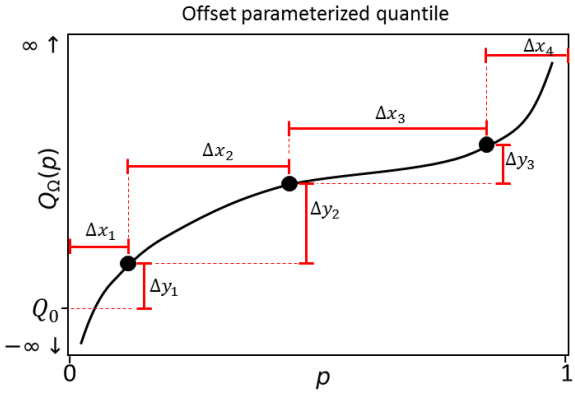
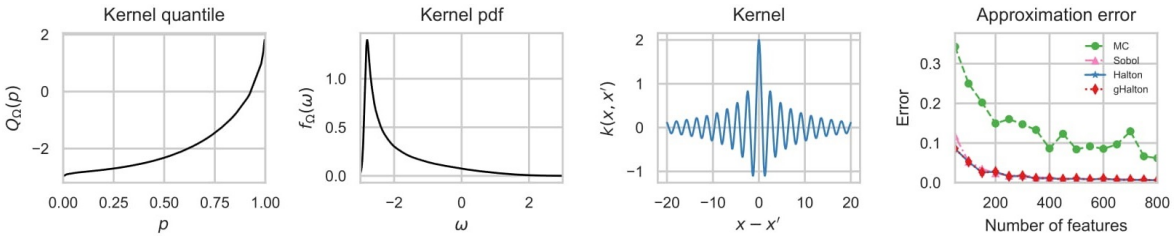


Figure 5.2. Diagram of the offset parameterised quantile

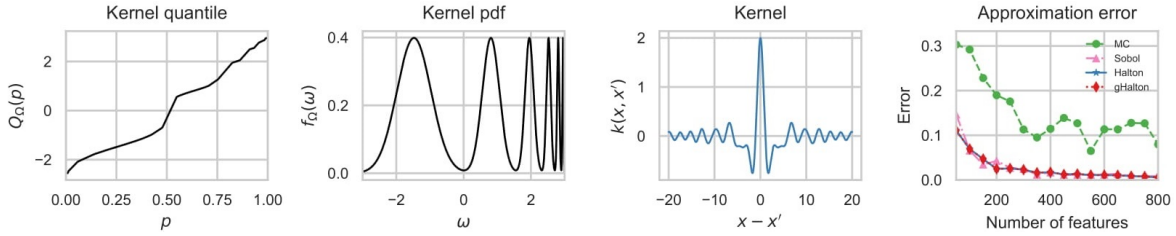
coordinates in the space of the quantile function $Q(p) \in \mathbb{R}$. This allows using isotonic regression or constrained interpolation on monotonically increasing points to interpolate the quantile function. Specifically, we use two interpolation methods that enjoy flexible coordinate based parameterisation and also ensure the necessary condition of functional monotonocity: *Linear* and *Monotonic Cubic Hermite* interpolation. While the simplest method of linear interpolation guarantees monotonicity on monotonic interpolating coordinates, naive cubic interpolation does not. Strict conditions must be set on the interpolant's tangents (Fritsch and Carlson 1980) in order to guarantee the induced kernel is valid. Specifically, we use the Piecewise Cubic Hermite Interpolation (PCHIP) method (Fritsch and Carlson 1980) however various other methods exist (Steffen 1990; Stineman 1980). As shown in Figure 5.2, we represent our N interpolating points in terms of horizontal and vertical *offsets* from some vertical origin Q_0. Specifically, for N interpolating points,

$$\begin{aligned} &\text{learn } \Delta x_1, \ldots, \Delta x_{N+1} \in [0, 1], \\ &\qquad \Delta y_1, \ldots, \Delta y_N \in \mathbb{R}, \\ &\text{s.t. } \sum_{i=1}^{N+1} \Delta x_i = 1, \\ &\qquad Q_0 \in \mathbb{R}, \end{aligned}$$

where Q_0 is a vertical origin from which all offsets are defined, Δx_i and Δy_i are offsets with respect to Q_0. To enforce the necessary constraint that all Δx_i sum up to 1, Δx_i are simply chosen in $[0, 1]$ and normalised.



(a) Skewed



(b) Combed

Figure 5.3. Section 5.5.1: Reconstructing corresponding kernels from two different quantile functions. Though not used in the algorithm, as visualised in the second column, kernels generated by BBQ can have corresponding pdfs that are arbitrarily complex and almost impossible to explicitly learn. BBQ parameterisation allows implicitly modelling such complex distributions. The last column shows normalised Frobenius errors for different sampling methods.

Even with this simple and explicit parameterisation of the quantile function, we demonstrate that we are able to learn complex kernels as well as avoid overfitting as the number of interpolating points increases, by using the negative log-marginal likelihood as a loss function.

Handling asymptotes. Valid quantile functions feature asymptotes at $p = 0$ and $p = 1$ which interpolation techniques do not guarantee. Asymptotes are constructed by using modified inverse functions $\frac{a_0}{p} + b_0$ and $\frac{a_1}{1-p} + b_1$ around $p = 0$ and $p = 1$ respectively. Parameters $a_0, b_0, a_1,$ and b_1 are chosen to ensure quantile function and derivative continuity at both interpolation end points.

5.4.4 Learning BBQ Parameters

Quantile parameters θ are learned by minimizing the inference algorithm loss $\mathcal{L}_{inference}$ (eg. regression loss). Examples of losses include Mean Square Error (MSE), computed on an

validation data, and Negative Log Marginal Likelihood (NLML), computed on training data. Depending on the quantile parameterisation used, constraints on θ may apply, and hence, one generally needs to solve,

$$\text{minimise}_\theta \quad \mathcal{L}_{inference}(\theta) \tag{5.5a}$$

$$\text{subject to} \quad g(\theta) \leq b, \tag{5.5b}$$

where g and b reflect constraints from the chosen quantile parameterisation. Extended literature on optimization provides a wide range of local and global derivative-free optimisers to efficiently solve this problem, such as ADAM, Bayesian optimization (Brochu, Cora and De Freitas 2010), DIRECT (Powell 1994) or COBYLA (Gablonsky and Kelley 2001).

Negative Log Marginal Likelihood in Bayesian Linear Regression:

For the inference setup with Bayesian Linear Regression we implement the Negative Log Marginal Likelihood with respect to the quantile parameters θ. Here, Φ is parameterised by θ and $\boldsymbol{A} := \boldsymbol{\Sigma}^{-1}$ as in Equation 2.9. Details as in (Lázaro-Gredilla et al. 2010; Bishop 2007):

$$\begin{aligned}\log p(\boldsymbol{y}|\theta) = &(\boldsymbol{y}^\top \boldsymbol{y} - \boldsymbol{y}^\top \hat{\Phi}^\top \boldsymbol{A}^{-1} \hat{(\Phi)} \boldsymbol{y})/(2\sigma_n^2) \\ &+ \frac{1}{2}\log|\boldsymbol{A}| - m\log\frac{m\sigma_n^2}{\sigma_0^2} + \frac{n}{2}\log 2\pi\sigma_n^2\end{aligned} \tag{5.6}$$

5.4.5 Relation with Spectral Mixtures

A non-negative definite kernel fully defines a probability distribution and this representation is commonly referred to as a characteristic function in statistics. It is possible to specify the uncertainty of a quantity using the characteristic function, cdf, pdf, or quantile, and with some assumptions the following equalities hold: $\phi(x) = \mathbb{E}[e^{-ix\Omega}] = \int_{\mathbb{R}} e^{-ix\omega} dF_\Omega(\omega) = \int_{\mathbb{R}} e^{-ix\omega} f_\Omega(\omega)\mathrm{d}\omega = \int_0^1 e^{-ixQ_\Omega(p)}\mathrm{d}p$. In this work, we demonstrated that the quantile representation can be conveniently used to learn complex kernels in a data-driven way. One key related work is the Spectral Mixture (SM) kernel (Wilson and Adams 2013).

BBQ features have inherent similarities to SM kernels – the former uses the quantile representation while the later uses the pdf representation. Although SM kernels are attractive, as

with any flexible kernel based methods, they are sensitive to initialization of hyperparameters. In a more generic Bayesian nonparametric setting, (Oliva et al. 2016) further scale this as a mixture of pdfs by exploiting sampling based techniques which require expensive Markov chain Monte Carlo (MCMC) techniques. In BBQs, we attempt to circumvent these issues by a different spectral representation which leads to similarly flexible parameterisation and optimization but from the orthogonal perspective of kernel approximation with Fourier features and a parametric (as opposed to a non-parametric) model.

5.5 Experiments

The BBQ algorithm with parameterisations discussed in Section 5.4 are implemented in python. We conducted a series of experiments on a computer with 16 GB RAM to validate the proposed method. All datasets were normalised between -1 and 1. Bayesian Linear Regression (BLR) (Bishop 2007) was used for inference, similar to (Lázaro-Gredilla et al. 2010). The code is available at `github.com/MushroomHunting/black-box-quantile-kernels`.

5.5.1 Quality of Kernel Approximation

In this section, we verify that the parameterised quantile functions can easily construct expressive black box kernels with popular kernels. Furthermore, by utilizing QMC sequences, we also demonstrate that the true kernel can be approximated significantly faster than standard MC samples which are used in pdf-based methods such as (Rahimi and Recht 2007a; Oliva et al. 2016). To show this, as shown in Figure 5.3, complex probability distributions whose kernels are known in closed from were chosen. In order to show the importance of QMC sampling on BBQ kernel learning, kernels were approximated using three low-discrepancy sampling methods – Sobol, Halton, and generalised Halton sequences – in addition to Monte-Carlo sampling.

In order to assess the quality of approximation, we calculated the normalised Frobenius norm error $\|K_t - \hat{K}\|_F / \|K_t\|_F$ where K_t and $\hat{K}$ are the true and approximated kernel Gram

matrices, respectively. For the true and approximate kernel Gram matrices 4000 points were sampled on the interval $[-10, 10]$ and vary the number of features over the range $[50, 1000]$.

While low-discrepancy sequences have been considered for standard kernels (Yang et al. 2014), their efficacy has not been demonstrated for the more general family of kernels induced by arbitrary quantile functions. For the various complicated quantiles in Figure 5.3, and their corresponding pdfs and kernels, irrespective of the number of features, the error is always smaller with QMC.

5.5.2 Optimisation Settings for Various Experiments

C02 Optimiser: COBYLA, Number of steps: 200, Number of interpolating points: 2.
Passenger Optimiser: COBYLA, Number of steps: 200, Number of interpolating points: 2.
Concrete Optimiser: Controlled Random Search (CRS) (Johnson 2014), Number of steps: 200, Number of interpolating points: 12.
Noise Optimiser: Controlled Random Search (CRS), Number of steps: 200, Number of interpolating points: 12.
Pores Texture Optimiser: adam, adam β_1: 0.9, adam β_2: 0.999, Learning Rate: 0.5e-2, Number of steps: 225, Number of interpolating points: 15
Rubber Texture Optimier: adam, adam β_1: 0.9, adam β_2: 0.999, Learning Rate: 0.5e-3, Number of steps: 225, Number of interpolating points: 25
Tread Texture Optimiser: adam, adam β_1: 0.9, adam β_2: 0.999, Learning Rate: 0.5e-3, Number of steps: 225, Number of interpolating points: 31

5.5.3 Various Aspects of Learning BBQ Kernels

Using two toy datasets (Figure 5.6) we illustrate that Bayesian linear regression with BBQ features can model nonlinear patterns. Interestingly, rather than handcrafting and composing periodic and RBF kernels, the BBQ parameterisation allowed automatically learning the appropriate kernel, making both interpolation and extrapolation possible. Figure 5.7a shows

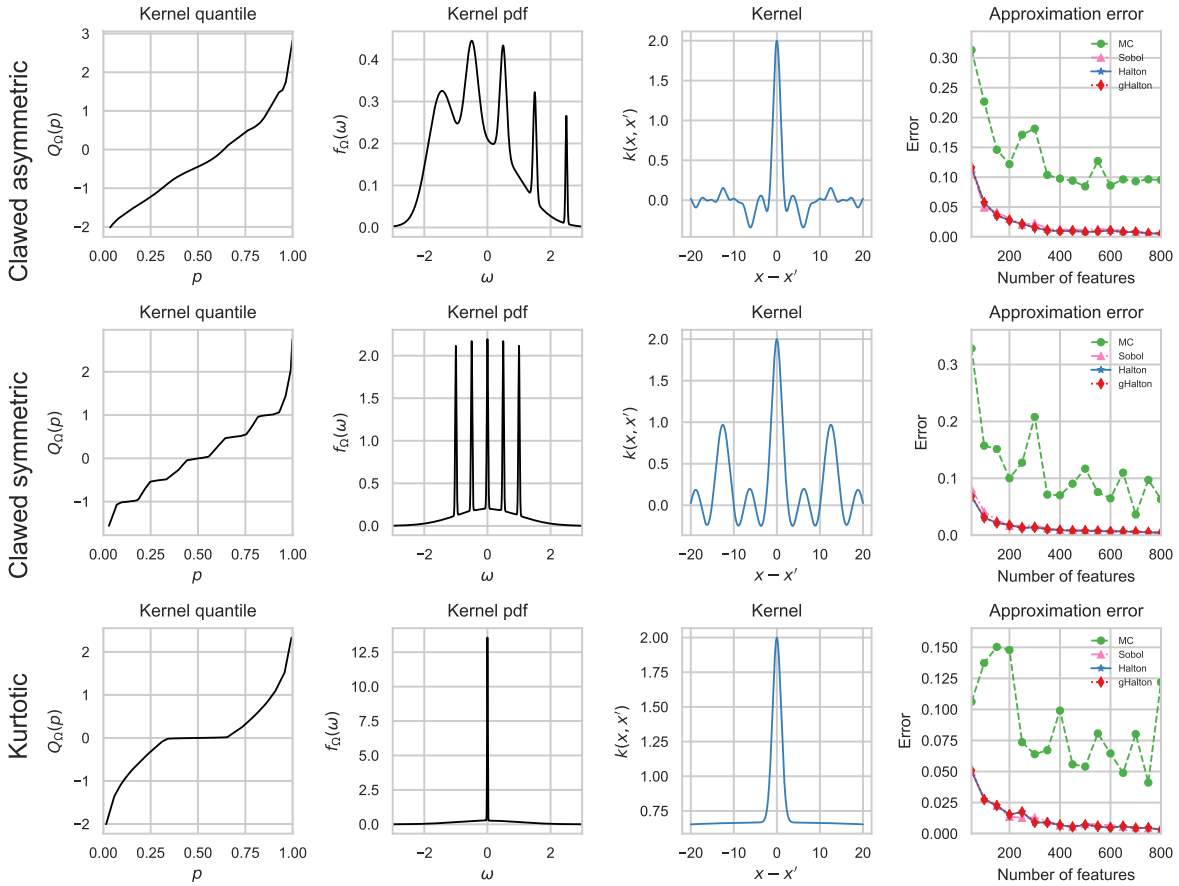


Figure 5.4. Gram matrix reconstruction error corresponding to various quantile, pdf, and kernels. Relative errors are measured with the approximating Gram matrix measure in the normalised Frobenius norm.

the error surface. The objective is to determine quantile parameters θ that minimise the loss. An instance of gradient descent to find an optima is shown in Figure 5.7b.

In order to demonstrate the BBQ algorithm's capacity to learn highly complex kernels, two toy datasets (Figure 5.6) were used to learn the quantile functions. Even though pdfs and quantiles are merely two different representions of the probability, the proposed quantile parameterisation leads to generating flexible and arbitrarily complex kernels. Highlighting this property, as shown in Figure 5.8, the proposed method was able to learn complex quantiles that would otherwise have been challenging, if not impossible, to pragmatically learn even with a large finite mixture of pdfs or a composition of known kernels.

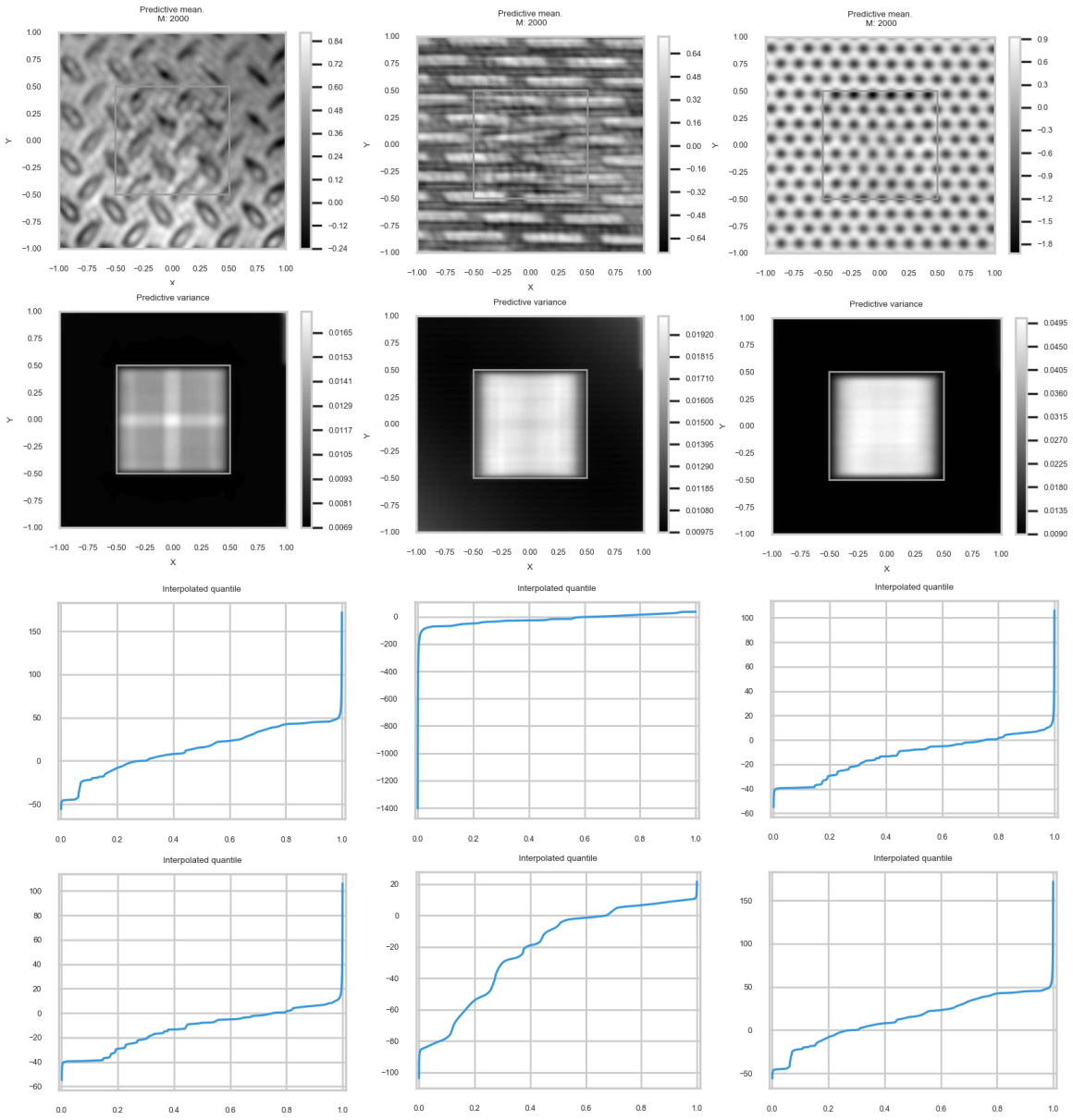


Figure 5.5. Predictive mean and variance for the texture datasets from models learned with BBQ quantiles. The two quantiles (for the vertical and horizontal patterns) for each pattern is shown below each dataset. Note that the quantiles are similar for the third pattern because it is symmetric.

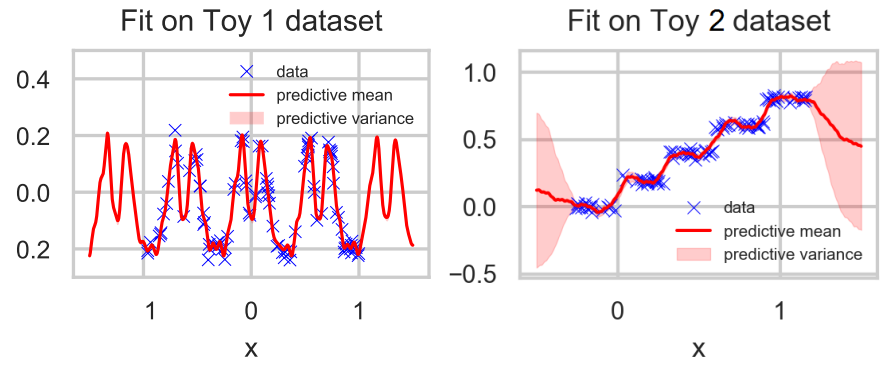
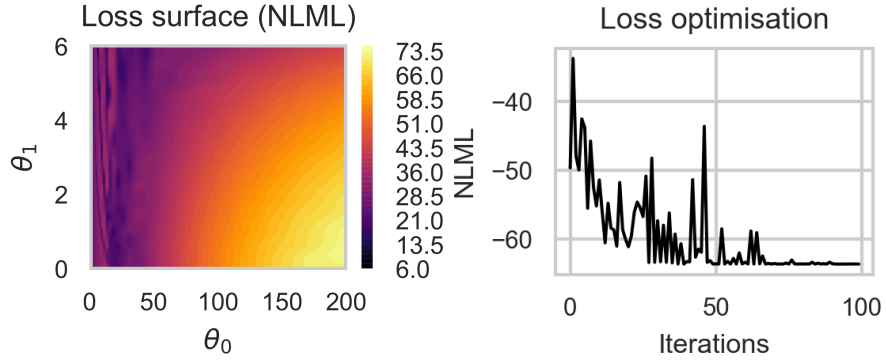


Figure 5.6. Section 5.5.3: Fits on toy datasets (periodic and steps) using BBQ features.



(a) (b)

Figure 5.7. Section 5.5.3: An example of loss (NLML) surface for 2-parameter quantile and loss per optimization iteration.

5.5.4 Learning Complex Patterns and Extrapolation

We experiment on various real-world datasets from the UCI machine learning repository*. *CO2* and *passenger* are periodic datasets evaluating extrapolation capabilities. Datasets *concrete* and *noise* feature higher dimensions of 5 and 8 respectively. We further tests on three in-filling texture datasets from (Wilson et al. 2014) *pores*, *rubber tread*. For multidimensional datasets, one black-box quantile per dimension was learned. For extrapolation datasets, *passenger* and *CO2* we compose BBQ features with a linear kernel $k_{lin} + k_{BBQ}$ and $k_{lin} + k_{lin} \times k_{BBQ}$, respectively.

*https://archive.ics.uci.edu/ml/index.php

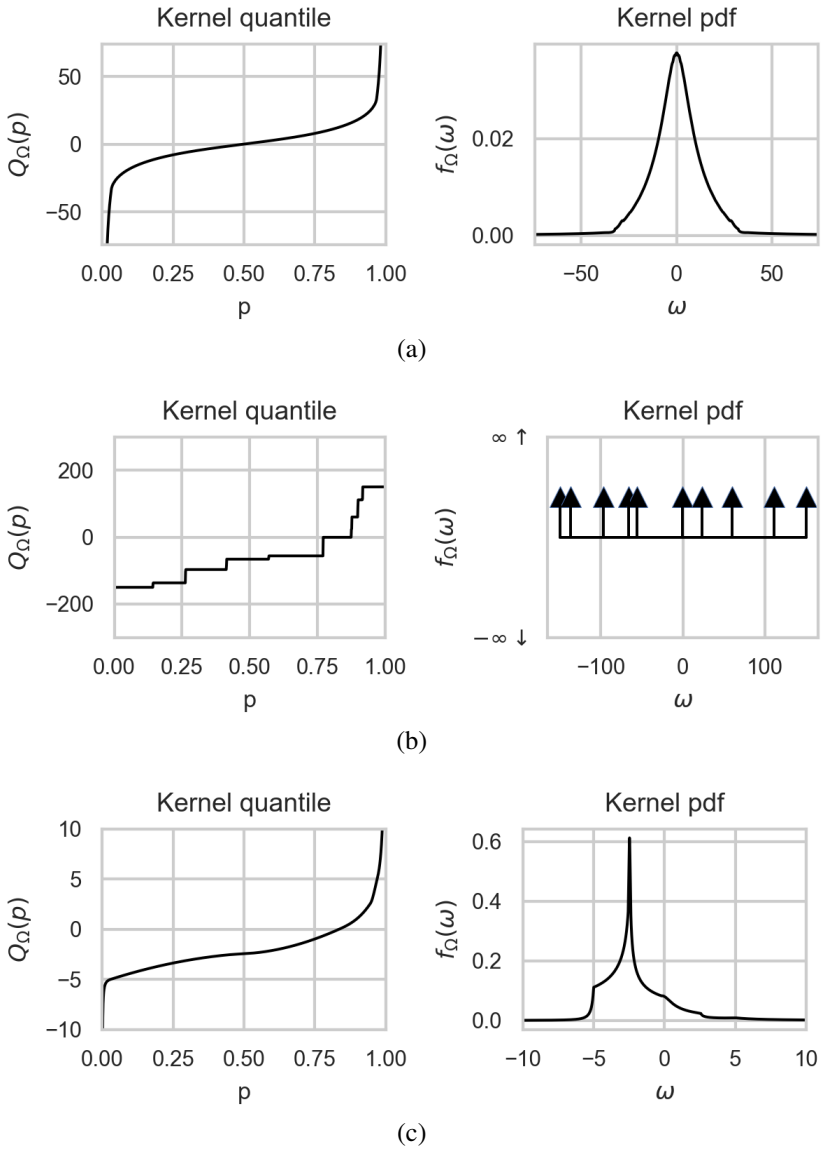


Figure 5.8. Section 5.5.3: Learned kernel quantile functions and their corresponding kernel pdfs. Slight variation in quantile function result varied PDFs: unimodal, diracs, multimodal or skewed.

Conventionally, different Gaussian process methods such as (Hensman, Matthews and Ghahramani 2015; Titsias 2009) with fixed kernels are used to capture nonlinear patterns. In addition to these, we compare Bayesian linear regression augmented with BBQ features against the standard Random Fourier Features (RFFs) for the RBF kernel (Rahimi and Recht 2007a) as well as Spectral Mixture (SM) kernels (Wilson and Adams 2013). Since SM kernels are somewhat sensitive to initialization, we run SM kernels 10 times and report the best

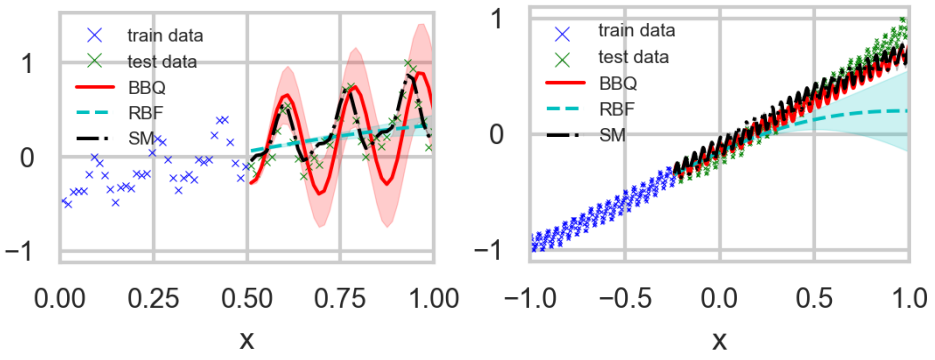
result. In order to compare textures that are in a regular grid, we used 3000 inducing points with bicubic interpolation, in a sparse approximation method akin to, though not exactly the same, KISS-GP (Wilson and Nickisch 2015). We also compare with Doubly Stochastic Deep GPs (Salimbeni and Deisenroth 2017) which can learn complex patterns in data because of the deep structure, though a complex kernel is not explicitly learned.

Methods are evaluated in terms of RMSE and Mean Negative Log Loss (MNLL). The smaller these metrics are the more accurate the model is. Unlike RMSE, MNLL takes into account both the mean and variance of predictions (Bishop 2007). Occasionally full GPs with SM kernels perform better which could be explained by better estimates of uncertainty. Results displayed in Table 5.1 show the superior performance of using BBQ features. In comparison, indicating the importance of learning the kernel, the standard RFFs (RFF-RBF) consistently scores higher errors.

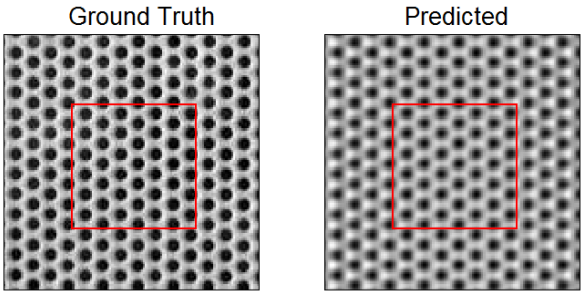
Fits on individual data sets for various methods are displayed in Figure 5.9, showing both SM and BBQ identify the data periodicity, while RFF-RBF only manages to follow the global trend. All three textures with corresponding quantiles (one quantile per dimension) are shown in Figure 5.5. Finally, BLR with BBQ features has complexity $\mathcal{O}(M^2N)$ resulting in much faster runtime than SM of complexity $\mathcal{O}(N^3)$. This difference is especially noticeable on moderately sized datasets such as textures, where BLR-BBQ runs in minutes on a desktop computer compared to hours for SM.

5.5.5 Effect of the Number of Quantile Parameters

We designed an experiment to show the empirical influence of increasing number of quantile parameters on BBQ regression error. See Figure 5.10 for results on CO2 dataset. The training and testing errors decrease with the increased number of parameters and then levels off. In this case, training by optimizing NLML (using Bayesian linear regression as the model) does not lead to overfitting, even when the model is overparameterised. Nevertheless, note that this is not a straightforward comparison because the errors depend not only on



(a) Airline dataset (b) CO2 dataset



(c) Pores dataset

Figure 5.9. Section 5.5.4: (a)-(b) Extrapolation on two real datasets. On both datasets, BLR with BBQ, SM both discover periodicity while RBF only finds the general trend. This shows BBQ features can easily be composed with nonstationary kernels (here linear) to learn global trends. (c) Intra-filling task on *pores* texture dataset (train set outside red square, test set inside) and prediction with 300 BBQ features respectively.

the quantile parameterisation, but also on the inference model, loss function, and the optimiser.

5.5.6 Kernel Composition

It is straightforward to combine BBQ features (which are approximating a stationary kernel) with other potentially kernels (that are nonstationary) to enable additional expressiveness. We support this notion with Claim 1.

Table 5.1. Section 5.5.4: Loss metrics on all real datasets. We used RMSE and MNLL for spectral mixture (SM), RBF with random Fourier features (RFF-RBF), the proposed technique (BBQ), Sparse Gaussian Process Regression (SGPR) (Titsias 2009), Sparse Variational Gaussian Process (SVGP) (Hensman, Matthews and Ghahramani 2015), and Doubly Stochastic Deep GP (DSDeepGP) (Salimbeni and Deisenroth 2017).

Loss	Method	CO2	Passenger	Concrete	Noise	Rubber	Pores	Tread
RMSE	BBQ	**0.068**	**0.096**	**0.124**	0.138	**0.248**	**0.256**	**0.114**
	SM	0.083	0.102	0.465	**0.132**	0.395	0.795	0.513
	RFF-RBF	0.245	0.270	0.164	0.184	0.687	1.739	0.326
	SGPR	0.190	0.262	0.138	0.164	0.315	0.586	0.276
	SVGP	0.191	0.262	0.176	0.201	0.3176	0.5853	0.1436
	DSDeepGP	0.446	0.396	0.178	0.174	0.3256	0.5853	0.1508
MNLL	BBQ	**-1.242**	**-0.610**	**-0.577**	-0.173	1.336	**0.337**	**-0.754**
	SM	-0.604	-0.441	0.743	**-0.570**	**0.523**	1.386	1.022
	RFF-RBF	-0.368	14.310	3.173	7.569	18.351	122.689	1.057
	SGPR	-0.695	0.516	-0.545	-0.392	0.268	0.885	0.328
	SVGP	-0.686	0.503	-0.308	-0.181	0.274	0.885	-0.501
	DSDeepGP	1.454	1.361	0.111	0.032	0.306	0.884	-0.339

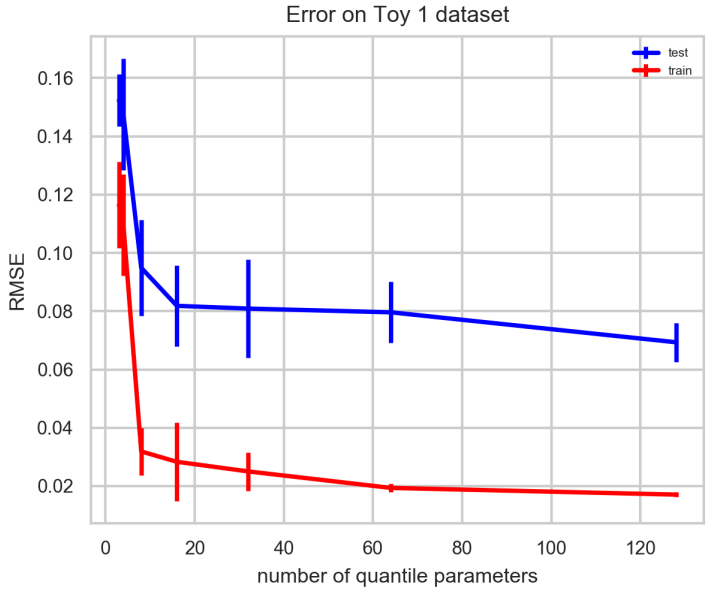


Figure 5.10. Effect of increasing number of quantile parameters.

CLAIM 1. *We have the equivalence of kernel composition operations in the kernel space and feature space (Shawe-Taylor and Cristianini 2004).*

$$\begin{aligned}(k_1 + k_2)(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \\ &= [\phi_1(\mathbf{x})\phi_2(\mathbf{x})][\phi_1(\mathbf{x}')\phi_2(\mathbf{x}')]^\top,\end{aligned} \tag{5.7}$$

defines the sum of two feature maps, and

$$\begin{aligned}(k_1 \times k_2)(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') \times k_2(\mathbf{x}, \mathbf{x}') \\ &= \sum_{i}^{n,m} \phi_{1,2}^{(i)}(\mathbf{x}) \phi_{1,2}^{(i)}(\mathbf{x}'),\end{aligned} \tag{5.8}$$

defines the product of two feature maps, where $\phi_{1,2}(\mathbf{x}) = \phi_1 \times \phi_2$ is the Cartesian product.

In experiments, to handle multiple dimensions, different quantiles were used on a per-dimension basis analogous to Automatic Relevance Determination (ARD) in kernel based methods (MacKay 1998; Neal 2012). As an alternative method to deal with multi-dimentional data, it is possible to concatenate feature vectors for multiple dimensions. Consider a D-dimensional dataset having N data points $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D) \in \mathbb{R}^{N \times D}$. If f is an inference model such as linear regression, a composite model similar to Generalised Additive Model (GAM) (Hastie 2017) $f(\mathbf{X}) = \sum_{d=1}^{D} f_d(\mathbf{x}_d)$ can be composed. For instance, consider the linear model $f(\mathbf{x}) = \mathbf{w}\hat{\phi}^\top(\mathbf{x}; \theta)$ with BBQ features $\hat{\phi} \in \mathbb{R}^{N \times 2M}$ and corresponding coefficients $\mathbf{w} \in \mathbb{R}^{1 \times 2M}$. Then, the model for multidimensional data is $f(\mathbf{X}) = \sum_{d=1}^{D} \mathbf{w}_d \hat{\phi}^\top(\mathbf{x}_d; \theta_d) = \mathbf{W}^\top \hat{\Phi}(\mathbf{X}; \Theta)$ where $\hat{\Phi}(\mathbf{X}; \Theta) = \|_{d=1}^{D} \hat{\phi}(\mathbf{x}_d; \theta_d) \in \mathbb{R}^{N \times 2MD}$ and $\mathbf{W} = \|_{d=1}^{D} \mathbf{w}_d \in \mathbb{R}^{1 \times 2MD}$ with $\|$ indicating vector concatenation. Algorithmically, it is possible to learn an individual quantile for each dimension and then concatenate corresponding features to create a $2MD$ dimensional feature vector for the inference algorithm.

Although the aforementioned treatments to handle multi-dimensional data are straightforward, note that they cannot capture correlation between covariates. Though the out of scope of this chapter, the parameterised quantile representation naturally opens the door to explicitly representing multidimensional variations using copulas that are widely studied in statistics (Nelsen 2007).

5.6 Summary

In summary, this chapter introduces a novel technique to automatically represent and learn highly expressive kernels that fit the data best. To do this, we propose a parameterised a

quantile function, that represents an *arbitrary* kernel in its Fourier domain, and then learn its parameters using stochastic gradient descent. With the use of Bayesian linear regression, we have shown that inducing a kernel by a quantile function allows one to additionally take advantage of Quasi-Monte Carlo sampling with Random Fourier features to reduce approximation error.

We demonstrated the efficacy of the propose BBQ features on a large set of synthetic and real datasets highlighting the various possibilities that a flexible kernel allows. Our algorithm performed competitively with similar state-of-the-art nonparametric and parametric kernel learning methods from the literature. One burden that remains on our quest to relax kernel learning to theme of *let the data speak* is that while arbitrarily flexible, the BBQ parameterisation is still only approximating a *stationary* kernel. That is to say, it is *fundamentally* unable to correctly represent nonstationary processes. This motivates us towards the next chapter which aims to address this limitation through a novel nonstationary kernel learning technique.

CHAPTER 6

Warped Input Measures for Nonstationary Kernel Learning

In this final chapter we establish a general form of explicit, input-dependent, measure-valued input warpings for learning nonstationary kernels. This chapter expands upon the capabilities of the previous Chapter 5 by addressing kernel learning to nonstationary representations while remaining in the paradigm of kernels in the Fourier domain. While stationary kernels are ubiquitous and simple to use, they struggle to adapt to functions that vary in smoothness with respect to the input. The proposed learning algorithm warps inputs as conditional Gaussian measures that control the smoothness of a standard stationary kernel resulting in an implicit nonstationary kernel representation. This construction allows us to capture nonstationary patterns in the data while providing an intuitive inductive bias. The resulting method is based on sparse spectrum Gaussian processes, enabling closed-form solutions, and is extensible to a stacked construction to capture more complex patterns. The method is extensively validated alongside related algorithms on synthetic and real-world datasets. We demonstrate a remarkable efficiency in the number of parameters of the warping functions in learning problems with both small and large data regimes.

6.1 Introduction

Many interesting real-world phenomena exhibit characteristics, such as varying smoothness, across their domain. Simpler phenomena that *do not* exhibit such variation may be called *stationary*. The typical kernel-based learner canonically relies on a *stationary* kernel function, a measure of 'similarity', to define the prior beliefs over the function space. Such a kernel, however, cannot represent desirable *nonstationary* nuances, like varying spatial smoothness

and sudden discontinuities. Restrictive stationary assumptions do not generally hold and limit applicability to interesting problems, such as robotic control and reinforcement learning (Ng et al. 2006), spatial mapping (Ton et al. 2018), genetics (Friedman et al. 2000), and Bayesian optimisation (Martinez-Cantin 2017). One obvious way to alleviate the problem of finding the appropriate kernel function given one's data is hyperparameter optimisation. However, for a GP with a stationary kernel, even if the *optimal* set of hyperparameters were found, it would be insufficient if our underlying response were fundamentally nonstationary with respect to the observed inputs. This provides a general motivation for the algorithmic contribution of this work.

In this chapter we propose a method for nonstationary kernel learning, based on sparse spectral kernel representations. Our method is linear in complexity with respect to the number of data points and is simultaneously able to extract nonstationary patterns. In our setup, we consider the problem of learning a function $f : \mathcal{X} \to \mathbb{R}$ as a nonstationary Gaussian process. We decompose f as:

$$f(\mathbf{x}) = \mathbb{E}[u \circ \mathbf{m}(\mathbf{x})|u] \, , \quad \mathbf{x} \in \mathcal{X} \, , \tag{6.1}$$

where $\circ$ denotes function composition, $u : \mathcal{Q} \to \mathbb{R}$ is a function over a latent space $\mathcal{Q}$, and $\mathbf{m}(\mathbf{x})$ represents the warped input. If u has covariance function k_u, the resulting f follows a GP with covariance $k_f(\mathbf{x}, \mathbf{x}') = \mathbb{E}[k_u(\mathbf{m}(\mathbf{x}), \mathbf{m}(\mathbf{x}'))]$. The latter constitutes a nonstationary kernel.

To model u as a stationary GP on $\mathcal{Q}$, we propose a formulation for $\mathbf{m} : \mathcal{X} \to \mathcal{Q}$, which is based on a locally affine stochastic transform:

$$\mathbf{m}(\mathbf{x}) = \mathbf{G}(\mathbf{x})\mathbf{x} + \mathbf{h}(\mathbf{x}) \, , \tag{6.2}$$

where $\mathbf{G}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$ are Gaussian processes. Intuitively, the matrix $\mathbf{G}$ scales the inputs, with a similar effect to what length-scales have on stationary kernels (Rasmussen 2004), but which now varies across the space, while $\mathbf{h}$ allows for arbitrary shifts.

The conditional expectation (6.1) also corresponds to the composition of a function on $\mathcal{Q}$ with a measure (Bauer 1981) on $\mathcal{Q}$, which is actually a function of $\mathbf{x} \in \mathcal{X}$. In our case,

the measure-valued warpings are Gaussian probability measures, which we parameterise as Gaussian process conditioned on *pseudo-training points*. In particular, we use sparse spectrum Gaussian processes (Lázaro-Gredilla et al. 2010) due to their scalability and availability of closed-form results for Gaussian inputs (Pan et al. 2017).

6.2 Contributions

This chapter focuses on the problem of learning nonstationary spectral approximations of kernel embeddings. While most kernels and their approximations assume stationary underlying processes this is an often unrealistic assumption to hold for real-world data in general. Furthermore, it is often difficult to learn approximations of conventional nonstationary kernel functions simply because they are non-separable. We will demonstrate a way to unify conventional stationary spectral kernel approximations with input warpings. Specifically, we show it is possible to effectively bootstrap off simpler stationary kernels composed with input-dependent warping functions. To this end we present the following contributions:

(1) We propose a new method to learn nonstationary Gaussian process models via input warping. We introduce the use of a measure-valued, self-supervised and input-dependent warping function as a natural improvement for sparse spectrum Gaussian processes. We term this *sparse spectrum warped input measures* (SSWIM);
(2) We propose a self-supervised training scheme for representing the warping function allowing us to cleanly represent the latent measure valued warping; and
(3) We propose a simple extension to multiple levels of warping by propagating moments.

6.3 Related Work

Foundational work (Higdon, Swall and Kern 1999; Paciorek and Schervish 2004) on kernel-based nonstationarity necessitated manipulation of the kernel function with expensive inference procedures. Recent spectral representation of kernel functions have emerged with

Bochner's theorem (Bochner 1932). In this paradigm, one constructs kernels in the Fourier domain via *random Fourier features* (RFFs) (Rahimi and Recht 2007a; Rahimi and Recht 2008) and extensions for nonstationarity via the generalised Fourier inverse transform (Samo and Roberts 2015; Remes, Heinonen and Kaski 2017; Ton et al. 2018; Sun et al. 2018). While general, these methods suffer from various drawbacks such as expensive computations and overfitting due to over-parameterised models (Ton et al. 2018).

More expressive modelling frameworks (Calandra et al. 2016; Wilson, Knowles and Ghahramani 2012; Sampson and Guttorp 1992; Anderes, Stein et al. 2008) have played a major role in expanding the efficacy of kernel-based learning. Perhaps the most well known in the recent literature is Deep Kernel Learning (Wilson et al. 2016) and the *deep Gaussian process* (Damianou and Lawrence 2013) and heretofore its various extensions (Salimbeni and Deisenroth 2017; Cutajar et al. 2017; Bui et al. 2016). While functionally elegant, methods like DKL and DGP often rely on increasing the complexity of the composition to produce expressiveness and are often unsuitable or unwieldy in practice occasionally resulting in performance worse than stationary inducing point GPs (Salimbeni and Deisenroth 2017). We remark a notable difference between DGP and SSWIM is one should interpret our pseudo-training points as hyperparameters of the kernel as opposed to parameters of a variational approximation.

Simple bijective input warpings were considered in (Snoek et al. 2014) for transforming nonstationary functions into more well behaved functions. In (Heinonen et al. 2016) the authors augment the standard GP model by learning nonstationary data-dependent functions for the *hyperparameters* of a nonstationary squared-exponential kernel (Gibbs 1997). Their method, however, is limited to low dimensions. More recently, the work of (Hegde et al. 2019) has explored a dynamical systems view of input warpings by processing the inputs through time-dependent differential fields. Less related models presented in (Wang and Neal 2012; Dutordoir et al. 2018; Snelson, Ghahramani and Rasmussen 2004) involve *output warping* non-Gaussian likelihoods and heteroscedastic noise.

6.4 Preliminaries

We start by reviewing relevant background with regards to kernel methods for Gaussian process regression. In particular, we focus on the sparse spectrum approximation to GPs (Lázaro-Gredilla et al. 2010), which we use to formulate nonstationary kernels.

6.4.1 Gaussian Processes

As described in Section 2.2.2, GPs are a method for representing a distribution over functions. We provide the definition here for completeness. Suppose our goal is to learn a function $f : \mathbb{R}^D \to \mathbb{R}$ given IID data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, with each data pair related through

$$y = f(\mathbf{x}) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma_n^2), \tag{6.3}$$

where ϵ is IID additive Gaussian noise. A Gaussian process is a distribution on functions f over an input space $\mathcal{X} \subseteq \mathbb{R}^D$ such that any finite set of inputs $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathcal{X}$ produces a multivariate normal distribution of response variables $\mathbf{f}_N := [f(\mathbf{x}_1), ..., f(\mathbf{x}_N)]^\mathsf{T}$:

$$\mathbf{f}_N \sim \mathcal{N}(\mathbf{m}_N, \mathbf{K}_N) \,, \tag{6.4}$$

where $\mathbf{m}_N = m(\mathbf{x}_1, ..., \mathbf{x}_N)$ is the mean vector, and $\mathbf{K}_N = \{k(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j}$ with kernel k.

6.4.2 Approximate GP in Feature Space

Considering the inner product structure $\langle \cdot, \cdot \rangle$ we can represent the kernel as $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$ for $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, where $\phi : \mathcal{X} \mapsto \mathcal{H}$ is a mapping from the input space $\mathcal{X}$ into potentially infinite-dimensional Hilbert space $\mathcal{H}$. Perhaps the most popular kernel is the stationary squared-exponential kernel $k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp(-2\gamma^2 \|\mathbf{x} - \mathbf{x}'\|_2^2)$ with parameters σ^2 and γ^2. It is also well known that it is possible to combine standard kernels to arrive at more elaborate kernel structures (Duvenaud et al. 2013). Full GP inference is a challenging problem naively occurring in $\mathcal{O}(N^3)$ complexity as a consequence of having to invert an (N, N) Gram matrix. Much work as gone into scaling GP inference using pseudo-inputs or

inducing points in which we avoid operating on the full data space and work with a lower complexity subset of size M where $M \ll N$ (Hensman, Fusi and Lawrence 2013b). An alternative perspective on approximate GP inference is to consider the *feature space* view of the kernel function using Bochner's theorem (Bochner 1932). Under this view, *random Fourier features* (Rahimi and Recht 2007a; Rahimi and Recht 2008) decompose the kernel function in terms of Fourier features based on a finite approximation to the kernel's spectrum.

As presented by (Rahimi and Recht 2007a), the Fourier transform of any shift-invariant positive-definite kernel $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ yields a valid probability distribution p_k, so that k is approximately:

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{\boldsymbol{\omega} \sim p_k}[\cos(\boldsymbol{\omega}^\mathsf{T}(\mathbf{x} - \mathbf{x}'))] \approx \boldsymbol{\phi}(\mathbf{x})^\mathsf{T} \boldsymbol{\phi}(\mathbf{x}')\,, \tag{6.5}$$

where $\boldsymbol{\phi}$ corresponds to the approximate feature map:

$$\boldsymbol{\phi}(\mathbf{x}) = \sqrt{\frac{2}{M}} \left[\cos\left(\boldsymbol{\omega}_1^\mathsf{T}\mathbf{x}\right), \ldots, \cos\left(\boldsymbol{\omega}_M^\mathsf{T}\mathbf{x}\right), \sin\left(\boldsymbol{\omega}_1^\mathsf{T}\mathbf{x}\right), \ldots, \sin\left(\boldsymbol{\omega}_M^\mathsf{T}\mathbf{x}\right) \right] \in \mathbb{R}^{2M}. \tag{6.6}$$

where $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^{2M}$. The Fourier interpretation of kernels has gained attention in recent years because of the simplicity, solid theoretical basis (Sutherland and Schneider 2015; Choromanski et al. 2018), and superiority in various applications (Rajeswaran et al. 2017). Using the feature map above we are able to define an approximate GP termed the Sparse Spectrum Gaussian Process (SSGP) (Lázaro-Gredilla et al. 2010).

DEFINITION 2. *(Sparse Spectrum Gaussian Process) (Lázaro-Gredilla et al. 2010) The Sparse Spectrum Gaussian Process (SSGP) is a GP with kernels defined on finite-dimensional and randomised feature map $\boldsymbol{\phi}$,*

$$k(x, x') = \boldsymbol{\phi}(x)^\mathsf{T} \boldsymbol{\phi}(x') + \sigma_n^2 \delta(x - x') \tag{6.7}$$

where the function δ denotes the Kronecker delta.

The second additive term accounts for additive zero mean noise from (6.3). Due to the feature map from (6.6), the SSGP is a Gaussian distribution over the feature weights $\mathbf{w} \in \mathbb{R}^{2M}$. If

we assume the weight prior is $\mathcal{N}(\mathbf{0}, \mathbf{I})$, after conditioning on data $\mathcal{D}$ the posterior distribution of $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\alpha},\, \sigma_n^2 \mathbf{A}^{-1})$, where:

$$\boldsymbol{\alpha} = \mathbf{A}^{-1}\boldsymbol{\Phi}\mathbf{y}, \tag{6.8}$$

$$\mathbf{A} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathsf{T}} + \sigma_n^2 \mathbf{I}, \tag{6.9}$$

following from Bayesian Linear Regression (Bishop 2007). The design matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), ..., \boldsymbol{\phi}(\mathbf{x}_N)]$ and column vector $\mathbf{y} = [y_1, ..., y_N]^{\mathsf{T}}$ are given directly by the data $\mathcal{D}$. The posterior distribution over the response y given an $\mathbf{x}$ is exactly Gaussian:

$$p(f(\mathbf{x})|\mathbf{x}) = \mathcal{N}\left(\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(x), \sigma_n^2 \|\boldsymbol{\phi}(x)\|^2_{\mathbf{A}^{-1}}\right), \tag{6.10}$$

where we define $\|\mathbf{v}\|^2_{\boldsymbol{\Sigma}} := \mathbf{v}^{\mathsf{T}}\boldsymbol{\Sigma}\mathbf{v}$. Multivariate outputs can be modelled as conditionally independent GPs for each output dimension or jointly by encoding the covariance between the outputs as a vector-valued GP (Alvarez, Rosasco and Lawrence 2011).

6.4.3 Kernels with Gaussian Inputs

Input warping methods for nonstationarity require some *functional form* of the warping. For example in (Wilson et al. 2016) the warping is a deterministic neural network, in (Snoek et al. 2014) it is a deterministic monotonic function, and in (Hegde et al. 2019) the warping is defined through a stochastic differential equation. In contrast, with our method, we propose to explicitly learn an operator-valued input-dependent function $\mathbf{G}$ that combines with the original input data before being passed as uncertain inputs into an SSGP. The only difference is that we also wish to propagate any uncertainty on the warping function into the top-level function u which will produce the final predictions taking that underlying uncertainty into account. In our formulation of nonstationary kernel, we take form the kernel-based on expectations with respect to distributions conditioned on the inputs. In the sparse-spectrum formulation, the expected kernel is simply the result of the inner product between the expected feature map of each input, due to the linearity of expectations. For the case of Gaussian inputs, results by (Pan et al. 2017) then allow us to compute the expected feature map in closed form. For a

Gaussian input $\tilde{\mathbf{x}} \sim \mathcal{N}(\hat{\mathbf{x}}, \boldsymbol{\Sigma})$, we have:

$$\mathbb{E}[\cos(\boldsymbol{\omega}^\mathsf{T}\tilde{\mathbf{x}})] = \exp\left(-\frac{1}{2}\|\boldsymbol{\omega}\|^2_{\boldsymbol{\Sigma}}\right)\cos(\boldsymbol{\omega}^\mathsf{T}\hat{\mathbf{x}})\,, \tag{6.11}$$

$$\mathbb{E}[\sin(\boldsymbol{\omega}^\mathsf{T}\tilde{\mathbf{x}})] = \exp\left(-\frac{1}{2}\|\boldsymbol{\omega}\|^2_{\boldsymbol{\Sigma}}\right)\sin(\boldsymbol{\omega}^\mathsf{T}\hat{\mathbf{x}}). \tag{6.12}$$

What is important to note here is the exponential constant which scales the standard feature by a value proportional to the uncertainty in the warped input. That is to say, expectations that take on larger (predictive) uncertainties will be *smaller* than if we did not take this uncertainty into account.

6.5 Sparse Spectrum Warped Input Measures

In this section we introduce the main contribution: *sparse spectrum warped input measures* (SSWIM). The key idea in our work is based on two crucial steps. First, we construct a stochastic vector-valued mapping modelling the input warping $\mathbf{m} : \mathcal{X} \to \mathcal{Q}$, where $\mathcal{X} \subseteq \mathbb{R}^D$ represents the raw input space and $\mathcal{Q}$ is the resulting warped space. A top-level GP modelling $u : \mathcal{Q} \to \mathbb{R}$ then estimates the output of the regression function $f : \mathcal{X} \to \mathbb{R}$. To learn the warping, each lower-level SSGP is provided with *pseudo-training* points, which are learned jointly with the remaining hyper-parameters of both GP models.

It is important to note that the pseudo-training points are *free parameters* of the latent warping function and therefore *hyperparameters of the top-level function*. Furthermore, while our construction and implementation assumes a pseudo-training dimensionality equal to that of the original data $\dim \mathcal{X} = \dim \mathcal{Q}$, nothing preventing us from embedding the input warping into a lower $\dim \mathcal{Q} \ll \dim \mathcal{X}$ or higher $\dim \mathcal{Q} \gg \dim \mathcal{X}$ dimensional manifold.

6.5.1 Warped Input Measures

To model and propagate the uncertainty on the warping operator $\mathbf{G}$ through the predictions, we start by modelling $\mathbf{G} : \mathcal{X} \to \mathcal{L}(\mathcal{X})$ as a Gaussian process. Then every linear operation on $\mathbf{G}$ results in another GP (Jidling et al. 2017), so that $\mathbf{m}(\mathbf{x}) = \mathbf{G}(\mathbf{x})\mathbf{x} + \mathbf{h}(\mathbf{x})$, for a

deterministic $\mathbf{x} \in \mathcal{X}$, is Gaussian. Similarly, as expectations constitute linear operations, the *expected value* of the GP u under the random input given by the warping is also Gaussian (Oliveira, Ott and Ramos 2019). Marginalising $\mathbf{m}$ out of the predictions, i.e. inferring the expected value of f under the distribution of $\mathbf{m}$, $\hat{f}(\mathbf{x}) = \mathbb{E}[u \circ \mathbf{m}(\mathbf{x})|u]$, we end up with a final GP, which has analytic solutions.

From Section 6.4.3, the uncertain-inputs predictions from $\hat{u} = \mathbb{E}[u(\tilde{\mathbf{x}})|u]$ for $\mathbf{m}(\mathbf{x}) \sim \mathcal{N}(\hat{\mathbf{m}}(\mathbf{x}), \boldsymbol{\Sigma}_{\mathbf{m}}(\mathbf{x}))$ are given by the SSGP predictive equations in (6.10) using the expected feature map for $\mathbb{E}[\boldsymbol{\phi}(\mathbf{m}(\mathbf{x}))]$. Equation (6.12) then allows us to compute $\mathbb{E}[\boldsymbol{\phi}(\mathbf{m}(\mathbf{x}))]$ in closed form for a given mean $\hat{\mathbf{m}}(\mathbf{x})$ and covariance matrix $\boldsymbol{\Sigma}_{\mathbf{m}}(\mathbf{x})$. The general form of the covariance matrix $\boldsymbol{\Sigma}_{\mathbf{m}}(\mathbf{x})$ for $\mathbf{m}(\mathbf{x})$ involves dealing with a fourth order tensor describing the second moment of $\mathbf{G}$. For this work, however, we consider a particular case with a more elegant formulation and yet flexible enough to accommodate for a large variety of warpings.

LEMMA 1. *Let* $\mathbf{v} \sim \mathcal{N}(\hat{\mathbf{v}}, \boldsymbol{\Sigma}_{\mathbf{v}})$ *denote a Gaussian random vector and* $\mathbf{x} \in \mathbb{R}^D$ *an arbitrary point. Then* $\mathbf{z} := \mathbf{v} \odot \mathbf{x}$ *is Gaussian,* $\mathbf{z} \sim \mathcal{N}(\hat{\mathbf{z}}, \boldsymbol{\Sigma}_{\mathbf{z}})$*, with mean and covariance matrix given by:*

$$\hat{\mathbf{z}} = \hat{\mathbf{v}} \odot \mathbf{x} \tag{6.13}$$

$$\boldsymbol{\Sigma}_{\mathbf{z}} = \mathbf{x}\mathbf{1}^{\mathsf{T}} \odot \boldsymbol{\Sigma}_{\mathbf{v}} \odot \mathbf{1}\mathbf{x}^{\mathsf{T}} \tag{6.14}$$

Let $\mathbf{G}(\mathbf{x})\mathbf{x} := \mathbf{g}(\mathbf{x}) \odot \mathbf{x}$, where $\odot$ denotes the element-wise product and $\mathbf{g}$ is a vector-valued Gaussian process. This type of warping is equivalent to $\mathbf{G}(\mathbf{x})$ map to a diagonal matrix. The mean and covariance matrix of the warped input $\mathbf{m}(\mathbf{x})$, can be derived as:

$$\hat{\mathbf{m}}(\mathbf{x}) = \hat{\mathbf{g}}(\mathbf{x}) \odot \mathbf{x} + \hat{\mathbf{h}}(\mathbf{x}) \tag{6.15}$$

$$\boldsymbol{\Sigma}_{\mathbf{m}}(\mathbf{x}) = \mathbf{x}\mathbf{1}^{\mathsf{T}} \odot \boldsymbol{\Sigma}_{\mathbf{g}}(\mathbf{x}) \odot \mathbf{1}\mathbf{x}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\mathbf{h}}(\mathbf{x}) \,, \tag{6.16}$$

where $\hat{\mathbf{g}}(\mathbf{x})$ and $\boldsymbol{\Sigma}_{\mathbf{g}}(\mathbf{x})$ are the predictive mean and covariance, respectively, of the GP defining $\mathbf{g}$, while $\hat{\mathbf{h}}(\mathbf{x})$ and $\boldsymbol{\Sigma}_{\mathbf{h}}(\mathbf{x})$ are the same for the GP on $\mathbf{h}$.

Algorithm 1 Sparse Spectrum Warped Input Measures

Input: $\{\mathbf{X}, \mathbf{y}\}$
Output: $\boldsymbol{\theta} = \{\boldsymbol{\theta}_u, \boldsymbol{\theta}_\mathbf{g}, \boldsymbol{\theta}_\mathbf{h}, \mathbf{X}_\mathbf{g}, \mathbf{Y}_\mathbf{g}, \mathbf{X}_\mathbf{h}, \mathbf{Y}_\mathbf{h}\}$
Initialise pseudo-training points $\{\mathbf{X}_\mathbf{g}, \mathbf{Y}_\mathbf{g}\}$, $\{\mathbf{X}_\mathbf{h}, \mathbf{Y}_\mathbf{h}\}$
for $t \in \{1, \ldots, T\}$ **do**
  Fit $\mathbf{g}$ and $\mathbf{h}$ to $\{\mathbf{X}_\mathbf{g}, \mathbf{Y}_\mathbf{g}\}$, $\{\mathbf{X}_\mathbf{h}, \mathbf{Y}_\mathbf{h}\}$
  Compute $\hat{\mathbf{m}}$ and $\boldsymbol{\Sigma}_\mathbf{m}$ for $\mathbf{X}$
  Fit u using expected feature map
  Calculate $\log p(\mathbf{y}|\boldsymbol{\theta})$
  Update gradients and take new step.
end for

PROOF. The element-wise vector product, which is the Hadamard product for single-column matrices, is symmetric and linear, i.e. $\mathbf{a} \odot \mathbf{b} = \mathbf{b} \odot \mathbf{a}$ and $\mathbf{a} \odot (\mathbf{b} + \mathbf{c}) = \mathbf{a} \odot \mathbf{b} + \mathbf{a} \odot \mathbf{c}$, for any $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^D$. By the linearity of the expectation, we then have $\mathbb{E}[\mathbf{z}] = \mathbb{E}[\mathbf{v} \odot \mathbf{x}] = \mathbb{E}[\mathbf{v}] \odot \mathbf{x}$. From the properties of the Hadamard product, one can also show that $\mathbf{a}(\mathbf{b} \odot \mathbf{c})^\mathsf{T} = \mathbf{a}\mathbf{b}^\mathsf{T} \odot \mathbf{1}\mathbf{c}^\mathsf{T}$, where $\mathbf{1}$ is a vector of 1's. Therefore, the covariance matrix $\boldsymbol{\Sigma}_\mathbf{z} := \mathbb{V}[\mathbf{z}] = \mathbb{E}[(\mathbf{z} - \hat{\mathbf{z}})(\mathbf{z} - \hat{\mathbf{z}})^\mathsf{T}]$, is given by:

$$\begin{aligned} \mathbb{V}[\mathbf{z}] &= \mathbb{E}[((\mathbf{v} - \hat{\mathbf{v}}) \odot \mathbf{x})((\mathbf{v} - \hat{\mathbf{v}}) \odot \mathbf{x})^\mathsf{T}] \\ &= \mathbf{x}\mathbf{1}^\mathsf{T} \odot \mathbb{E}[(\mathbf{v} - \hat{\mathbf{v}})(\mathbf{v} - \hat{\mathbf{v}})^\mathsf{T}] \odot \mathbf{1}\mathbf{x}^\mathsf{T} \\ &= \mathbf{x}\mathbf{1}^\mathsf{T} \odot \mathbb{V}[\mathbf{v}] \odot \mathbf{1}\mathbf{x}^\mathsf{T}\,, \end{aligned} \tag{6.17}$$

which concludes the proof. □

6.5.2 Latent Self-supervision With Pseudo-training

In order to fully specify our latent function, we utilise *pseudo-training* pairs $\{\mathbf{X}_\mathbf{g}, \mathbf{Y}_\mathbf{g}\}$ and $\{\mathbf{X}_\mathbf{h}, \mathbf{Y}_\mathbf{h}\}$, somewhat analogous to the well known *inducing-points* framework for sparse Gaussian processes (Titsias 2009). Conditioning on these virtual observations allows us to implicitly control the Gaussian measure determined by the warping SSGP.

We model the multiplicative warping $\mathbf{g} : \mathbb{R}^D \to \mathbb{R}^D$ using a standard, multi-output, SSGP that is analytically fit on virtual *pseudo-training* points $\{\mathbf{X}_\mathbf{g}, \mathbf{Y}_\mathbf{g}\}$. Assuming coordinate-wise

output independence, we model $\mathbf{g}$ as $\mathbf{g}(\mathbf{x}) \sim \mathcal{N}(\hat{\mathbf{g}}(\mathbf{x}), \boldsymbol{\Sigma}_{\mathbf{g}}(\mathbf{x}))$, where:

$$\hat{\mathbf{g}}(\mathbf{x}) = \boldsymbol{\phi}_{\mathbf{g}}(\mathbf{x})^{\mathsf{T}} \mathbf{A}_{\mathbf{g}}^{-1} \boldsymbol{\Phi}_{\mathbf{g}} \mathbf{Y}_{\mathbf{g}} \tag{6.18}$$

$$\boldsymbol{\Sigma}_{\mathbf{g}}(\mathbf{x}) = \sigma_{n,g}^2 \boldsymbol{\phi}_{\mathbf{g}}(\mathbf{x})^{\mathsf{T}} \mathbf{A}_{\mathbf{g}}^{-1} \boldsymbol{\phi}_{\mathbf{g}}(\mathbf{x}) \mathbf{I} \,, \tag{6.19}$$

with $\boldsymbol{\Phi}_{\mathbf{g}} := \boldsymbol{\Phi}_{\mathbf{g}}(\mathbf{X}_{\mathbf{g}})$ as the matrix of Fourier features for the pseudo-inputs $\mathbf{X}_{\mathbf{g}}$, and $\mathbf{A}_{\mathbf{g}} = \boldsymbol{\Phi}_{\mathbf{g}} \boldsymbol{\Phi}_{\mathbf{g}}^{\mathsf{T}} + \sigma_{n,g}^2 \mathbf{I}$. The pseudo-inputs $\mathbf{X}_{\mathbf{g}}$ are initially sampled uniformly across the data domain, $\mathbf{X}_{\mathbf{g}} \sim \mathcal{U}(\min(\mathbf{X}), \max(\mathbf{X}))$. The pseudo-training targets $\mathbf{Y}_{\mathbf{g}}$ are initialised $[\mathbf{Y}_{\mathbf{g}}]_{i,j} \sim \mathcal{N}(1, \sigma_\gamma^2)$ where σ_γ^2 mimics a prior variance for the latent warping function. The mean at 1 keeps the initial warping close to identity.

We adopt a similar construction for the GP on the additive component of the warping $\mathbf{h}$. However, we initialise the pseudo-training targets $\mathbf{Y}_{\mathbf{h}}$ with zero-mean values $[\mathbf{Y}_{\mathbf{h}}]_{i,j} \sim \mathcal{N}(0, \sigma_\gamma^2)$, so that we favour a null effect initially. In summary, the complete expected kernel is thus given as:

$$k_f(\mathbf{x}, \mathbf{x}') := \mathbb{E}[\boldsymbol{\phi}(\mathbf{m}(\mathbf{x})]^{\mathsf{T}} \mathbb{E}[\boldsymbol{\phi}(\mathbf{m}(\mathbf{x}'))] \,, \tag{6.20}$$

where the expectation is taken over $\mathbf{m}$, whose distribution is recursively defined by equations Equation (6.15) to Equation (6.19).

6.5.3 A Layered Warping

We have thus far considered a single warping $\mathbf{m}$ of the input $\mathbf{x}$. It is natural to ask: *can we warp the warpings?* A simple way to answer this is to revisit how we interpret a single warping: we are transforming the original input space, with which our response varies in a nonstationary way, to a new space a GP with a stationary kernel can easily represent. We could thus intuit a warping of a warping to mean that we are transforming the first level of warping to a second one to which our response variable is simply *more stationary* than if we had just relied on the first warping alone. We present now an extension to SSWIM which lets us perform this measure value warping of a measure valued warping. Let us begin by defining the J^{th} warping as:

$$\mathbf{m}^{(J)}(\mathbf{x}^{(J-1)}) = \mathbf{g}^{(J)}(\mathbf{x}^{(J-1)}) \odot \mathbf{x}^{(J-1)} + \mathbf{h}^{(J)}(\mathbf{x}^{(J-1)}), \tag{6.21}$$

where:

$$\mathbf{x}^{(J-1)} = \mathbf{m}^{(J-1)}(\mathbf{x}^{(J-2)}) \quad , \; J \geq 2 \tag{6.22}$$

While multiplication of a known vector by a Gaussian random matrix keeps Gaussianity, after the first warping layer, the product of two Gaussians is no longer Gaussian in (6.21). For the layered formulation, we therefore apply moment matching to approximate each layer's warped input as a Gaussian $\mathbf{x}^{(J)} \sim \mathcal{N}(\hat{\mathbf{x}}^{(J)}, \boldsymbol{\Sigma}_{\mathbf{x}}^{(J)})$. Making independence assumptions on (6.21) and applying known results for the Hadamard product of independent random variables (Neudecker, Liu and Polasek 1995), we have:

$$\hat{\mathbf{x}}^{(J)} = \hat{\mathbf{g}}^{(J)} \odot \hat{\mathbf{x}}^{(J-1)} + \hat{\mathbf{h}}^{(J)} \tag{6.23}$$

$$\boldsymbol{\Sigma}_{\mathbf{x}}^{(J)} = \boldsymbol{\Sigma}_{\mathbf{x}}^{(J-1)} \odot \boldsymbol{\Sigma}_{\mathbf{g}}^{(J)} + \boldsymbol{\Sigma}_{\mathbf{x}}^{(J-1)} \odot \hat{\mathbf{g}}^{(J)}\hat{\mathbf{g}}^{(J)\mathsf{T}} + \boldsymbol{\Sigma}_{\mathbf{g}}^{(J)} \odot \hat{\mathbf{x}}^{(J-1)}\hat{\mathbf{x}}^{(J-1)\mathsf{T}} + \boldsymbol{\Sigma}_{\mathbf{h}}^{(J)} \,, \tag{6.24}$$

where $\mathbf{g}^{(J)} \sim \mathcal{N}(\hat{\mathbf{g}}^{(J)}, \boldsymbol{\Sigma}_{\mathbf{g}}^{(J)})$ and $\mathbf{h}^{(J)} \sim \mathcal{N}(\hat{\mathbf{h}}^{(J)}, \boldsymbol{\Sigma}_{\mathbf{h}}^{(J)})$ are the SSGP predictions using the expected feature map (Equation (6.12)) of the previous layer's output $\mathbf{x}^{(J-1)}$.

The layered warping allows for more complex input transformations. The drawback, however, is an increased computational cost due to the additional hyper-parameters, i.e. the pseudo-training points. In addition, we are taking a non-linearly transformed Gaussian input, which leads to a non-Gaussian result, and moment-matching it with a Gaussian. This distribution mismatch leads to compounding effects across several layers which could make the top-level Gaussian tend to a high-variance flat distribution. However, the training process should compensate for this increase in variance by adjusting the pseudo-training points according to a loss that takes the data into account, e.g. the GP marginal likelihood.

6.5.4 Joint Training

The goal of optimisation in learning our warping with uncertainty is to quickly discover hyper-parameters whose models explain the variation in the data. We also want to avoid pathologies that may manifest with an arbitrarily complex warping function. We do this by minimising the model's negative log-marginal likelihood. Given a set of observations

$\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, we learn the hyper-parameters $\boldsymbol{\theta}$ by minimising the negative log-marginal likelihood:

$$-\log p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{2\sigma_n^2}(\mathbf{y}^\mathsf{T}\mathbf{y} - \mathbf{y}^\mathsf{T}\hat{\boldsymbol{\Phi}}_F^\mathsf{T}\mathbf{A}_F^{-1}\hat{\boldsymbol{\Phi}}_F\mathbf{y}) + \frac{1}{2}\log|\mathbf{A}_F| - \frac{D}{2}\log\sigma_n^2 + \frac{M}{2}\log 2\pi\sigma_n^2\,, \quad (6.25)$$

where $\hat{\boldsymbol{\Phi}}_F$ is the matrix whose rows to expected feature maps for the top-level SSGP, i.e. $[\hat{\boldsymbol{\Phi}}_F]_i = \mathbb{E}_{\mathbf{m}}[\boldsymbol{\phi}_F(\mathbf{m}(\mathbf{x}_i))]^\mathsf{T}$, and $|\mathbf{A}_F|$ denotes the determinant of $\mathbf{A}_F$. The expectation is taken under the warping $\mathbf{m}$, whose parameters are computed from the predictive mean and covariance functions of the lower-level GPs (cf. (6.15) and (6.16)), and available in closed form via Equation (6.12).

6.5.5 Computational Complexity

The top-level function u and two warping functions $\mathbf{G}$ and $\mathbf{h}$ all inherit the complexity of SSGP with and without predictions under uncertainty (Pan et al. 2017) which is $\mathcal{O}(nm^2 + m^3)$ for n samples and m features. For multiple warping levels this cost is multiplied by the number of levels J therefore the overall complexity remains $\mathcal{O}(nm^2 + m^3)$. In practice m is very small with $m < 1000$. For SSWIM, a single pseudo-training point has dimensionality D which is the same as the raw input $\mathbf{x}$. Therefore $\mathbf{G}$ and $\mathbf{h}$ consist of $D \times N_\mathbf{G}$ and $D \times N_\mathbf{h}$ pseudo-training points respectively. The functions u, $\mathbf{G}$ and $\mathbf{h}$ contain model and kernel hyperparameters of size $|\boldsymbol{\theta}_u|$, $|\boldsymbol{\theta}_\mathbf{G}|$ and $|\boldsymbol{\theta}_\mathbf{h}|$ respectively which should each not exceed much more than D for conventional stationary kernels.

6.5.6 On Function Classes of the Warping

It has been remarked in prior work on deep GP models that degenerate covariance matrices may arise after consecutive compositions (Duvenaud et al. 2014; Hegde et al. 2019). Recent works, such as (Hegde et al. 2019; Ustyuzhaninov et al. 2020), that employ a dynamical systems based formulation can demonstrate improved uncertainty propagation under an *injective* warping (by maintaining monotonic constructions) as opposed to conventional deep GP models (Damianou and Lawrence 2013). Indeed our proposed SSWIM is not guaranteed

to be injective and falls into the most general class of functions. An interesting consequence of this is that one could argue injectivity is *not necessarily ideal* for learning latent mappings and furthermore it certainly is not a necessary condition for preventing collapse of uncertainty although such phenomena may be correlated. To comment further, by relaxing from certain function types it is plausible for multiple different input values in a prior warping layer to *warp into the same input location* in the next layer; i.e. in a surjective function. This may ultimately be a desirable property – it suggests compressiblity of the input domain – in that there might be an underlying non-monotonic, nonstationary covariance function at play in the latent representation. Such expressiveness would not be able to be directly captured by a purely injective mapping. Injectivity and even bijectivity could be enforced as an additional constraint and this perspective undoubtedly deserves future investigation.

6.5.7 On Kernel Priors

One may enquire about the choice of kernels for latent and top level functions. Our methodology is, generally speaking, "kernel prior agnostic" in the sense that the nonstationarity is accomplished through the affine warpings. We remark that kernel choice undoubtedly may play a role in performance. One could indeed use extremely expressive kernels, like the stationary spectral mixture (Wilson and Adams 2013) or quantile kernel representations (Tompkins et al. 2019). However, to restrict the space of analysis to measure the effect of the warping construction, we aimed to forgo kernel discovery.

6.6 Experiments

We experimentally validate SSWIM alongside various state of the art methods in both small and large data regimes as well as expand upon the intuition in Section 6.6.1 by examining specific aspects of the model. Section Section 6.6.2 analyses computational complexity and model performance with respect to the pseudo-training points. We investigate increasing the number of warping levels in Section 6.6.3. Large scale comparison alongside various algorithms is presented in Section 6.6.4.

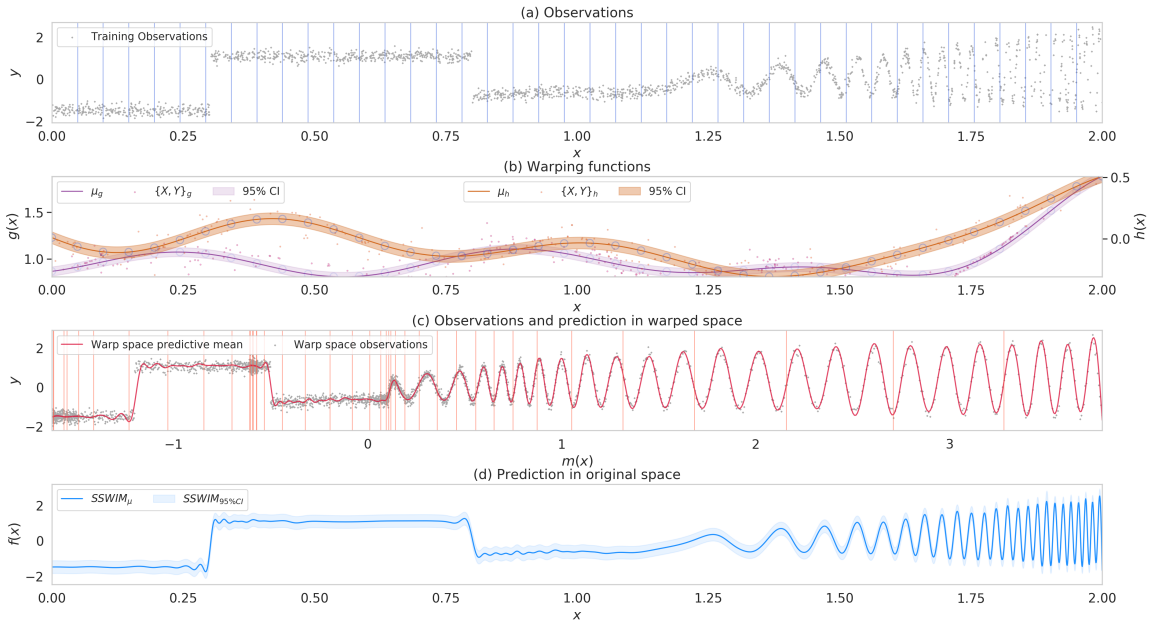


Figure 6.1. Visualisation of SSWIM learning an input warping. (a) Noisy training data. Going left to right, the signal observations exhibit abrupt steps, periodic, and spatial frequency nonstationarity. (b) The learned warping functions. (c) The training data after input warping, and (d) Final prediction with respect to the warped inputs. The key observation here is how the spatially varying frequencies and steps in the original training data from (a) have been transformed to (c) where the warped data varies in a more uniform (stationary) manner.

Table 6.1. RMSE and MNLP metrics for various real-world datasets.

	(D, N)	$(8, 1030)$	$(16, 5875)$	$(15, 17379)$	$(379, 53500)$	$(81, 21263)$	$(9, 45730)$	$(77, 583250)$	$(90, 515345)$
	Method	concrete	parkinsons	bikeshare	ct slice	supercond	protein	buzz	song
RMSE ($\times 10^{-1}$)	$SSWIM_1$	3.05 ± 0.26	7.63 ± 0.20	0.13 ± 0.04	0.46 ± 0.02	3.44 ± 0.14	5.91 ± 0.07	2.98 ± 0.04	8.12 ± 0.05
	$SSWIM_2$	$\mathbf{3.01 \pm 0.31}$	$\mathbf{7.55 \pm 0.15}$	0.11 ± 0.03	$\mathbf{0.23 \pm 0.01}$	$\mathbf{3.02 \pm 0.04}$	$\mathbf{5.80 \pm 0.08}$	$\mathbf{2.40 \pm 0.01}$	$\mathbf{7.97 \pm 0.03}$
	DSDGP	5.88 ± 1.24	7.94 ± 0.20	0.33 ± 0.55	4.81 ± 1.18	5.10 ± 0.84	5.96 ± 0.06	3.65 ± 0.75	8.46 ± 0.03
	DKL	3.18 ± 0.38	8.84 ± 0.74	0.24 ± 0.03	0.52 ± 0.08	3.46 ± 0.18	7.15 ± 1.10	4.11 ± 3.33	16.66 ± 8.14
	RFFNS	3.46 ± 0.24	8.15 ± 0.15	0.05 ± 0.01	4.39 ± 0.27	3.85 ± 0.05	6.87 ± 0.06	5.70 ± 0.84	8.35 ± 0.03
	SVGP	3.32 ± 0.26	8.14 ± 0.12	0.06 ± 0.03	1.16 ± 0.02	4.06 ± 0.05	7.32 ± 0.08	9.98 ± 0.02	12.19 ± 0.18
	SGPR	5.55 ± 0.58	7.86 ± 0.22	0.67 ± 0.18	1.79 ± 0.04	4.27 ± 0.06	6.45 ± 0.07	2.89 ± 0.02	8.40 ± 0.04
	RFFS	3.33 ± 0.30	8.24 ± 0.17	$\mathbf{0.03 \pm 0.00}$	2.34 ± 0.05	3.89 ± 0.06	6.91 ± 0.07	3.78 ± 0.14	8.36 ± 0.04
MNLP ($\times 10^{-1}$)	$SSWIM_1$	10.22 ± 4.15	11.95 ± 0.47	-11.89 ± 0.15	-11.24 ± 0.05	3.55 ± 0.32	8.95 ± 0.12	2.03 ± 0.13	12.08 ± 0.05
	$SSWIM_2$	5.19 ± 2.59	12.50 ± 0.44	-11.78 ± 0.07	$\mathbf{-11.79 \pm 0.02}$	$\mathbf{2.82 \pm 0.29}$	$\mathbf{8.82 \pm 0.15}$	$\mathbf{-0.09 \pm 0.04}$	$\mathbf{11.93 \pm 0.04}$
	DSDGP	11.02 ± 1.06	$\mathbf{11.91 \pm 0.24}$	-23.28 ± 8.29	6.62 ± 2.61	7.36 ± 1.62	9.04 ± 0.10	3.80 ± 2.02	12.52 ± 0.04
	DKL	7.69 ± 0.20	13.17 ± 1.12	6.82 ± 0.01	6.83 ± 0.01	7.76 ± 0.10	11.02 ± 1.53	9.01 ± 4.65	42.64 ± 44.77
	RFFNS	3.31 ± 0.45	12.18 ± 0.18	-11.97 ± 0.00	5.95 ± 0.66	4.66 ± 0.12	10.39 ± 0.08	8.78 ± 1.87	12.39 ± 0.04
	SVGP	$\mathbf{2.83 \pm 0.56}$	12.21 ± 0.14	$\mathbf{-27.70 \pm 1.24}$	-5.98 ± 0.13	5.32 ± 0.12	11.10 ± 0.09	63.31 ± 3.44	18.02 ± 0.09
	SGPR	8.48 ± 2.10	12.39 ± 0.30	-13.67 ± 0.98	-3.14 ± 0.26	5.58 ± 0.10	10.05 ± 0.13	1.11 ± 0.12	11.97 ± 0.07
	RFFS	3.05 ± 0.96	12.29 ± 0.20	-11.98 ± 0.00	-0.33 ± 0.22	4.79 ± 0.13	10.45 ± 0.09	4.41 ± 0.37	12.40 ± 0.05

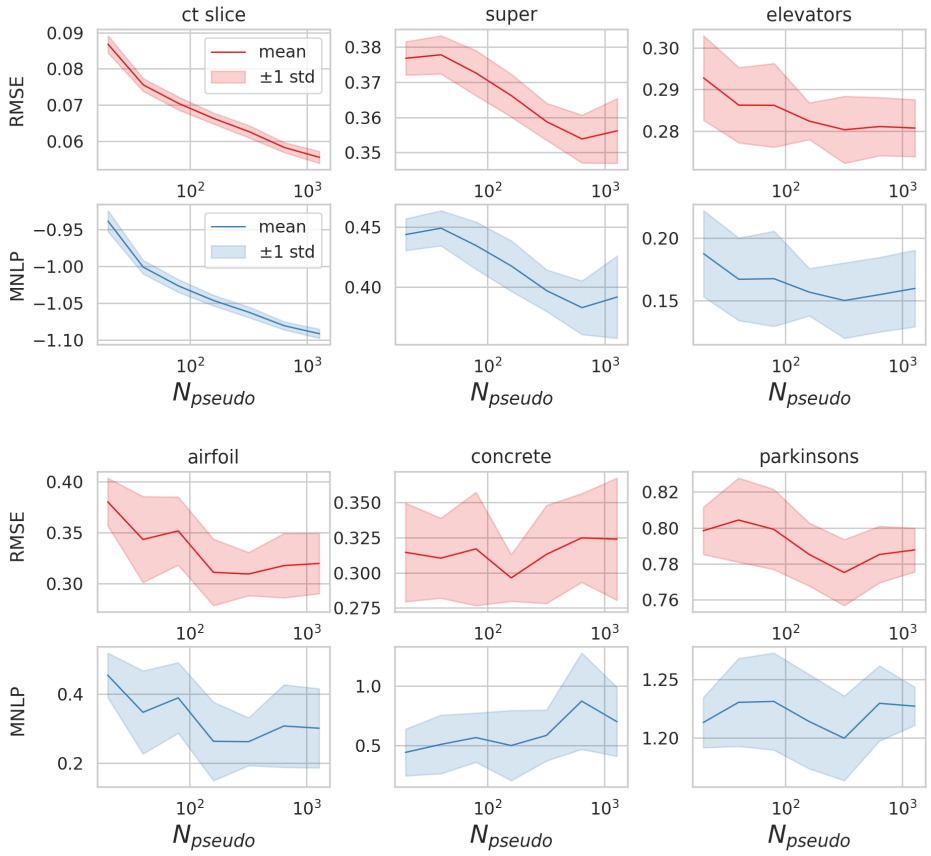


Figure 6.2. Performance in RMSE and MNLP as the number of pseudo-training points increases.

For every quantitative experiment, we report the mean and standard deviation over ten repeats. Metrics are presented in the standardised data scale. In all experiments the Matern $\frac{3}{2}$ is used as the base kernel. For performance evaluation we use the test Root Mean Square Error (RMSE) and test Mean Negative Log Probability (MNLP). These are defined as RMSE = $\sqrt{\langle (y_{*j} - \mu_{*j})^2 \rangle}$ and MNLP = $\frac{1}{2}\langle (\frac{y_{*j}-\mu_{*j}}{\sigma_{*j}})^2 + \log \sigma_{*j}^2 + \log 2\pi \rangle$ where y_{*j} is the true test value, and μ_{*j} and σ_{*j}^2 are the predictive mean and variance respectively for the j^{th} test observation. Mean is denoted as $\langle \cdot \rangle$.

6.6.1 Inductive Bias and a Geometric Interpretation

An intuitive interpretation of SSWIM is by imagining it as learning a conditional affine transformation. The quintessential affine transformation of some vector x is described as

$\mathbf{A}x + b$ for some multiplication matrix $\mathbf{A}$ and addition vector b. Such transformations are typically interpreted geometrically (González and Woods 2008) as *translation*, *rotation*, *reflection*, *scale* and *shear*. SSWIM learns a *conditional* affine map that *depends on the input*. I.e. $\mathbf{A}$ and b become maps $\mathbf{A}(x)$ and $b(x)$. By directly applying a learned warping to the original input data we transform the inputs into a locally Euclidean manifold which ultimately preserves any structure with respect to the input resulting in a convenient inductive bias. Observe in Figure 6.1 (c) how we have non-uniformly "stretched out" out the left and rightmost parts of the original data in (a) to produce a new warped dataset. What was original spatially nonstationary becomes spatially homogeneous resulting excellent prediction as in Figure 6.1 (d).

6.6.2 How Many Pseudo-training Points?

To understand the overall sensitivity of our method we visualise the predictive performance as a function of the number of pseudo-training points. Figure 6.2 shows performance, in log scale, with respect to the number of pseudo-training points on real-world datasets. While we observe a trending improvement, very few pseudo-targets are required to get excellent performance, even in much higher dimensional problems like *superconductivity* ($D = 81$) and *ct slice* ($D = 379$), suggesting that there is significant expressiveness in the underlying warping function.

We remark that a possible drawback of pseudo-training points and fitting a stochastic model over those points is the question of how to set the prior of their locations. Furthermore, how do we initialise them? To answer this, it is natural to set $\mathbf{G}$ and $\mathbf{h}$ to be fit to noisy pseudo-targets with mean $\mathbf{I}$ and $\mathbf{0}$ respectively. This has a nice interpretation as the matrices corresponding to the identity operations of an affine transformation.

We now make an observation of our framework in terms of the required *hyperparameter complexity* to perform learning. We consider the highly performing deterministic warping approach of Deep Kernel Learning (Wilson et al. 2016) (DKL). In DKL a large neural network is used to encode the input data before the outputs are passed into a standard GP. The neural

network is completely deterministic. As seen in Table 6.1, DKL offers similar performance to our method and is among the overall best performing. The default network structure is (input, output) layer dimensionalities of $[(D, 1000), (1000, 500), (500, 50), (50, 2)]$. In addition to this, there are GP hyperparameters. If we probe the parameter count for the model used for the *airfoil* dataset, we find there are a total of 531656 free parameters for the optimiser to tune. The majority of these come from the neural network. For comparable results of our method SSWIM, we use 512 features for both the warped mapping and top-level predictive function. The total number of free parameters including additional kernel and SSGP hyperparameters accounts to 3510 which is more than 2 orders of magnitude fewer parameters for similar performance.

6.6.3 Increased Warping Depth

In this experiment we evaluate the predictive performance of SSWIM as we increase the number of levels of consecutive input warping from 0 to 3. A depth of 0 simply corresponds to the *stationary* SSGPR specification. Figure 6.3 summarises the results. For all the datasets we can see that adding just a single level of input warping increases predictive performance. Adding additional levels of warping seems to consistently improve performance however it adds additional variance to all results which could be explained by the additional complexity required for optimisation.

6.6.4 Real Datasets

For completeness, we specify for each dataset used, the data dimensionality and sample size, the raw data source, the modelling objective (i.e. the target) as defined for the original problem, and any target variable pre-processing excluding standardisation. The dimensionality D is reported for the inputs X (i.e. excluding the target variable y). All problems are single output regression tasks. Number of samples N is reported for the entire dataset before train/test splitting is applied. Note that we do not alter the raw data files and pre-processing is applied as defined and in the provided supplementary code as per *dataloader.py*.

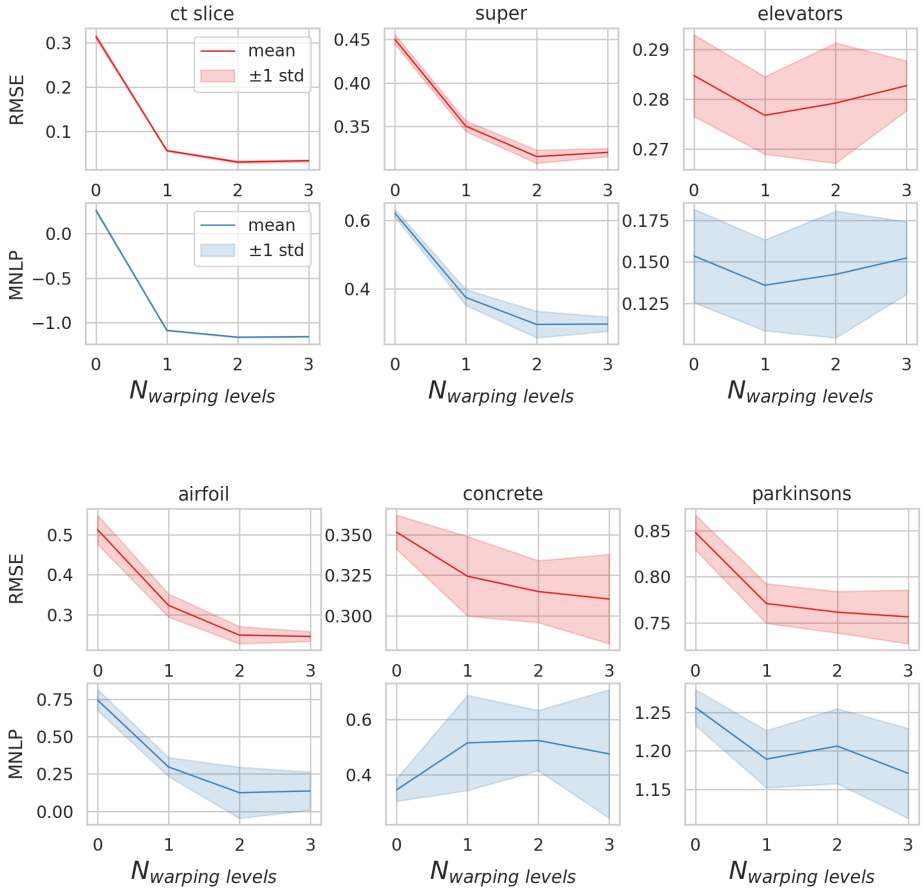


Figure 6.3. Performance in RMSE and MNLP as the number of warping levels increases.

Table 6.2. Summary of datasets used.

Name	D	N	Preprocessing	Target	Source
elevators	18	8751	None	goal	(Torgo 2019)
airfoil	5	1503	None	sound pressure in decibels	(Dua and Graff 2017)
concrete	8	1030	None	compressive strength	(Dua and Graff 2017)
parkinsons	16	5875	Drop the first 5 columns as they are not used in the original problem	total udpr	(Dua and Graff 2017)
bikeshare	15	17379	None	number of bike shares per hour	(Dua and Graff 2017)
ct slice	379	53500	Drop Patient ID. Drop columns which have constant value throughout entire dataset.	reference (relative location)	(Dua and Graff 2017)
supercond	81	21263	None	critical temperature	(Dua and Graff 2017)
protein	9	45730	$\log(1+y)$ transform for target y	rmsd	(Dua and Graff 2017)
buzz	77	583250	$\log(1+y)$ transform of target y	mean number of active discussion (nad)	(Dua and Graff 2017)
song	90	515345	None	year of song release	(Dua and Graff 2017)
abalone	9	4177	None	number of rings	(Dua and Graff 2017)
creep	30	2066	None	rupture stress	(Brun and Yoshida n.d.)
ailerons	40	7154	None	goal	(Dua and Graff 2017)

We compare our model on various real-world datasets including multiple regression tasks (Dua and Graff 2017; Torgo 2019; Cole et al. 2000). All datasets are standardised using the train set. We use $\frac{2}{3}$ of the samples for training and the remaining $\frac{1}{3}$ for testing. We compare multiple related algorithms alongside our proposed method SSWIM using both one level of warping (SSWIM_1) and two levels of warping (SSWIM_2), Deep Kernel Learning (Wilson

et al. 2016) (DKL), SSGP with stationary random fourier features kernel (RFFS), SSGP with nonstationary kernel features (RFFNS) with freely variable mean and width for the Matern $\frac{3}{2}$ spectral frequencies (Remes, Heinonen and Kaski 2017; Ton et al. 2018), Sparse Gaussian Process Regression (SGPR) (Titsias 2009), Sparse Variational Gaussian Process (SVGP) (Hensman, Matthews and Ghahramani 2015), and Doubly Stochastic Deep GP with 2 layers (DSDGP) (Salimbeni and Deisenroth 2017). All experiments were performed on a Linux machine with a single Titan V GPU. We ran all methods for 150 iterations with stochastic gradient descent and the library GPyTorch was used for DKL, DSDGP, SGPR, and SVGP. We have provided implementations for RFFS, RFFNS, and SSWIM. PyTorch Code is provided at https://github.com/MushroomHunting/SSWIM to reproduce the experimental results.

In the main experimental results given in Table 6.1 we can observe a consistent high performance across all datasets for SSWIM in all tasks for the RMSE metric. For *concrete, parkinsons* and *bikeshare* SSWIM is outperformed in MNLP by DSDGP, SVGP and RFFS suggesting they were more capable of representing the predictive distribution rather than the mean. For the remaining datasets SSWIM has performed extremely well, most notably on the high dimensional problem *ct slice*. SSWIM_2 with two levels of warping comprehensively outperforms other methods as well as SSWIM_1 which also performs competitively. These results further corroborate the analysis given in Figure 6.2 and Figure 6.3.

6.6.5 Overfitting Analysis

We ran an overfitting analysis of $SSWIM_1$ to observe the effect of over-optimising with respect to the marginal likelihood. We ran with 256 features, 1280 pseudo-training points, for 150 steps, with 10 repeats, and evaluated the test RMSE and test MNLP on the test set for every single epoch of optimisation. The results are averaged with mean and standard deviations of the training curves displayed in Figure 6.4. We can see that we are quite resistant to overfitting except for RMSE in the *elevators* dataset and the MNLP in the *concrete* and *parkinsons* datasets. The causes of this could be explained by the underlying flexibility of the proposed method which allows the model to become overconfident in what it has learned with respect to the data it has observed. In fact, we observed similar overfitting

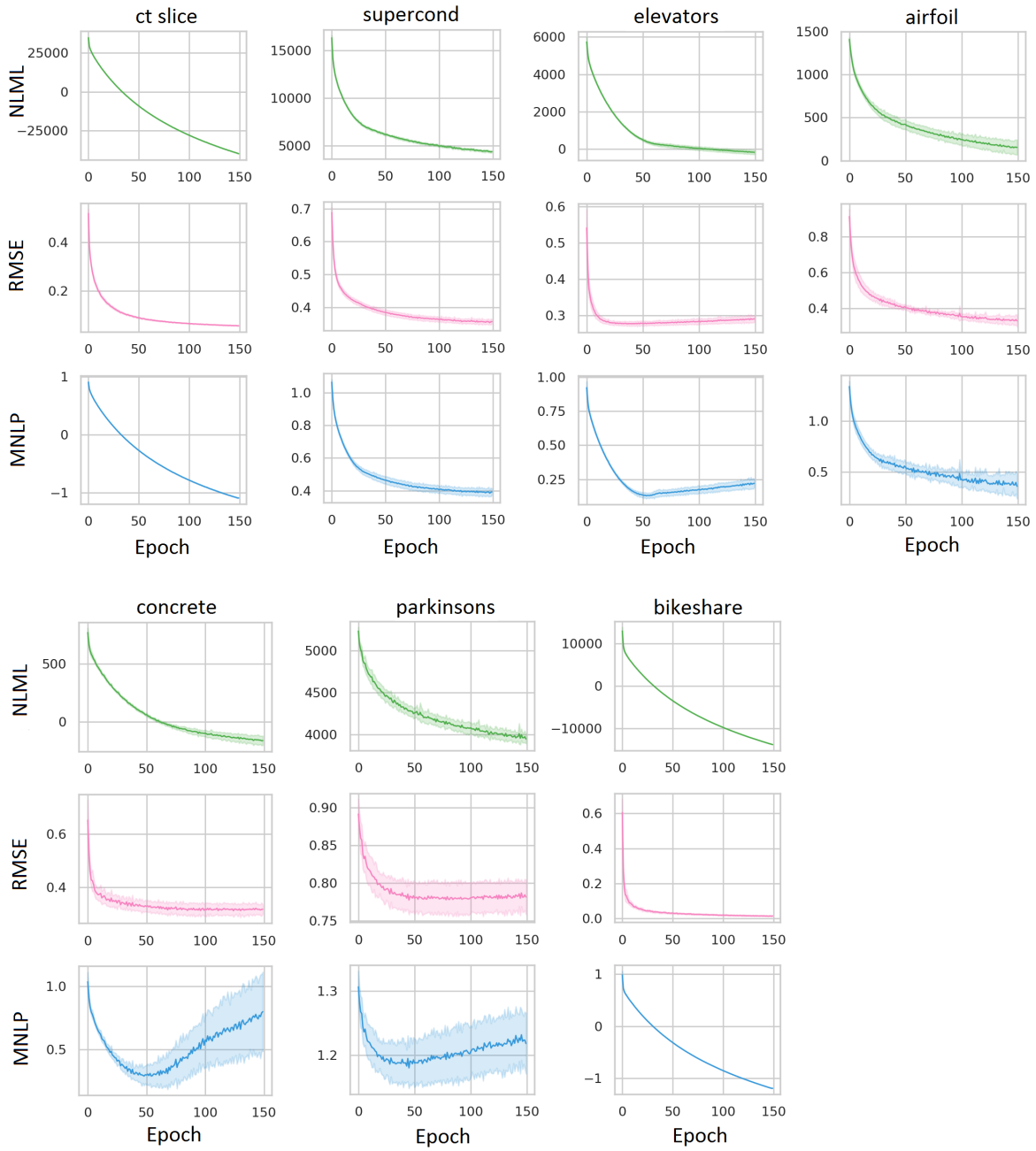


Figure 6.4. Empirical analysis of overfitting behaviour in SSWIM

behaviour for similar optimisation periods with DKL and DSDGP. This analysis leads to some interesting observations and recommendations for future algorithm development in more expressive GP methodologies: 1. the marginal likelihood is no panacea although it is easy to think it is, and 2. other loss functions and training schemes, such as leave-one-out cross validation may be equally or more effective. These issues corroborate long known discussions

from (Rasmussen 2004) about the risk of overfitting from trusting the marginal likelihood with standard optimisation procedures, however their importance seems to have been largely ignored in evaluation of recent methodology innovations in the GP literature. We believe that a more open discussion should be on the table for analysing the interplay between model expressiveness and the effect this has on overfitting; this is especially pertinent to the GP literature which has placed a large emphasis on the importance of the marginal likelihood has a valid hyperparameter optimisation loss.

6.6.6 Relationship to Output Warped GPs

Table 6.3. MSE and MNLP metrics for comparison with Warped and Bayesian Warped GPs (Lázaro-Gredilla 2012). MSE results for *ailerons* are $\times 10^{-8}$.

	Method	abalone	creep	ailerons
MSE	GP	4.55 ± 0.14	584.9 ± 71.2	2.95 ± 0.16
	BWGP	4.55 ± 0.11	491.8 ± 36.2	2.91 ± 0.14
	MLWGP3	4.54 ± 0.10	502.3 ± 43.3	$\mathbf{2.80 \pm 0.11}$
	MLWGP20	4.59 ± 0.32	506.3 ± 46.1	3.42 ± 2.87
	$SSWIM_1$	4.64 ± 0.13	483.69 ± 64.12	2.96 ± 0.08
	$SSWIM_2$	$\mathbf{4.50 \pm 0.11}$	$\mathbf{279.86 \pm 31.88}$	2.83 ± 0.06
MNLP	GP	2.17 ± 0.01	4.46 ± 0.03	-7.30 ± 0.01
	BWGP	1.99 ± 0.01	4.31 ± 0.04	-7.38 ± 0.02
	MLWGP3	$\mathbf{1.97 \pm 0.02}$	$\mathbf{4.21 \pm 0.03}$	-7.44 ± 0.01
	MLWGP20	1.99 ± 0.05	$\mathbf{4.21 \pm 0.08}$	$\mathbf{-7.45 \pm 0.08}$
	$SSWIM_1$	2.18 ± 0.01	4.45 ± 0.03	-7.24 ± 0.01
	$SSWIM_2$	2.17 ± 0.02	4.27 ± 0.03	-7.00 ± 0.02

A body of work has previously been developed under the title of *warped* Gaussian processes (Snelson, Ghahramani and Rasmussen 2004). As noted, this contrasts from our modelling problem because while we proceed to expand the GP's capabilities to warp the *inputs*, the WGP and extensions warps explicitly the *output* distribution of the Gaussian process. We now juxtapose the efficacy of our input warping formulation with WGP by applying SSWIM to the three challenging datasets *abalone*, *creep*, and *ailerons* experimented upon in (Snelson, Ghahramani and Rasmussen 2004; Lázaro-Gredilla 2012).

Our results in Table 6.3 suggest that with SSWIM we are able to improve upon both WGP and BWGP in MSE: marginal improvements for *abalone* and a significant improvement

for *creep* with comparable performance in *ailerons*. Contrasting this, the output warping methods outperform unanimously on the MNLP metric. This is expected because output warping may allow one to capture non-Gaussian conditional distributions which an input warping formulation cannot with a standard Gaussian process. The discussion we wish to raise here is that both aspects of manipulating the inputs and outputs of a GP can result in major improvements respectively across different metrics.

6.6.7 Parameter Variations

Pseudo-training points. For the "increasing number of pseudo-training points" experiment we used 1 layer of warping with 256 features for both the warping and top-level predictive functions.

Warping Depth. We used 256 features and 1280 pseudo-training points for all of the experiments.

6.6.8 Pseudo-training Points in 2D

Figure 6.5 contains an interpretation of input warping and the pseudo-training points in higher dimensions. We use the exponential 2D function from (Gramacy 2005) as a case where it is intuitive to observe how SSWIM reacts to topological differences in the underlying function. The function consists of a mostly flat surface with abrupt spiking occurring at a corner of the domain as seen in Figure 6.5 (b). If we were to assume a homogeneous domain, and model our data with a standard SSGP with stationary RBF kernel, the kernel's hyperparameters would be optimised to provide a homogeneous representation resulting in conflict between the spiked area in the corner and flat areas elsewhere. By directly manipulating the input domain with our warping, we are able to transform the input domain with a continuous warping that consequently allows accurate representation as seen in Figure 6.5 (c).

The intuition and utility of our proposed *pseudo-training* as virtual training points is visualised from a birds-eye view in Figure 6.5 (d). Before optimisation, we initialise these points

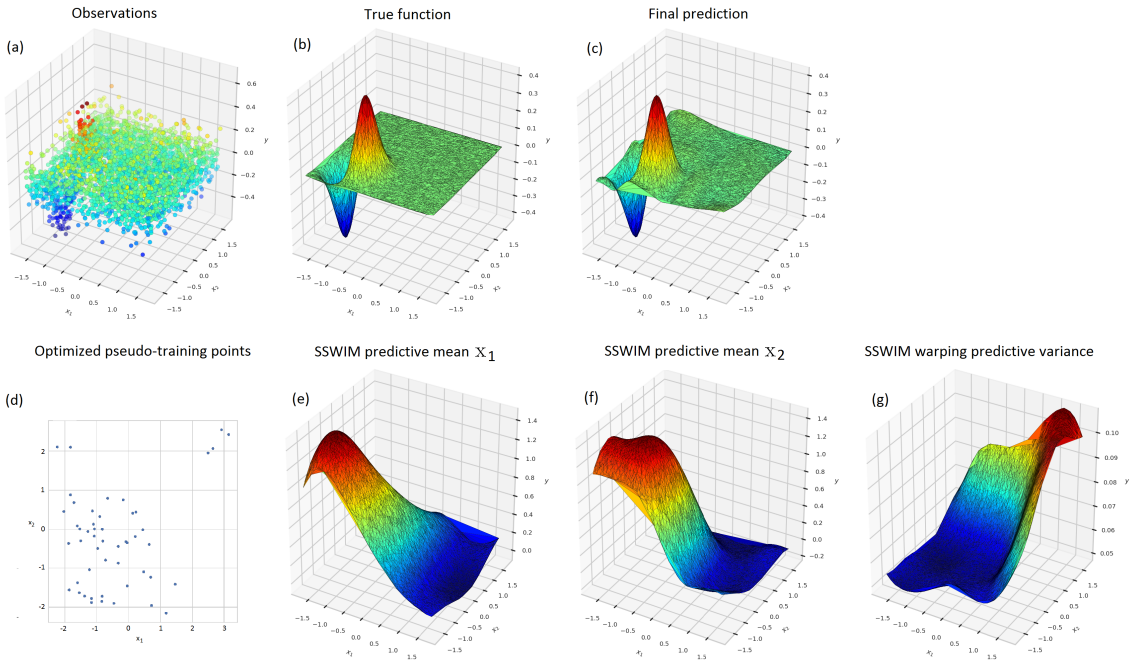


Figure 6.5. Visualisation of learned pseudo training points in 2D. We demonstrate spatial nonstationarity with the exponential 2D function from (Gramacy 2005). (a) Noisy training data, (b) True function surface, (c) SSWIM prediction conditioned on training data, (d) Learned pseudo-training point positions, (e) Learned warping predictive mean for x_1, (f) Learned warping predictive mean for x_2, (g) Learned warping predictive variance.

uniformly across the domain of the training space. It is clear that the learned positions of the pseudo-training data have transformed their spatial locations away from uniform. At the bottom left they appear to have clustered near the discontinuity of the test function while in the remaining corners of the space the points have spread away. The predictive mean of the learned warping function is thus visualised across the domain, for both x_1 and x_2, in Figure 6.5 (e) and (f). Furthermore, Figure 6.5 (g) shows the predictive variance of the warping function and we can see how a lower amount of spatial uncertainty arises both from the noisy pseudo-targets as well as the pseudo-inputs from (d).

6.7 Summary

We have proposed a crucial advance to the sparse spectrum Gaussian process framework to account for nonstationarity through a novel input warping formulation. Our model analytically incorporates complete Gaussian measures in the functional input warping with the concept of *pseudo-training* data and latent *self-supervision*. We have further extended this core contribution with the necessary results to extend the warping to multiple levels resulting in higher levels of model expressiveness.

Experimentally, the methodology we propose has demonstrated excellent results in the total number of hyperparameters for various low and high dimensional real-world datasets when compared to deterministic and neural network based approaches but also performing exceptionally well in contrast to deep Gaussian processes. Our model suggests an interesting and effective inductive bias when interpreted as a learned conditional affine transformation. This perspective invites a fresh take on how to discover more effective representations of nonstationary data.

CHAPTER 7

Conclusion

On our journey to discover a more nuanced view of kernel representations and learning we encountered two learning perspectives: i) explicitly local kernel placements, and ii) implicitly global Fourier kernel. In the first perspective we accommodate a *local* approach where kernels are explicitly located in the same space of the original data. The advantage of this approach is that our kernels are both local, analytically computable, and highly intuitive. This allows us to optimize individual kernels with respect to the known input space. Conversely, this approach potentially suffers from the necessity to add kernels cumulatively if the convex hull of our observation space increases which is the case for higher dimensional problems. In the second approach, which uses Fourier representations of kernels, we approximate placing an infinite amount of kernels and simply learn a vector of linear weights and a single kernel parameter. Using this alternative spectral perspective on kernel representation, we observe that we can exploit the distributional representation of the kernel. That is to say, we can completely relax the conventional assumptions that canonical kernels impose and learn how to adapt the distributional form of the kernel to data. Furthermore, while remaining in the spectral domain, instead of adapting the kernel to fit the data, we demonstrate it is also possible to adapt the data to fit the kernel.

7.1 Summary of Contributions

Our first major contribution in Chapter 3 was **a probabilistic framework for joint learning of model and kernel parameters in a unified model that can capture nonstationarity in observations in an automatic manner.** In this chapter we proposed a theoretical kernel

learning framework that works well in practice to learn all parameters of a classification model that uses explicitly placed kernels in the data domain. We showcased the method on the contemporary problem of probabilistic occupancy mapping in robotics. Experimental results validate the method with significant improvements in representational performance for robotic mapping in highly unstructured and nonstationary environments.

In Chapter 4 we introduced **a theoretical framework for online parameter transfer using the theory of optimal transport.** This contribution improved on the work from the previous chapter by proposing a way to adapt previously trained kernel models, on demand and in real time, using a parallelized decomposition of the learning problem. We demonstrated the process on the domain adaptation paradigms of *intra* and *inter*-domain transfer in a robotic mapping problem. The most important takeaway from this work was the realisation that if there is a clear *geometric* coupling between observations and model then transforming the online learning problem that is approximately equivalent to the original computationally demanding optimization problem.

Switching gears to the spectral domain, in Chapter 5, we presented **a new method that learns quantile representations of generalised stationary kernels in the Fourier domain.** We introduced a relaxation of an approximate Fourier representation of a kernel that adapts to the observed data. We additionally showed that the quantile representation naturally allows for improved approximation efficiency through quasi-Monte Carlo integral sampling. We validated the method on various datasets and problems that exhibit complex phenomena like pseudo-periodicity that cannot be modelled with conventional stationary kernels within limited data.

Finally, in Chapter 6 we closed the set of contributions with **an approach to learning non-stationary approximate Gaussian process models via input warping.** We introduced an intuitive measure-value input-dependent warping function alongside its multi-level warping extension by propagating moments. Comprehensive experiments showed competitive performance with many deep learning and alternative approximate Gaussian process methods.

7.2 Future Work

7.2.1 Black Box Variational Inference and Kernel Learning

Since kernel methods have also been successfully used in a variety of nonlinear path planning methods (Mukadam, Yan and Boots 2016; Marinho et al. 2016; Vallicrosa and Ridao 2018) we plan to extend these ideas to path planning so that mapping and path planning can be performed simultaneously in real-world in an end-to-end fashion under one framework.

7.2.2 Parameter Optimal Transport

Although our method presented in Chapter 3 was demonstrated in the context of occupancy mapping, there are many other potential applications in robotics. For example, the theory can be potentially used for domain adaptation of policy parameters where a policy is trained in one environment and needs to be transferred to another. For example, say a robotic arm is trained to grasp objects on a table and performs well on this particular task. One could then, in principle transport these prior learned policies for use in another arm without retraining the policy from scratch. Finally, one could envision the same methodology being used for sim2real where models are learned in simulation and transferred to the physical world, saving significant time and cost in running real robots.

7.2.3 Quantile Kernel Features

Inspired by recent ideas in automated machine learning, fundamental connections with harmonic analysis and measure theory, we believe more general and flexible representations of kernels will open doors to compelling new directions in Bayesian inference techniques and kernel learning. As future work, it would be interesting to investigate connections with Copula methods (Nelsen 2007), non-stationary kernels (Remes, Heinonen and Kaski 2017), and alternative quantile function parameterizations.

7.2.4 Input Warped Kernel Learning

The functional warping scheme presented in Chapter 6 are not monotonic increasing warpings; in other words they are *not* bijective mappings. While not a necessary condition, non-bijective mappings may introduce unnecessary complexity in the optimization loss surface due to the potential for many-to-many inputs warpings to occur. We conjecture this would introduce arbitrary discontinuities in the loss landscape making optimization much harder than it need be. It would be worthwhile investigating *strictly* monotone mappings of the input space to improve both the convergence speed and stability of the optimization problem. Finally, there would be scope to investigate connections between our work and the recently proposed Neural Tangent Kernel (Jacot, Gabriel and Hongler 2018) perspective on kernel learning with neural networks.

7.3 La Fin

So I have just one wish for you – the good luck to be somewhere where you are free to maintain the kind of integrity I have described, and where you do not feel forced by a need to maintain your position in the organization, or financial support, or so on, to lose your integrity. May you have that freedom.

$\sim$ Richard P. Feynman

Bibliography

Akametalu, A. K., Kaynama, S., Fisac, J. F., Zeilinger, M. N., Gillula, J. H. and Tomlin, C. J. (2014). 'Reachability-based safe learning with Gaussian processes'. In: *IEEE Conference on Decision and Control (CDC)*.

Alvarez, M. A., Rosasco, L. and Lawrence, N. D. (2011). *Kernels for Vector-Valued Functions: a Review*. Tech. rep. MIT - Computer Science and Artificial Intelligence Laboratory.

Ammar, H. B., Eaton, E., Luna, J. M. and Ruvolo, P. (2015). 'Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning'. In: *International Joint Conference on Artificial Intelligence (IJCAI)*.

Anderes, E. B., Stein, M. L. et al. (2008). 'Estimating deformations of isotropic Gaussian random fields on the plane'. In: *The Annals of Statistics*.

Arbuckle, D., Howard, A. and Mataric, M. (2002). 'Temporal occupancy grids: a method for classifying the spatio-temporal properties of the environment'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Arjovsky, M., Chintala, S. and Bottou, L. (2017). 'Wasserstein gan'. In: *arXiv preprint arXiv:1701.07875*.

Bach, F. (2017). 'On the equivalence between kernel quadrature rules and random feature expansions'. In: *Journal of Machine Learning Research*.

Bach, F. R., Lanckriet, G. R. and Jordan, M. I. (2004). 'Multiple kernel learning, conic duality, and the SMO algorithm'. In: *International Conference on Machine Learning (ICML)*. ACM.

Bauer, H. (1981). *Probability theory and elements of measure theory*. Probability and mathematical statistics. Academic Press.

Besl, P. J. and McKay, N. D. (1992). 'Method for registration of 3-D shapes'. In: *Sensor fusion IV: control paradigms and data structures*. International Society for Optics and Photonics.

Bishop, C. (2007). 'Pattern Recognition and Machine Learning (Information Science and Statistics), 1st edn. 2006. corr. 2nd printing edn'. In: *Springer, New York*.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bochner, S. (1933). 'Monotone Funktionen, Stieltjessche Integrale und harmonische Analyse'. In: *Mathematische Annalen*. URL: http://eudml.org/doc/159644.

Bochner, S. (1932). *Vorlesungen über Fouriersche Integrale: von S. Bochner*. Akad. Verl.-Ges.

Bohg, J. (2011). 'Multi-modal scene understanding for robotic grasping'. PhD thesis. KTH Royal Institute of Technology.

Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., Konolige, K. et al. (2018). 'Using simulation and domain adaptation to improve efficiency of deep robotic grasping'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Brochu, E., Cora, V. M. and De Freitas, N. (2010). 'A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning'. In: *arXiv preprint arXiv:1012.2599*.

Brun, F. and Yoshida, D. T. (n.d.). *creeprupt dataset*. URL: https://www.phase-trans.msm.cam.ac.uk/map/data/materials/creeprupt-b.html.

Bui, T., Hernández-Lobato, D., Hernandez-Lobato, J., Li, Y. and Turner, R. (2016). 'Deep Gaussian processes for regression using approximate expectation propagation'. In: *International Conference on Machine Learning (ICML)*.

Burchfiel, B. and Konidaris, G. (2017). 'Bayesian Eigenobjects: A Unified Framework for 3D Robot Perception.' In: *Robotics: Science and Systems (RSS)*.

Calandra, R., Peters, J., Rasmussen, C. E. and Deisenroth, M. P. (2016). 'Manifold Gaussian processes for regression'. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE.

Campbell, J. and Amor, H. B. (2017). 'Bayesian Interaction Primitives: A SLAM Approach to Human-Robot Interaction'. In: *Conference on Robot Learning (CoRL)*.

Casella, G. and Berger, R. L. (2002). *Statistical inference*. Duxbury Pacific Grove, CA.

Chambers, C. P. (2009). 'An axiomatization of quantiles on the domain of distribution functions'. In: *Mathematical Finance*.

Choromanski, K., Rowland, M., Sarlos, T., Sindhwani, V., Turner, R. and Weller, A. (2018). 'The Geometry of Random Features'. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Chowdhary, G., Kingravi, H. A., How, J. P. and Vela, P. A. (2014). 'Bayesian nonparametric adaptive control using gaussian processes'. In: *IEEE Transactions on Neural Networks and Learning Systems*.

Cole, D., Martin-Moran, C., Sheard, A., Bhadeshia, H. and MacKay, D. (2000). 'Modelling creep rupture strength of ferritic steel welds'. In: *Science and Technology of Welding and Joining*.

Cortes, C. and Vapnik, V. (1995). 'Support-vector networks'. In: *Machine learning*.

Courty, N., Flamary, R., Tuia, D. and Rakotomamonjy, A. (2017). 'Optimal transport for domain adaptation'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*.

Cristianini, N. and Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press.

Cutajar, K., Bonilla, E. V., Michiardi, P. and Filippone, M. (2017). 'Random feature expansions for deep Gaussian processes'. In: *International Conference on Machine Learning (ICML)*.

Cuturi, M. (2013). 'Sinkhorn distances: Lightspeed computation of optimal transport'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Dai, B., Xie, B., He, N., Liang, Y., Raj, A., Balcan, M.-F. F. and Song, L. (2014). 'Scalable kernel methods via doubly stochastic gradients'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Damianou, A. and Lawrence, N. (2013). 'Deep Gaussian processes'. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Deisenroth, M. and Rasmussen, C. E. (2011). 'PILCO: A model-based and data-efficient approach to policy search'. In: *International Conference on Machine Learning (ICML)*.

Dick, J., Kuo, F. Y. and Sloan, I. H. (2013). 'High-dimensional integration: the quasi-Monte Carlo way'. In: *Acta Numerica*.

Dick, J. and Pillichshammer, F. (2010). *Digital nets and sequences: discrepancy theory and quasi–Monte Carlo integration*. Cambridge University Press.

Dissanayake, M. G., Newman, P., Clark, S., Durrant-Whyte, H. F. and Csorba, M. (2001). 'A solution to the simultaneous localization and map building (SLAM) problem'. In: *IEEE Transactions on robotics and automation*.

Doherty, K., Wang, J. and Englot, B. (2016). 'Probabilistic Map Fusion for Fast, Incremental Occupancy Mapping with 3D Hilbert Maps'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A. and Koltun, V. (2017). 'CARLA: An open urban driving simulator'. In: *arXiv preprint arXiv:1711.03938*.

Dua, D. and Graff, C. (2017). *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml.

Duong, T., Yip, M. and Atanasov, N. (2022). 'Autonomous navigation in unknown environments with sparse bayesian kernel-based occupancy mapping'. In: *IEEE Transactions on Robotics*.

Dutordoir, V., Salimbeni, H., Hensman, J. and Deisenroth, M. (2018). 'Gaussian Process Conditional Density Estimation'. In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett. Curran Associates, Inc.

Duvenaud, D. (2014). 'Automatic model construction with Gaussian processes'. PhD thesis. University of Cambridge.

Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J. B. and Ghahramani, Z. (2013). 'Structure discovery in nonparametric regression through compositional kernel search'. In: *arXiv preprint arXiv:1302.4922*.

Duvenaud, D., Rippel, O., Adams, R. and Ghahramani, Z. (2014). 'Avoiding pathologies in very deep networks'. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Elfes, A. (1987). 'Sonar-based real-world mapping and navigation'. In: *IEEE Journal of Robotics and Automation*.

Elfes, A. (1989). 'Occupancy grids: a probabilistic framework for robot perception and navigation'. PhD thesis. Carnegie Mellon University.

Fang, X., Bai, H., Guo, Z., Shen, B., Hoi, S. and Xu, Z. (2018). 'DART: Domain-Adversarial Residual-Transfer Networks for Unsupervised Cross-Domain Image Classification'. In: *arXiv preprint arXiv:1812.11478*.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M. and Hutter, F. (2015). 'Efficient and robust automated machine learning'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Finn, C., Abbeel, P. and Levine, S. (2017). 'Model-agnostic meta-learning for fast adaptation of deep networks'. In: *International Conference on Machine Learning (ICML)*.

Friedman, N., Linial, M., Nachman, I. and Pe'er, D. (2000). 'Using Bayesian networks to analyze expression data'. In: *Journal of computational biology*.

Fritsch, F. N. and Carlson, R. E. (1980). 'Monotone piecewise cubic interpolation'. In: *SIAM Journal on Numerical Analysis*.

Gablonsky, J. M. and Kelley, C. T. (2001). 'A locally-biased form of the DIRECT algorithm'. In: *Journal of Global Optimization*.

Galliani, P., Dezfouli, A., Bonilla, E. and Quadrianto, N. (2017). 'Gray-box Inference for Structured Gaussian Process Models'. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Geiger, A., Lenz, P., Stiller, C. and Urtasun, R. (2013). 'Vision meets robotics: The KITTI dataset'. In: *The International Journal of Robotics Research*.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A. and Rubin, D. B. (2013). *Bayesian data analysis*. CRC press.

Genevay, A., Peyré, G. and Cuturi, M. (2018). 'Learning generative models with sinkhorn divergences'. In: *1608-1617*.

Ghahramani, Z. (2014). 'The Automatic Statistician'. In.

Ghifary, M., Kleijn, W. B., Zhang, M., Balduzzi, D. and Li, W. (2016). 'Deep reconstruction-classification networks for unsupervised domain adaptation'. In: *European Conference on Computer Vision (ECCV)*. Springer.

Gibbs, M. N. (1997). 'Bayesian Gaussian Processes for Regression and Classification'. In: *Ph. D. Thesis, Department of Physics, University of Cambridge*.

Gittens, A. and Mahoney, M. W. (2013). 'Revisiting the Nyström method for improved large-scale machine learning'. In: *J. Mach. Learn. Res*.

González, R. and Woods, R. (2008). 'Digital Image Processing. ISBN: 9780131687288'. In: *Prentice Hall*.

Gramacy, R. B. (2005). 'Bayesian treed Gaussian process models'. PhD dissertation.

Gronau, Q. F., Van Erp, S., Heck, D. W., Cesario, J., Jonas, K. J. and Wagenmakers, E.-J. (2017). 'A Bayesian model-averaged meta-analysis of the power pose effect with informed and default priors: The case of felt power'. In: *Comprehensive Results in Social Psychology*.

Guizilini, V. and Ramos, F. (2017a). 'Learning to Reconstruct 3D Structures for Occupancy Mapping'. In: *Proceedings of Robotics: Science and Systems (RSS)*. Cambridge, MA, USA.

Guizilini, V. C. and Ramos, F. T. (2017b). 'Unsupervised Feature Learning for 3D Scene Reconstruction with Occupancy Maps.' In: *AAAI Conference on Artificial Intelligence (AAAI)*.

Han, S., Liao, X., Dunson, D. and Carin, L. (2016). 'Variational Gaussian copula inference'. In: *Artificial Intelligence and Statistics*.

Hastie, T. J. (2017). 'Generalized additive models'. In: *Statistical models in S*. Routledge.

Hegde, P., Heinonen, M., Lähdesmäki, H. and Kaski, S. (2019). 'Deep learning with differential Gaussian process flows'. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Heinonen, M., Mannerström, H., Rousu, J., Kaski, S. and Lähdesmäki, H. (2016). 'Non-stationary Gaussian process regression with hamiltonian monte carlo'. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Hensman, J., Fusi, N. and Lawrence, N. D. (2013a). 'Gaussian Processes for Big Data'. In: *Conference on Uncertainty in Artificial Intelligence (UAI)*. Citeseer.

Hensman, J., Fusi, N. and Lawrence, N. D. (2013b). 'Gaussian Processes for Big Data'. In: *Conference on Uncertainty in Artificial Intelligence (UAI)*.

Hensman, J., Matthews, A. G. d. G. and Ghahramani, Z. (2015). 'Scalable Variational Gaussian Process Classification'. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Higdon, D., Swall, J. and Kern, J. (1999). 'Non-stationary spatial modeling'. In: *Bayesian statistics*.

Hoffman, J., Wang, D., Yu, F. and Darrell, T. (2016). 'Fcns in the wild: Pixel-level adversarial and constraint-based adaptation'. In: *arXiv preprint arXiv:1612.02649*.

Hofmann, T., Schölkopf, B. and Smola, A. J. (2008). 'Kernel methods in machine learning'. In: *The Annals of Statistics*.

Inc., G. (2018). 'Cloud AutoML'. In.

Isele, D., Rostami, M. and Eaton, E. (2016). 'Using Task Features for Zero-Shot Knowledge Transfer in Lifelong Learning.' In: *International Joint Conference on Artificial Intelligence (IJCAI)*.

Jaakkola, T. and Jordan, M. (1997). 'A variational approach to Bayesian logistic regression models and their extensions'. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Jacot, A., Gabriel, F. and Hongler, C. (2018). 'Neural tangent kernel: Convergence and generalization in neural networks'. In: *Advances in neural information processing systems*.

Jidling, C., Wahlström, N., Wills, A. and Schön, T. B. (2017). 'Linearly constrained Gaussian processes'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Johnson, S. G. (2014). *The NLopt nonlinear-optimization package*.

Kim, S. and Kim, J. (2013). 'Occupancy mapping and surface reconstruction using local Gaussian processes with kinect sensors'. In: *IEEE Transactions on Cybernetics*.

Kim, S., Kim, J. et al. (2014). 'Recursive bayesian updates for occupancy mapping and surface reconstruction'. In.

Kim, T., Cha, M., Kim, H., Lee, J. K. and Kim, J. (2017). 'Learning to discover cross-domain relations with generative adversarial networks'. In: *International Conference on Machine Learning (ICML)*.

Kingma, D. P. and Welling, M. (2013). 'Auto-encoding variational bayes'. In: *arXiv preprint arXiv:1312.6114*.

Kingma, D. P. and Welling, M. (2014). *Auto-Encoding Variational Bayes*. Electronic Article.

Kingravi, H. A., Chowdhary, G., Vela, P. A. and Johnson, E. N. (2012). 'Reproducing kernel Hilbert space approach for the online update of radial bases in neuro-adaptive control'. In: *IEEE Transactions on Neural Networks and Learning Systems*.

Kingravi, H. A., Maske, H. R. and Chowdhary, G. (2016). 'Kernel observers: systems-theoretic modeling and inference of spatiotemporally evolving processes'. In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Kulis, B. and Grauman, K. (2009). 'Kernelized locality-sensitive hashing for scalable image search'. In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE.

Lanckriet, G. R., Cristianini, N., Bartlett, P., Ghaoui, L. E. and Jordan, M. I. (2004). 'Learning the kernel matrix with semidefinite programming'. In: *Journal of Machine Learning Research*.

Lasota, P. A., Fong, T., Shah, J. A. et al. (2017). 'A survey of methods for safe human-robot interaction'. In: *Foundations and Trends® in Robotics*.

Lázaro-Gredilla, M. (2012). 'Bayesian warped Gaussian processes'. In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C. E. and Figueiras-Vidal, A. R. (2010). 'Sparse spectrum Gaussian process regression'. In: *Journal of Machine Learning Research*.

Liu, M.-Y., Breuel, T. and Kautz, J. (2017). 'Unsupervised image-to-image translation networks'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Lopez-Paz, D., Muandet, K. and Recht, B. (2015). 'The randomized causation coefficient'. In: *Journal of Machine Learning Research*.

MacKay, D. J. (1998). 'Introduction to Gaussian processes'. In: *NATO ASI Series F Computer and Systems Sciences*.

Mania, H., Guy, A. and Recht, B. (2018). 'Simple random search provides a competitive approach to reinforcement learning'. In: *arXiv preprint arXiv:1803.07055*.

Marinho, Z., Boots, B., Dragan, A., Byravan, A., Gordon, G. J. and Srinivasa, S. (2016). 'Functional Gradient Motion Planning in Reproducing Kernel Hilbert Spaces'. In: *Proceedings of Robotics: Science and Systems (RSS)*. Michigan, USA.

Martinez-Cantin, R. (2017). 'Bayesian optimization with adaptive kernels for robot control'. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Meier, F., Hennig, P. and Schaal, S. (2014). 'Efficient Bayesian local model learning for control'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Meyer-Delius, D., Beinhofer, M. and Burgard, W. (2012). 'Occupancy Grid Models for Robot Mapping in Changing Environments'. In: *AAAI Conference on Artificial Intelligence (AAAI)*.

Monge, G. (1781). 'Mémoire sur la théorie des déblais et des remblais'. In: *Histoire de l'Académie Royale des Sciences de Paris*.

Mukadam, M., Yan, X. and Boots, B. (2016). 'Gaussian process motion planning'. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE.

Neal, R. M. (1993). 'Probabilistic inference using Markov chain Monte Carlo methods'. In.

Neal, R. M. (2012). *Bayesian learning for neural networks*. Springer Science & Business Media.

Nelsen, R. B. (2007). *An introduction to copulas*. Springer Science & Business Media.

Neudecker, H., Liu, S. and Polasek, W. (1995). 'The Hadamard product and some of its applications in statistics'. In: *Statistics*.

Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E. and Liang, E. (2006). 'Autonomous inverted helicopter flight via reinforcement learning'. In: *Experimental robotics IX*. Springer.

Nguyen, T. V. and Bonilla, E. V. (2014). 'Automated variational inference for Gaussian process models'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Nguyen-Tuong, D., Seeger, M. and Peters, J. (2009). 'Model learning with local gaussian process regression'. In: *Advanced Robotics*.

Norouzi, M., Miro, J. V. and Dissanayake, G. (2017). 'Planning stable and efficient paths for reconfigurable robots on uneven terrain'. In: *Journal of Intelligent & Robotic Systems*.

Norouzii, M., Miro, J. V. and Dissanayake, G. (2016). 'Probabilistic stable motion planning with stability uncertainty for articulated vehicles on challenging terrains'. In: *Autonomous Robots*.

O'Callaghan, S. T. and Ramos, F. T. (2012). 'Gaussian process occupancy maps'. In: *The International Journal of Robotics Research (IJRR)*.

O'Callaghan, S. T. and Ramos, F. T. (2014). 'Gaussian process occupancy maps for dynamic environments'. In: *The 14th International Symposium on Experimental Robotics (ISER)*. Marrakesh, Morocco.

O'Callaghan, S. T., Ramos, F. T. and Durrant-Whyte, H. (2009). 'Contextual occupancy maps using Gaussian processes'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Kobe, Japan.

Oliva, J. B., Dubey, A., Wilson, A. G., Póczos, B., Schneider, J. and Xing, E. P. (2016). 'Bayesian nonparametric kernel-learning'. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Oliveira, R., Ott, L. and Ramos, F. (2019). 'Bayesian optimisation under uncertain inputs'. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. Naha, Okinawa, Japan.

Paciorek, C. J. and Schervish, M. J. (2004). 'Nonstationary covariance functions for Gaussian process regression'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Paisley, J., Blei, D. and Jordan, M. (2012). 'Variational Bayesian inference with stochastic search'. In: *arXiv preprint arXiv:1206.6430*.

Pan, S. J. and Yang, Q. (2009). 'A survey on transfer learning'. In: *IEEE Transactions on Knowledge and Data Engineering*.

Pan, Y., Yan, X., Theodorou, E. A. and Boots, B. (2017). 'Prediction under uncertainty in sparse spectrum Gaussian processes with applications to filtering and control'. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. Proceedings of Machine Learning Research.

Parzen, E. (1980). *Quantile Functions, Convergence in Quantile, and Extreme Value Distribution Theory.* Tech. rep. Texas A and M univ college station inst of statistics.

Pastur, L. A. (1973). 'Spectra of random self adjoint operators'. In: *Russian mathematical surveys*.

Perrot, M., Courty, N., Flamary, R. and Habrard, A. (2016). 'Mapping estimation for discrete optimal transport'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Powell, M. J. (1994). 'A direct search optimization method that models the objective and constraint functions by linear interpolation'. In: *Advances in optimization and numerical analysis*. Springer.

Rahimi, A. and Recht, B. (2007a). 'Random features for large-scale kernel machines'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Rahimi, A. and Recht, B. (2008). 'Weighted sums of random kitchen sinks: Replacing minimization with randomisation in learning'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Rahimi, A. and Recht, B. (2007b). 'Random Features for Large-Scale Kernel Machines'. In: *Conference on Neural Information Processing Systems*.

Rajeswaran, A., Lowrey, K., Todorov, E. V. and Kakade, S. M. (2017). 'Towards generalization and simplicity in continuous control'. In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Ramos, F. and Ott, L. (2015). 'Hilbert maps: scalable continuous occupancy mapping with stochastic gradient descent'. In: *Robotics: Science and Systems (RSS)*.

Rana, M. A., Mukadam, M., Ahmadzadeh, S. R., Chernova, S. and Boots, B. (2017). 'Towards Robust Skill Generalization: Unifying Learning from Demonstration and Motion Planning'. In: *Conference on Robot Learning (CoRL)*.

Rasmussen, C. E. (2004). 'Gaussian processes in machine learning'. In: *Advanced lectures on machine learning*. Springer.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Remes, S., Heinonen, M. and Kaski, S. (2017). 'Non-Stationary Spectral Kernels'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Robert, C. P. (2004). *Monte carlo methods*. Wiley Online Library.

Rudi, A. and Rosasco, L. (2017). 'Generalization properties of learning with random features'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Sadeghi, F., Toshev, A., Jang, E. and Levine, S. (2017). 'Sim2real view invariant visual servoing by recurrent control'. In: *arXiv preprint arXiv:1712.07642*.

Salimbeni, H. and Deisenroth, M. (2017). 'Doubly stochastic variational inference for deep Gaussian processes'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Samo, Y.-L. K. and Roberts, S. (2015). 'Generalized spectral kernels'. In: *arXiv preprint arXiv:1506.02236*.

Sampson, P. D. and Guttorp, P. (1992). 'Nonparametric estimation of nonstationary spatial covariance structure'. In: *Journal of the American Statistical Association*.

Sankaran, P., Nair, N. U. and Midhu, N. (2016). 'A new quantile function with applications to reliability analysis'. In: *Communications in Statistics-Simulation and Computation*.

Schölkopf, B., Smola, A. and Müller, K.-R. (1998). 'Nonlinear component analysis as a kernel eigenvalue problem'. In: *Neural computation*.

Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. the MIT Press.

Senanayake, R., O'Callaghan, S. and Ramos, F. (2017). 'Learning highly dynamic environments with stochastic variational inference'. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Senanayake, R., Ott, L., O'Callaghan, S. and Ramos, F. (2016a). 'Spatio-Temporal Hilbert Maps for Continuous Occupancy Representation in Dynamic Environments'. In: *Neural Information Processing Systems (NIPS)*.

Senanayake, R., Ott, L., O'Callaghan, S. and Ramos, F. T. (2016b). 'Spatio-temporal Hilbert maps for continuous occupancy representation in dynamic environments'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Senanayake, R. and Ramos, F. (2017). 'Bayesian hilbert maps for dynamic continuous occupancy mapping'. In: *Conference on Robot Learning (CoRL)*.

Senanayake, R. and Ramos, F. (2018). 'Building continuous occupancy maps with moving robots'. In: *AAAI Conference on Artificial Intelligence (AAAI)*.

Senanayake, R., Tompkins, A. and Ramos, F. (2018). 'Automorphing Kernels for Nonstationarity in Mapping Unstructured Environments'. In: *Conference on Robot Learning (CoRL)*.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge university press.

Sinkhorn, R. and Knopp, P. (1967). 'Concerning nonnegative matrices and doubly stochastic matrices'. In: *Pacific Journal of Mathematics*.

Snelson, E., Ghahramani, Z. and Rasmussen, C. E. (2004). 'Warped Gaussian processes'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Snoek, J., Swersky, K., Zemel, R. and Adams, R. (2014). 'Input warping for Bayesian optimization of non-stationary functions'. In: *International Conference on Machine Learning (ICML)*.

Solomon, J., De Goes, F., Peyré, G., Cuturi, M., Butscher, A., Nguyen, A., Du, T. and Guibas, L. (2015). 'Convolutional wasserstein distances: Efficient optimal transportation on geometric domains'. In: *ACM Transactions on Graphics (TOG)*.

Sriperumbudur, B. and Szabó, Z. (2015). 'Optimal rates for random fourier features'. In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Stachniss, C. and Burgard, W. (2005). 'Mobile robot mapping and localization in non-static environments'. In: *AAAI Conference on Artificial Intelligence (AAAI)*.

Steffen, M. (1990). 'A simple method for monotonic interpolation in one dimension'. In: *Astronomy and Astrophysics*.

Stein, M. L. (1999). 'Interpolation of spatial data: some theory for kriging'. In.

Steinbrecher, G. and Shaw, W. T. (2008). 'Quantile mechanics'. In: *European journal of applied mathematics*.

Stineman, R. W. (1980). 'A consistently well-behaved method of interpolation'. In: *Creative Computing*.

Sun, S., Zhang, G., Wang, C., Zeng, W., Li, J. and Grosse, R. (2018). 'Differentiable Compositional Kernel Learning for Gaussian Processes'. In: *International Conference on Machine Learning (ICML)*.

Sutherland, D. J. and Schneider, J. (2015). 'On the error of random Fourier features'. In.

Thrun, S. (2000). 'Probabilistic algorithms in robotics'. In: *AI Magazine*.

Thrun, S., Burgard, W. and Fox, D. (2005). *Probabilistic robotics*. MIT press.

Titsias, M. K. (2009). 'Variational Learning of Inducing Variables in Sparse Gaussian Processes.' In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. Clearwater Beach, FL, USA.

Tompkins, A. (2018). 'Bayesian Spatio-Temporal Modelling with Fourier Features'. PhD thesis. The University of Sydney.

Tompkins, A., Senanayake, R., Morere, P. and Ramos, F. (2019). 'Black Box Quantiles for Kernel Learning'. In: *International Conference on Artificial Intelligence and Statistics*.

Ton, J.-F., Flaxman, S., Sejdinovic, D. and Bhatt, S. (2018). 'Spatial mapping with Gaussian processes and nonstationary Fourier features'. In: *Spatial statistics*.

Torgo, L. (2019). *Regression datasets*. https://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html.

Tran, D., Hoffman, M. D., Saurous, R. A., Brevdo, E., Murphy, K. and Blei, D. M. (2017). 'Deep probabilistic programming'. In: *International Conference on Learning Representations*.

Tran, D., Kucukelbir, A., Dieng, A. B., Rudolph, M., Liang, D. and Blei, D. M. (2016). 'Edward: A library for probabilistic modeling, inference, and criticism'. In: *arXiv preprint arXiv:1610.09787*.

Uhlenbeck, G. E. and Ornstein, L. S. (1930). 'On the theory of the Brownian motion'. In: *Physical review*.

Unhelkar, V., Guan, C., Roy, N. and Shah, J. (2018). 'Enabling Robot Teammates to Learn Latent States of Human Collaborators'. In.

Unhelkar, V. V. and Shah, J. A. (2018). 'Learning Models of Sequential Decision-Making without Complete State Specification using Bayesian Nonparametric Inference and Active Querying'. In: *Massachusetts Institute of Technology*.

Ustyuzhaninov, I., Kazlauskaite, I., Ek, C. H. and Campbell, N. (2020). 'Monotonic Gaussian Process Flows'. In: *International Conference on Artificial Intelligence and Statistics*. PMLR.

Valera, I. and Ghahramani, Z. (2017). 'Automatic discovery of the statistical types of variables in a dataset'. In: *International Conference on Machine Learning (ICML)*.

Vallicrosa, G. and Ridao, P. (2018). 'H-SLAM: Rao-Blackwellized Particle Filter SLAM Using Hilbert Maps'. In: *Sensors*.

Villani, C. (2003). *Topics in optimal transportation*. American Mathematical Soc.

Villani, C. (2008). *Optimal transport: old and new*. Springer Science & Business Media.

Wainwright, M. J., Jordan, M. I. et al. (2008). 'Graphical models, exponential families, and variational inference'. In: *Foundations and Trends® in Machine Learning*.

Wang, C. and Neal, R. M. (2012). *Gaussian Process Regression with Heteroscedastic or Non-Gaussian Residuals*. Tech. rep. University of Toronto. eprint: 1212.6246. URL: http://arxiv.org/abs/1212.6246.

Wang, J. and Englot, B. (2016). 'Fast, Accurate Gaussian Process Occupancy Maps via Test-Data Octrees and Nested Bayesian Fusion'. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Wasserman, L. A. (2006). *All of nonparametric statistics: with 52 illustrations*. Springer.

Whitman, J. and Chowdhary, G. (2017). 'Learning Dynamics Across Similar Spatiotemporally-Evolving Physical Systems'. In: *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*. Proceedings of Machine Learning Research. PMLR.

Williams, C. K. and Seeger, M. (2001). 'Using the Nyström method to speed up kernel machines'. In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Wilson, A. G. and Adams, R. (2013). 'Gaussian process kernels for pattern discovery and extrapolation'. In: *International Conference on Machine Learning (ICML)*.

Wilson, A. G., Gilboa, E., Nehorai, A. and Cunningham, J. P. (2013). 'Gpatt: Fast multidimensional pattern extrapolation with Gaussian processes'. In: *arXiv preprint arXiv:1310.5288*.

Wilson, A. G., Gilboa, E., Nehorai, A. and Cunningham, J. P. (2014). 'Fast kernel learning for multidimensional pattern extrapolation'. In: *Advances in Neural Information Processing Systems (NIPS)*.

Wilson, A. G., Hu, Z., Salakhutdinov, R. and Xing, E. P. (2016). 'Deep kernel learning'. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Wilson, A. G., Knowles, D. A. and Ghahramani, Z. (2012). 'Gaussian process regression networks'. In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*.

Wilson, A. G. and Nickisch, H. (2015). ‘Kernel interpolation for scalable structured Gaussian processes (KISS-GP)’. In: *International Conference on Machine Learning (ICML)*.

Wulfmeier, M., Bewley, A. and Posner, I. (2018). ‘Incremental adversarial domain adaptation for continually changing environments’. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Wüthrich, M., Cifuentes, C. G., Trimpe, S., Meier, F., Bohg, J., Issac, J. and Schaal, S. (2016). ‘Robust Gaussian Filtering using a Pseudo Measurement’. In: *American Control Conference (ACC)*.

Yang, J., Sindhwani, V., Avron, H. and Mahoney, M. (2014). ‘Quasi-Monte Carlo feature maps for shift-invariant kernels’. In: *International Conference on Machine Learning (ICML)*.

Zhu, J.-Y., Park, T., Isola, P. and Efros, A. A. (2017). ‘Unpaired image-to-image translation using cycle-consistent adversarial networks’. In: *International Conference on Computer Vision (ICCV)*.