Spatio-Temporal Inference in Non-Homogenous Poisson Processes

Ben Morris
Student ID: 1865136
Email: ben.morris@warwick.ac.uk

A thesis submitted to the University of Warwick
in partial fulfillment of the requirements for the degree of
MSc in Statistics

Department of Statistics
University of Warwick
Coventry, CV4 7AL
United Kingdom

4th **September 2019**

Abstract

The aim of this thesis is to explore the Multi-task Cox Process Models (MCPM) framework, and to extend the model into the spatio-temporal setting. I explore kernel methods in the spatial and temporal dimensions, looking at both stationary and non-stationary kernels, before combining them to fascilitate spatio-temporal inference. I first consider separable kernels with the assumption of independence between space and time, and provide an implementation of these. I then review current literature on non-separable kernels, which aim to model interactions between space and time, and discuss their suitability. The methods implemented are tested on synthetic data before being applied on a real world dataset.

Acknowledgements

I would like to thank my co-supervisors, Virginia Aglietti and Dr Theo Damoulas, for their support and guidance throughout my thesis. The expertise and advice of Ollie Hamelijnck and Dr Kangrui Wang from the Alan Turing Intstitute also proved invaluable.

I would also like to thank my family for their support and encouragement throughout my dissertation.

Contents

List of Figures

List of Tables

1 Introduction

Multi-task problems are settings involving processes that are highly correlated with one another. The Multi-task Cox Process Model (MCPM) methodology presented in (Aglietti et al., 2019) is a generalization of the Log Gaussian Cox process (LGCP) framework (Moller, et al., 1998) that exploits the dependencies between these processes to model multiple correlated point data jointly. The work in (Aglietti et al., 2018) considers models for either spatial or temporal data separately, and the main aim of this thesis is to extend the model into the spatio-temporal direction by utilising spatio-temporal kernels to measure both spatial and temporal correlations of input data.

A prominent area of application for these types of model is Geostatistics, an area of statistics concentrated on spatio-temporal datasets with many problems characterized by correlated point data. The UK Crime dataset I use in my experiments is an example of a multi-task setting for which these models prove useful. The data contains counts of the occurences of different categories of crime, and as these counts are correlated they can be predictive of one another. The MCPM model can exploit these dependencies.

One of the most extensively developed frameworks for modelling this data is the LGCP model, a non-homogenous Poisson process (NHPP) that can be used to model count or point processes. LGCP's are a NHPP that uses a Gaussian Process (GP) prior as its intensity function, and are one of the most extensively researched frameworks of these modelling approaches. However, the model suffers from serious computational problems due to the scalability problems of GP's, which is exacerbated when considering multiple correlated tasks. Therefore, scalable inference algorithms for these models have been proposed and are an important area of current research (Aglietti et al., 2019).

The MCPM framework improves on some of the problems experienced by LGCP's. The introduction of stochastic mixing weights provides additional model flexibility and the ability to propagate uncertainty, whilst posterior estimation over both latent and mixing processes using variational inference (VI) ensures the model is orders of magnitude faster than competing approaches. Closed-form expectations in the variational objective ensure Monte Carlo estimation is completely avoided, unlike many other VI methods. The MCPM model in (Aglietti et al., 2019) can be extended to higher dimensions, however so far it has only been tested in a one-dimensional setting.

The work in this paper looks to build on (Aglietti et al., 2019) by utilising spatio-temporal kernels in order to conduct inference on multi-task problems in the spatio-temporal setting. I review different classes of one-dimensional kernels, both stationary and non-stationary, before combining them for spatio-temporal inference. Initially, I consider independence between space and time by exploring separable kernels, taking ideas from (Flaxman, 2015). I then review non-separable kernels, considering work in (Cressie et al., 1999), (Gneiting, 2002) and (Fonseca, 2010).

This thesis is organized as follows. In Chapter 2, I discuss important background material including GP's, VI, and scalable VI in the context of GP models with non-Gaussian likelihoods. In Chapter 3 I cover non-homogenous Poisson processes, including Log Gaussian Cox processes and Multi-task Cox Process models in detail. Chapter 4 provides an overview of kernel methods, including their properties, and examples of common stationary and non-stationary kernels. In chapter 5 I explore inference in the spatio-temporal direction. I first consider Separable kernels by combining individual spatial and temporal kernels presented in chapter 4, before considering non-separable kernels. I present my experiments in Chapter 6, including a synthetic experiment, as well as an experiment on a real world UK Crime dataset.

2 Background

2.1 Gaussian Processes

2.1.1 Definition

A GP is a collection of random variables, any finite number of which have consistent Gaussian distributions. GP's are a generalization of the Gaussian probability distribution, they extend multivariate Gaussian distributions into an infinite dimensionality (Rasmussen et al., 2006). A Gaussian distribution is specified as: $\mathbf{f(x)} \sim \mathbf{N(}\mu\mathbf{,}\ \sigma\mathbf{)}$. In contrast, a GP is as follows: $\mathbf{f(x)} \sim \mathbf{GP(m(x),\ K(x,x'))}$, where $\mathbf{m(x)}$ is the mean function, and $\mathbf{K(x,x')}$ the covariance or kernel.

GP's are non-parametric models, which makes them more flexible than linear models, however we still need to be carefuly in terms of the covariance function chosen when fitting them to a dataset. The prior specification for a GP model, its mean and covariance functions, is important as it fixes the properties of the functions considered for inference. The mean of a GP is often assumed to be 0 everywhere, whilst a range of different covariance functions can be selected from. Changing the properties and parameters of the covariance function determines the variation of the process, and learning in a GP involves finding suitable properties in a covariance function.

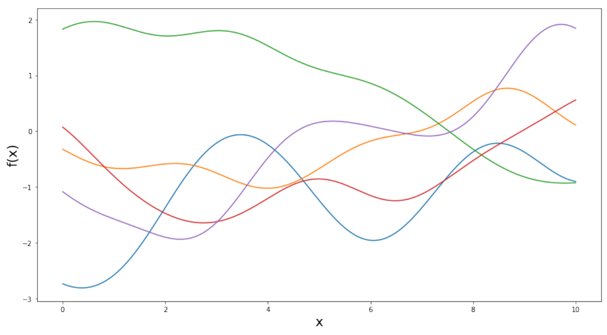

Figure 1: Draws from a GP prior with mean 0 and Gaussian RBF kernel.

GP's have been used extensively in several areas, particularly with data that contains correlation between points for example spatial or temporal data. Two paticular models we will look at later on, LGCP's and MCPM's, make use of GP's.

A specific example of the use of a GP is a task where we want to predict temperature at different points in time. If temperatures were recorded at midday each day for a month, we would have a series of datapoints assumed to be normally distributed. However, if we wanted to use these recordings to predict temperature at different times of day this data would not be useful. Instead, repeating the above experiment each hour of the day would give us 24 different Gaussian distributions. In this scenario a GP can be used to model the correlation between these random vairables, allowing us to make predictions about the temperature at any time of the day, including outside of the times we have taken measurements from.

2.1.2 Useful Properties

GP's with Gaussian likelihoods have several useful properties that makes their use particularly appealing, some of these highlighted nicely in (Duvenaud, 2014):

- **Closed-form.** The predictive distribution of a GP at a set of test points is simply a multivariate Gaussian distribution. This allows GP's to be easily composed with other models.
- **Covariance function.** The covariance function, or kernel, of a GP allows us to express a wide range of modelling assumptions dependent on the class of kernel used and the hyperparameters selected.
- **Inference.** Given a kernel and observations, the predictive posterior distribution can be computed exactly in closed form, a rare property for non-parametric models.
- **Integration over hypothesis.** A GP posterior, given a fixed kernel, lets us integrate over a wide range of hypothesis ensuring overfitting is less of an issue than in comparable model classes.
- **Model selection.** A benefit of being able to integrate over all hypotheses' is that we can compute the marginal likelihood of the data given a model. This allows us to compare different models.
- **Non-parametric.** GP's are non-parametric models as the number of parameters grows with the number of data points, therefore they can model arbitrary functions of the input points.
- **Simplicity.** GP's provide a relatively limited class of models, which can prove restrictive. However, more simple models can be used as well-understood building

blocks to enable the construction of more interesting models. They also lead to relatively straightforward linear algebra implementations.

2.1.3 Gaussian Process Regression

Regression is a supervised learning problem, where we are concerned with predicting continuous quantities. Linear regression is the simplest and most commonly used regression, and is a model where the output is a linear combination of the inputs. The model is simple to implement and easily interpretable, however a drawback is its limited flexibility, especially if the relationship between input and output cannot be reasonably explained by a linear relationship.

GP regression is a non-parametric, Bayesian approach to the regression model, that is becomming increasingly popular in the machine learning community. Non-Bayesian regression involves learning exact values for every parameter, the Bayesian approach infers a probability distribution over all possible values. The Bayesian analysis of the linear regression model is as follows:

$$y = f(x) + \epsilon, f(x) = x^T w, \tag{1}$$

where x is the input vector, w is a vector of weights, f is the function value and y the observed target value.

The Bayesian approach involves specifying a prior distribution, p(w), and updating probabilities based on evidence (observed data) using Bayes' rule. A common choice of prior is a zero-mean Gaussian distribution with covariance matrix Σ_p on the weights. To update the probabilites based on the observed data we use the likelihoood p(y|X,w), the probability density of the observations (data) given the parameters:

$$p(y|X, w) = \prod_{i=1}^{n} p(y_i|x_i, w) = ... = N(X^T w, \sigma_n^2 I). \tag{2}$$

The posterior distribution, p(w|y,X), is the updated distribution that incorporates information from both the prior distribution and the dataset, and is calculated as follows:

$$p(w|y, X) = \frac{p(y|X, w).p(w)}{p(y|X)} \tag{3}$$

To get predictions at unseen data points, we calculate the predictive distribution by weighting all possible predictions by their calculated posterior distribution:

$$p(f^*|x^*, y, X) = \int_w p(f^*|x^*, w)p(w|y, X)dw, \tag{4}$$

where p($f^*|x^*$,y,X) is the predictive distribution.

The prior and the likelihood are assumed to be Gaussian, and this makes the integration analytically tractable. With these assumptions, the predictive distribution must also be Gaussian, so from this we can obtain a point prediction using the mean and variance.

As with the simple linear regression model, Bayesian linear regression also sufferes from limited expressiveness. A simple idea to overcome this problem is to first project the inputs into a higher-dimensional space using a set of basis functions, and then apply a linear model in this space rather than directly to the inputs. If we introduce a function, ϕ(x), that maps our input into an N dimensional feature space. Now our model is:

$$f(x) = \phi(x)^T w. \tag{5}$$

The analysis for this model is analogous to the standard model, except we now substitute X for ϕ(X) everywhere. If an algorithm is defined solely in terms of inner products in input space then it can be lifted into feature space by replacing occurences of those inner products by k(x,x'), often called the kernel trick. This is particularly valuable in situations where it is more convinient to compute the kernel than the feature vectors themselves.

2.1.4 Gaussian Process Classification

Classification is another supervised learning problem, where we aim to predict discrete attributes. We will focus on probabilistic classification, where test predictions take the form of class probabilities, contrasting with methods that provide only a guess at the class label. We will start with the joint probability p(y,x), where y is the class label. Using Bayes' theorem, the joint probability can be decomposed as either p(y)p(x|y) or as p(x)p(y|x), giving rise to two different approaches to classification problems (Rasmussen et al., 2006).

The first is called the generative approach, and models the class-conditional distributions p(x|y) for y = C_1,...,C_C and also the prior probabilities of each class, and then computes the posterior for each class using:

$$p(y|x) = \frac{p(y)p(x|y)}{\sum_{c=1}^{C} p(C_c)p(x|C_c)}. \tag{6}$$

The alternative approach, the discriminative approach, involves modelling p(y|x) directly.

To turn both of these approaches into practical methods we need to create models for either p(x|y) or p(y|x). A simple choice for the generative case is to model class-conditional densities as Gaussians: p(x | C_c) = $N(\mu_c$,$\Sigma_c)$. A Bayesian treatment would involve placing appropriate priors on the mean and covariance of each of the Gaussians. For the binary discriminative case a simple idea is to turn the output of a regression model into a class probability using a 'response' function, which 'squashes' its argument into the range [0,1], guarunteeing a valid probabilistic interpreation. An example is the linear logistic regression model:

$$p(C_1|x) = \lambda(x^T w), where \lambda(z) = \frac{1}{1 + exp(-z)} \tag{7}$$

Perhaps the biggest question in classification is which approach should we prefer, generative or discriminative. The discriminative approach is appealing in that it is directly modelling what we want, p(y|x). Also density estimation of the class-conditional distributions is a hard problem, so if we are just interested in classification then the generative approach may mean we are trying to solve a harder problem than we need to. However, to deal with missing input values, outliers and unlabelled data it is very helpful to have access to p(x), which is obtained by marginalizing out the class label y from the joint distribution. The GP classifiers we will look at are discriminative.

For binary classification the idea behind GP prediction is simple, we place a GP prior over the latent function f(x) and then 'squash' this through the logistic function to obtain a prior on π(x). π is a deterministic function of f, and since f is stochastic, so is π. GP classification is a natural generalisation of the linear logistic regression model. We replace the linear f(x) function by a GP, and the Gaussian prior on the weights by a GP prior.

The latent function f is a nuissance function: we don't observe the values of f itself, only the inputs X and class labels y. We aren't interested in the values of f, instead the values of π. The purpose of f is just to allow a convinient formulation of the model. Inference is divided into two steps: first computing the distribution of the latent variable corresponding to a test case:

$$p(f_*|X, y, x_*) = \int p(f_*|X, y, x_*)p(f|X, y)df, \tag{8}$$

where p(f|X,y) is the posterior over the latent variables. The second step involves using this distribution over the latent f_* to produce a probabilistic prediction:

$$\pi_* = p(y_* = +1|X, y, x_*) = \int \sigma(f_* p(f_*|X, y, x_*)df_*. \tag{9}$$

In the regression case, computing the prediction was straightforward as the integrals were Gaussian and so could be computed analytically. With classification the non-Gaussian likelihood makes the integral analytically intractable. Therefore we need to use either analytical approximations of these integrals, or solutions based on Monte Carlo sampling. We will discuss an approximation approach later.

2.1.5 Model Selection

In practical applications, choosing the covariance function for a GP and all its hyper-parameters can be a difficult task. From (Rasmussen et al., 2006) we know that we only have limited information about the properties of a covariance function, for example choosing the values of free hyper-parameters. For GP's to be useful we must therefore solve the problem of selecting these values.

As we will see in Chapter 4, there are a number of different families of kernels, each with their own properties. The majority of these covariance functions have free hyper-parameters, and their values need to be chosen. Model selection consists of two problems; choosing the covariance function, and choosing its' hyper-parameters. The properties of a covariance function will be determined by the values of its hyperparameters.

It is often the case that the value of the hyper-parameters can be easily interpreted, which can be useful when trying to understand a model. An example, as in (Rasmussen et al., 2006), is the squared exponential covariance function. Here, the length scale, roughly speaking, specifies how large the distance must be in input space for the function values to become uncorrelated.

As we can see, model selection is an open ended problem with no direct end point. This provides an opportunity to learn the covariance function and its hyper-parameters for a model, and this requires a practical approach. (Rasmussen et al., 2016) outlines three general principles of model selection: (1) compute the probability of the model given the data, (2) estimate the generalization error, and (3) bound the generalization error. Generalization error is the average error on unseen test examples. We use this as opposed to the training error as the model may be fit the noise in the training data, which would lead to a low training error but poor generalization error.

2.1.6 Limitations

Several problems exist with GP's, highlighted in (Duvenaud, 2014), and these can cause problems in practical applications:

- **Light tails.** In a GP a Gaussian likelihood can be chosen, which is a distribution known to have light tails. This may not be suitable for our data so a non-Gaussian likelihood should be used instead.
- **Non-Gaussian Likelihood.** Non-Gaussian likelihoods can make a model more robust to outliers. However, their selection requires the use of approximate inference.
- **Slow inference.** The computation of inverting matrices in a GP has an O(N^3) computation time, which makes exact inference extremely slow when working with large datasets. However, approximate inference schemes have been developed to improve this.

2.2 Variational Inference

2.2.1 Definition

A core problem of Bayesian statistics is how to approximate difficult to compute probability densities (Blei et al., 2018). Modern Bayesian statistics relies on models for which the posterior is not easy to compute and corresponding algorithms for approximating them. VI is a method originating from machine learning used for approximating probability densities. VI is widely used to approximate posterior densities and is seen as an alternative strategy to MCMC sampling.

The Kullback-Leibler (KL) divergence is a key statistic used in the VI algorithm. We first introduce the KL divergence below, before introducing VI itself in more detail along with some of its extensions.

2.2.2 Kullback-Leibler Divergence

KL divergence is a measure of how similar or dissilmilar two probability distributions are. It was first introduced in 1951 by *Kullback, S and Leibler, R.* From (MacKay, 2003), the KL divergence between two probability distributions P(x) and Q(x) defined over the same alphabet is:

$$D_{KL}(P||Q) = \sum_{x} p(x) \log(\frac{p(x)}{q(x)})dx \tag{10}$$

The divergence is very similar to entropy, and can be described as a measure of the information lost when an approximation distribution is used to approximate the posterior distribution. More specifically, the expected number of extra bits required to code a sample from the posterior when using a code based on the approximation distribution.

2.2.3 Description & Explanation

As highlighted above, some properties of GP's require that we utilise VI for the model to be practical and effective. In the past, MCMC has been one of the dominant approaches in the field of approximate inference. However, as detailed in (Blei et al., 2018), VI is now an alternative strategy to MCMC. VI tends to be faster and easier to scale to large data than MCMC, but has so far been studied less rigorously than MCMC so its properties are less well understood.

The idea behind MCMC is to first construct an ergodic Markov chain on $\mathbf{z}$, the prior density, that has a stationary distribution equivalent to the posterior $\mathbf{p(z||x)}$. Sampling from the chain gives us samples from the stationary distribution. The posterior can therefore be approximated from an emperical estimate from the samples. There are, however, problems for which MCMC sampling is not appropriate, e.g when working with large datasets or complex models.

The main idea behind VI is optimization. First, we posit a family of approximate densities. We then try to find the member of that family of densities that minimizes the KL divergence to the exact posterior. The final step is to approximate the posterior distribution with the optimized member of the family. Therefore, VI turns inference into an optimization problem. The composition of the approximate family determines the complexity of the optimization problem that we must solve. The family must be flexible, or broad enough, so that the optimization will ensure the resulting approximation is close enough to the posterior, but also simple enough to ensure the optimization will be efficient (Blei et al., 2018).

The choice of method, MCMC or VI, depends on the tasks at hand and the properties needed. MCMC tends to be more computationally intensive than VI, however it does provide a guarantee of producing exact sample from the target density. VI tends to be faster, but it can't produce exact samples from the target, only a density close to the target. These properties suggest that VI is particularly suited to large datasets, or scenarios where we want to quickly explore several models. MCMC is suitable when working with smaller datasets, or situations where we are happy to have a bigger computational cost in exchange for more precise samples (Blei et al., 2018).

The relative accuracy of the two is still unknown. We know that VI generally underestimates the variance of the posterior as a consequence of the objective function, but depending on the task underestimating the variance may be acceptable. These techniques can also be applied more generally to computation of intractable densities, not just in Bayesian settings.

The goal of VI is to approximate the conditional density of latent variables given observed variables. Optimization allows us to find the member of the approximate family that is closest in KL divergence to the target distribution (Blei et al., 2018). Inference is now the problem of minimising the KL divergence:

$$q^*(x) = argmin_{q(x)\epsilon\Omega} KL(q(z)||p(z|x)) \tag{11}$$

This objective, $q^*(\mathbf{.})$, is not computable as it requires computing $\mathbf{log(p(x))}$, which can be hard to compute and is the reason we use approximate inference in the first place, (Blei et al., 2018). Recall that KL divergence is:

$$KL(q(z)|p(z\|x)) = E[log(q(z))] - E[log(p(z\|x))] \tag{12}$$

where expectations are taken in respect to q(z). We then expand the conditional to get:

$$KL(q(z)|p(z\|x)) = E[log(q(z))] - E[log(p(z,x))] + log(p(x)). \tag{13}$$

This reveals the dependence on log(p(x)). As we cant compute the KL, we instead optimize an alternative objective that is equivalent to KL divergence up to an added constant:

$$ELBO(q) = E[log(p(z,x))] – E[log(q(z))] \tag{14}$$

This is known as the Evidence Lower Bound (ELBO). It is equivalent to the negative KL divergence plus log(p(x)), a constant with respect to q(z). Maximising the ELBO is equivalent to minimising the KL divergence. We can rewrite the ELBO as a sum of the expected log likelihood of the data and the KL divergence between the prior q(z) and p(z):

$$ELBO(q) = E[log(p(x|z))] - KL(q(z)\|p(z)). \tag{15}$$

The ELBO is the variational objective function of Equation (2). We will now briefly describe an example of a variational family, the mean-field variational family, to allow us to complete the specification of the optimization problem. As mentioned above, the complexity of the variational family determines the complexity of the optimization problem. The mean-field variational family is a family where the latent variables are mutually independent and each is governed by a distinct factor in the variational density. A generic member of this family is as follows (Blei et al., 2018):

$$q(z) = \prod_{j=1}^{m} q_j(z_j) \tag{16}$$

Each latent variable z_j is governed by its own variational factor, the density $q_j(z_j)$. In optimization, these variational factors are chosen to maximize the ELBO equation. Each variational factor can take on any parametric form, e.g. a continuous variable may have a Gaussian factor and a categorical variable will have a categorical factor. The mean-field family is expressive because it can capture any marginal density of the latent variables. However, it can not capture correlation between them, suggesting some of the intuitions and limitations of the family.

Co-ordinate ascent mean-field variational inference (CAVI) is a common algorithm used for solving the optimization problem above. CAVI iteratively optimizes each factor of the mean-field variational density, while holding the other fixed. It moves up the ELBO to a local optimum (Blei et al., 2018). We will not cover the CAVI algorithm in depth here.

In comparison to competing methods, the main advantage of VI relates to its speed and scalability and mentioned above. However, many variational approximations can suffer from slow convergence, see (King et al., 2006). The main disadvantage of using this method surrounds the fact that we are approximating the posterior rather than sampling from it directly. VI also doesn't guarantee that we will find the optimum member of the approximation family that we propose.

2.2.4 Scalable Variational Inference

The cubic scaling on the number of observations of GP models as discussed earlier has become a problem for their use in a wide variety of applications. Solutions so far have included selecting informative datapoints to facilitate sparse approximations to the posterior, to considering these inducing points as continuous parameters and optimizing them within a coherent probabilistic framework. (Bonilla et al., 2018) introduces a scalable inference approach that we will cover below.

Scalable inference begins with an inital input of N x D dimension, and involves reducing N by selecting a subset, M, of the N points and performing inference on this subset. There are a number of methods to select this subset, including: random sampling, using a nearest neighbours approach, or utilising an optimisation method. We then perform inference on these M inducing variables for each latent process. We are now performing inference on a M x M matrix, instead of a N x N matrix, with M < N. This produces an algorithm with time complexity O(M^3) (Bonilla et al., 2018).

Our prior must be redefined in terms of some auxiliary variables, know as inducing variables, in order to make the inference scalable. These inducing variables are drawn from the same zero-mean GP priors. Most GP approximations can be formulated using this augmented prior, with additional conditions on the training and test conditional distributions. We refer to these approaches as sparse approximations. We also refer to models with a large number of inducing inputs as denser models (Bonilla et al., 2018).

The posterior inference given the prior and likelihood is analytically intractable (Bonilla et al., 2018), so we must resort to VI as above. We seek to approximate the joint posterior p(f, u$\|$y) with a variational distribution q(f, u $\|\lambda$). As our true joint posterior, p(f, u$\|$y) = p(f$\|$u, y) p(u$\|$y), we choose an approximate posterior of the form:

$$q(f, u\|\lambda) = p(f\|u)q(u\|\lambda), \tag{17}$$

where p(f$\|$u) is the conditional prior and q(u$\|\lambda$) is our approximate posterior. Hence, we can define our variational distribution using a mixture of Gaussians:

$$q(u|\lambda) = \sum_{k=1}^{K} \pi_k q_k(u|m_k, S_k) = \sum_{k=1}^{K} \pi_k \prod_{j=1}^{Q} N(u_j; m_{kj}, S_{kj}), \tag{18}$$

where $\lambda = \{\pi_k, m_{kj}, S_{kj}\}$ are the variational parameters, π_k the mixture proportions, m the posterior means and S the posterior covariances of the inducing variables of mixture components k and latent function j.

We have already shown that minimizing the KL divergence between our approximate and true posterior is equivalent to maximizing the ELBO. For the KL objective function, (Bonilla et al., 2018) exploits the structure of the variational posterior in order to avoid computing KL-divergences over distributions involving all the data. We obtain a lower bound for this objective function in the general case of q(u) being a mixture of Gaussians shown above. For the expected log-likelihood term we use a MC approach in order to estimate the required expectation and its gradients.

We decompose the KL-divergence term as, $L_{kl}(\lambda) = L_{ent}(\lambda) + L_{cross}(\lambda)$, where:

$$L_{ent}(\lambda) = -E_{q(u|\lambda)}[log(q(u|\lambda))], and L_{cross}(\lambda) = E_{q(u|\lambda)}[log(p(u))]. \tag{19}$$

L_{ent} denotes the differential entropy of the approximating distribution q(u$\|$y), and L_{cross} denotes the negative cross-entropy between the approximating distribution and the prior.

Next, we address the computation of the expected log likelihood (Bonilla et al., 2018). The main difficulty in its computation, unlike the KL-divergence objective where we have full knowledge of the prior and approximate posterior, here we don't assume a specific form for the conditional likelihood. We expand the equation using our definitions of approximate posterior and the factorization of the conditional likelihood:

$$L_{ell}(\lambda) = E_{q(f,u|\lambda)}[log(p(y|f,\phi))] = E_{q(f|\lambda)}[log(p(y|f,\phi))], \tag{20}$$

where the distribution q(f|y) resulting from marginalizing u from this joing posterior can be obtained analytically.

The scalable VI framework presented by (Bonilla et al., 2018) has the potential to be a powerful tool for researchers devising models with GP priors for which VI is not available.

3 Non-Homogenous Poisson Processes

3.1 Definition

Poisson processes are a widely used stochastic counting process (Gallager, 2011). They are used in scenarios where we are counting the occurrences of events that appear to occur at a certain rate, but actually occur completely at random. A Poisson process with rate λ and t $>$ 0 has the following PMF for N(t) (the number of arrivals in time (0, t]):

$$P_{N(t)}(n) = \frac{(\lambda t)^n exp(-\lambda t)}{n!} \tag{21}$$

The Poisson process defined above is characterized by a constant arrival rate λ, and is known as a homogenous Poisson process. It can often be more useful to consider a type of process in which the rate varies as a function of time. A Non-Homogenous Poisson Process is a process of this type with time varying rate, λ(t).

The types of data we will be looking at are characterized by count or point data in the spatio-temporal region. These settings can be modeled as non-homogenous Poisson processes where space-time varying intensity determines event occurences (Aglietti et al., 2019). Below, I will introduce some commonly used modeling approaches for these settings.

3.2 Log Gaussian Cox Processes

Cox processes provide useful and frequently applied models for aggregated spatial point patterns where the aggregation is due to a stochastic environmental heterogeneity (Moller et al., 1998). A Cox process is doubly stochastic as it arises from a non-homogenous Poisson process with a random intensity measure. The intensity measure is often specified by a random intensity function, or intensity process. Log Gaussian Cox processes (LGCP's) are a particular type of Cox process where the logarithm of the intensity function is a GP.

Stationary LGCP's have some appealing characteristics (Moller et al., 1998):

- The distribution is completely characterized by the intensity and the pair correlation function. This makes parametric models easy to interpret.

- Theoretical properties are easily derived. For instance, higher-order properties are simply expressed by the intensity and the pair correlation of the LGCP.
- The underlying GP and intensity surface can be predicted from a realization of the LGCP observed within a bounded window using Bayesian methods.
- There is no problem with edge effects as the distribution of a LGCP restricted to a bounded subset is known.

The LGCP model can also be extended into the multi-dimensional case (Moller et al., 1998). A simple example is the 2-dimensional input case of a Cox process X = (X_1, X_2) with random intensity process $\lambda_j = \{\lambda_j(\text{s}) = \exp(Y_j(\text{s}))\text{: s } \epsilon\ R^2\}$ and j = 1,2. Y = $\{(Y_1(\text{s}),\ Y_2(\text{s}))\text{: } \epsilon\ R^2\}$ is a bivariate stationary GP with mean (μ_1, μ_2) and covariance functions $c_{i,j}(\text{a})$. The conditional on Y, X_1 and X_2 are independent Poisson processes with intensity functions λ_1 and λ_2 respectively. The covariance function matrix of the multivariate GP must be a positive semi-definite. Many of the characteristics and properties of the LGCP model can be extended to the multivariate case as well.

3.3 Multi-Task Cox Process Models

3.3.1 MCPM Model

I now introduce the Multi-task Cox process model (MCPM) detailed in (Aglietti et al., 2018). The MCPM model is a generalization of the LGCP framework that allows us to model multiple correlated point data jointly. The observations are treated as realizations of multiple LGCP's whose log intensities are given by linear combinations of latent functions drawn from GP priors. The combination coefficients are also drawn from GP's and can incorporate additional dependencies.

The issues with the LGCP model predominantly involve the challenging inference due the doubly-stochastic nature and scalability issues with GP models. These computational problems are exacerbated when considering multiple correlated tasks jointly (Aglietti et al., 2018). MCPM involves posterior estimation over both latent and mixing processes using VI to create a method much faster than competing approaches. Existing multivariate LGCP's for point processes have intensities given by deterministic combinations of latent GP's. These approaches fail to propagate uncertainty in the weights of the linear combination, leading to statistical deficiencies that are addressed by MCPM's.

(Aglietti et al., 2019) proposes the following characteristics for the MCPM model:

- **Stochastic mixing weights.** MCPM consider correlated count data as realizations of multiple LGCP's, where the log intensities are linear combinations of latent GP's and the combination coefficients are also GP's.
- **Efficient inference.** Posterior estimation over both latent and mixing processes using VI ensures MCPM is orders of magnitude faster than competing approaches.
- **Closed-form expectations in the variational objective.** The required expectations in the VI objective are expressed in terms of moment generating functions of the log intensities. This completely avoids Monte Carlo estimates.

When detailing the model, we consider a dataset D $= \{(x_n, y_n)\}_{n=1}^{N}$, where $x_n \epsilon R^D$ represents the input and $y_n \epsilon R^P$ gives the event counts for P tasks. MCPM aims to learn the latent intensity functions and to make probabilistic predictions of the event counts. MCPM is characterized by Q latent functions which are all uncorrelated and drawn from Q zero-mean GP's. Therefore, the prior over all the N x Q latent function values is (Aglietti et al., 2018):

$$p(F|\theta_f) = \prod_{q=1}^{Q} p(F_{*q}|\theta_f^q) = \prod_{q=1}^{Q} N(F_{*q}; 0, K_{xx}^q), \tag{22}$$

where θ_q is the set of hyperparameters for the q^{th} latent function and F_{*q} is the values of latent function q for observations x_n.

MCPM models tasks' correlation by linearly combining the above latent functions with a set of stochastic task-specific mixing weights, determining the contribution of each latent function to the overall LGCP intensity. (Aglietti et al., 2018) considers two possible prior distributions; an independent prior and a correlated prior.

We now specify the prior over weights; it is assumed theses weights are drawn from Q zero-mean GP's:

$$p(W|\theta_w) = \prod_{q=1}^{q} p(W_{*q}|\theta_w^q) = \prod_{q=1}^{Q} N(W_{*q}; 0, K_w^q), \tag{23}$$

where W_{*q} represents the P weights for the q^{th} latent function and θ_w denotes the hyperparameters.

The likelihood of events at locations x_{n_p} under P independent non-homgenous Poisson processes each with rate function $\lambda_p(.)$ is:

$$exp[-\sum_{p=1}^{P} \int_{\tau} \lambda_p(x)dx] \prod_{n=1}^{P} \prod_{n_p=1}^{N_p} \lambda_p(x_{n_p}), \tag{24}$$

where τ is the observation domain.

Under MCPM, the likelihood of the observed counts is defined as:

$$p(Y|F,W) = \prod_{n=1}^{N} \prod_{p=1}^{P} Poisson(y_{np}; exp(W_{p*}F_{n*} + \phi_p)), \tag{25}$$

where y_{np} denotes the event counts for the p^{th} task at x_n, W_{*p} represents the Q weights for the p^{th} task, F_{n*} denotes the Q latent function values corresponding to x_n and ϕ_p indicates the task-specific offset to the log-mean of the Poisson process.

Introducing a GP prior poses significant computational challenges during posterior estimation as inference is dominated by O(N^3) computations. To make inference scalable, (Aglietti et al., 2019) follows the inducing-variable approach proposed by Titsias (2009) and further developed by Bonilla (2016). They augment the prior of the latent functions in (13) with M underlying inducing variables for each latent process. We see that major computational gains are realized when M $\leq\leq$ N. Hence, we have prior distributions for the inducing variables and the latent functions are:

$$p(U|\theta) = \prod_{q=1}^{Q} N(U_{*q}; 0, K_{zz}^q), and \tag{26}$$

$$p(F|U,\theta) = \prod_{q=1}^{Q} N(K_{zz}^q(K_{zz}^q)^{-1}U_{*q}, K_{zz}^q - A_q K_{zz}^q), \tag{27}$$

where $A_q = K_{xz}^q \ (K_{zz}^q)^{-1}$, U is the set of all the inducing variables; K_{xx}^q, K_{xz}^q, K_{zx}^q and K_{zz}^q are the covariances induced by evaluating the corresponding covariance function at all pairwise rows of training inputs X and the inducing inputs Z_q; and θ represents the set of hyperparameters for the Q latent functions.

3.3.2 Inference

Inference in MCPM involves estimating the posterior distribution over all latent variables given the data. As the posterior is analytically intractable, (Aglietti et al., 2018) resort to VI to estimate the posterior. The variational distribution is defined as:

$$q(F, U, W|v) = p(F|U) \prod_{q=1}^{Q} N(m_q, S_q) \prod_{q=1}^{Q} N(\omega_q, \Omega_q), \tag{28}$$

where $v_u = \{m_q, S_q\}$ and $v_w = \{\omega_q, \Omega_q\}$ are the variational parameters.

The variational distribution in Equation (19) allowed (Aglietti et al., 2018) to significantly simplify the computation of the KL divergence term, where the terms containing the latent function vanish. This gives $L_{kl}(v) = L^u_{ent}(v_u) + L^u_{cross}(v_u) + L^w_{ent}(v_w)$ $+L^w_{cross}(v_w)$, with each term given by:

$$L^u_{ent}(v_u) = \frac{1}{2}\sum_{q=1}^{Q}[Mlog(2\pi) + log(\|S_q\|) + M], \tag{29}$$

$$L^u_{cross}(v_u) = \sum_{q=1}^{Q}[log(N(m_q; 0, K^q_{zz}) - \frac{1}{2}tr(K^q_{zz})^{-1}S_q], \tag{30}$$

$$L^u_{ent}(v_w) = \frac{1}{2}\sum_{q=1}^{Q}[Plog(2\pi) + log(\|\Omega_q\|) + M], \tag{31}$$

$$L^u_{cross}(v_w) = \sum_{q=1}^{Q}[log(N(\omega_q; 0, K^q_w) - \frac{1}{2}tr(K^q_w)^{-1}\Omega_q]. \tag{32}$$

The MCPM framework in (Aglietti et al., 2018) also allows the computation of moments of the intensity function in a closed form, however we will not cover that in detail here.

The time complexity of operations in the MCPM algorithm are dominated by algebraic operations on K^q_{zz} which are O(M^3), while the space complexity is dominated by storing K^q_{zz}, which is O(M^2) where M is the number of inducing variables per latent process. L_{ent} and L_{cross} only depend on distributions over M dimensional variables, so their computational complexity is independent of N. L_{ell} decomposes as a sum of expectations over inidividual data points so stochastic optimization techniques can be used to evaluate this term making it independent of N. Finally, the algorithm complexity is independent on the number of tasks, P, and only dependent on Q making it scalable to multi-task datasets.

4 Kernels

4.1 Definition

The covariance function or kernel is the crucial ingredient in a GP predictor as it encodes our assumptions about the functions that we wish to learn (Rasmussen et al., 2006). Kernels allow us to map data into a high dimensional feature space in order to increase the computational power of linear machines. A GP is just one example of many algorithms that make use of kernels, and in a GP they are used to define the similarity between data points.

In the MCPM framework covered above, the matrices K in Equation (18) are the covariances induced by evaluating the covariance functions on the training inputs and the inducing inputs. The covariance functions produce a similarity value for each pair of inputs, with the value dependent on the covariance function chosen. In the next section we discuss some properties of covariance functions, or kernels.

4.2 Properties

A kernel is a function of two arguments that maps a pair of inputs, x and x'. Kernels have the form:

$$K(x, x') =< \phi(x), \phi(x') >, \tag{33}$$

where ϕ is a (usually nonlinear) mapping from the input space to feature space, and $<, >$ is an inner product (Genton, 2001).

Kernels are said to be symmetric if K(x,x') = K(x',x) (Rasmussen et al., 2006), and clearly from the symmetry of the inner product this condition is satisfied. They must also satisfy the Cauchy-Schwartz inequality: K^2(x,x') $\leq$ K(x,x) K(x',x') (Rasmussen et al., 2006). However, these conditions aren't sufficient for a kernel, it must also be a positive semidefinite (PSD). A symmetric matrix is PSD if and only if all of its eigenvalues are non-negative. Therefore, for any set of examples x_1, ..., x_n and any set of real numbers λ_1, ..., λ_n, a kernel K must satisfy (Genton, 2001):

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j K(x_i, x_j) \geq 0. \tag{34}$$

Given a set of inputs, x_1, ..., x_n, we can compute the Gram matrix K whose entries are K_{ij} = k(x_i, x_j). If k is a covariance function then K is known as the covariance matrix (Rasmussen et al., 2006).

A stationary kernel is one which is invariant to translations in the input space (Rasmussen et al., 2006):

$$K(x, x') = K(x - x'), \tag{35}$$

that is, it depends only on the difference between x and x', not on the values of x and x' themselves. A special case of a stationary kernel is when the kernel depends only on the norm of the difference between x and x'. In this case the kernel is said to be isotropic and is only a function of the distance:

$$K(x, x') = K(\|x - x'\|). \tag{36}$$

A useful property of stationary kernels is that they can be constructed from their spectral representation derived by (Bochner, 1955). He proved that a stationary kernel K(x,x') is positive definite in R^d if and only if it has the form:

$$K(x, x') = \int_{R^d} cos(w^T(x - x'))F(dw), \tag{37}$$

where F is a positive finite measure. The quantity F/K(0) is called the spectral distribution function. The equation above is simply the Fourier transform of F. Below we state Bochner's theorem.

Bochner's theorem: A complex-valued function k on R^D is the covariance function of a weakly stationary mean square continuous complex-valued random process of R^D if and only if it can be represented as:

$$k(\tau) = \int_{R^D} e^{2\pi i s \tau} d\mu(s), \tag{38}$$

where μ is a positive finite measure.

If μ has a density S(s), then S is known as the spectral density corresponding to k. When the spectral density exists, the covariance function and the spectral density are

Fourier duals of each other. This is known as the Wiener-Khintchine theorem, and is shown below (Rasmussen et al., 2006):

$$k(\tau) = \int S(s)e^{2\pi i s\tau} ds, \quad S(s) = \int k(\tau)e^{2\pi i s\tau} d\tau. \tag{39}$$

S(s) is the amount of power allocated on average to eigenfunction $e^{2\pi i s\tau}$ with frequency s. S(s) must eventually decay sufficiently fast as $|s| \to \infty$ so that it is integrable; the rate of decay of the power spectrum gives important information about the smoothness of the associated stochastic process (Rasmussen et al., 2006).

We will now cover examples of some of the most common families of both stationary and non-stationary kernels.

4.3 Kernel Function Examples

4.3.1 Stationary Kernels

The Radial Basis Function (RBF), or Squared Exponential kernel, is a stationary isotropic kernel with the following function (Duvenaud, 2014):

$$K(x, x') = \sigma^2 exp(\frac{-(x - x')^2}{2l^2}), \tag{40}$$

where σ^2 is the variance and l is the lengthscale. It is also known as the Gaussian kernel.

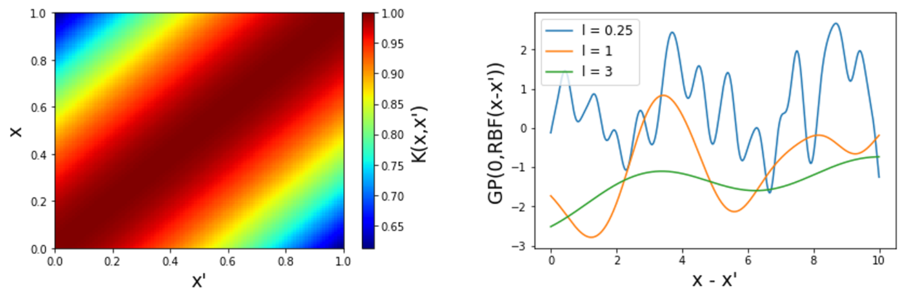

Figure 2: RBF Kernel
Left: a heatmap of the RBF function. Right: draws from a GP with RBF kernel with differing lengthscales.

The lengthscale parameter determines how smooth the function is; the larger the value the smoother the function is as we can see from the plot above. The variance determines the distance of the function from the mean, it is essentially a scale factor (Duvenaud, 2014).

The Rational Quadratic kernel is equavalent to adding together many Squared Exponential kernels with differing lengthscales (Duvenaud, 2014). GP priors with this kernel expect to see functions that vary smoothly across many lengthscales. The function is:

$$K(x, x') = \sigma^2 (1 + \frac{(x - x')^2}{2\alpha l^2})^{-\alpha}, \tag{41}$$

where σ^2 is the variance, l is the lengthscale and α is a parameter that determines the weighting of large-scale and small-scale variations (Duvenaud, 2014). The variance and lengthscale parameters have a similar effect as the RBF kernel. When $\alpha \to \infty$, the kernel is identical to the RBF function.

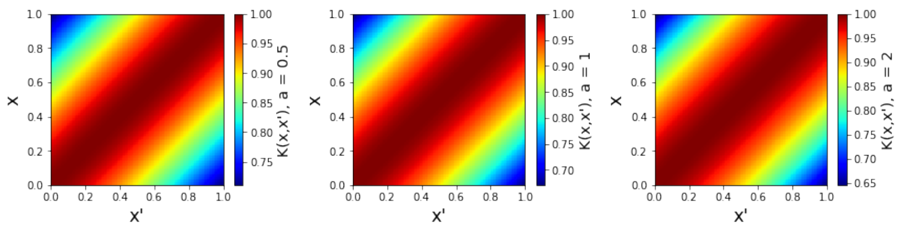

Figure 3: Rational Quadratic Kernel
A heatmap of the Rational Quadratic function with different values of parameter α.

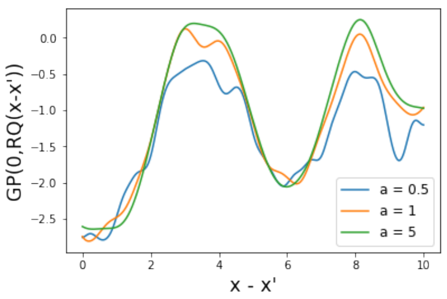

Figure 4: Rational Quadratic Kernel
Draws from a GP with Rational Quadratic kernel with differing values of α.

One of the benefits of using a RBF or Rational Quadratic kernel is that they usually work well for smooth functions when N is a multiple of D. If the function has a discontinuity at any point, then either the lengthscale will become extremely short and the posterior mean will become zero almost everywhere, or the posterior mean will have 'ringing' effects (Duvenaud, 2014). Even if the function has just a small non-smooth region, the lengthscale can be determined by this region, failing to extrapolate to smooth areas. In this scenario other kernels may be appropriate.

The next class of kernels we look at are the Matérn class of kernels. They are also isotropic stationary:

$$K(x, x') = \sigma^2 \frac{2^{1-v}}{\Gamma(v)} (\frac{\sqrt{2vr}}{l})^v K_v(\frac{\sqrt{2vr}}{l}), \tag{42}$$

where σ is the variance, l the length parameter, v a positive parameter, and K_v is a modified Bessel function (Rasmussen et al., 2006).

Matérn covariance functions simplify nicely when v is a half-integer, with the most interesting cases being v = $\frac{3}{2}$ and v = $\frac{5}{2}$, known as the Matérn-$\frac{3}{2}$ and the Matérn-$\frac{5}{2}$ kernel respectively. Another interesting case is when v = $\frac{1}{2}$, as the Matérn kernel reduces to the Exponential function (Rasmussen et al., 2006):

$$v = \frac{3}{2}: \quad K(x, x') = \sigma^2(1 + \frac{\sqrt{3}(x - x')}{l})\ exp(-\frac{\sqrt{3}(x - x')}{l}), \tag{43}$$

$$v = \frac{5}{2}: \quad K(x, x') = \sigma^2(1 + \frac{\sqrt{5}(x - x')}{l} + \frac{5(x - x')^2}{3l^2})\ exp(-\frac{\sqrt{5}(x - x')}{l}),\ and \tag{44}$$

$$v = \frac{1}{2}: \quad K(x, x') = \sigma^2 exp(-\frac{x - x'}{l}), \tag{45}$$

where σ^2 is the variance and l is the length parameter.

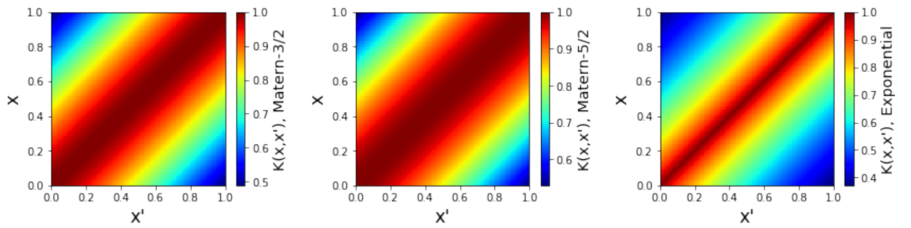

Figure 5: Matérn Kernel
Left: a heatmap of the Matén-$\frac{3}{2}$ function. Middle: a heatmap of the Matén-$\frac{5}{2}$ function. Right: a heatmap of the Exponential function.

The Matérn class of kernels have proved popular recently due to their ability to control the smoothness of the function with the value of parameter v. The larger the value of v the smoother the function is, however the three cases of the family highlighted above are the only ones regularly used due to the significantly higher computation costs of other members of this kernel family. The plot below highlights the difference in smoothness between the kernels.

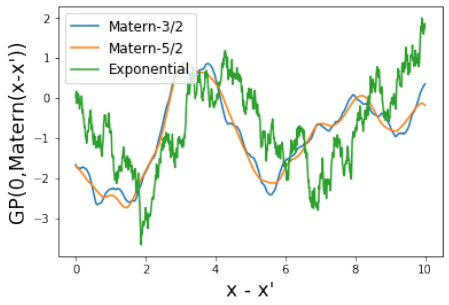

Figure 6: Matérn Kernel
Draws from a GP with different classes of Matérn kernel.

The different families of kernels presented above are all predominantly used in the spatial setting, calculating the similarity between different spatial data points. I will now introduce a couple of kernels often applied in the temporal dimension. The first we will present is the Periodic kernel:

$$K(x, x') = \sigma^2 exp(-(\frac{2sin^2(\pi(x - x')/p)}{l^2})), \tag{46}$$

where σ^2 is the variance, p gives the period of the function, and l is a length parameter.

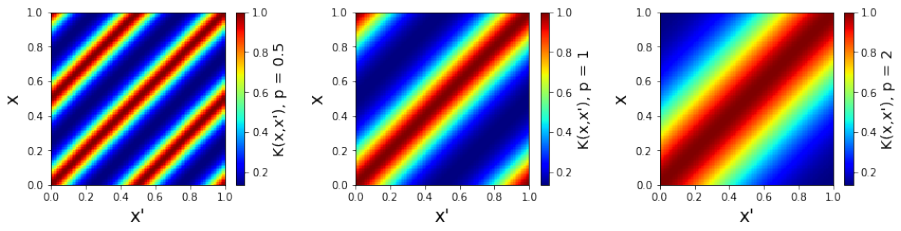

Figure 7: Periodic Kernel
Heatmaps of the Periodic function with different period values.

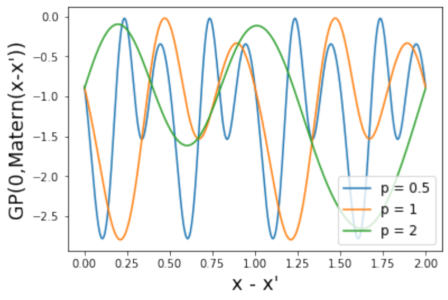

Figure 8: Periodic Kernel
Draws from a GP with different period values.

The period determines the distance between each period of the function, and the length-scale and variance have a similar effect to the kernels above (Duvenaud, 2014). From the plots above we can see visually how the different values of the period parameter affect the kernel, and it is therefore quite intuitive as to what value to set this parameter do if we have some knowledge about the temporal data.

The periodic kernel is very repetitive. In order to add some flexibility to a model, we could add or multiply one of the spatial kernels introduced above with the Periodic kernel. This allows the shape of the repeating part of the function to change over time (Duvenaud, 2014).

We now examine the Spectral Mixture (SM) kernel, a powerful class of kernels formed by linearly combining spectral mixture components:

$$K(x,x') = \sum_{q=1}^{Q} w_q exp(-\frac{1}{2}(x-x')^2\Sigma_q\tau)cos(\mu_q^T(x-x')), \tag{47}$$

where Q is the number of latent functions, P is the dimension of the data, $\mu_q = (\mu_q^{(1)}$, ...,$\mu_q^{(P)})$ is the mean vector, Σ_q = diag($\sigma_1^{(q)}$, ..., $\sigma_n^{(q)}$) is the covariance matrix and w_q is the weight vector that specifies the relative contribution of each mixture component (Parra et al., 2017).

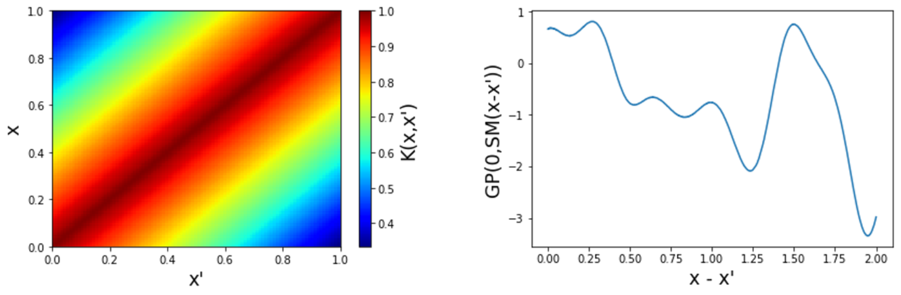

Figure 9: Spectral Mixture Kernel
Left: a heatmap of the SM function. Right: draws from a GP with SM kernel.

The SM kernel is derived by modelling a spectral density, the Fourier transform of a kernel, with Gaussian mixtures. With enough components, the SM kernel can approximate any other stationary kernel. The kernel is also able to learn different periodicities.

4.3.2 Non-Stationary Kernels

Stationarity is an important property of kernels, and implies that the covariance function is only a function of the distance between two points rather than the value of the points. Stationary kernels are unable to encode input-dependent function dynamics or long-range correlations in input space (Remes et al., 2017). I now review a class of non-stationary spatial covariance functions, as well as a non-stationary temporal kernel.

(Paciorek, 2003) and (Paciorek et al., 2004) extend the non-stationary approach of Higdon (1999), described as the HSK covariance, to provide a closed form correlation function. They give a non-stationary Gaussian covariance function from which a class of non-stationary kernels can be derived. The non-stationary Gaussian covariance function has the following form:

$$Q_{ij} = (x_i - x_j)^T (\frac{(\Sigma_i + \Sigma_j)}{2})^{-1} (x_i - x_j), \tag{48}$$

where Σ_i, the kernel matrix, is the covariance matrix of the Gaussian kernel at input x_i (Paciorek et al., 2004).

From the non-stationary covariance above, (Paciorek et al., 2004) introduces a thoerem from which a class of non-stationary kernels can be built.
Theorem: Let Q_{ij} be defined as above. If a stationary correlation function, $R^S(x_i, x_j)$,

is a positive definite on R^P for every p = 1,2,..., then:

$$R^{NS}(x_i, x_j) = |\Sigma_i|^{\frac{1}{4}}|\Sigma_j|^{\frac{1}{4}}|\frac{(\Sigma_i + \Sigma_j)}{2}|^{-\frac{1}{2}}R^S(\sqrt{Q_{ij}}), \tag{49}$$

is a non-stationary correlation function, that is positive definite on R^P.

From the theorem above, (Paciorek et al., 2004) introduce a non-stationary version of the Gaussian kernel:

$$C^{NS}(x_i, x_j) = \sigma^2|\Sigma_i|^{\frac{1}{4}}|\Sigma_j|^{\frac{1}{4}}|\frac{(\Sigma_i + \Sigma_j)}{2}|^{-\frac{1}{2}}exp(-Q_{ij}). \tag{50}$$

The non-stationary nature of the covariance means that it will change with different input values, not just the difference between values. Kernel matrices, Σ, drop off quickly producing locally short correlation scales.

Another spatial non-stationary kernel constucted in a similar way to above is the non-stationary Rational Quadratic covariance function (Paciorek, 2003):

$$R(x_i, x_j) = \frac{2^{\frac{P}{2}}|\Sigma_i|^{\frac{1}{4}}|\Sigma_j|^{\frac{1}{4}}}{|\Sigma_i + \Sigma_j|^{\frac{1}{2}}}(\frac{1}{1 + Q_{ij}})^v, \tag{51}$$

where v and P are hyper-parameters.

The final class of non-stationary spatial covariance functions I will consider is the non-stationary Matérn class of kernels:

$$R(x_i, x_j) = \frac{2^{\frac{P}{2}}|\Sigma_i|^{\frac{1}{4}}|\Sigma_j|^{\frac{1}{4}}}{|\Sigma_i + \Sigma_j|^{\frac{1}{2}}}\frac{1}{\Gamma(v)2^{v-1}}(\sqrt{2vQ_{ij}})^v K_v(\sqrt{2vQ_{ij}}), \tag{52}$$

with v a hyper-parameter as present in the stationary version, K_v the Bessel function, and Q_{ij} as defined in Equation (48) (Paciorek et al., 2004).

I will implement a particular sub-class of the non-stationary Matérn kernel above, the non-stationary Matérn-$\frac{3}{2}$ kernel:

$$R(x_i, x_j) = \frac{2^{\frac{P}{2}}|\Sigma_i|^{\frac{1}{4}}|\Sigma_j|^{\frac{1}{4}}}{|\Sigma_i + \Sigma_j|^{\frac{1}{2}}}(1 + \frac{\sqrt{3}(Q_{ij})}{l})\ exp(-\frac{\sqrt{3}(Q_{ij})}{l}) \tag{53}$$

Below are plots of the non-stationary covariance functions introduced above between two input points x and x'.

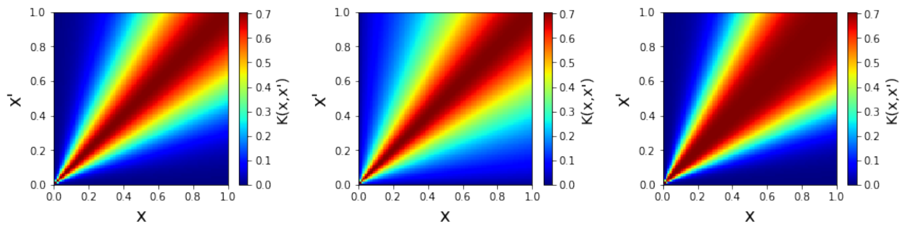

Figure 10: Non-Stationary Spatial Covariance Functions
Left: non-stationary Gaussian kernel. Middle: non-stationary Rational Quadratic kernel. Right: non-stationary Matérn-$\frac{3}{2}$ kernel.

The non-stationary nature of the kernels is obvious from the differing intensities for the values where x and x' are equal, e.g the diagonal from the top left to the bottom right. In the case of a stationary kernel we would expect all values where x and x' are identical to have the same intensity.

The non-stationary temporal kernel I will cover is the Generalized Spectral Mixture (GSM) kernel. It is a generalisation of the SM kernel into the non-stationary case, introduced by (Remes et al., 2017). The non-stationary GSM kernel has the following function:

$$K(x,x') = \sum_{i=1}^{Q} w_i(x)w_i(x')k_{gibbs,i}(x,x')cos(2\pi(\mu_i(x)x - \mu_i(x')x')),\ where \tag{54}$$

$$k_{Gibbs,i}(x,x') = \sqrt{\frac{2l_i(x)l_i(x')}{l_i(x)^2 + l_i(x')^2}}exp(-\frac{(x-x')^2}{l_i(x)^2 + l_i(x')^2}). \tag{55}$$

In this kernel the mixture weights, lengthscales and frequencies are parameterised as GP's:

$$log(w_i(x)) \sim GP(0, k_w(x,x')), \tag{56}$$

$$log(l_i(x)) \sim GP(0, k_l(x,x')), \tag{57}$$

$$logit(\mu_i(x)) \sim GP(0, k_\mu(x,x')), \tag{58}$$

where the log transform is used to insure the weights (w(x)) and lengthscales (l(x)) are non-negative, and the logit transform is used to keep the mean between 0 and the

Nyquist frequency (Remes et al., 2017).

The kernel in Equation (43) is a product of three PSD terms. It encodes the similarity between two data points based on their combined signal variance (w(x)w(x')), and the frequency surface based on the frequencies $(\mu(x), \mu(x'))$ and frequency lengthscales (l(x)l(x')) associated with both inputs. The kernel reduces to the stationary SM kernel with constant functions; w_i(x) $= w_i$, μ_i(x) $= \mu_i$ and l_i(x) $= 1/(2\pi\sigma_i)$.

5 Spatio-Temporal Inference

So far, the MCPM model has been used in either the spatial or temporal dimension. I now look to extend this by combining the dimensions to perform spatio-temporal inference. In order to do this, I will implement the MCPM model with a two dimensional kernel. This kernel will enable us to calculate the correlation between two 2-dimensional observations, (s,t) and (s',t'), where each observation is a fixed space-time location, with s the spatial observation and t the temporal observation.

If we are able to repeatedly draw samples, (s, t), then we assume the sample to be iid. However, the fundamental nature of spatiotemporal data is that observations at nearby locations in space and time are similar, violating the classical assumption of independence. In classical statistics we would obtain multiple iid observations and use these to perform inference. In spatiotemporal statistics a single set of spatiotemporal observations is the only realization we have of the process.

5.1 Separable Kernels

The first class of two dimensional kernels I will cover are separable kernels. They are a multi-dimensional kernel in which the different dimensions are independent of one another. When considering a separable spatio-temporal kernel, the kernel effectively has separate covariance functions for space and time, there is no dependence on each other.

Separable kernels are typically constructed in two ways. I first present a kernel built from multiplying a spatial and temporal kernel, we will refer to it as a Separable Product kernel:

$$k((s,t),(s',t')) = k_s(s,s')k_t(t,t'), \tag{59}$$

where k_s is a spatial kernel, k_t is a temporal kernel, s and s' are spatial observations, and t and t' are temporal observations, as seen in (Flaxman, 2015).

In order to calculate the Gram matrix, K, for this kernel, we have to first calculate the separate Gram matrics K_s and K_t of the constituent spatial and temporal kernels. K is constructed from the Kronecker product of the two smaller Gram matrics: K = K_s $\otimes$ K_t. Utilising the Kronecker product exploits some efficiency gains that are possible when using this kernel (Flaxman, 2015).

The Kronecker product of two 2 x 2 dimensional matrices A and B is calculated as:

$$\begin{bmatrix} a1 & a2 \\ a3 & a4 \end{bmatrix} \otimes \begin{bmatrix} b1 & b2 \\ b3 & b4 \end{bmatrix} = \begin{bmatrix} a1*b1 & a1*b2 & a2*b1 & a2*b2 \\ a1*b3 & a1*b4 & a2*b3 & a2*b4 \\ a3*b1 & a3*b2 & a4*b1 & a4*b2 \\ a3*b3 & a3*b4 & a4*b3 & a4*b4 \end{bmatrix}$$

From the equation above, we can visually see the computational savings made through the use of the Kronecker product. This method uses operations on the smaller K_s and K_t matrics, before constucting the full n x n matrix, instead of relying on computations of the full n x n matrix (Flaxman, 2015).

The second type of separable kernel I consider is built from the sum of a spatial and temporal kernel. It is referred to as the Separable Sum kernel:

$$k((s,t),(s',t')) = k_s(s,s') + k_t(t,t'), \tag{60}$$

where k_s is a spatial kernel, k_t is a temporal kernel, s and s' are spatial observations, and t and t' are temporal observations.

Similar to the Separable Product above, the Gram matrix for the Separable Sum kernel is constructed by calculating the Kronecker sum of the two smaller Gram matrices: K $= K_s \oplus K_t$. The Kronecker product of two 2 x 2 dimensional matrics A and B is:

$$\begin{bmatrix} a1 & a2 \\ a3 & a4 \end{bmatrix} \oplus \begin{bmatrix} b1 & b2 \\ b3 & b4 \end{bmatrix} = \begin{bmatrix} a1+b1 & a1+b2 & a2+b1 & a2+b2 \\ a1+b3 & a1+b4 & a2+b3 & a2+b4 \\ a3+b1 & a3+b2 & a4+b1 & a4+b2 \\ a3+b3 & a3+b4 & a4+b3 & a4+b4 \end{bmatrix}$$

I will now present some examples of both stationary and non-stationary kernels below.

5.1.1 Stationary Separable Kernels

In order for a separable kernel to be stationary, both of the constituent kernels must also be stationary. In the spatial dimension, I consider the RBF and Exponential. I also looked at two temporal kernels: Periodic and Spectral Mixture. This gives a large combination of separable kernels that can be constructed, with an option of whether to use a Separable Product or Separable Sum kernel, each of these with very different properties.

I plotted the covariance functions of different stationary separable kernels, and these plots are shown below:

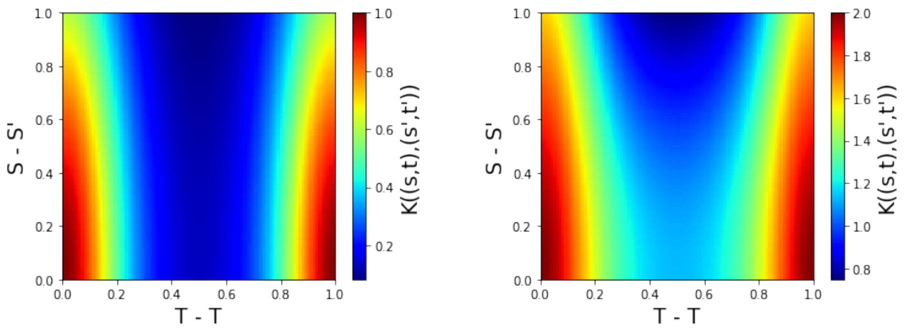

Figure 11: RBF & Periodic Separable Function
Left: Separable Product. Right: Separable Sum.

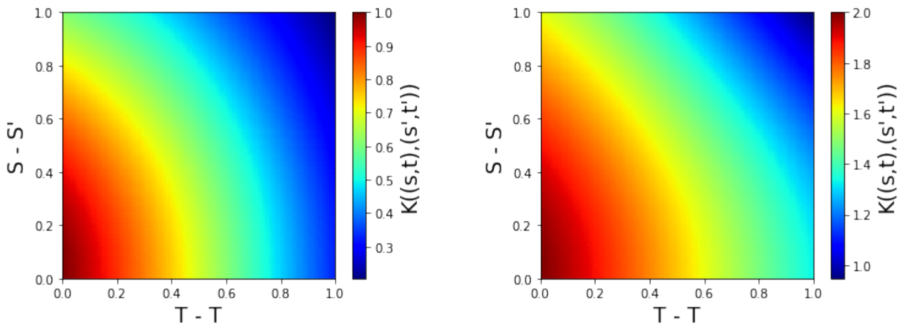

Figure 12: RBF & Spectral Mixture Separable Function
Left: Separable Product. Right: Separable Sum.

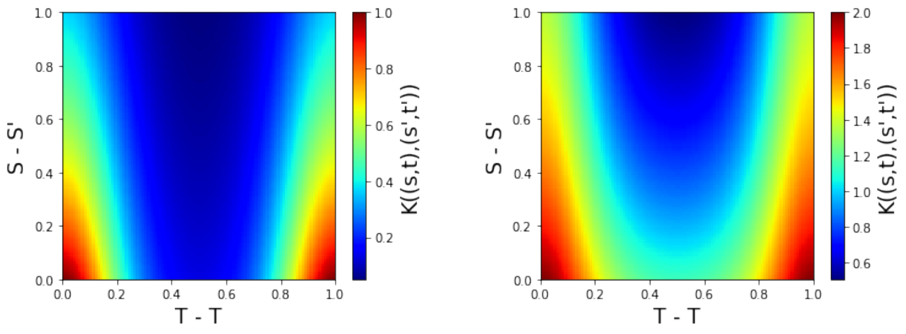

Figure 13: Exponential & Periodic Separable Function
Left: Separable Product. Right: Separable Sum.

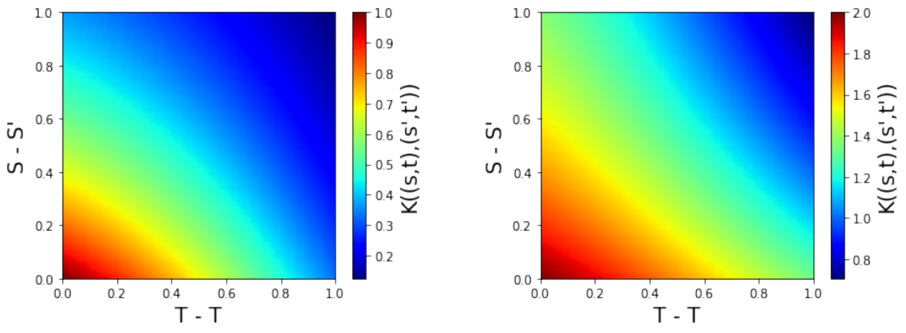

Figure 14: Exponential & Spectral Mixture Separable Function
Left: Separable Product. Right: Separable Sum.

From the plots, we can see that the change in spatial covariance functions only has a small effect on the overall kernel. We can see that the gradient of separable functions with Exponential spatial covariance is steeper than RBF spatial covariance. In contrast, there is a large difference in the overall covariance function depending on what temporal covariance is chosen. When a Periodic kernel is used, as expected, a periodic pattern appears, as opposed to a SM kernel that slowly decreases through time.

The most interesting observation of our plots are differences between the separble product and the separable sum kernel. The product kernel has a slightly steeper gradient compared to the sum kernel, particularly with values where the product is close to 0. Another interesting point to note is that the separable sum kernel only has values close to 0 when values of both the constituent kernels are close to zero. In the case of the product, only one of the constituent kernels must have a value close to 0 for the overall separble kernel to be very small.

I now plot a draw from a GP with each of the separable kernels plotted above, and plot this draw as a heatmap:

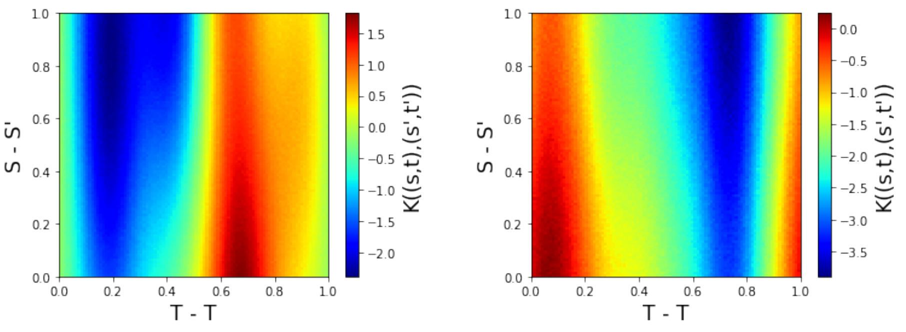

Figure 15: GP with separable RBF and Periodic kernels.
Left: separable product. Right: separable sum.

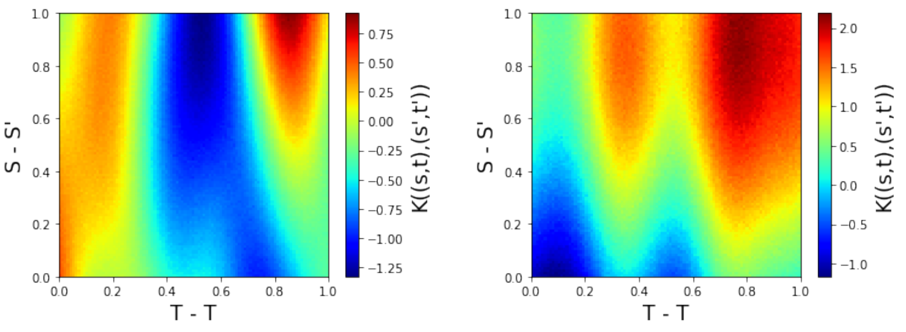

Figure 16: GP with separable RBF and SM kernels.
Left: separable product. Right: separable sum.

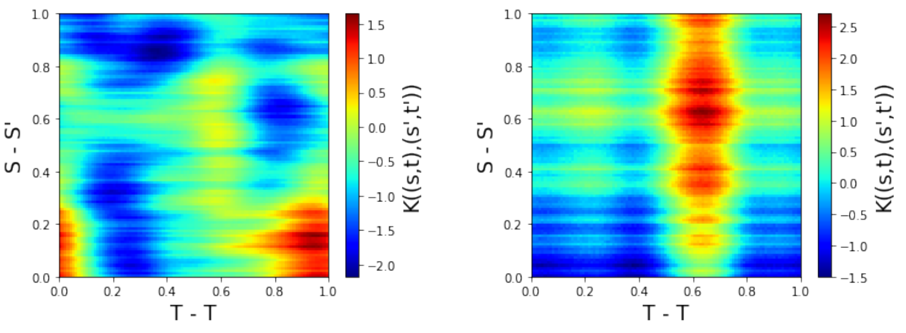

Figure 17: GP with separable Exponential and Periodic kernels.
Left: separable product. Right: separable sum.

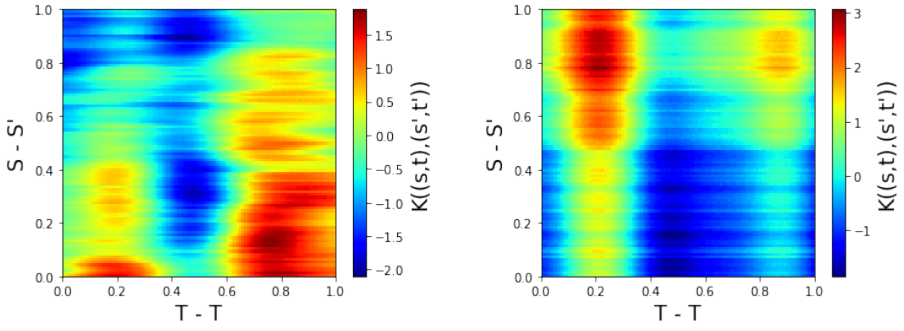

Figure 18: GP with separable Exponential and SM kernels.
Left: separable product. Right: separable sum.

The plots above provide some interesting observations about the draws from a GP with the different separable kernels. The spatial kernel chosen can be seen to have a large effect on the smoothness of the function, with the Exponential function causing the roughest Gaussina processes and RBF function the smoothest.

In the case of the separable product kernels, the choice of temporal kernel, Periodic or SM, appeaers to have only a small effect on shape of the GP. This is particulary interesting due to the obvious difference between the two functions and draws from a GP with each as a kernel. Interestingly, the GP with separable sum kernels for each of the spatial covariances show vast differences depending on the class of temporal kernel used. The reason for this difference could be to do with the fact that a large proportion of the function for the separable product covariances has a value close to 0, so the difference in temporal kernel isn't so large in this scenario.

The contrast between the GP draws with separable product and sum kernels formed of identical constituent kernels is quite large, suggesting that the way the kernel is constructed makes a big difference to the final model.

5.1.2 Non-Stationary Separable Kernels

Above we have looked at several separable kernels, all that are stationary. I will now look to extend this to consider non-stationary separable kernels. For a serparable kernel to be stationary, the constituent kernels must both be stationary. In the non-stationary case, we can either consider just one of the constituent kernels to be non-stationary, or both non-stationary. Both of these scenarios will generate a non-stationary separable kernel.

The non-stationary separable kernel I implement in the experiments below is constructed from a non-stationary Matérn-$\frac{3}{2}$ spatial covariance, along with a either a Periodic or Spectral Mixture temporal covariance. I now plot a sample from a GP with the covariance taken from the non-stationary separable kernels discussed:

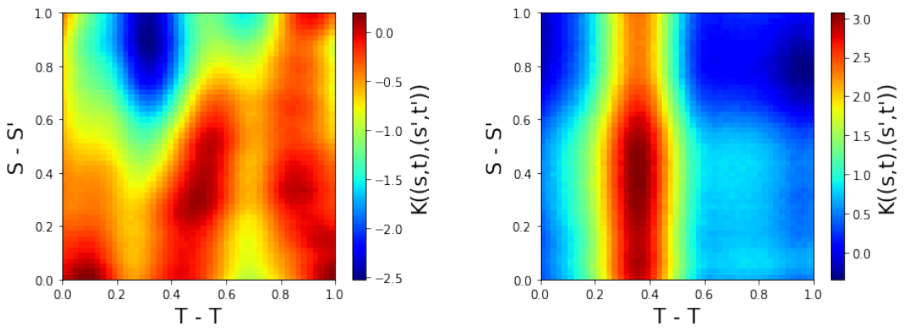

Figure 19: GP Draw with Non-Stationary Matérn-$\frac{3}{2}$ & Periodic Separable Kernels
Left: separable product. Right: separable sum.

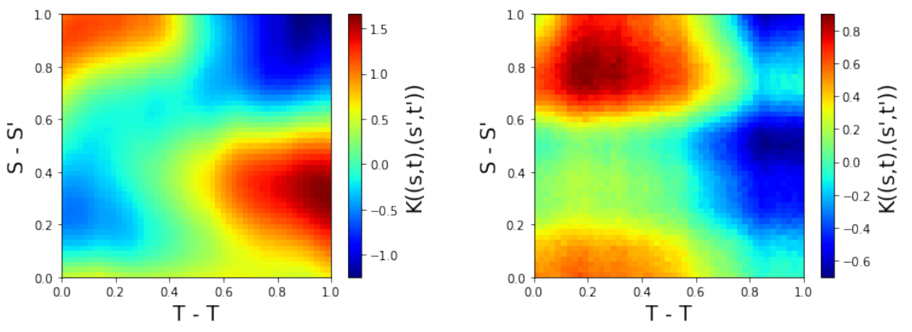

Figure 20: GP Draw with Non-Stationary Matérn-$\frac{3}{2}$ & Spectral Mixture Separable Kernels
Left: separable product. Right: separable sum.

The plots above show the effect a non-stationary constituent kernel has on the overall separable kernel. The non-stationary GP draws appear a lot rougher than their stationary counterparts.

5.2 Non-Separable Kernels

In the section above we reviewed separable space-time kernels, where we assume an independence between space and time. However, whilst this produces attractive results it is not always convenient to assume independence in practice. Separability tells us that for two different points in time, the correlation structure between the same two spatial locations will be identical. The same is true for the correlation between two temporal locations and two different points in space. This assumption is likely to be voilated by real data (Flaxman, 2015), and this provides a question of whether we should use this assumption or not.

I will now review some methods to measure the separability of data, before introducing some examples of non-separable kernels used to model to correlation between data when this assumption does not apply.

5.2.1 Measure of Separability

I will first review the Knox and Mantel tests for separability, before exploring the Kernel Space-Time test introduced by (Flaxman, 2015).

The Knox test (Knox et al., 1964) is a space-time interaction test. It involves first picking threshold distances for 'near in space' and 'near in time', s_0 and t_0 respectively, given data P. We consider every pair of distinct points s, s' ϵ P. Let d_s(p,p') measure the Euclidean distance between p and p': $\sqrt{(x-x')^2+(y-y')^2}$, and d_t(p,p') measure the time interval: |t - t'|. Then the table below can be filled in as follows:

	near in sapce	far in space
near in time	X	n1
far in time	n2	N - (X + n1 + n2)

If there are N = n(n - 1)/2 pairs of points, the test statistic is given by the difference between the number of pairs that we observe to be near in both time and space, X, and the number of pairs that we would expect to be near in both time and space if time and space are independent: N$\frac{X+n1}{N}\frac{X+n2}{N}$. Together this is: X - $\frac{1}{N}$(X + n1)(X + n2). Since the null hypothesis is the there is independence between space and time, we can empirically find the distribution of X under the null by randomly permuting the time lavels and recomputing the test statistic (Flaxman, 2015).

The Knox test is relatively straightforward, however does have some limitations. Particularly, correctly specifying the spatial and temporal ranges can become difficult, and considering a range of different values leads to problems of multiple hypothesis testing (Flaxman, 2015).

I will next state the Mantel test, introduced in (Mantel, 1967), and summarised nicely in (Flaxman, 2015). Given data P, we create an n x n spatial distance matrix D_s with entries given by $d_s(p_i,\ p_j)$ for row i and column j and a n x n temporal distance matrix D_t with entries given by $d_t(p_i,p_j)$. As with the Knox test, we want to test whether space and time, represented by two matrices, are independent.

We flatten the entries above the diagonal of each matrix as a vector of n(n - 1)/2 entries, and calculate the Pearson correlation between these vectors. However, the usual significance test for Pearson's correlation is not valid as the observations are not independent. To derive the null distribution we turn to randomization testing, applying a given permutation to the rows and columns of one of the matrics in order to preserve dependence among the observations.

We are often concerned with short space and time distances, but the Mantel test can be significant due to longer range features. The mantel test is essentially a linear test of dependence, so we expect it to have the same shortcomings as Pearson correlation i.e. zero correlation implies no linear relationship but does not imply independence (Flaxman, 2015).

(Flaxman, 2015) introduces a kernalized version of the Mantel test in order to improve on some of the limitations of the tests described above. The Mantel test measures the correlation between a pair of dissimilarity matrices. Given objects P = $(p_1,\ ...,\ p_n)$ and two kernels k and l, he constructs the Gram matrices K and L and asks, as in the Mantel test, whether the two kernels are measuring independent properties of the objects of P.

Once the Gram matrices are calculated, Flaxman proceeds as in the Mantel test, defining the test statistic T = $\Sigma_{i \neq j}$ k(i,j) l(i,j), and obtaining significance levels by randomization testing. This is called the kernalized Mantel test. With this approach, Flaxman defines a new test for space-time interaction based on kernel embeddings. The most straightforward approach would be to apply Hilbert-Schmidt Independence Criterion (HSIC), a statistical test for the null hypothesis of independence, as a black box test to test whether P and Q are independent. However, computationally this is not attractive as it leads to O(n^4) computations because HSIC considers pairs of observations, and in this case observations are themselves pairs of points.

(Flaxman, 2015) considers a more computationally efficient approach known as the Kernel Space-Time (KST) test. The last term of the HSIC estimator, whcih estimates $||\mu_p\mu_q||^2$, is unchanged by randomization testing, so the key difference is the cross-term, $< \mu_p\mu_q, \mu_{pq} >$. Recalling the definition of the covariance operator, an equivalent definition is:

$$\Sigma_{PQ} = E_{xy}[(\phi(x) - \mu_p) \otimes (\psi(y) - \mu_q)], \tag{61}$$

where ϕ is the feature embedding for H_K, and ψ is the feature embedding for H_L. Therefore we see that the cross-term in $|\Sigma_{PQ}|^2_{HS}$ arises because the feature vectors ϕ(x) and ψ(y) are centered before being multiplied together.

In comparison to the Mantel test, this is the critical difference - the Mantel test measures dependence by calculating the inner product between two matrices as vectors, where these vectors are centered by subtracting the mean of their entries. But this is not equivalent to the centering done by HSIC. This suggests a simple fix for the Mantel test, which can even be applied to the classic version. Given similarity, dissimilarity, or Gram matrices K and L, calculate K' and L' and then apply the Mantel test. Since this is proportional to $\frac{1}{n^2}$ tr(K'L'), Flaxman's KST test takes the same form as HSIC.

In cases where the space and time dimensions are not separable, we utilise non-separable kernels to model this data. Below I introduce some examples of both stationary and non-stationary separable kernels and discuss their different characteristics.

5.2.2 Non-Separable Kernel Examples

We have seen in the section above that separability is a convinient property of spatio-temporal kernels with great computational benefits, however the assumption of separability between space and time is not usually realistic. Often separable models are chosen for their convenience rather than how well they fit the model, as pointed out by (Cressie et al., 1999).

(Fonseca, 2010) and (Gneiting, 2002) both introduce a class of stationary non-separable kernels, building on the work of (Cressie et al., 1999). The main limitation of the approach by (Cressie et al., 1999) is the dependence on Fourier transform pairs in R^d, and (Gneiting, 2002) avoids this limitation to introduce a very general class of space-time covariance functions. (Fonseca, 2010) proposes a class of non-separable covariance models that allow for different degrees of smoothness across space and time.

The stationary non-serparable class of kernels proposed by (Gneiting, 2002) are as follows:

$$C(h,u) = \frac{\sigma^2}{\psi(|u|^2)^{d/2}}\varphi(\frac{\|h\|^2}{\psi(|u|^2)}), \tag{62}$$

where h and u are space and time observations, σ^2 the variance of the process, a and λ smoothing parameters, $\varphi(.)$ a completely monotone function, and $\psi(.)$ a positive function. Some examples that the function φ and ψ can take are detailed in the tables below:

Function	Parameters
φ(t) = exp(-ct^γ)	c > 0, 0 < $\gamma \leq$ 1
φ(t) = ($2^{v-1}\Gamma$(v))$^{-1}$ (c$t^{1/2}$)$^v K_v$(c$t^{1/2}$)	c > 0, v > 0
φ(t) = (1 + ct^γ)$^{-v}$	c > 0, 0 < $\gamma \leq$ 1, v $\leq$ 0
φ(t) = 2^v(exp(c$t^{1/2}$) + exp(-c$t^{1/2}$))$^{-v}$	c > 0, v > 0

Function	Parameters
ψ(t) = (at^α + 1)$^\beta$	a > 0, 0 < $\alpha \leq$ 1, 0 $\leq \beta \leq$ 1
ψ(t) = ln(at^α + b)/ln(b)	a > 0, b > 1, 0 < $\alpha \leq$ 1
ψ(t) = (at^α + b)/(b(at^α + 1))	a > 0, 0 < b $\leq$ 1, 0 < $\alpha \leq$ 1

Though the model detailed above is non-separable, the choice of the functions φ and ψ is associated to the spatial and temporal covariance structures of the data respectively. The correlation model derived from the covariance function depends on a space-time interaction parameter $\beta\epsilon$ [0,1], where β = 0 corresponds to a separable model. As β increases, space-time interaction strengthens, and the spatial correlations at nonzero temporal lags fall off less and less rapidly (Gneiting, 2002).

The class of functions introduce by (Fonseca, 2010) allow for different degrees of smoothness across space and time, and for any given location in space the pure temporal process can have long-range dependence in time.

(Fonseca, 2010) considers the covariance function C(s,t) = $\int C_1$(su)C_2(tv)dμ(u,v), which is obtained by mixing the pure spatial and pure temporal covariance functions and the mix is defined by a bivariate random vector with joint distribution μ(u,v). The fundamental step in constructing the model is how U and V interact with each other, as this generates the interaction between the space and time dimensions.

An important feature in the model specification is the desired flexibility of the class of models, and (Fonseca, 2010) proposes a general way to define the non-negative bivariate random vector (U,V) that leads to flexible non-separable covariance functions with useful properties. To build a general class of bivariate distributions for (U,V), we need to choose univariate distributions for space, time and space-time respectively. This choice defines the dependence in space, time and space-time. (Fonseca, 2010) defines the three univariate distributions to obtain Cauchy and Matérn classes for the temporal and spatial components respectively due to their attractive properties. For this purpose the generalized inverse Gaussian (GIG) distribution is used.

Consider $X_i \sim$ Ga(λ_i,a_i) for i = 0,2 and $X_1 \sim$ GIG(λ_1, ω, a_1), then the corresponding space-time covariance function generated is:

$$C(s,t) = \sigma^2\{1 + \frac{\gamma_1 + \gamma_2}{a_0}\}^{-\lambda_0}\{1 + \frac{\gamma_1}{a_1}\}^{-\frac{\lambda_1}{2}}\frac{K_{\lambda_1}(2\sqrt{(a_1 + \gamma_1)\omega})}{K_{\lambda_1}(2\sqrt{a_1\omega})}\{1 + \frac{\gamma_2}{a_2}\}^{-\lambda_2}, \qquad (63)$$

where $K_\lambda(.)$ is the modified Bessel function of order λ. The σ^2 parameter is the space-time variance, a_1 and ω are parameters explaining the rate of decay for the spatial correlation, and a_2 the same in the temporal dimension.

The spatial structure in space is more complex than the one defined in time. However, if the main interest is the temporal dimension, we could also consider a GIG distribution on X_2 instead to generate a more complex temporal structure. In theory GIG distributions could be used for both.

6 Experiments

I now analyse the MCPM model with Normal and GP mixing, and the LGCP model, on both a synthetic and a real world dataset with multiple correlated tasks. Both of these are missing data experiments, and we will compare the transfer capabilities of the MCPM and the LGCP model. I will implement a variety of kernels and compare their prediction capabilities.

The three measures we will use to evaluate performace are the root mean square error (RMSE), the negative log predicted likelihood (NLPL), and the computation time. The RMSE and NLPL are calculated as follows:

$$RMSE_p = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_{np} - E(\lambda_{np}))^2}, \tag{64}$$

$$NLPL_p = -\frac{1}{S}\sum_{s=1}^{S}\frac{\sum_{n=1}^{N} log(p(y_{np}|\lambda_{np}^s))}{n}, \tag{65}$$

where E(λ_{np}) represents the posterior mean estimate for the p^{th} intensity at x_n and S denotes the number of samples from the variational distributions q(F) and q(W) (Aglietti et al., 2018). To computation time is calculated as the average time per epoch.

As well as the performance measures discussed above, I also plotted the actual intensities along with the predicted intensities. This allows us to visualise the predicitive capability of the different models.

6.1 Synthetic Experiment

In the synthetic experiment I construct four correlated tasks by sampling from a multivariate point process with intensities obtained as the linear combination of two latent functions via task-specific mixing weights. The final count observations are obtained by adding noise to Poisson counts generated from the constructed intensities.

I use 400 input observations, constructed as a 20 x 20 dimension space-time meshgrid with 20 unique space and time observations. In each of the tasks we introduce some missing data values, and use the models to reconstruct the intensities in these missing data regions by learning inter-task correlations and transferring information across tasks.

I perform two different experiments, one for the stationary case and one for the non-stationary case. In both cases, the data is constructed from a separable product kernel with a RBF spatial covariance and a Periodic temporal covariance.

6.1.1 Stationary Experiments

In this experiment, I use the MCPM model with both Normal and GP prior weights, as well as the LGCP model, on the synthetic data. I run these models with both a separable product and separable sum kernel to compare the differences. A Matérn-$\frac{3}{2}$ function is used for the spatial covariance, and a Spectal Mixture kernel is used as the temporal covariance.

The results from the separable product experiments are shown below.

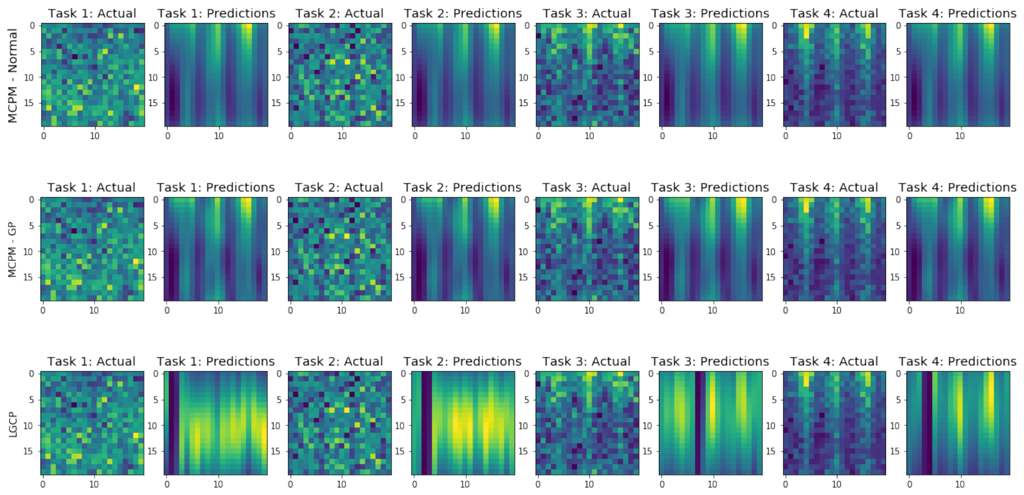

Figure 21: Stationary Synthetic Experiment - actual and predicted intensity plots with a separable product kernel.

RMSE	Task 1	Task 2	Task 3	Task 4
MCPM - Normal	7.96	8.89	7.11	20.83
MCPM - GP	7.52	7.46	7.30	18.07
LGCP	16.35	19.64	20.29	46.95

NLPL	Task 1	Task 2	Task 3	Task 4
MCPM - Normal	4.13	4.37	3.47	7.16
MCPM - GP	3.90	3.77	3.51	5.90
LGCP	17.20	19.88	16.89	75.20

	Time per Epoch (s)
MCPM - Normal	0.0824
MCPM - GP	0.0836
LGCP	0.3530

Table 1: Stationary Synthetic Experiment - performance statistics with a separable product kernel.

The results from the separable sum experiment are shown below:

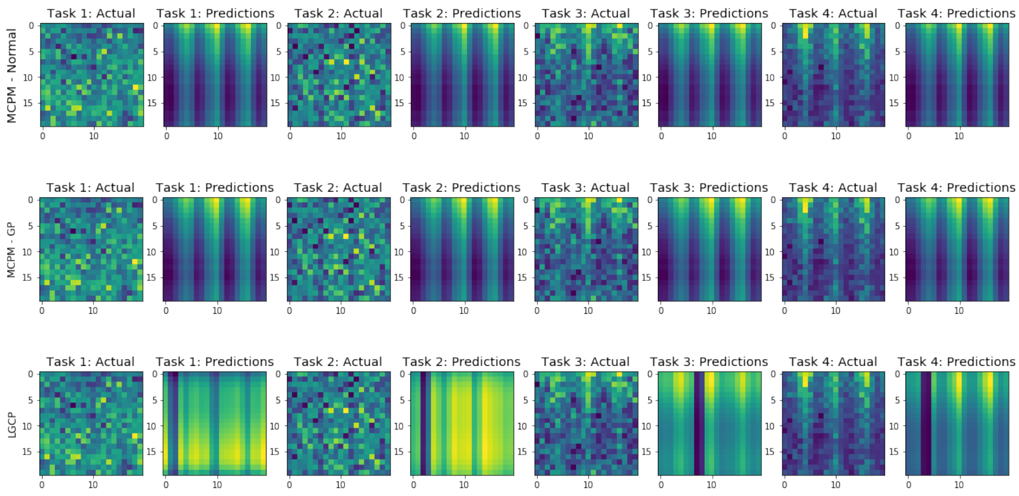

Figure 22: Stationary Synthetic Experiment - actual and predicted intensity plots with a separable sum kernel.

RMSE	Task 1	Task 2	Task 3	Task 4
MCPM - Normal	7.40	8.27	6.94	18.81
MCPM - GP	7.26	7.79	6.99	19.57
LGCP	7.88	13.54	19.51	44.84

NLPL	Task 1	Task 2	Task 3	Task 4
MCPM - Normal	3.85	4.06	3.43	6.22
MCPM - GP	3.79	3.86	3.44	6.56
LGCP	4.13	8.15	14.58	55.90

	Time per Epoch (s)
MCPM - Normal	0.0817
MCPM - GP	0.0819
LGCP	0.3471

Table 2: Stationary Synthetic Experiment - performance statistics with a separable sum kernel.

In the synthetic experiment, the MCPM model showed very little difference in performance when either Normal or GP prior weights are used. GP prior weights performed better in three of the four tasks for both RMSE and NLPL for each type of separable kernel, although only slightly. The experiments with Normal prior weights have a very slightly faster computation time, although this difference is almost negligible.

From our results, the superior performance of the MCPM model is quite evident. The LGCP model performed worse than MCPM for all tasks, with tasks two, three and four showing significantly worse peformance for the RMSE and NLPL. Also obvious from the results is the difference in computation time, with computations for the LGCP model have a significantly higher cost, over four times longer.

Another interesting element to consider from the experiment is how the separable product and separable sum kernels compare. The experiment with a separable sum kernel performed better in the majority of areas, although these differences were not large. This is particularly interesting considering that the synthetic data was constructed from a separable product kernel so we would maybe expect this to perform better. Different spatial and temporal covariances were used to construct the data than were used in the model, which is another factor we should consider. The computation time was marginally better for the sum kernel, although again with an almost negligible difference.

6.1.2 Non-Stationary Experiments

I now repeat the synthetic experiment from above, now considering the non-stationary case. In order to do this I run the MCPM model with both Normal and GP mixing weights, again with separable product and sum kernels, but use a non-stationary version of the Matérn-$\frac{3}{2}$ spatial covariance function. This will force the kernel as a whole to be non-stationary, and will allow us to look for any non-stationary patterns in the data. The results for the non-stationary separable product experiment are shown below.

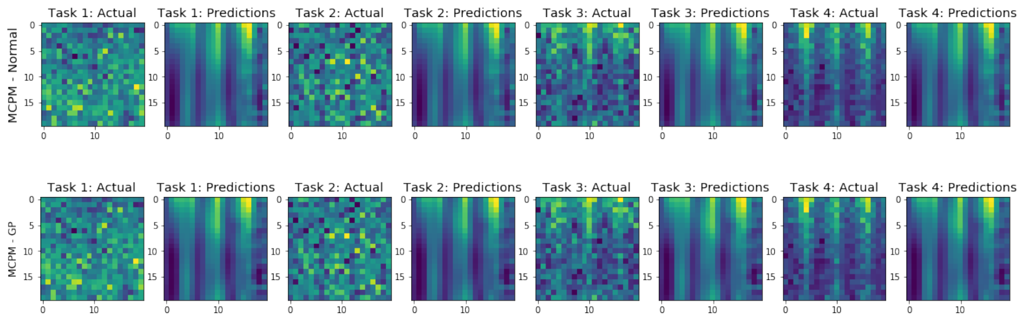

Figure 23: Non-Stationary Synthetic Experiment - actual and predicted intensity plots with a separable product kernel.

RMSE	Task 1	Task 2	Task 3	Task 4
MCPM - Normal	8.01	8.85	7.56	21.06
MCPM - GP	7.54	7.48	7.48	18.45

NLPL	Task 1	Task 2	Task 3	Task 4
MCPM - Normal	4.16	4.36	3.57	7.29
MCPM - GP	3.91	3.78	3.55	6.05

	Time per Epoch (s)
MCPM - Normal	0.0816
MCPM - GP	0.0818

Table 3: Non-Stationary Synthetic Experiment - performance statistics with a separable product kernel.

The results for the non-stationary separable sum experiment are shown below.

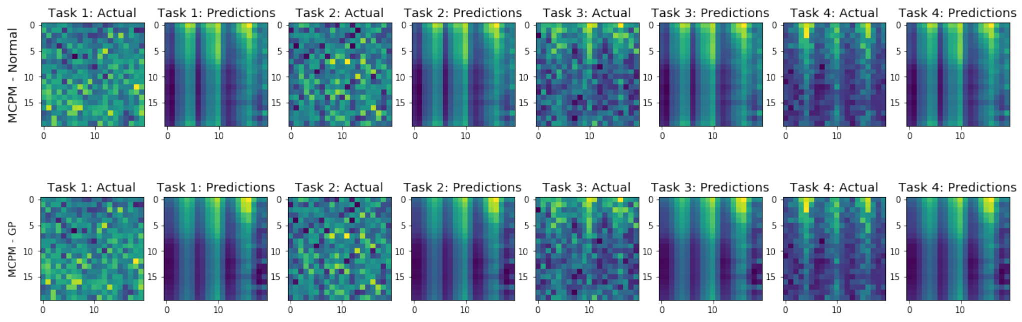

Figure 24: Non-Stationary Synthetic Experiment - actual and predicted intensity plots with a separable sum kernel.

RMSE	Task 1	Task 2	Task 3	Task 4
MCPM - Normal	8.30	10.25	7.48	21.69
MCPM - GP	7.40	7.77	7.21	19.60

NLPL	Task 1	Task 2	Task 3	Task 4
MCPM - Normal	4.32	5.07	3.55	7.66
MCPM - GP	3.85	3.85	3.49	6.56

	Time per Epoch (s)
MCPM - Normal	0.0816
MCPM - GP	0.0821

Table 4: Non-Stationary Synthetic Experiment - performance statistics with a separable sum kernel.

Looking at the experiments with the separable product kernel, the stationary experiment achieves a slightly better performance than the non-stationary case. However, the difference in values of RMSE and NLPL are extremely similar, with no big differences.

In the case of the separable sum kernel, the differences are a lot more obvious. The stationary experiment for the MCPM model with Normal mixing weights outperforms its non-stationary equivalent for all tasks. The results are much closer in the MCPM model with GP weights, with the stationary kernel not performing significantly better than its non-stationary alternative.

It is worth noting that the synthetic data used was constructed from non-stationary kernels, so it is not surprising to see superior performance in the non-stationary case.

6.2 UK Crime Data Experiments

I now move on to evaluate the performance of the model with different kernels on a real world dataset. The data used was an open dataset about crimes in England, Wales and Northern Ireland, and was obtained from data.police.uk. The data specifically focussed on street-level crime and was taken from all 45 police forces for the period August 2016 to July 2019.

The data was contained in separate csv files for each police force for every month. Each row of the data contains information about a single crime, including: the police force, the month, the longitude and latitude, the type of crime, the crime ID, and a description of the location.

The first pre-processing step involved creating dummy variables for each of the crime types, giving 14 columns, one for each crime type. Then, for each csv file, I added a data row containing the count of occurences of each crime type in that csv. For each row I then added columns for the data, the longitude and the latitude. As each csv file contained data from one month only, the date was copied over. For the crimes recorded by each police force, the longitude and latitude locations were calculated as the average locations of all the crimes recorded for that police force, giving 45 spatial locations.

I removed the data recorded by the British Transport Police due to large variety of locations for which they recorded crimes, clearly due to the nature of the force. I converted the dates into a unix timestamp to give integer values, and then standardised the date, latitude and longitude obsevations.

My final dataset consisted of three input dimensions, (date, longitude and latitude), and fourteen tasks (each of the crime types). The data consists of 1584 observations, with 44 unique spatial locations and 36 unique temporal locations. I removed 50 consecutive observations at different indexes for each of the 14 tasks in order to generate missing data values that I would use my model to predict.

The mapping between task number and crime type is shown in the table below:

Task Number	Crime Type
Task 1	Anti-Social Behaviour
Task 2	Bicycle Theft
Task 3	Burglary
Task4	Criminal Damage & Arson
Task 5	Drugs
Task 6	Other Crime
Task 7	Other Theft
Task 8	Possession of Weapons
Task 9	Public Order
Task 10	Robbery
Task 11	Shoplifting
Task 12	Theft From the Person
Task 13	Vehicle Crime
Task 14	Violence & Sexual Offences

When evaluating the performance of the model, I have plotted the police force of the area the crime occured against time. Whilst this does not allow us to visualise the distance in space, this would be very difficult without multiple plots for different points in time. The main objective of the plots is to visualise how well our model predicts the crime counts, and this can be seen clearly.

6.2.1 Stationary Experiments

I tested multiple combinations of spatial and temporal kernels on the UK Crime dataset. The combination of a Matérn-$\frac{3}{2}$ spatial covariance, and Periodic temporal covariance, appeared to fit the data best. Below are some visualisations and performance statistics of experiments using both a separable product and separable sum kernel formed from the individual kernels discussed above.

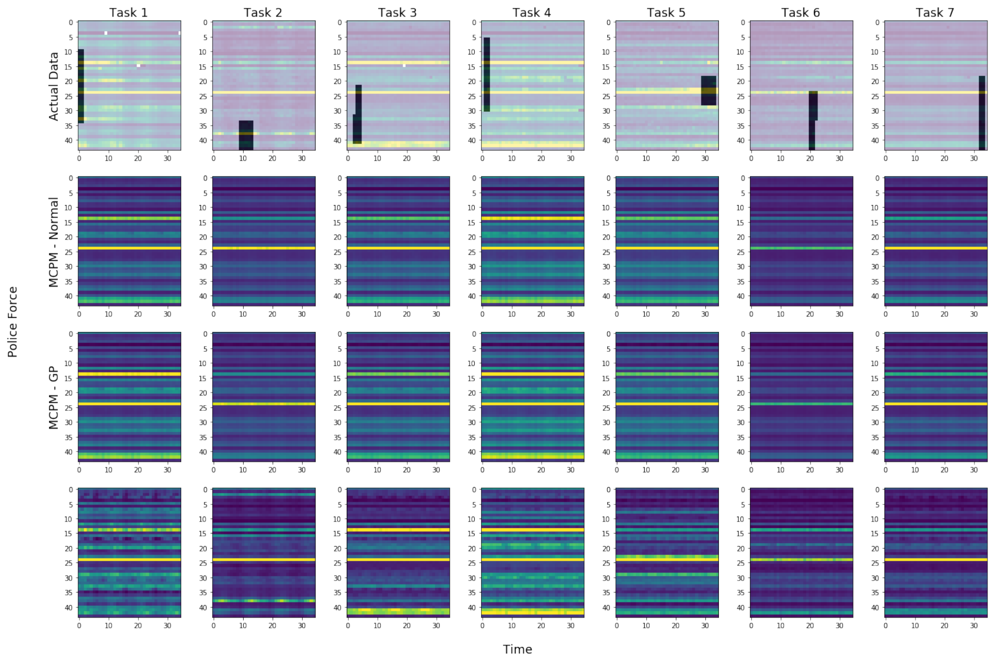

Figure 25: UK Crime Experiment - Actual and Predicted Intensity Plots with Separable Product Kernel (Task (1) - (7))

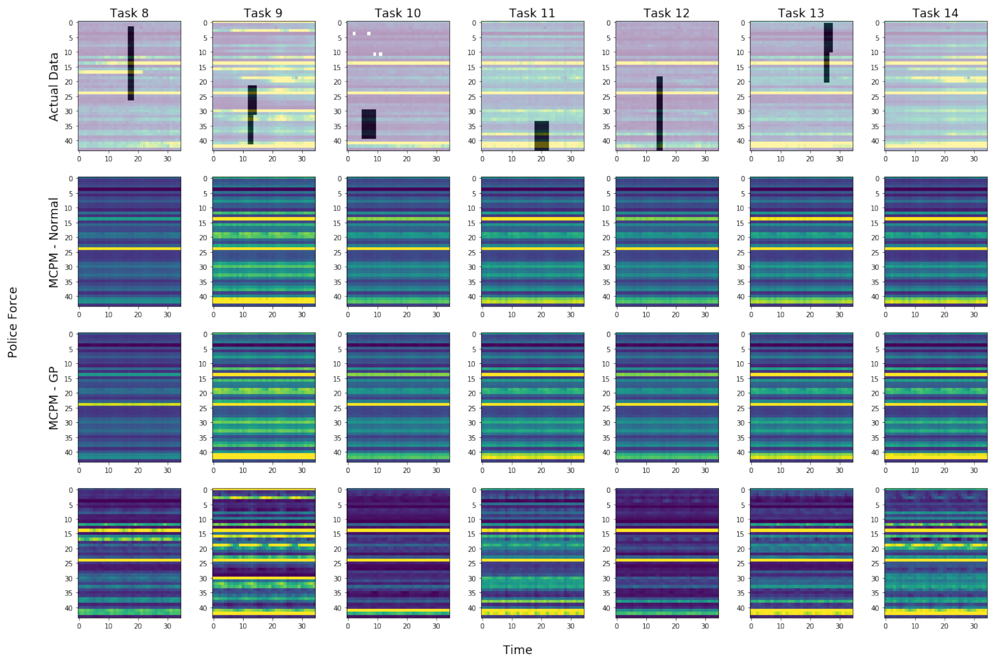

Figure 26: UK Crime Experiment - Actual and Predicted Intensity Plots with Separable Product Kernel (Task (8) - (14))

RMSE	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
MCPM - Normal	2675.5	173.4	463.7	342.0	745.7	88.4	1118.4
MCPM - GP	1997.8	171.5	405.3	380.3	760.0	94.6	1022.5
LGCP	2200.3	111.1	125.4	229.6	286.5	55.8	444.3

RMSE	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14
MCPM - Normal	80.1	400.1	99.0	242.0	705.2	255.5	1536.0
MCPM - GP	84.2	402.0	104.3	206.0	714.2	255.5	1329.1
LGCP	23.0	233.1	16.3	120.9	200.5	167.0	1178.9

NLPL	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
MCPM - Normal	844.6	48.1	45.4	35.0	182.5	15.0	98.3
MCPM - GP	527.0	46.7	40.6	29.3	191.1	16.3	85.2
LGCP	429.4	25.1	14.0	21.3	31.9	9.1	21.7

NLPL	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14
MCPM - Normal	25.0	112.6	44.4	31.6	184.6	50.9	396.1
MCPM - GP	27.3	117.8	47.9	23.9	195.5	57.6	236.6
LGCP	5.2	27.9	5.0	13.4	14.1	33.4	367.8

	Total Computation Time (s)	Time per Epoch (s)
MCPM - Normal	3230.8	6.46
MCPM - GP	3349.7	6.70
LGCP	25638.3	51.3

Table 5: UK Crime Experiment - Performance Statistics with Separable Product Kernel

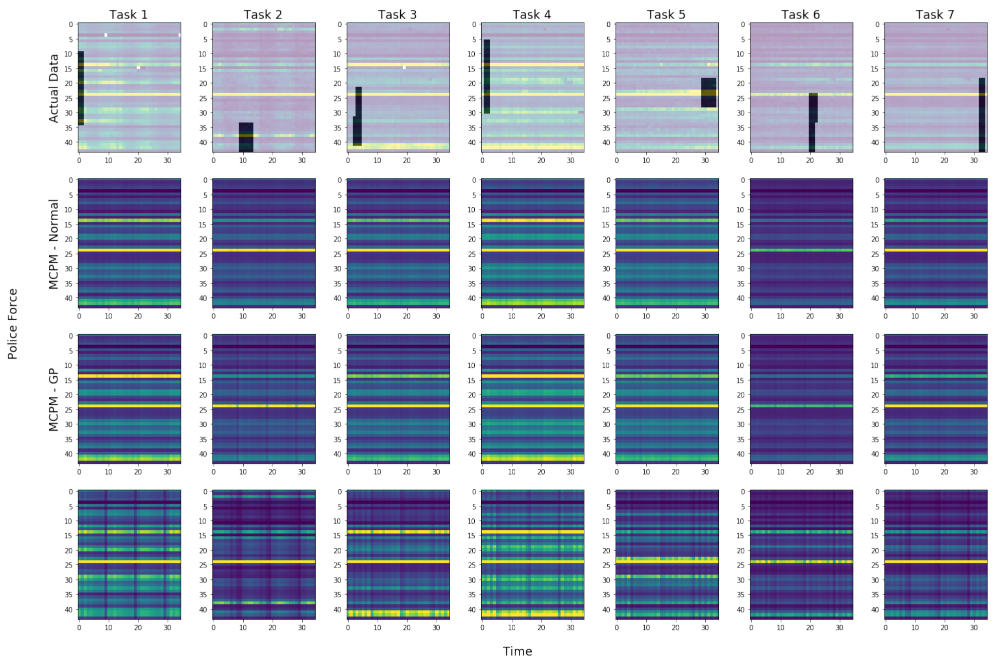

Figure 27: UK Crime Experiment - Actual and Predicted Intensity Plots with Separable Sum Kernel (Task (1) - (7))

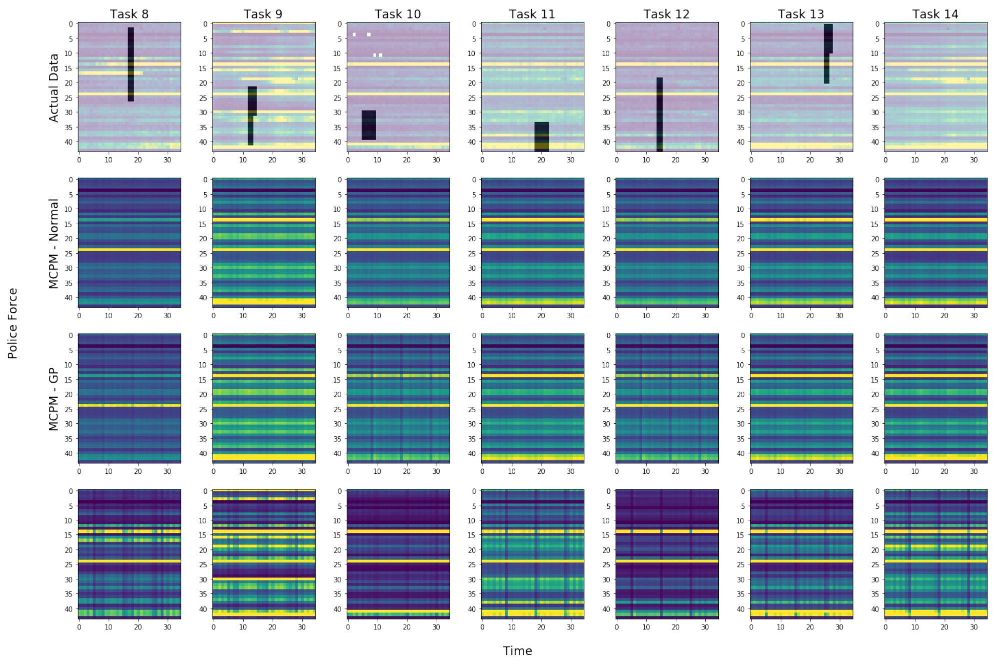

Figure 28: UK Crime Experiment - Actual and Predicted Intensity Plots with Separable Sum Kernel (Task (8) - (14))

RMSE	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
MCPM - Normal	2626.5	173.8	459.0	351.6	740.7	86.4	1100.4
MCPM - GP	1976.1	171.0	413.0	375.8	748.0	91.1	1004.7
LGCP	1538.3	102.9	178.4	145.1	164.2	41.2	211.2

RMSE	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14
MCPM - Normal	79.8	392.2	99.5	238.8	705.0	254.2	1486.2
MCPM - GP	84.0	393.5	105.6	206.6	702.0	250.7	1266.1
LGCP	31.8	147.6	15.4	183.1	381.9	261.2	946.8

NLPL	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
MCPM - Normal	795.2	48.4	44.2	34.2	178.7	14.7	94.6
MCPM - GP	497.9	46.4	41.1	27.1	182.6	15.8	81.5
LGCP	240.9	22.4	12.6	11.7	16.6	6.7	14.5

NLPL	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14
MCPM - Normal	24.9	108.5	44.8	30.7	183.9	49.0	366.3
MCPM - GP	26.9	114.1	48.3	24.1	183.0	53.3	200.9
LGCP	6.0	16.0	4.4	24.2	51.6	45.1	154.9

	Total Computation Time (s)	Time per Epoch (s)
MCPM - Normal	3410.0	6.82
MCPM - GP	3503.1	7.00
LGCP	66182.3	132.4

Table 6: UK Crime Experiment - Performance Statistics with Separable Product Kernel

The stationary separable product experiment produces some very interesting results. The LGCP model outperforms both versions of the MCPM model on all tasks for RMSE, and all but one task for NLPL. This is in stark contrast to the results of the synthetic experiment. The results of the stationary separable sum experiment reinforce this, with LGCP performing better in 13 of the 14 tasks for both RMSE and NLPL.

Whilst performing worse with regards to the RMSE and NLPL statistics, the MCPM model does perform significantly better in regards to the computation time. In both the separable product and sum experiments, the average time per epoch is roughly 6.5 to 7 seconds. In the case of the LGCP model, the average time is over 50 seconds per epcoh. This illustrates the computational benefits of the MCPM model. It is possible that for a similar computational cost (for example using more epochs in the MCPM model), the performance difference between the two models would actually be reversed.

Comparing the performance difference between using Normal or GP mixing weights, in the separable product experiment the results are very similar. For both the RMSE and NLPL, the best performance is split fairly equally between the tasks. In the separable sum experiment, the GP prior weights can be seen to perform better in roughly 8 of the tasks.

Looking at the comparison between the separable product and the separable sum kernel, the separable sum experiment has superior performance in the majority of cases.

6.2.2 Non-Stationary Experiments

I now perform a non-stationary version of the experiment above on the same UK Crime dataset. I implement the MCPM model with both Normal and GP mixing weights, with separable product and sum kernels. To make the kernels non-stationary, I use a non-stationary version of the Matérn-$\frac{3}{2}$ kernel as in the synthetic experiment above. The results for the separable product experiment are shown below.

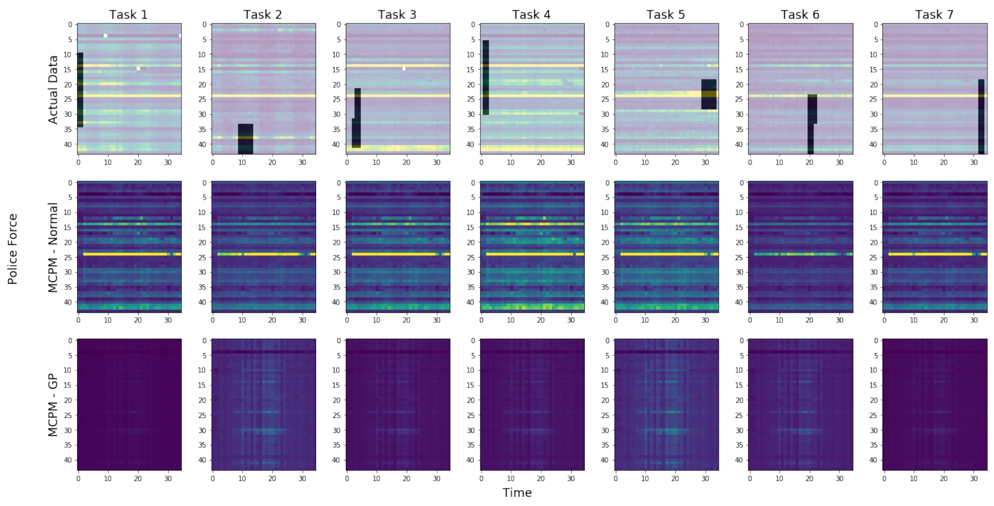

Figure 29: Non-Stationary UK Crime Experiment - Actual and Predicted Intensity Plots with Separable Product Kernel (Task (1) - (7))

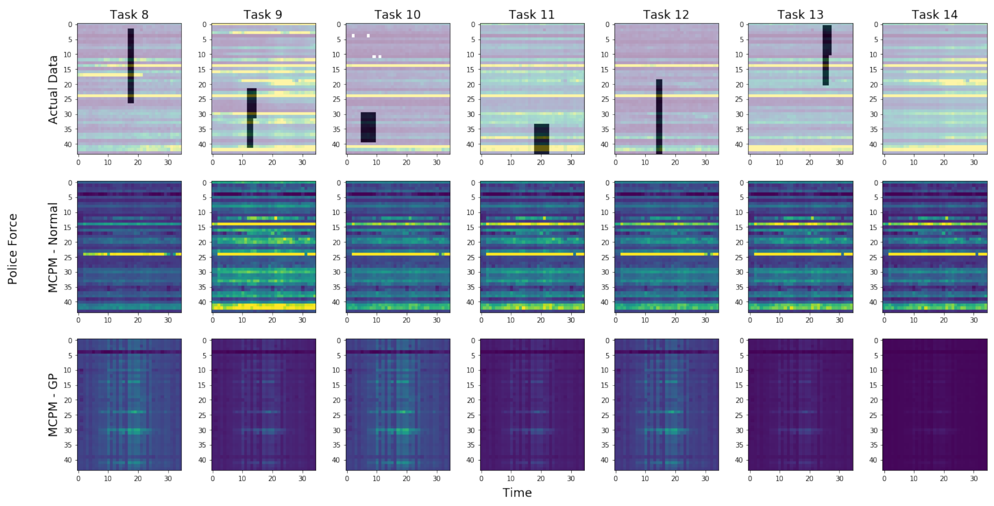

Figure 30: Non-Stationary UK Crime Experiment - Actual and Predicted Intensity Plots with Separable Product Kernel (Task (8) - (14))

RMSE	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
MCPM - Normal	5552.0	182.7	696.4	807.3	918.2	90.6	1568.5
MCPM - GP	6478.9	218.3	1444.0	1501.0	1084.7	206.7	2390.3

RMSE	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14
MCPM - Normal	78.5	413.6	79.6	291.6	689.5.0	312.5	2444.0
MCPM - GP	105.3	1128.5	56.8	756.2	827.4	696.1	5309.1

NLPL	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
MCPM - Normal	4023.0	58.3	102.0	251.4	390.1	17.7	263.3
MCPM - GP	13925.9	103.0	1411.3	1952.8	896.2	72.3	2468.3

NLPL	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14
MCPM - Normal	22.5	114.5	32.3	52.7	169.1	68.4	1007.7
MCPM - GP	43.9	832.6	23.0	586.0	407.2	543.4	10008.7

	Total Computation Time (s)	Time per Epoch (s)
MCPM - Normal	981.6	1.96
MCPM - GP	995.3	1.99

Table 7: Non-Stationary UK Crime Experiment - Performance Statistics with Separable Product Kernel

The results of the non-stationary separable sum experiment are shown below.

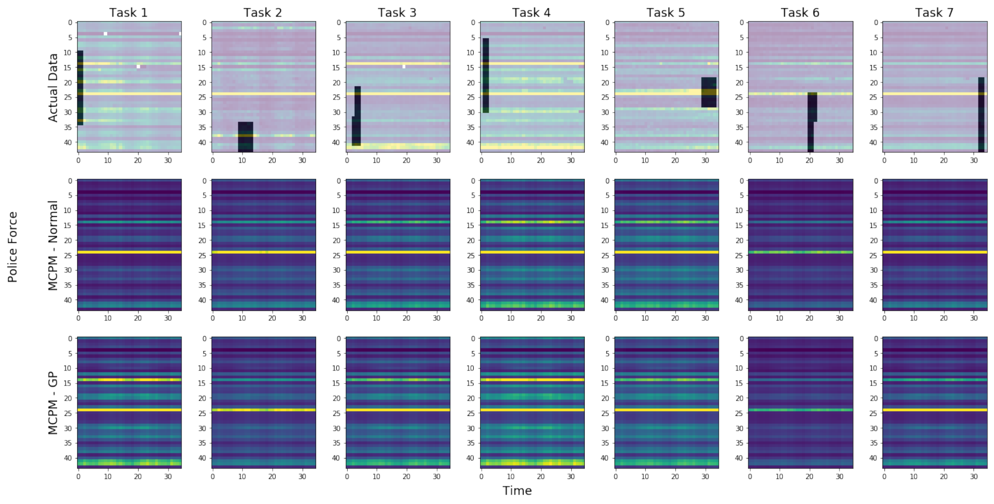

Figure 31: Non-Stationary UK Crime Experiment - Actual and Predicted Intensity Plots with Separable Sum Kernel (Task (1) - (7))

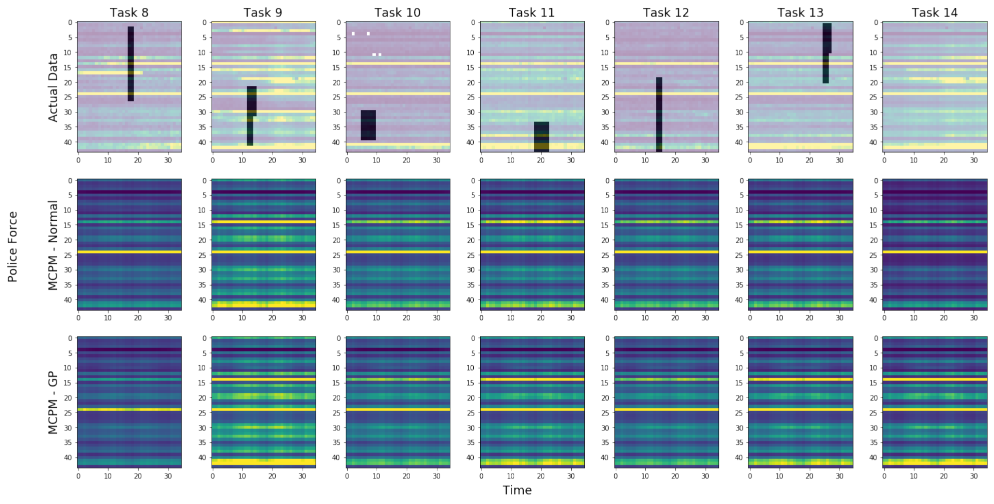

Figure 32: Non-Stationary UK Crime Experiment - Actual and Predicted Intensity Plots with Separable Sum Kernel (Task (8) - (14))

RMSE	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
MCPM - Normal	4602.9	175.7	564.3	519.4	735.1	81.5	1290.3
MCPM - GP	2781.8	171.8	487.9	391.1	755.9	90.9	1056.1

RMSE	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14
MCPM - Normal	77.8	398.1	89.5	305.5	697.5	261.8	3114.9
MCPM - GP	85.9	381.7	99.8	241.0	713.5	240.8	1433.0

NLPL	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
MCPM - Normal	2684.0	50.1	66.0	116.9	178.7	13.5	149.8
MCPM - GP	904.0	46.9	47.5	49.0	188.5	15.5	87.6

NLPL	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14
MCPM - Normal	23.3	103.1	38.4	51.0	175.8	47.8	1368.0
MCPM - GP	27.6	108.3	45.1	31.3	195.7	51.8	317.2

	Total Computation Time (s)	Time per Epoch (s)
MCPM - Normal	990.1	1.98
MCPM - GP	1049.5	2.10

Table 8: Non-Stationary UK Crime Experiment - Performance Statistics with Sum Product Kernel

Looking at the visualisations of the experiments above, we can immediately see that the non-stationary separable product experiment, particularly with GP prior weights, performs particularly poorly on the UK Crime dataset. This is supported by the performance statistics, with the RMSE and NLPL significantly worse than in the stationary case for the majority of tasks. The MCPM model with Normal mixing weights achieves slightly better results, however still has a worse performance than its stationary counterpart.

The visualisations for the separable sum non-stationary experiment suggest improved predictive performance on the dataset, however in the majority of tasks the non-stationary separable sum still does not perform as well as its stationary counterpart. Here, the MCPM model with GP prior weights performs better.

The overall poor performance of the non-stationary experiment could be caused by the composition of the data. Another reason could be poor initialisation of non-stationary kernel, or even down to the effect of the non-stationary spatial covariance and stationary temporal covariance interacting with each other.

7 Conclusion

The aim of the MCPM framework developed in (Aglietti et al., 2019) was to address the modelling and inference limitations of current approaches to model count data in a spatio-temporal setting. (Aglietti et al., 2019) implemented the MCPM model in a spatial setting, and the aim of this paper was to implement the model into a spatio-temporal setting, extending its use.

The paper reviews some background material including GP's and VI, before covering the LGCP and MCPM models. I then presented a variety of one dimensional kernel functions, both stationary and non-stationary. I reviewed seperable and non-separable methods for combining spatail and temporal covariances in order to implement spatio-temporal inference. Finally, I implemented the MCPM and LGCP models on a synthetic and real world spatio-temporal dataset consisting of correlated count processes. I implemented both stationary and non-stationary separable kernels.

The difference in performance between the LGCP and MCPM models produce some interesting observations. As in (Aglietti et al., 2019), we showed that the MCPM model has far superior computational performance than the LGCP model. The reduced computational costs of the MCPM model are exacerbated as the number of tasks increases, as shown by the difference in computation time for the synthetic experiment with four correlated tasks and the real dataset with 14 correlated tasks.

The stationary and non-stationary experiments were run on different days, and whilst the computation times within each experiment are comparable, the computation times between different experiments are not reliable due to the different speeds of the server over these days. This leads to a question of the suitability of computation time as a measure of computation cost unless a baseline is run before each experiment. An area of consideration for future experiments would be to consider an alternative measure of compuational cost to ensure standardisation between experiments.

When it comes to the performance statistics considered in the experiments above, the RMSE and NLPL, the performance differences between the two models are particularly challenging to evaluate. In the synthetic experiments, the MCPM model clearly outperforms the LGCP model for both performance statistics. In the UK Crime experiment, however, the performance of the models are reveresed with the LGCP performing best for the majority of tasks. To draw firm conclusions on the performance differences, both models would need to be tested on a wider variety of datasets with several different kernels. An interesting area to consider is how the performance of each model would compare for similar computational costs. For example, implementing the MCPM model with a higher number of epochs could improve its performance further, whilst still maintaining reasonable compuational cost.

Another interesting area of comparison in this paper is the performance difference between the separable product and separable sum kernels. In Chapter 5 I have plotted draws from a GP with different separable product and sum covariances, as well as the separable functions themselves, allowing us to visually compare the differences. A particular area of interest is where one of the constituent covariances has values close to zero. In the separable product case this forces the overall covariance to be very close to 0, however in the separable sum scenario the overall covariance would be very close to the value of covariance for the remaining constituent kernel. This means the separable product kernel is likely to experiences larger areas of very low covariance than the separable sum kernel.

In the synthetic experiment, the separable sum kernel performs better in the stationary case, but worse in the non-stationary case. The separable sum kernel generally performs better on the UK Crime dataset, significantly better for the non-stationary case. It would be easy to conclude that the separable sum covariance is superior to the separable product, however further experimentation is needed before drawing hard conclusions. I believe that the constituent spatial and temporal covariances chosen will determine the type of separable kernel that is appropriate.

I would like to extend this paper with further research into non-stationary kernels. Further areas of work would start by implementing a non-stationary temporal covariance, before comparing the effects of constructing a separable kernel from: a non-stationary spatial and stationary temporal covariance; a stationary spatial and non-stationary temporal covariance; and finally both non-stationary spatial and temporal covariances. Research would look at how the different constituent kernels interact with each other, as well as their performance on various datasets.

Another area of further work is the implementation of non-separable kernels. It is often convenient to assume the independence of space and time, however research has shown this is not usually the case. Both stationary and non-stationary non-separable kernels could be implemented, and their comparison with each other as well as their separable counterparts would be interesting to investigate.

A Source Code

Code and data for all of the experiments and plots are available at:
`https://github.com/BenMorris1/Spatio_Temporal_MCPM`

Important Note: the experiments were run on a remote server, and the speed of this server fluctuated as different experiments were run. For example, the stationary and non-stationary crime experiments were run on different days, and the average time per epoch for Normal MCPM model is around 6.5 seconds in the stationary case and less than two seconds in the non-stationary case. This difference is not due to a lower computational cost of the non-stationary experiment, and therefore computation times should be treated with caution. The computation times within each experiment are comparable, however they are not comparable between experiments.

References

[1] Aglietti, V, Damoulas, T, and Bonilla, E. (2019). *Efficient Inference in Multi-Task Cox Process Models.* arXiv preprint arXiv:1805.09781.

[2] Álvarez, A., M, Rosasco, L, and Lawrence, D., N. (2012) *Kernels for Vector-Valued Functions: a Review*' Foundations and Trends in Machine Learning, 4(3), 195-266.

[3] Blei, M. D, Kucukelbir, A and McAuliffe, D. J. (2018). *Variational Inference: A Review for Statisticians.* Journal of the American Statistical Association, 112(518), 859-877.

[4] Bonilla, E, Krauth, K and Dezfouli, A. (2018). *Generic Inference in Latent Gaussian Process Models.* arXiv preprint arXiv:1609.00577.

[5] Chen, K, Groot, P, Chen, J and Marchiori, E. *Generalized Spectral Mixture Kernels for Multi-Task Gaussian Processes.* arXiv preprint arXiv:1808.01132.

[6] Cheng, C. A., and Boots, B. (2017). *Variational Inference for Gaussian Process Models with Linear Complexity.* Advances in neural information processing systems (pp. 342-350).

[7] Cho, Y and Saul, K., L. (2019). *Kernel Methods for Deep Learning.* University of California, San Diego.

[8] Cressie, N, and Huang, H., C. (1999). *Classes of Non-Separable Spatio-Temporal Stationary Covariance Functions.* Journal of the American Statistical Association, 94(448), 1330-1339.

[9] Do, C. (2007). *Gaussian Processes.* Stanford University, Stanford, CA.

[10] Duvenaud, K., D. (2014). *Automatic Model Construction with Gaussian Processes* Doctoral dissertation, Ph. D. thesis, University of Cambridge.

[11] Ebden, M. (2008). *Gaussian Processes for Regression: A Quick Introduction.* arXiv preprint arXiv:1505.02965.

[12] Flaxman, S. (2015). *Machine learning in space and time.* Doctoral dissertation, Ph. D. thesis, Carnegie Mellon University.

[13] Flaxman, S, Wilson, G., A, Neill, B., D, Nickisch, H, and Smola, J., A. (2015). *Fast Kronecker Inference in Gaussian Processes with Non-Gaussian Likelihoods.* International Conference on Machine Learning (pp. 607-616).

[14] Fonseca, O., C., T. (2010). *On Flexible Modelling of Spatiotemporal Processes.* Doctoral dissertation, Ph. D. thesis, University of Warwick.

[15] Gallager, R. G. (2012). *Discrete Stochastic Processes* Springer Science & Business Media.

[16] Genton, G., M. (2001). *Classes of Kernels for Machine Learning: A Statistics Perspective.* Journal of Machine Learning Research 2, 299-312.

[17] Gneiting, T. (2011). *Nonseparable Stationary Covariance Functions for Space-Time Data.* Journal of the American Statistical Association, 97:458, 590-600.

[18] Heinonen, M. (2017). *Learning with Spectral Kernels.* Aalto University, CS department.

[19] King, N and Lawrence, N. (2006). *Fast Variational Inference for Gaussian Process Models through KL-Correction.* European Conference on Machine Learning.

[20] Knox, G., E, and Bartlett, S., M. (1964). *The Detection of Space-Time Interactions.* Journal of the Royal Statistical Society. Series C (Applied Statistics).

[21] MacKay, C. J. D. (2003) *Information Theory, Inference and Learning Algorithms.* Cambridge University Press.

[22] Mantel, N. (1967) *The Detection of Disease Clustering and a Generalized Regression Approach.* Cancer research, 27(2 Part 1):209–220.

[23] Møller, J, Syversveen, R., A, and Waagepetersen, P., R. (1998). *Log Gaussian Cox Processes.* Scandinavian journal of statistics, 25(3), 451-482.

[24] Paciorek, J., C. (2003). *Nonstationary Gaussian Processes for Regression and Spatial Modelling.* Doctoral dissertation, PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania.

[25] Paciorek, J., C, and Schervish, J., M. (2004) *Nonstationary Covariance Functions for Gaussian Process Regression.* Advances in neural information processing systems (pp. 273-280).

[26] Paciorek, J., C, and Schervish, J., M. (2006) *Spatial Modelling Using a New Class of Nonstationary Covariance Functions.* Environmetrics: The official journal of the International Environmetrics Society, 17(5), 483-506.

[27] Parra, G and Tobar, F. (2017). *Spectral Mixture Kernels for Multi-Output Gaussian Processes.* Advances in Neural Information Processing Systems (pp. 6681-6690).

[28] Rasmussen, C and Williams, K. (2006). *Gaussian Processes for Machine Learning.* MIT Press.

[29] Remes, S, Heinonen, M and Kaski, S. (2017). *Non-Stationary Spectral Kernels.* Advances in Neural Information Processing Systems (pp. 4642-4651).

[30] Ton, J., F, Flaxman, S, Sejdinovic, and Bhatt, S. (2017) *Spatial Mapping with Gaussian Processes and Nonstationary Fourier Features.* Spatial statistics, 28, 59-78.