Clustering Gaussian Processes: A Modified EM Algorithm for Functional Data Analysis with Application to British Columbia Coastal Rainfall Patterns

CLUSTERING GAUSSIAN PROCESSES: A MODIFIED EM ALGORITHM FOR FUNCTIONAL DATA ANALYSIS WITH APPLICATION TO BRITISH COLUMBIA COASTAL RAINFALL PATTERNS

BY

FORREST PATON, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF MATHEMATICS & STATISTICS

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

Master of Science (2018) McMaster University
(Mathematics & Statistics) Hamilton, Ontario, Canada

TITLE:	Clustering Gaussian Processes: A Modified EM Algorithm for Functional Data Analysis with Application to British Columbia Coastal Rainfall Patterns
AUTHOR:	Forrest Paton B.Sc., (Statistics) Simon Fraser University, Vancouver, Canada
SUPERVISOR:	Dr. Paul D. McNicholas
NUMBER OF PAGES:	viii, 39

Abstract

Functional data analysis is a statistical framework where data are assumed to follow some functional form. This method of analysis is commonly applied to time series data, where time, measured continuously or in discrete intervals, serves as the location for a function's value. In this thesis Gaussian processes, a generalization of the multivariate normal distribution to function space, are used. When multiple processes are observed on a comparable interval, clustering them into sub-populations can provide significant insights. A modified EM algorithm is developed for clustering processes. The model presented clusters processes based on how similar their underlying covariance kernel is. In other words, cluster formation arises from modelling correlation between inputs (as opposed to magnitude between process values). The method is applied to both simulated data and British Columbia coastal rainfall patterns. Results show clustering yearly processes can accurately classify extreme weather patterns.

Acknowledgements

I would first like to thank Paul McNicholas—his advice and support throughout my degree was invaluable. He is a great mentor and constant reminder of where hard work and passion can take you. I'm grateful to be part of such a fantastic research group.

I would also like to thank Luke Bornn for sparking my interest in applied statistics and his advice during undergraduate studies. Thanks to Dave Campbell, Jinko Graham, Rachel Altman, and Daniela Blettner for their support and help.

Last but not least, thanks to Fred Hoppe and Shui Feng for being on my exam committee.

Contents

List of Figures

Chapter 1

Introduction

Functional data analysis (FDA) is a branch of statistics that deals with modeling data as a function over some smooth space. The FDA framework models data that are assumed to take some functional form. Today, more and more streams of data are generated over continuous or discrete time points, both examples of functional data (Wang *et al.*, 2016). Thus, methods to analyze such data are becoming increasingly popular. Specifically, the use of Gaussian processes (GPs) gives a probabilistic starting point. A GP is a stochastic process that generalizes a finite-dimensional normal distribution to function space. GPs have been used to successfully solve complex non-linear regression and classification problems (Rasmussen, 2005).

When multiple functions exist on the same interval (usually compact $[0, T]$ and finite) it can be useful to classify them into a finite number of mutually exclusive groups. For example, grouping patients into distinct groups based off of some time measurement (heart rate, blood pressure, etc.) to specify a more group specific intervention or treatment. Here, the process would be defined on the index set time; furthermore, in Chapter 4, the months of the year will provide the necessary index. In this context, using GPs (as opposed to the multivariate normal distribution) provides some immediate advantages. First, it allows processes with different index sets to be compared. Second, the use of a kernel function can model arbitrarily nonlinear complex pairwise correlation between indexes.

In an unlabelled, non-functional context, mixture models are popular methods used to estimate clusters where the data is assumed to have a “missing” or latent variable that explains the inter-group variation. A complete data log likelihood is then maximized and data are then often assigned to a cluster using maximum a posteriori classification (MAP).

In this thesis, the mixture model framework is extended to cluster functional data. In iterative steps, the hyper-parameters that define each GP are optimized via a Gradient Ascent algorithm. Then, the complete data log likelihood parameters are maximized. This is continued until some convergence criterion is satisfied. This method is then applied to rainfall data and results are discussed in the context of classification.

Chapter 2

Background

2.1 Gaussian Processes

2.1.1 Kernel Function

In the monograph text by Rasmussen (2005) a GP is defined as: "...a collection of random variables, any finite number of which have a joint Gaussian distribution". A GP, $f(x)$, creates a set of random variables evaluated at x. In essence, a GP is a distribution over functions, i.e.,

$$\mathbf{f}(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}),\ k(\mathbf{x}, \mathbf{x}')\big). \tag{2.1}$$

An example of a GP is given in Figure 2.1.

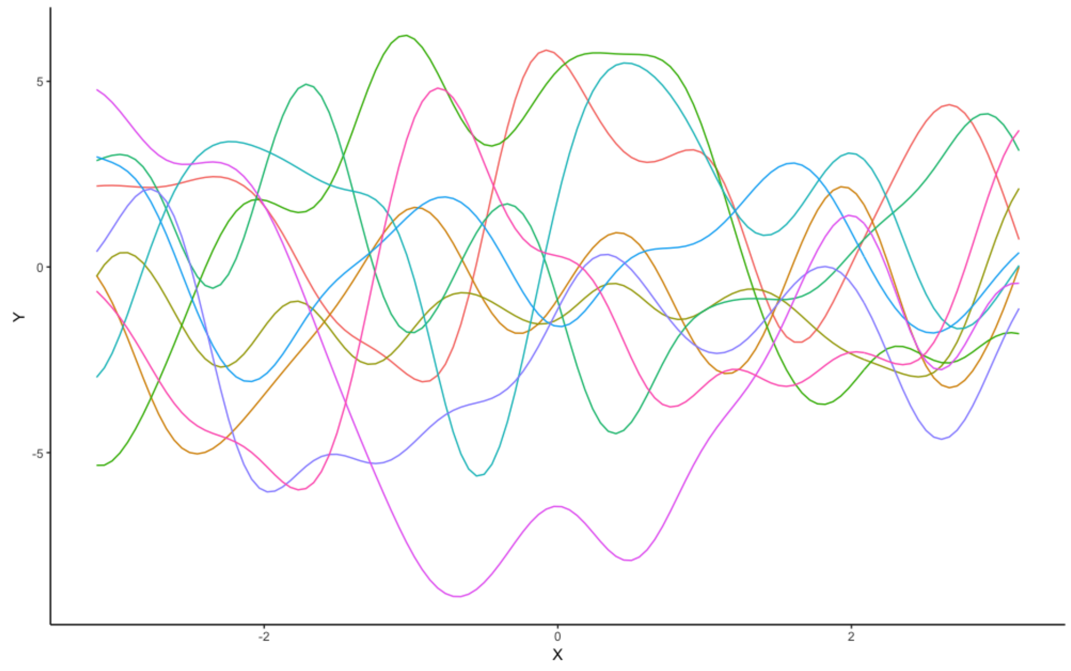

Figure 2.1: Ten GPs randomly drawn with mean function $m(x) = 0$ and SE covariance kernel.

A GP is defined by its mean function and covariance kernel:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \tag{2.2}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \tag{2.3}$$

A common GP, and the kernel considered in this thesis, is defined with mean function $\mathbf{0}$ and squared exponential (SE) covariance function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}|\mathbf{x}_i - \mathbf{x}_j|^2\right) + \sigma_n^2 \delta_{ij}. \tag{2.4}$$

The SE kernel is a widely used kernel (Rasmussen, 2005). One convenient property of the SE kernel is being able to infinitely differentiate it, which is useful because the first derivative is needed for hyper-parameter estimation. Here, the use of the term hyper-parameter refers to a set of parameters that make up the "non-parametric model". That is, the hyper-parameters only appear in the model's prior, and as shown in (2.7), are integrated out of the final model (posterior). The covariance kernel for example is defined by the following set of hyper-parameters which control the shape of the process:

$$\begin{aligned}
\sigma_f^2 &\text{ is the height parameter,} \\
l &\text{ is the length-scale parameter,} \\
\sigma_n^2 &\text{ is the measurement noise,} \\
\delta_{ij} &= \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise} \end{cases} .
\end{aligned}$$

In general, different kernel functions can be used, with the SE being the most popular. Because the kernel is needed to populate a multivariate normal distribution covariance matrix, the kernel is restricted to produce positive semi-definite matrices. While the SE covariance kernel is a popular choice, modelling the shape of the covariance kernel is an open ended problem. One systematic solution can be formed by considering a Bayesian model selection framework as discussed in Rasmussen (2005). For the SE covariance kernel, σ_f^2 controls the height or the amplitude of the GP, l controls the length-scale. The hyper-parameter σ_n^2 adds measurement noise to the function output y. Where y is the function's value f with the additive noise, $y = f + \sigma_n^2$. If the observed value y is perfectly interpolated from the GP i.e. $y = f$ then $\sigma_n^2 = 0$.

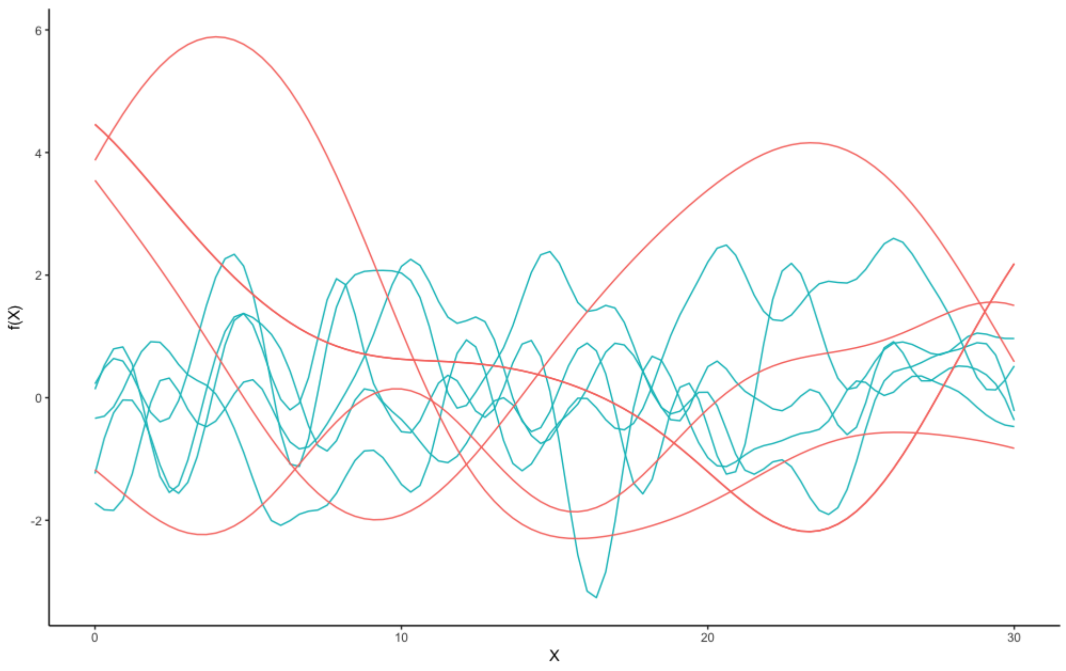

Figure 2.2: Effect of changing SE kernel hyper-parameters, blue lines generated with $l = 1$, red lines generated with $l = 5$. While σ_f is held constant for both cases ($\sigma_f = 1$).

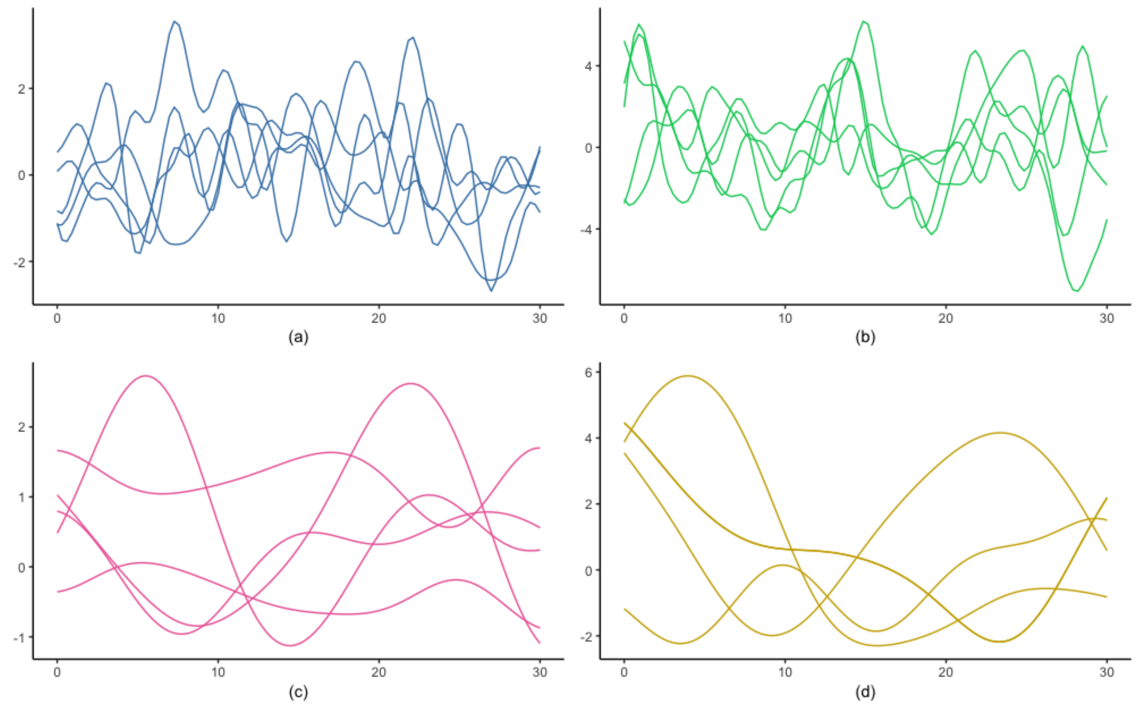

Figure 2.3: Effect of changing SE kernel hyper-parameters. (a) $l = 1$, $\sigma_f = 1$, (b) $l = 1$, $\sigma_f = 5$, (c) $l = 5$, $\sigma_f = 1$, (d) $l = 5$, $\sigma_f = 5$.

This kernel is used to construct a matrix, $\mathbf{K}$, which will serve as the variance covariance matrix in a multivariate normal distribution introduced in the next section

$$\mathbf{K} = \begin{pmatrix} k(x_1,\ x_1) & \cdots & k(x_1,\ x_n) \\ \vdots & \ddots & \vdots \\ k(x_n,\ x_1) & \cdots & k(x_n,\ x_n) \end{pmatrix}.$$

While a GP is defined on the entire real line, we only observe some finite set of realizations $\mathbf{x} = \{x_1, \ldots, x_n\}$, and corresponding $\mathbf{y} = \{y_1, \ldots, y_n\}$. $\mathbf{x}$ is commonly called the input and represents the location of the the process, i.e. observation $y_i = f(x_i)$. $\mathbf{y}$ is referred to as the output, and is the function evaluated at location $\mathbf{x}$. This allows for a generalization to a multivariate normal distribution: $f(\mathbf{x}) \sim \mathcal{MVN}\big(\mathbf{0},\ \mathbf{K}\big)$.

This is possible because marginalizing a Gaussian distribution is trivial: the resulting distribution is Gaussian and we can ignore the (x, y) pairs that are unobserved or missing. Here changing the kernel effects the shape of the function, effectively controlling the magnitude of which observations x_i and x_j are correlated. Formulating the problem this way we can see the kernel is our prior on the function space, and the (marginal) likelihood for the GP comes after conditioning on the realized points. As shown in the next section, the hyper-parameters are often estimated to maximize the GPs likelihood.

2.1.2 Likelihood

The previous section introduced the kernel function and how it relates to a prior on function space and how the hyper-parameters effect the correlation between the input $\mathbf{x}$ and outcome $\mathbf{y}$. Here the likelihood for a GP will be introduced and strategies for choosing the hyper-parameters will be shown. From the definition of a GP, $\mathbf{y} \sim \mathcal{N}(0, \mathbf{K})$, which will be shown formally below. First we let $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_*)$, where $\mathbf{K}_*$ is the covariance matrix constructed from the noiseless SE kernel:

$$k(x_i, x_j) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}|x_i - x_j|^2\right),$$

such that $\mathbf{K} \triangleq \mathbf{K}_* + \mathbf{I}\sigma_n^2$.

The likelihood for a GP is conditioned on the observed values to obtain a marginal likelihood, using ϕ to denote the normal density function:

$$p(\mathbf{f}\,|\mathbf{x}) = \underbrace{\phi(\mathbf{f}\,|\mathbf{0}, \mathbf{K}_*)}_{\text{function prior}}, \tag{2.5}$$

$$p(\mathbf{y}|\mathbf{f}) = \underbrace{\prod_{i=1}^{N} \phi(y_i|f_i, \sigma_n^2)}_{\text{likelihood}}. \tag{2.6}$$

We get the marginal for $\mathbf{y}$ by using Bayes' rule and integrating over $\mathbf{f}$:

$$p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{x})\ p(\mathbf{f}\,|\mathbf{x}) d\mathbf{f}. \tag{2.7}$$

Taking the log of (2.7) gives

$$\log p(\mathbf{y}_i|\mathbf{x}) = -\frac{1}{2}\Big\{\mathbf{y}_i\mathbf{K}^{-1}\mathbf{y}_i^\top + \log|\mathbf{K}| + N\log 2\pi\Big\}. \tag{2.8}$$

This likelihood can be broken down into three main components, the data fit term, model complexity term, and a constant term:

$$\log p(\mathbf{y}_i|\mathbf{x}) = -\frac{1}{2}\Big\{\underbrace{\mathbf{y}_i\mathbf{K}^{-1}\mathbf{y}_i^\top}_{\text{data fit}} + \underbrace{\log|\mathbf{K}|}_{\text{complexity}} + \underbrace{N\log 2\pi}_{\text{constant}}\Big\}. \tag{2.9}$$

The data fit and complexity component share an interesting tradeoff. For small length-scale values l, the model will fit the data well and the data fit component will be small. However, points will not be considered "near" each other, resulting in a high model complexity. Conversely, if l is large (suggesting no correlation between points), then the the complexity will be small but the data fit term will be large. This is because the SE kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ will converge to σ_f^2, turning $\mathbf{K}$ into a diagonal matrix. Because GPs have these inherent penalty terms for over-/under-fitting, cross validation methods are generally not needed.

2.1.3 Predictive Distribution

GPs are commonly used in supervised regression tasks for their ability to non-parametrically approximate complex functions and solve functional engineering problems (Bin and Wenlai, 2013). It is often of interest to infer the functions value outside of the paired training data $(\mathbf{x}, \mathbf{y})$. To do this, a predictive distribution can be constructed. Let $\mathbf{y}_* = \mathbf{f}(\mathbf{x}_*)$ be the unobserved outputs at locations $\mathbf{x}_*$ to be inferred. The distribution can easily be derived through probabilistic terms. From properties of joint Gaussians

$$\mathbf{f}(\mathbf{x}) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}(\mathbf{x}, \mathbf{x})\right), \tag{2.10}$$

$$\mathbf{f}(\mathbf{x}_*) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}(\mathbf{x}_*, \mathbf{x}_*)\right), \tag{2.11}$$

$$\begin{bmatrix} \mathbf{f}(\mathbf{x}) \\ \mathbf{f}(\mathbf{x}_*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{x}, \mathbf{x}) & \mathbf{K}(\mathbf{x}, \mathbf{x}_*) \\ \mathbf{K}(\mathbf{x}_*, \mathbf{x}) & \mathbf{K}(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix}\right). \tag{2.12}$$

Equation 2.12 is the joint distribution of the observed pairs $(\mathbf{x}, \mathbf{y})$ and unobserved $\mathbf{f}(\mathbf{x}_*)$ at location $\mathbf{x}_*$. The expected value for $\mathbf{f}(\mathbf{x}_*)$ can be derived using conditional properties which leads to

$$\hat{\mathbf{f}}(\mathbf{x}_*) \triangleq \mathbb{E}[\mathbf{f}(\mathbf{x}_*) | \mathbf{x}, \mathbf{y}, \mathbf{x}_*] = \mathbf{K}(\mathbf{x}_*, \mathbf{x})[\mathbf{K}(\mathbf{x}, \mathbf{x})]^{-1}\mathbf{y}, \tag{2.13}$$

full derivation is available in Rasmussen (2005). Equation (2.13), is then used to compute estimates for the function's value $\mathbf{f}(\mathbf{x}_*)$ at location $\mathbf{x}_*$.

2.2 Supervised, Unsupervised, and Semi-supervised Algorithms

Machine learning algorithms may be broken down into three general subcategories: unsupervised, supervised, and semi-supervised. Which subcategory to use depends on the data labelling and given problem. If the “label”, the variable we are interested in predicting, is present in the data then supervised learning is often used (Hastie *et al.*, 2001). For example, the output or y variable in linear regression acts as the “labelled” point. Parameters can then be estimated to produce accurate estimates for $\hat{y}$ using the independent variables x. Estimates $(\hat{y})$ can then be checked against the true values y, and adjustments can be made (supervision). The labelled data are often thought of as the training data and the in sample error, $\sum_{i=1}^{N} (\hat{y}_i - y_i)^2$, can be arbitrarily reduced. Most regression and classification problems can be thought of as supervised machine learning problems. Unsupervised learning refers to problems where the data does not have labels. Instead, unlike in supervised learning, we are interested in modelling the independent variables. Inference is made on variables without knowing the “true” value, such as group membership, where no confirmation of the right value exists. Clustering is an example of unsupervised learning. Semi-supervised learning refers to problems where some proportion of the data are labelled.

2.3 Model-Based Clustering

2.3.1 Clustering

Clustering is an unsupervised machine learning task which attempts to classify unlabelled data points into distinct groups. Commonly, clustering is defined as assigning data into groups such that data in the same cluster are more similar to each other than to data in a different cluster. Initially this definition seems intuitive; however, practically there are some problems. Namely, grouping each data point into its own cluster would satisfy this definition. Instead, McNicholas (2016a) provides a definition not based on similarity:

> A cluster is a unimodal component within an appropriate finite mixture model.

The use of the word appropriate here requires consideration of the data:

> It means that the model has the necessary flexibility, or parameterization, to fit the data

(McNicholas, 2016b). In either case, many methods have been developed to tackle this problem of unsupervised learning. Model-based refers to using probability distributions to model the clusters (as opposed to hierarchical, k-means, etc.).

2.3.2 Finite Mixture Model

The finite mixture model is a popular model-based technique that assumes data are generated in a sequential process: first a random draw chooses which cluster the data point is generated from with probability π_g. Next, a data point is drawn from distribution g. Formally the density can be written as:

$$f(y|\boldsymbol{\varphi}) = \sum_{g=1}^{G} \pi_g f_g(y|\boldsymbol{\theta}_g). \tag{2.14}$$

Here, the mixing proportion must satisfy the following constraints: $\pi_g > 0$ and $\sum_{g=1}^{G} \pi_g = 1$. The density parameters: $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_G)$ are the density specific parameters with $\boldsymbol{\varphi} = (\boldsymbol{\pi}, \boldsymbol{\theta})$, $\boldsymbol{\pi} = (\pi_1, ..., \pi_G)$. The likelihood is given by

$$\mathcal{L}(\boldsymbol{\varphi}) = \prod_{i=1}^{n} \sum_{g=1}^{G} \pi_g \ \phi(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g), \tag{2.15}$$

where $\boldsymbol{\varphi} = (\boldsymbol{\pi}, \boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_G, \boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_G)$. The cluster's soft probabilities ($\hat{z}_{ig} \in [0, 1]$) are converted into hard classifications, $\in \{0, 1\}$, by maximum a posteriori classification:

$$\text{MAP}(\hat{z}_{ig}) = \begin{cases} 1 & \text{if } g = \text{argmax}_h(\hat{z}_{ih}), \\ 0 & \text{otherwise.} \end{cases} \tag{2.16}$$

2.3.3 Expectation-Maximization

Model-based clustering requires estimating the unknown model parameters from the mixture density in (2.14). The expectation-maximization (EM) algorithm provides a good starting point for this problem. The two-step algorithm, introduced by Dempster *et al.* (1977) first starts by computing the expectation of the complete-data log-likelihood — which often amounts to taking the expectation of sufficient statistics of the latent variables conditioned on the data — then maximizes the expectation of the complete-data log-likelihood. Consider a Gaussian model-based clustering complete-data likelihood, denoted by $\mathcal{L}_c$, where the latent variable that denotes group membership $\mathbf{z}_i = (z_{i1}, \ldots, z_{iG})$ has been added:

$$\mathcal{L}_c(\boldsymbol{\varphi}) = \prod_{i=1}^{n} \sum_{g=1}^{G} [\pi_g \ \phi(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)]^{z_{ig}}, \tag{2.17}$$

where $z_{ig} = 1$ if observation i belongs to cluster g, and $z_{ig} = 0$ otherwise. Now, if $\mathbf{z}_1, \ldots, \mathbf{z}_n$ were known, then $\mathcal{L}_c$ could easily be maximized by splitting the data into their respective groups and MLE estimates or some continuous optimizer could be used. However, because $\mathbf{z}_1, \ldots, \mathbf{z}_n$ is unknown the EM algorithm can be used to handle the missing variable $\mathbf{z}_i$. In the first step ("expectation" step), the expected value of each z_{ig} is calculated:

$$\hat{z}_{ig} := \mathbb{E}[z_{ig} |\ \mathbf{x}_i] = \frac{\hat{\pi}_g \phi(\mathbf{x}_i \mid \hat{\boldsymbol{\mu}}_g, \hat{\boldsymbol{\Sigma}}_g)}{\sum_{h=1}^{G} \hat{\pi}_h \phi(\mathbf{x}_i \mid \hat{\boldsymbol{\mu}}_h, \hat{\boldsymbol{\Sigma}}_h)}. \tag{2.18}$$

Next, in the "maximization" step, the parameters are updated. This amounts to estimating the covariance matrix and mean vector for a Gaussian mixture model. As McNicholas (2016a) illustrates, the updates are weighted MLE estimates of $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$,

$$\hat{\boldsymbol{\mu}}_g = \frac{1}{n_g} \sum_{i=1}^{n} \hat{z}_{ig} \mathbf{x}_i, \tag{2.19}$$

$$\hat{\boldsymbol{\Sigma}}_g = \frac{1}{n_g} \sum_{i=1}^{n} \hat{z}_{ig} (\mathbf{x_i} - \hat{\boldsymbol{\mu}}_g)(\mathbf{x_i} - \hat{\boldsymbol{\mu}}_g)^\top, \tag{2.20}$$

where $n_g = \sum_{i=1}^{n} \hat{z}_{ig}$.

2.4 Gradient Based Optimization

Commonly, in machine learning tasks maxima and minima of some functions need to be found. When a function's gradient can be found in closed form and is unconstrained, an approach called gradient ascent is often used (Murphy, 2012). Gradient ascent is an algorithm that searches for a function's local maximum. In each iteration, the algorithm moves closer to the maximum by taking steps along the function's gradient. Let f be a function of θ, and let g be its gradient. Then, gradient ascent updates the move along θ by

$$\theta_{i+1} = \theta_i + \lambda \mathbf{g}_i, \tag{2.21}$$

where λ is the "step size" or "learning rate". It controls how big of a step the update will take. Choosing the correct step size λ effects how efficiently the method will converge. If λ is too small, then it will take many iterations and will be slow to converge as shown in Murphy (2012). If the step size is too large, then the method might diverge and not find a solution. The algorithm stops when it has found a sufficiently close value to the maximum (when the gradient approaches zero, a local optimum).

An illustration of gradient ascent, on a convex likelihood, is given in Figure 2.4.

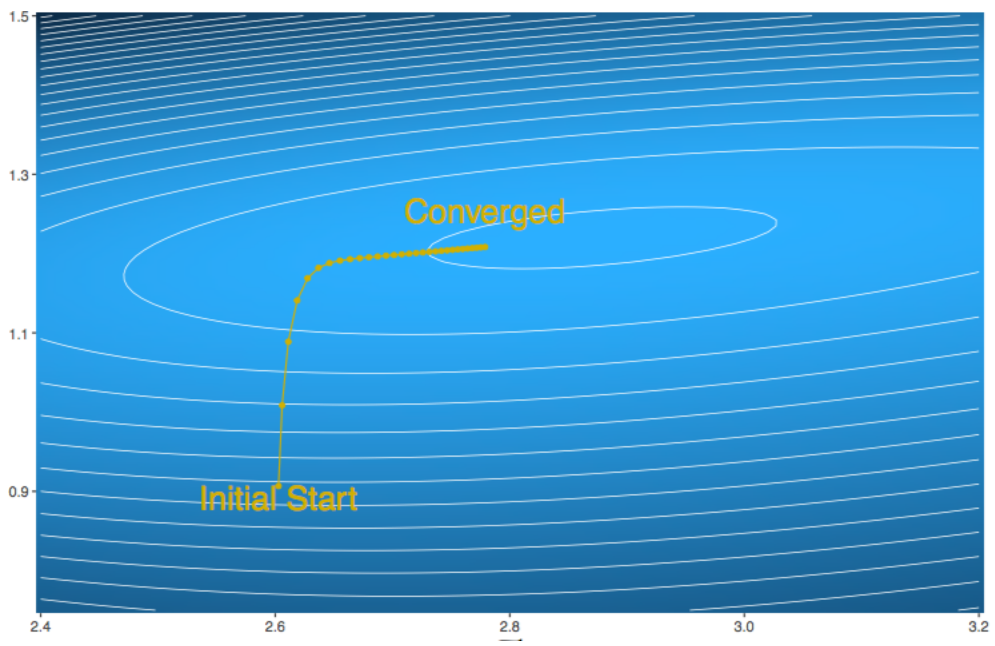

Figure 2.4: Illustration of gradient ascent on a convex likelihood function.

Chapter 3

Methods

3.1 Model

From the previous section we saw that the log-likelihood for a GP with the observed output vector $\mathbf{y}$ and corresponding input vector $\mathbf{x}$ was distributed according to a multivariate Gaussian (Equation 2.8). When clustering GPs, the goal will be to find clusters that contain processes which have similiar paths. The meaning of similar path refers to how close two processes values are but also how similar their shape is to one another (smooth, wiggly, etc.). Now let us define the notation used for the model, the ith GP will have the output vector $\mathbf{y}_i$ and input vector $\mathbf{x}_i$, and

$$\log p(\mathbf{y}_i|\boldsymbol{\theta}_i, \mathbf{x}_i) = -\frac{1}{2}\Big\{\mathbf{y}_i\mathbf{K}^{-1}\mathbf{y}_i^{\intercal} + \log|\mathbf{K}| + N\log 2\pi\Big\}. \tag{3.22}$$

This will be the likelihood used in the finite mixture model shown next:

$$p(\mathbf{y}_i|\boldsymbol{\theta}, \mathbf{x}_i) = \sum_{g=1}^{G} \pi_g p_g(\mathbf{y}_i|\boldsymbol{\theta}_g, \mathbf{x}_i), \tag{3.23}$$

where $\boldsymbol{\theta}_g = \{l_g, \sigma_{fg}, \sigma_{ng}\}$ is the set of hyper-parameters for the gth cluster, π_g is the mixing proportion for cluster g, $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_G)$, and $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_G)$ is a vector of G kernel hyper-parameters sets. Because the likelihood of a GP is a Gaussian distribution with covariance matrix $\mathbf{K}$, the complete-data likelihood is given by

$$\mathcal{L}_c(\boldsymbol{\varphi}) = \prod_{i=1}^{n} \sum_{g=1}^{G} [\pi_g \phi(\mathbf{y}_i\ |\mathbf{0}, \mathbf{K}_g)]^{z_{ig}}, \tag{3.24}$$

where $\mathbf{K}_g$ is the covariance matrix corresponding to cluster g and ϕ is the Gaussian density function. A SE covariance kernel is used as the prior on the function space

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}|\mathbf{x}_i - \mathbf{x}_j|^2\right) + \sigma_n^2 \delta_{ij}. \tag{3.25}$$

This means that observations are not perfectly interpolated from the GP; instead, they are corrupted by i.i.d. noise σ_n^2. The goal is to recover the G sets of kernel hyper-parameters $\boldsymbol{\theta_g} = (l_g, \sigma_{fg}^2, \sigma_{ng}^2)$ and the mixing parameters $\boldsymbol{\pi} = (\pi_1, ..., \pi_G)$ to estimate the latent variables $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$.

3.2 Parameter Estimation

3.2.1 GP Kernel Hyperparameters

The first step is to estimate each GP's kernel hyperparameters. The set of kernel hyper-parameters for GP i is denoted by: $\mathbf{\Theta}_i = \{l_i,\ \sigma_{fi},\ \text{and}\ \sigma_{ni}\}$. In this step, the maximized kernel hyper-parameters for each GP — $l_i^{\max}$, ${\sigma_{fi}}^{\max}$, and ${\sigma_{ni}}^{\max}$ — are estimated. To find these maximized hyper-parameters, a MLE solution is found using gradient ascent, starting with the log-likelihood i.e.,

$$\log p(\mathbf{y}_i|\mathbf{x}, \mathbf{\Theta}_i) = -\frac{1}{2}\left\{\mathbf{y}_i\mathbf{K}^{-1}\mathbf{y}_i^\top + \log|\mathbf{K}| + N\log 2\pi\right\}. \tag{3.26}$$

The derivative is then taken w.r.t. to the kernel hyper-parameters

$$\begin{aligned}\frac{\partial}{\partial\mathbf{\Theta}_i}\log p(\mathbf{y}|\mathbf{x}, \mathbf{\Theta}_i) &= \frac{1}{2}\mathbf{y}^\top\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial\mathbf{\Theta}_i}\mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial\mathbf{\Theta}_i}\right)\\ &= \frac{1}{2}\mathrm{tr}\left(\left[\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \mathbf{K}^{-1}\right]\frac{\partial\mathbf{K}}{\partial\mathbf{\Theta}_i}\right),\end{aligned} \tag{3.27}$$

where $\boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{y}$. The partial derivatives for l_i, σ_{fi}, and σ_{ni} are calculated from the first derivatives of the kernel function:

$$\frac{\partial \mathbf{K}}{\partial l_i} = \sigma_{fi}^2 \exp\Big\{ - \frac{1}{2l_i^2}(x_i - x_j)^2 \Big\}(x_i - x_j)^2 l_i^{-3}, \tag{3.28}$$

$$\frac{\partial \mathbf{K}}{\partial \sigma_{fi}} = 2\sigma_{fi} \exp\Big\{ - \frac{1}{2l^2}(x_i - x_j)^2 \Big\}(x_i - x_j)^2, \tag{3.29}$$

$$\frac{\partial \mathbf{K}}{\partial \sigma_{ni}} = \begin{cases} 2\sigma_{ni} & \text{if } x_i = x_j, \\ 0 & \text{otherwise.} \end{cases} \tag{3.30}$$

After finding the gradient for the likelihood, a gradient ascent algorithm is used to find a sufficiently close solution. This algorithm is given by repeating the following until a convergence criterion is attained:

$$\begin{aligned} l_i^{\text{update}} &:= l_i^{\text{old}} + \lambda \, \frac{\partial}{\partial l_i} \log p(\mathbf{y}_i|\mathbf{x}, l_i^{\text{old}}) \\ \sigma_{fi}^{\text{update}} &:= \sigma_{fi}^{\text{old}} + \lambda \, \frac{\partial}{\partial \sigma_{fi}} \log p(\mathbf{y}_i|\mathbf{x}, \sigma_{fi}^{\text{old}}) \\ \sigma_{ni}^{\text{update}} &:= \sigma_{ni}^{\text{old}} + \lambda \, \frac{\partial}{\partial \sigma_{ni}} \log p(\mathbf{y}_i|\mathbf{x}, \sigma_{ni}^{\text{old}}). \end{aligned} \tag{3.31}$$

After maximizing the kernel hyper-parameters, we have

$$\hat{\boldsymbol{\Theta}} = \{\hat{\boldsymbol{\Theta}}_1, \hat{\boldsymbol{\Theta}}_2, \ldots, \hat{\boldsymbol{\Theta}}_N\},$$

where $\hat{\boldsymbol{\Theta}}_1 = \{{l_1}^{\max}, {\sigma_{f1}}^{\max}, {\sigma_{n1}}^{\max}\}$, $\hat{\boldsymbol{\Theta}}_1$ is the maximized kernel hyper-parameters for the first GP, $\hat{\boldsymbol{\Theta}}_2$ is the hyper-parameters for GP 2, and so on.

3.2.2 Cluster Parameters

The model seeks to cluster the processes and make inferences on the latent variables. A modified EM approach is used. First, the mixing proportion and cluster hyper-parameters are initialized randomly:

$$\begin{aligned}
\boldsymbol{\pi} &\leftarrow \text{Initialize;}\\
\boldsymbol{l} &\leftarrow \text{Initialize;}\\
\boldsymbol{\sigma}_f &\leftarrow \text{Initialize;}\\
\boldsymbol{\sigma}_n &\leftarrow \text{Initialize;}
\end{aligned}$$

Where $\boldsymbol{l} = \{l_1, l_2, \dots, l_G\}, \boldsymbol{\sigma_f} = \{\sigma_{f1}, \sigma_{f2}, \dots, \sigma_{fG}\}, \boldsymbol{\sigma_n} = \{\sigma_{n1}, \sigma_{n2}, \dots, \sigma_{nG}\}$. Next, each GPs $(\mathcal{GP}_1, \dots, \mathcal{GP}_N)$ responsibilities are calculated for each of the G clusters to get $N \times G$ responsibilities

$$\hat{r}_{ig} = \frac{\pi_g \ \phi(\mathbf{y}_i|\mathbf{0}, \mathbf{K}_g)}{\sum_{h=1}^{G} \ \pi_h \ \phi(\mathbf{y}_i|\mathbf{0}, \mathbf{K}_h)}, \tag{3.32}$$

where $\hat{r}_{ig}$ represents the responsibility, or conditional expected value, of the ith process belonging to the gth cluster, $\hat{r}_{ig} \triangleq \hat{z}_{ig}$. After the responsibilities are calculated, the mixing proportions $\boldsymbol{\pi}$ are conditionally maximized on these responsibilities:

$$m_g = \sum_{i=1}^{N} r_{ig}, \tag{3.33}$$

$$\pi_g = \frac{m_g}{m}, \tag{3.34}$$

where m_g is the responsibility for cluster g and π_g is the gth mixing proportion with $m = \sum_{g=1}^{G} m_g$. The kernel hyper-parameters specific for each cluster: l_g, σ_{fg}, and σ_{ng} are then updated, where l_g is the length-scale parameter for cluster g, σ_{fg} is the height parameter for cluster g, and σ_{ng} is the noise parameter for cluster g:

$$l_g = \frac{1}{m_g} \sum_{i=1}^{N} \hat{r}_{ig} \; l_i^{\max}, \tag{3.35}$$

$$\sigma_{fg} = \frac{1}{m_g} \sum_{i=1}^{N} \hat{r}_{ig} \; \sigma_{fi}^{\max}, \tag{3.36}$$

$$\sigma_{ng} = \frac{1}{m_g} \sum_{i=1}^{N} \hat{r}_{ig} \; \sigma_{ni}^{\max}. \tag{3.37}$$

This is done by weighting the maximized hyper-parameters — $\sigma_{fi}^{\max}$, $l_i^{\max}$, and $\sigma_{ni}^{\max}$ — by their respective cluster responsibility $\hat{r}_{ig}$.

This scheme, of calculating the responsibilities then updating the cluster parameters, is repeated until some convergence criterion is met. In this case when the change in likelihood (3.24) between iterations becomes small, i.e., until

$$|\mathcal{L}_c(\boldsymbol{\varphi}_{i-1})) - \mathcal{L}_c(\boldsymbol{\varphi}_i)| < \epsilon.$$

3.3 Numerical Issues

At each iteration of gradient ascent the GP's likelihood gradient needs to be computed:

$$\frac{\partial}{\partial \boldsymbol{\Theta}_i} \log p(\mathbf{y}_i|\mathbf{x}, \boldsymbol{\Theta}_i) = \frac{1}{2}\mathrm{tr}\Big(\Big[\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \mathbf{K}^{-1}\Big]\frac{\partial \mathbf{K}}{\partial \boldsymbol{\Theta}_i}\Big). \tag{3.38}$$

This operation requires inverting a $t \times t$ matrix $\mathbf{K}^{-1}$. Inverting large matrices are notoriously computationally unstable. This is especially true when matrices are not full rank (or sufficiently close) and eigenvalues become very large or very small. One solution is to first decompose the matrix into lower triangular form:

$$\mathbf{K} = \mathbf{L}\mathbf{L}^\top. \tag{3.39}$$

The lower triangle $\mathbf{L}$ is then inverted:

$$\mathbf{K}^{-1} = (\mathbf{L}^{-1})^\top \mathbf{L}^{-1}. \tag{3.40}$$

The `R` (R Core Team, 2017) package used is `FastGP` (Gopalan and Bornn, 2016), which implements the package `RcppEigen` to invert the decomposed matrix.

Chapter 4

Analyses

4.1 Overview

This section will first look at two cases of simulated data. The hyper-parameters $\boldsymbol{\theta} = \{\boldsymbol{l}, \boldsymbol{\sigma_f}\}$ and the mixing proportions $\boldsymbol{\pi}$ will vary on the simulated sets. The method from Chapter 3 will then be applied to recover the hyper-parameters and classify each GP into their respective groups. For the two simulation studies, noiseless squared exponential covariance functions will be used. Meaning, a perfectly interpolated, noiseless process is observed for the simulation. Finally, rainfall data from the coastal region of Tofino, British Columbia will be modelled and analyzed. The rainfall analyses will use a squared exponential covariance function, with additive measurement noise assumed to be present. Thus, for the real data analysis, $\boldsymbol{\theta} = \{\boldsymbol{l}, \boldsymbol{\sigma}_f, \boldsymbol{\sigma}_n\}$ will be estimated and modelled. The observed outputs $\mathbf{y}$ will be (incorrectly) connected by lines for illustrative purposes.

4.2 Simulated Data

4.2.1 Simulation I

The first simulation starts with generating 30 GPs. The processes are generated on the interval $[0, 10]$ with $T = 7$ evenly spaced realizations, i.e., each process has seven values spread evenly on the interval. In all, 10 of the 30 GPs are generated from a multivariate normal distribution (using the R package: `mvrnorm`) where the covariance matrix was constructed using an SE covariance kernel with hyper-parameters $l = 1$ and $\sigma_f = 3$. The remaining 20 were generated similarly but with a covariance matrix constructed with hyper-parameters $l = 3$ and $\sigma_f = 3$.

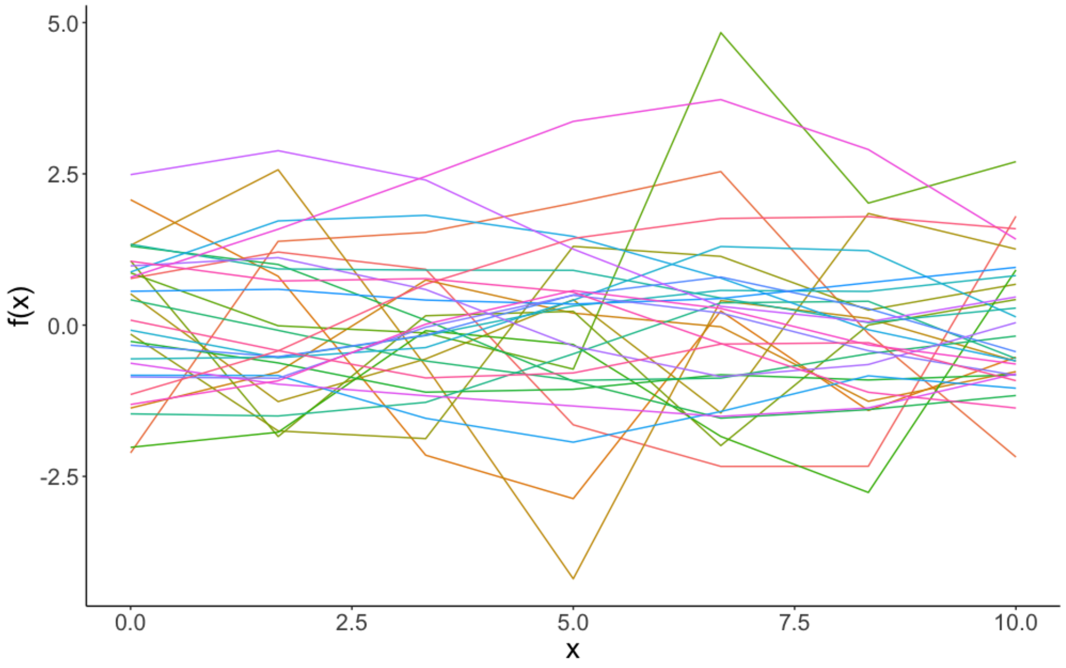

Figure 4.1: The 30 GPs from Simulation I, generated from two different kernel hyper-parameter settings.

After running the algorithm described in Chapter 3, estimates for the set of hyper-parameters and mixing proportion were recorded (Table 4.1). The mixing proportion is easily identified and accurately estimated. Using the MAP classification, the algorithm was able to correctly classify each process. Table 4.1 gives the mean parameter estimates and standard errors. This was done by randomly (initializing the parameters from a random uniform draw) starting the algorithm 10 times and calculating the mean and standard error from these 10 restarts.

Table 4.1: Simulation I, mean value for recovered hyper-parameters with standard error. Calculated by randomly restarting the algorithm 10 times.

Parameter	Truth	Mean Estimate	Standard Error
π_1	0.33	0.33	0
π_2	0.67	0.67	0
l_1	1	1.22	0.005
l_2	3	3.08	0.02
σ_{f1}	3	2.09	0.028
σ_{f2}	1	1.32	0.049

Once the processes are coloured by their MAP classification (Figure 4.1), one can visually see the difference between the two process clusters. The processes ($g = 2$, blue) with the larger length-scale $l = 3$ are smoother compared to those generated from the process with length-scale $l = 1$. The length-scale parameter l was also readily recovered in this scenario, producing similar estimates to the true hyper-parameter. The hyper-parameter σ_{f1}, which as a reminder controls the functions variance (in y), is not near the true parameter value. One reason for this could because this cluster has a comparatively small length-scale $l = 1$, which models the relative correlation

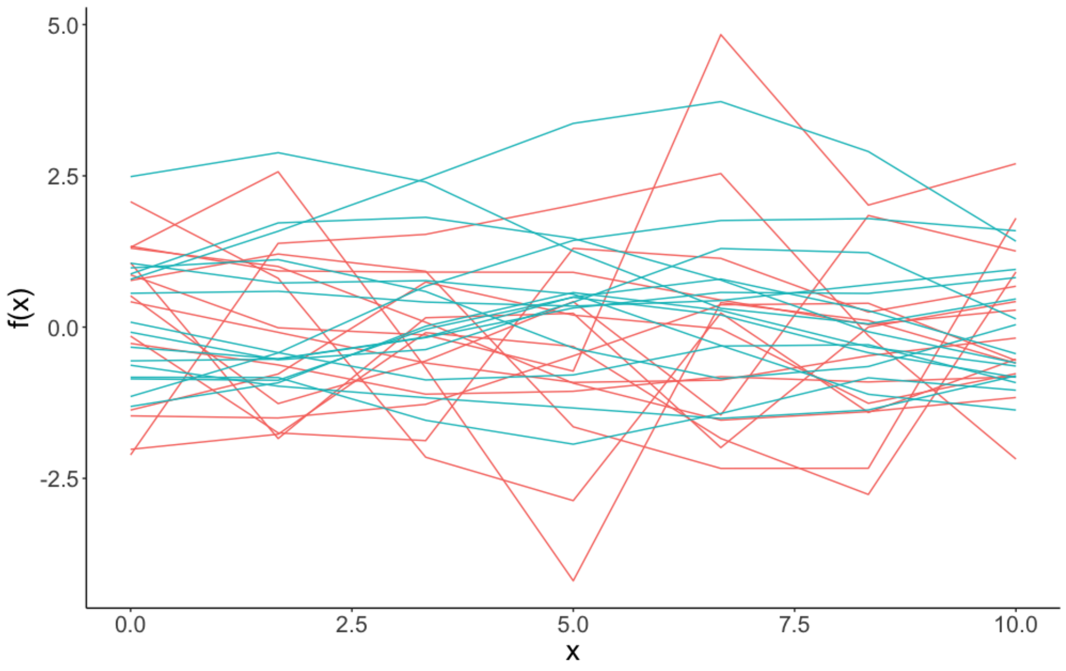

Figure 4.2: The 30 GPs from Simulation I, coloured by MAP classification. Red lines are cluster $g = 1$ and blue lines are cluster $g = 2$.

between the points. If this is true, preference to model the processes height variation is modelled more precisely with l than with σ_f.

4.2.2 Simulation II

The second simulation was done by first generating 20 GPs. Ten were generated from an SE covariance kernel with hyper-parameters $l = 1$ and $\sigma_f = 1$. The remaining 10 GPs were generated from a covariance kernel with hyper-parameters $l = 2$ and $\sigma_f = 2$. Similarly to Simulation I, the GPs were generated first by constructing the covariance matrix, then by generating random samples using the R package `mvrnorm`. In all, $T = 9$ equally spaced observed values were recorded for each GP (Figure 4.3).

Based on the plot in Figure 4.3, there seem to be no clear distinction or natural

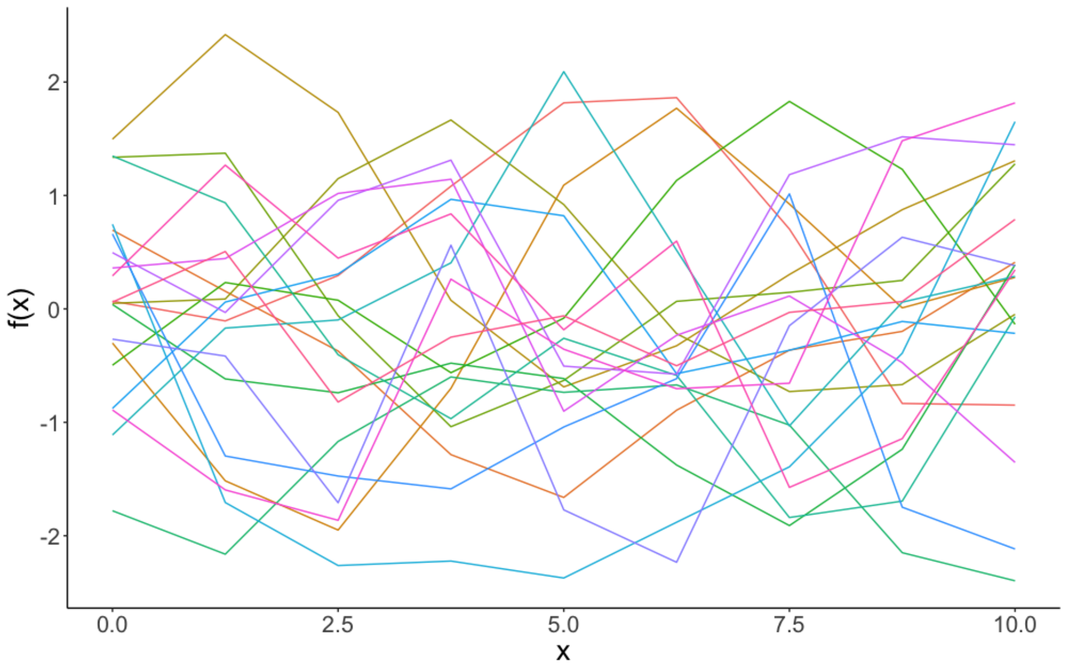

Figure 4.3: The 20 GPs from Simulation II, generated from two different kernel hyper-parameter settings.

groups of processes. After coloring the processes by their (correct) classifications (Figure 4.4), there is still ambiguity about the two groups separation.

Again, the method accurately recovers the mixing parameter and length-scale (Table 4.2). However, for cluster 1, the length-scale l is slightly overestimated and the method inflates σ_f to account for the height variance in y. The parameter estimates were calculated by randomly restarting the algorithm 10 times and using the mean estimate. The processes also look very similar between groups, and this solution might be unconvincing if true group labels were unknown.

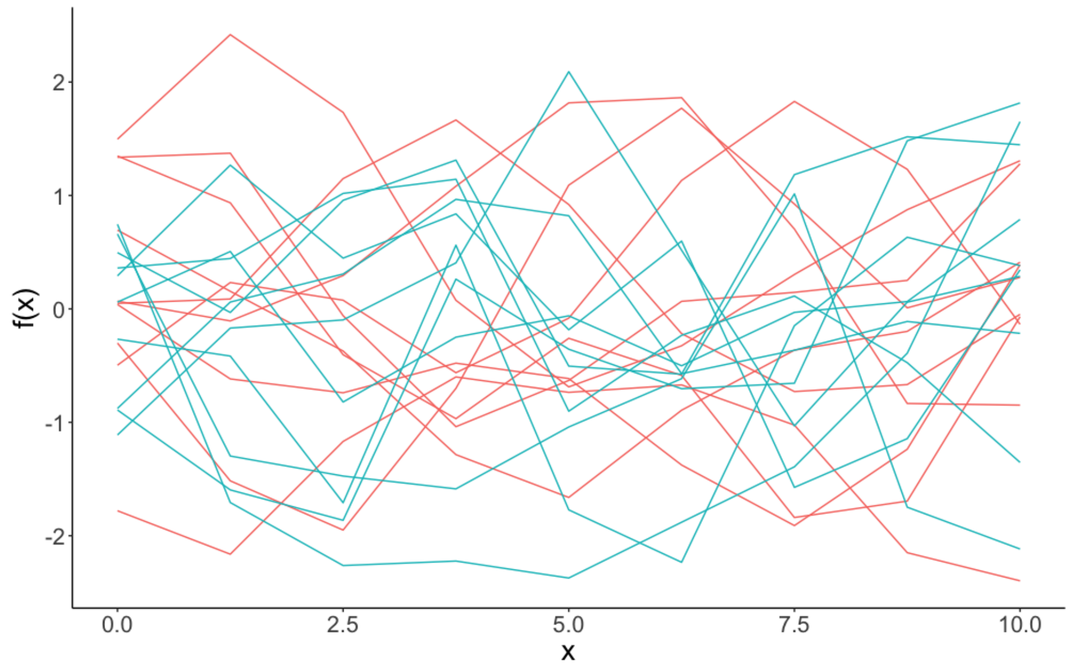

Figure 4.4: The 20 GPs from Simulation II, classified by their MAP. Red lines are cluster 1, blue lines are cluster 2.

Table 4.2: Simulation II, mean value for recovered hyper-parameters with standard error. Calculated by randomly restarting the algorithm 10 times.

Parameter	Truth	Mean Estimate	Standard Error
π_1	0.5	0.52	0.002
π_2	0.5	0.48	0.002
l_1	1	1.33	0.016
l_2	2	2.06	0.008
$\sigma_{f\ 1}$	1	1.82	0.034
$\sigma_{f\ 2}$	2	1.84	0.031

4.3 Tofino Rainfall Data

4.3.1 El Niño and La Niña

El Niño and La Niña are irregular recurring weather patterns lasting roughly a year in the Pacific Ocean and coastal region. El Niño is characterized by warming ocean temperatures and is officially classified using the Oceanic Niño Index (ONI), an index that measures positive increases in ocean temperature over a three month moving average (NOAA, 2018a). La Niña, the extreme opposite of El Niño, is characterized by cooling ocean temperatures and is also classified using ONI. A period of El Niño is often followed by a La Niña year. Apart from the significant impact these weather events have on precipitation and temperature, these events can drastically alter food prices and cause forest fires, leading to lasting economic and political consequences (NOAA, 2018b).

4.3.2 Analysis

This section will look at historical monthly precipitation data for the British Columbia (B.C.) coastal region of Tofino. Data is recorded by the Government of Canada (Government of Canada, 2007) and collected from the weather station Tofino A. Total monthly precipitation was recorded from January 1990 to December 2000 (Figure 4.5; Table 4.3). The ten years will be treated as independent GPs.

First, the data are centered and scaled such that the mean is 0 and standard deviation is 1. The maximized hyper-parameters are fitted as shown in Figure 4.6. There seem to be two groups emerging, six with a smaller length-scale (years 1991, 1992, 1993, 1996, 1997, 2000) and four years with a larger length-scale (years 1994,1995,1997,

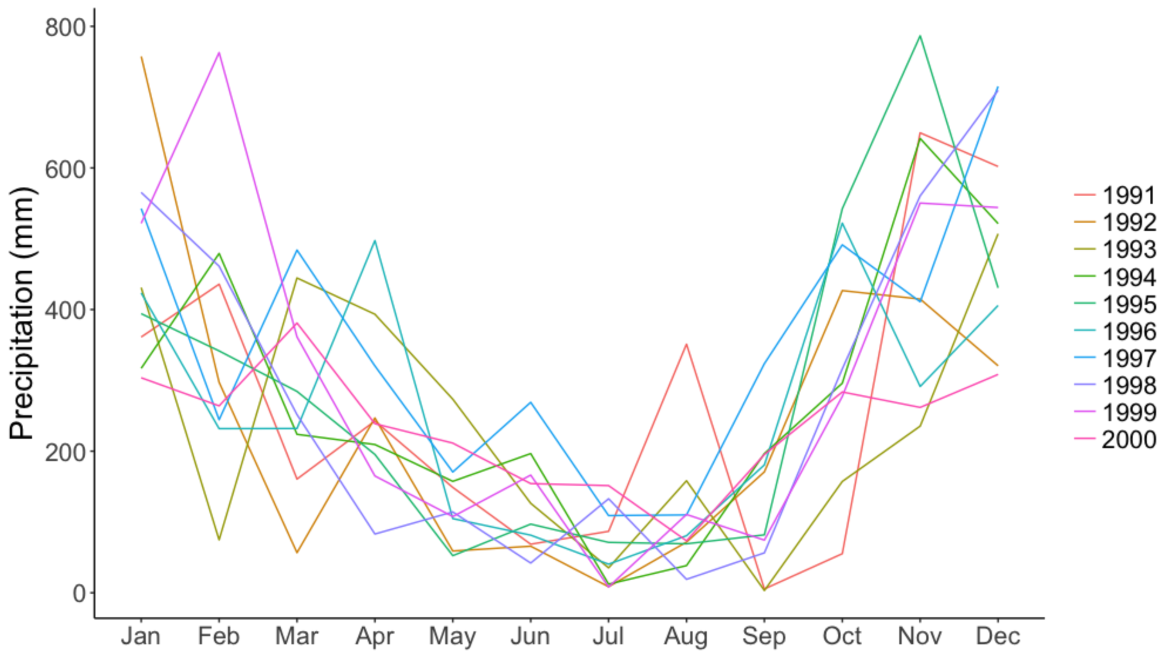

Figure 4.5: Monthly precipitation for the Tofino coastal region of B.C., Canada. The points are (incorrectly) connected between months for illustrative purposes, there are 12 measurements per year.

1998), where a smaller length-scale suggests time points are less correlated relative to time (month). In this case, the years with a smaller length-scale suggest monthly rainfall are less correlated month to month than those with a larger length-scale.

After clustering, the ten years are grouped by their MAP classification into two groups. Group two (“irregular”, $\pi_2 = 0.16$) contains the years 1995 and 1998, and the rest are classified into group one (“regular”, $\pi_1 = 0.84$). This is illustrated in Figure 4.7, where the years are coloured by their MAP classifications.

Cluster two contains only years where there was a change from El Niño to La Niña, i.e. the year started with warm enough ocean temperatures to classify it as an El Niño period, and by the end of the year the ocean had cooled enough to be classified as La Niña. The two years (1995, 1998) picked from the classification as belonging

Table 4.3: Year with their MAP classification, related weather events, and their cluster soft classification.

Year	Cluster	Special Event	Certainty/ P(cluster)
1991	1	El Niño	0.99
1992	1	El Niño	0.84
1993	1	None	1
1994	1	El Niño	0.93
1995	2	El Niño to La Niña	0.71
1996	1	La Niña	1
1997	1	El Niño	0.99
1998	2	El Niño to La Niña	0.80
1999	1	La Niña	0.73
2000	1	La Niña	1

to a different cluster correspond to years where rainfall patterns had comparatively larger length-scale parameters. Some years (1992, 1994, and 1999) had comparatively neutral optimized length-scale parameters ($0.72 \leq l \leq 0.84$), which is in the middle of the two clusters estimated length-scale parameters $\boldsymbol{l} = \{l_1 = 0.7,\ l_2 = 1.02\}$. The two clusters shared similar height variation (σ_f) and noise (σ_n) parameters as shown in Table 4.4. The parameter estimates shown represent the mean value and standard error after running the algorithm 10 times, randomly starting the parameters, $\boldsymbol{\theta} = \{\boldsymbol{l}, \boldsymbol{\sigma}_f, \boldsymbol{\sigma}_n\}$, with different values.

In Figure 4.8 two years are plotted side by side along with their predictive distribution $\mathbf{f}_*$ from Equation 2.13, the estimated function for unobserved values between data points. The blue line (1998, cluster two) is comparatively smoother, the length-scale is larger and changes in output (rainfall) can be accounted for by measurement noise. This is shown as the blue line passes nearby (but not through) some red crosses. Oppositely, the green line (1993, cluster one) is characterized with a smaller length-scale l and smaller noise σ_n. With a smaller l, the function has more flexibility to

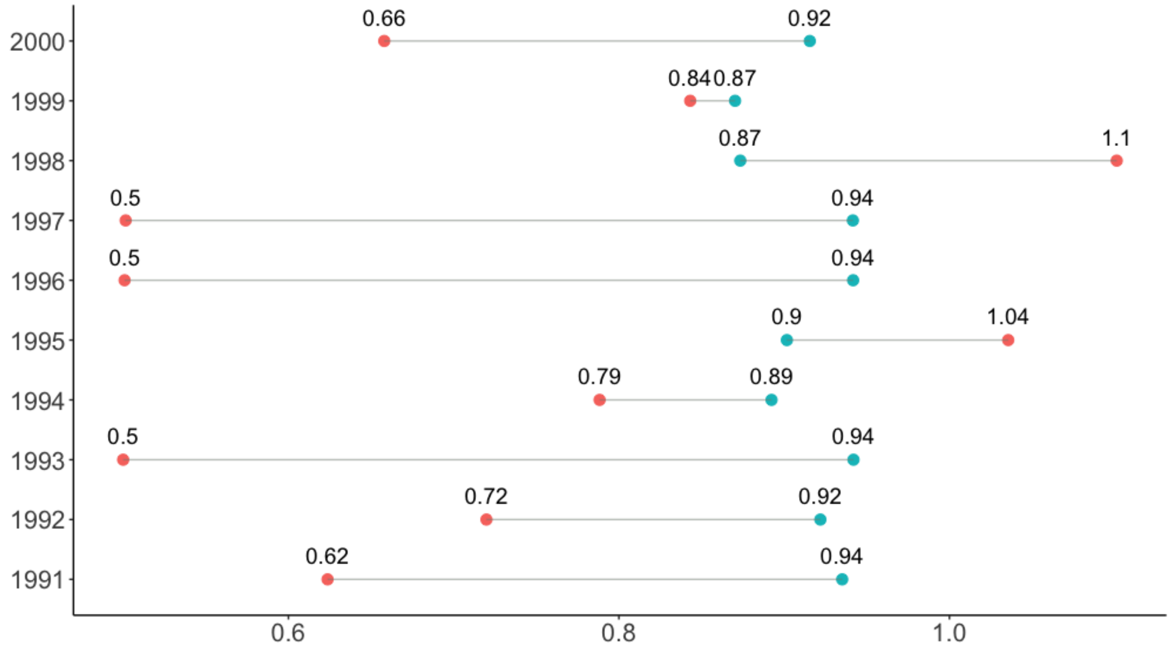

Figure 4.6: Optimized hyper-parameters for the ten years of precipitation data. Red dots are the l parameter, blue dots are σ_f. The years 1995 and 1998 were classified into a separate cluster, they're the only years in which σ_f is less than l.

model the sharp increases/decreases with little noise and, therefore, passes through the points very precisely.

From further consideration of the estimated cluster parameters in Table 4.4, cluster two's years tend towards a larger length-scale compared to cluster one. This suggests years where El Niño changes to La Niña, rainfall patterns change more smoothly (i.e., are more correlated) across months as opposed to regular weather years.

Table 4.4: Clustering results, parameters recovered. These estimates are the mean and standard error calculated by running the algorithm 10 times from random start points. The clusters differed mainly with their length-scale parameter l. The irregular cluster, which only contained years where ocean temperatures changed from irregularly warm (El Niño) to irregularly cold (La Niña).

Parameter	Mean Estimate	Standard Error
π_1	0.84	0.011
π_2	0.16	0.011
l_1	0.7	0.002
l_2	1.02	0.008
σ_{f1}	0.93	0.001
σ_{f2}	0.89	0.001
σ_{n1}	0.043	0.001
σ_{n2}	0.055	0.001

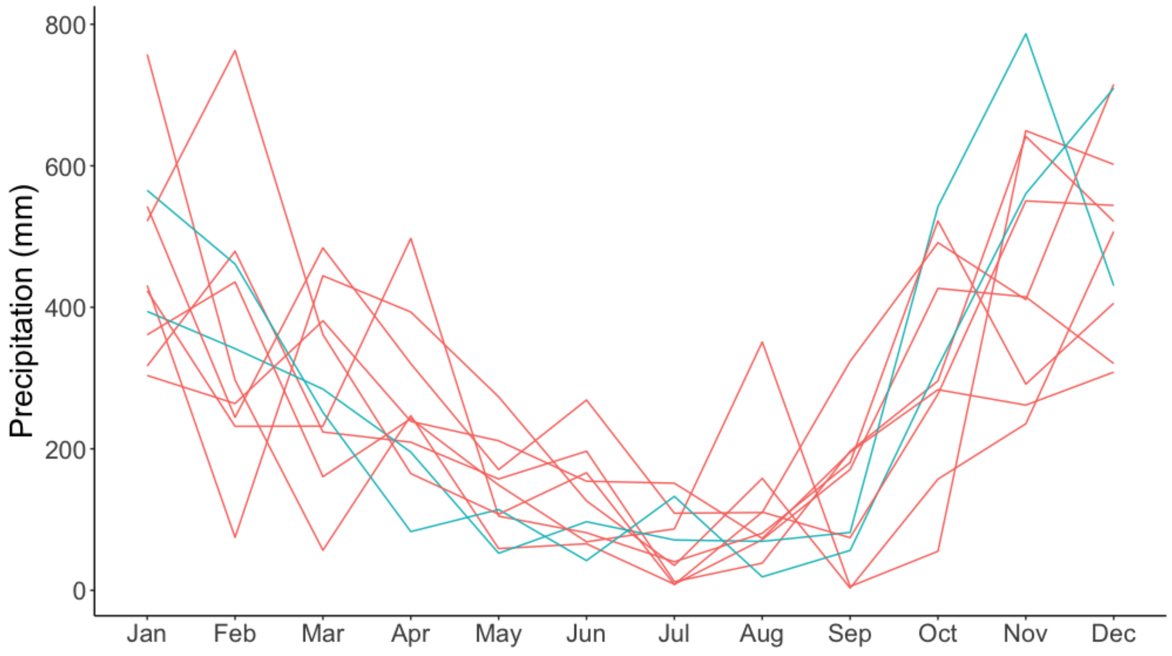

Figure 4.7: Tofino monthly precipitation coloured by their MAP classification.

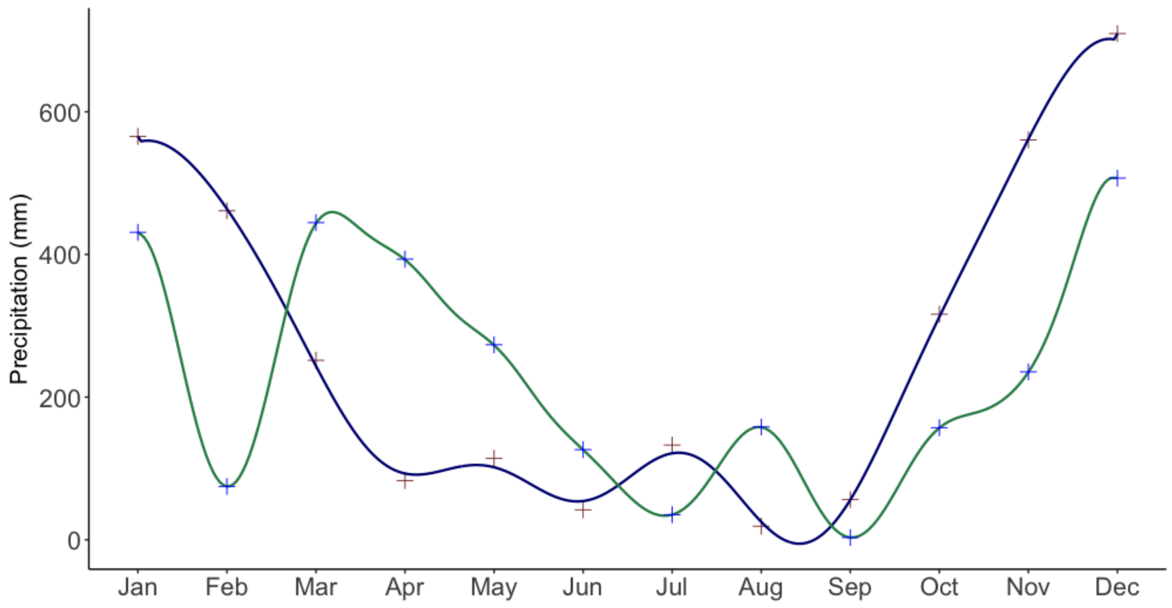

Figure 4.8: Two years of data plotted alongside their predictive distribution. The blue line (year 1998) is smoother than the green line (year 1993). Mainly because its characterized by a higher noise parameter (σ_n^2), which is able to account for change in height as measurement noise independent of the underlying function. Whereas the green line must move quicker (through a smaller length-scale) to fit the points.

Chapter 5

Conclusions

5.1 Discussion

This thesis presented a method for clustering functional data. First, the hyper-parameters that make up a GP where optimized through a gradient-based maximum likelihood optimizer. The EM algorithm for finite Gaussian mixture models was then modified. Instead of maximizing the standard covariance matrix, hyper-parameters for a kernel function that measures correlation in $\mathbf{x}$ were optimized. The covariance matrix was then constructed from the optimal kernel parameters. In this thesis G was considered known; however, future work should consider unknown number of groups using methods such as entropy. It is also noted that missing, or incomplete data can be handled by the model. Either by using the predictive distribution to impute the missing data or by ignoring the missing values. This is possible because the model makes inference on the underlying hyper-parameters of the kernel, and not the particular index set of the process.

Two simulation studies were performed. When GPs from different distributions had a large difference in their length-scale parameter l (i.e., 1 and 3), parameters were readily recovered. When GPs had similar length-scale parameters, l was recovered but σ_f tended to shrink towards a common estimate between both clusters.

The methods developed were then applied to rainfall data from the coastal region of Tofino, B.C. The method discovered two groups of years, one which contained "regular" years and the other "irregular" years. The irregular years consisted only of years that had both El Niño and La Niña events. These results suggest El Niño events can be classified based on their kernel hyper-parameters. The data were standardized to have a zero mean function, implying correlation between rainfall patterns month-to-month can discriminate El Niño events (as apposed to magnitude of rainfall).

Bibliography

Bin, S. and Wenlai, Y. (2013). Application of gaussian process regression to prediction of thermal comfort index. In *2013 IEEE 11th International Conference on Electronic Measurement Instruments*, volume 2, pages 958–961.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, **39**, 1–38.

Gopalan, G. and Bornn, L. (2016). *FastGP: Efficiently Using Gaussian Processes with Rcpp and RcppEigen.* R package version 1.2.

Government of Canada (2007). Monthly Precipitation Data, Tofino A. `http://climate.weather.gc.ca/climate_data/monthly_data_e.html`. Accessed 2018-08-05.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning.* Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

McNicholas, P. D. (2016a). *Mixture Model-Based Classification.* Chapman & Hall/CRC Press, Boca Raton.

McNicholas, P. D. (2016b). Model-based clustering. *Journal of Classification*, **33**, 331–373.

Murphy, K. P. (2012). *Machine Learning, A Probabilistic Approach*. MIT Press, Cambridge.

NOAA (2018a). Cold and warm episodes by season. `http://origin.cpc.ncep.noaa.gov/products/analysis_monitoring/ensostuff/ONI_v5.php`. Accessed 2018-08-15.

NOAA (2018b). El nino. `http://www.noaa.gov/resource-collections/el-nino`. Accessed 2018-08-16.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Rasmussen, C. E. (2005). *Gaussian Processes for Machine Learning*. MIT Press, Cambridge.

Wang, J.-L., Chiou, J.-M., and Muller, H.-G. (2016). Functional data analysis. *Annual Review of Statistics*, **3**, 257–295.