Rheinische
Friedrich-Wilhelms-Universität Bonn

Master thesis

Studying Dynamics of User Behavior. Heterocedastic Time Series Forecasting and Clustering of Inhomogeneous Poisson Process

Author:
Kostadin Cvejoski
Matr. Number: 2709674

First Examiner:
Prof. Dr. Christian Bauckhage
University of Bonn

Second Examiner:
Prof. Dr. Stefan Wrobel
University of Bonn

Submitted: August 24, 2016

Declaration of Authorship

I declare that the work presented here is original and the result of my own investigations. Formulations and ideas taken from other sources are cited as such. It has not been submitted, either in part or whole, for a degree at this or any other university.

Location, Date

Signature

Acknowledgements

First of all I would like to thank my thesis advisor Cesar Ali Ojeda Marin from the Fraunhofer-Institute for Intelligent Analysis and Informations System (IAIS). His door was always open for me and whenever I had a question he was always there offering me his support. I am also very grateful to Sifa Rafet for spending the time to read and discuss parts of my thesis. Without their passionate participation and input, the work could not have been successfully conducted.

I would like to also express my appreciation to my examiners: Prof. Dr. Christian Bauckhage and Prof. Dr. Stefan Wrobel. Thank you for giving me the opportunity to be a part of the Fraunhofer-Institute for Intelligent Analysis and Informations Systems research, and igniting my passion to dive deeper into the Machine Learning field.

Finally, I must express my very profound gratitude to my wife, for providing me with an inexhaustible support and continuous encouragement throughout my years of study and the whole process of researching and writing. This accomplishment would not have been possible without her. Thank you.

Abstract

Complex time series patterns are generated by the behavior of a large number of different users in so the called "question and answering" web platforms. This calls for flexible, accurate and descriptive techniques for studying the dynamics of such systems. In this study, we extend the Sparse Input Gaussian Process formalism, in order to incorporate functional description of the input dependent noise. Such procedure also provides a regularization method that improves the accuracy of the predictions. We compare our results with the results of the other Gaussian Process methods, and apply the methodology to time series from the questions and answer web site Stackoverflow.

For finding the common behavior between the users we propose the scale invariant Dynamic Piecewise Similarity measures and the K-PSC clustering algorithm for clustering time series in order to provide much more descriptive cluster centroids then the centroids from the K-Means clustering algorithm.

Contents

List of Figures

List of Algorithms

1. Introduction and Related Work

"Question answering" (QA) sites have gained considerable popularity over the last couple of years. In these Internet platforms users pose questions and answers to the public, in order to exchange knowledge. Yahoo Answers, Quora and the Stack Exchange family of sites provide platforms facilitating the formation of Internet communities that in turn provide natural and seamless ways for organizing and obtaining knowledge [Ada+08].

Dynamical aspects of such QA sites have been studied in different contexts so far. Previous works in this area include: studying causality aspects through quasi experimental designs [OTJ10]; user churn analysis through classification algorithms such as support vector machines or random forests [PAT14]; and predictions of the future value of question answer pairs, according to the initial activity of the question post [And+12].

In contrast to previous works, where long term activity of users are being predicted, we focus on time series analysis related to user-defined tags. This approach offers a possibility of a daily detailed analysis of users behavior. We concentrate on the QA site Stackoverflow in our work, because this platform has an already established reputation on the web, and boasts a community of over 5 million distinct active users, who have provided more than 18 million answers to more than 11 million questions. Thanks to the shear size of the corresponding data set, as well as the regular activity of the user base, we were able to mine temporal data in order to uncover defining aspects of the dynamics of the users behavior.

Our work of analyzing the temporal dynamics of the users is divided into two segments. The first segment is analyzing the collective users behavior related to a specific tag, through modeling changes in attention as a variability in the fluctuation of time series of occurrences of users defined tags, which can be categorized as a special case of *heterocedasticity* or input dependent noise. In the second segment we focus on finding common user posting behavior.

Due to the complexity of the user-system interaction (millions of people discuss thousands of topics), flexible and accurate models are required in order to guarantee a reliable analysis. The Bayesian setting and Gaussian Process (GP) framework [Ras06] have shown to provide an accurate and flexible tool for time series analysis in the recent years, and can be used to create accurate models. In particular, the possibility of incorporating error ranges, as well as different models trough the selection of different kernels, allows interpretability of the results.

We provide an extension of sparse input Gaussian Processes [SG12; SG05] which allow us to model functional dependence in the time variation of the fluctuations. In practical experiments, we study the top 10 different tags for the Stackoverflow data set over different years, spanning a data set of over 2.9 million questions. We find that our model outperforms the predictions made by the simple GP model under variable noise.

In particular, we discover weekly and seasonal periodicity patterns, as well as random behavior in monthly trends.

The goal of our second segment is to find common behavioral patterns among the users. This task is performed by clustering the time series of postings for each user, and cluster centroids are taken as representatives of common users behavioral patterns. Analyzing each user as time series of postings however is not possible, because the majority of the users (almost 94%) on an individual level show a posting behavior of less then ten postings in 2014, where the posting behavior of all the users together show a regular activity. This behavior may occur because of many reasons, but mainly because of the users limited working capacity. This leads to a sparsity of data, that makes it difficult to analyze the users behavior with a GP model, therefore we use a point process. Point processes are standard models when the objects of study are the number and repartition of otherwise identical points on a domain, usually time or space.

Similar work has been done for analyzing user behavior [Mal+08], where a Poisson Point Process (PPP) is used to explain the heavy tails in e-mail communication. However, the number of users that were analyzed is much smaller than we want to analyze, due to the computational complexity of the PPP. Recent development of a fast approximation [SR14] of the PPP makes it possible to apply this model on much bigger scale. In this approximated PPP model the intensity function is modeled by our extended sparse GP, which is learned by using Monte Carlo sampling.

For clustering the intensity functions of every PPP we have developed a new scale invariant similarity measure, called Dynamic Piecewise Time similarity measure, and K-PSC clustering algorithm that is a generalization of the K-SC [YL11]. Our algorithm is able to find more descriptive cluster centroids than the normal K-means clustering algorithm. Also, we do not find any seasonal behavior of the individual users like we have found in the collective behavior.

The theory behind the models used in this work is presented in the next chapter. We formally introduce the Gaussian Process framework and provide details regarding our extensions towards variable noise models. In section 2.4 we present brief introduction of Poisson Point process. Then, in section 2.5, we introduce the Dynamic Piecewise Time similarity measure and the K-PSC clustering algorithm. In chapter 3 we show an analysis of the periodicity of the time series of tag activity in the Stackoverflow data set. Next, we compare our prediction results with those of other models, and also discuss the advantages of introducing functional dependencies on noise terms. In chapter 4 we present the common patterns of the user behavior, that we discovered using the Dynamic Piecewise Time similarity measure and the K-PSC clustering algorithm. Finally, we provide conclusions and directions for future work.

2. Theory

2.1. Gaussian Process

For modeling the collective temporal dynamics of the users with respect to a tag, can be done by building a model that describes that dynamics. Gaussian process is a Bayesian non-parametric method which is commonly used for regression tasks. The main advantages of this method are its non-parametric nature, the possibility of an interpretation of a model through flexible kernel selection, and the confidence interval (error bar) it provides with every prediction.

The non-parametric nature of this model comes from defining a prior probability distribution over the infinite space of functions. Working with a distribution over the infinite space of functions may seem difficult. The practice shows however, that for a finite training set it is enough just to consider the values of the function evaluated at the input points. In this way we work with a finite space. For a better understanding of the Gaussian process model, we will derive one type of a Gaussian process model from a linear regression example, by working in terms of a distributions over functions $f(\mathbf{x}, \mathbf{w})$. More detailed information for the Gaussian process can be found in [Ras06] and [Bis06].

2.1.1. Linear Regression and Linear Basis Function Model

The *linear regression* model is a linear combination of input variables with the parameters of the model

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_D x_D \tag{2.1}$$

where $\mathbf{x} = (x_1, \ldots, x_D)^T$ are the input variables and $\mathbf{w} = (w_1, \ldots, w_D)$ are the model parameters. Due to the linear combination of the input variables and the model parameters, this model shows significant limitations (poor results with non-linear problems). Extending the linear regression model to the non-linear class of problems is possible by considering linear combination with a fixed number of non-linear functions of the input variables with the model parameters

$$f(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{m=1}^{M-1} w_m \phi_m(\mathbf{x}) \tag{2.2}$$

where $\phi_i(\mathbf{x})$ are called *basis functions*, and M is the total number of parameters in the model. The w_0 is called *bias* parameter, and it is often convenient for it additional basis

function to be defined $\phi_0(\mathbf{x}) = 1$, therefore we have

$$f(\mathbf{x}, \mathbf{w}) = \sum_{m=0}^{M-1} w_m \phi_m(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) \tag{2.3}$$

where $\mathbf{w} = (w_0, \ldots, w_{M-1})^T$ and $\boldsymbol{\phi} = (\phi_0, \ldots, \phi_{M-1})^T$.

Using non-linear basis functions allows the function $f(\mathbf{x}, \mathbf{w})$ to be non-linear function of the input vector $\mathbf{x}$. However, this function is called linear model because it is linear in $\mathbf{w}$. There are many different functions that can be used as basis functions, one example is

$$\phi_m(\mathbf{x}) = \exp\left(\frac{(x - \mu_n)^2}{2\sigma^2}\right) \tag{2.4}$$

where μ_i is the locations of the basis function with the input space and σ is the spatial scale of the function.

2.1.2. Gaussian Process for Regression

In section 2.1.1 we have defined the linear regression model, we shall continue with deriving the Gaussian process model by defining a distribution over the functions $y(\mathbf{x}, \mathbf{w})$. Let us consider the model defined by Eq. (2.3) and define prior probability over the parameters $\mathbf{w}$ given by an isotropic Gaussian of the form

$$p(\mathbf{w}) = \mathcal{N}\left(\mathbf{w} \mid \mathbf{0}, \alpha^{-1}\mathbf{I}\right) \tag{2.5}$$

where α^{-1} is the precision of the distribution. Different values of $\mathbf{w}$ define different functions over $\mathbf{x}$, therefore, by defining a probability distribution over $\mathbf{w}$ we are implicitly defining a probability distribution over the functions $f(\mathbf{x})$. We evaluate the function $f(\mathbf{x})$ at the training points $\mathbf{x}_1, \ldots, \mathbf{x}_N$, therefore, we are interested in the joint distribution of the function values $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)$, which we denote with $\mathbf{f}$ where $f_n = f(\mathbf{x}_n)$ for $n = 1, \ldots, N$. Using Eq. (2.3) this vector is calculated by

$$\mathbf{f} = \boldsymbol{\Phi}\mathbf{w} \tag{2.6}$$

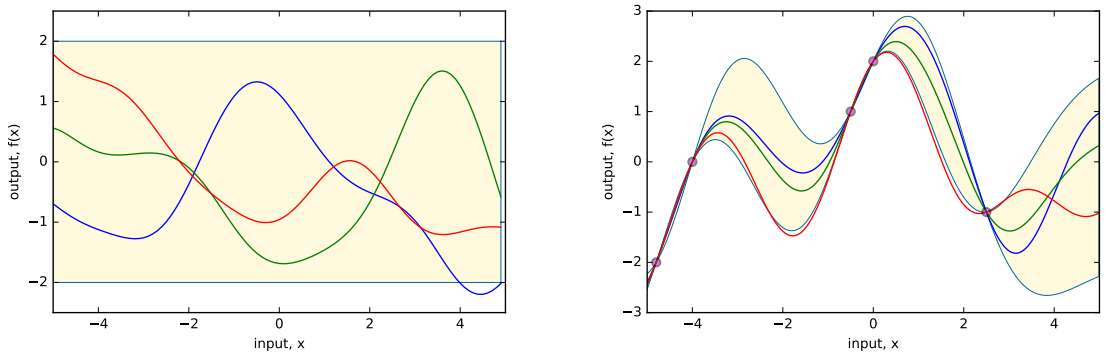
where $\boldsymbol{\Phi}$ *design matrix* with elements $\Phi_{nm} = \phi_m(\mathbf{x}_n)$. Because, $\mathbf{f}$ is a linear combination of Gaussian distributed variables given by the elements of $\mathbf{w}$, also $\mathbf{f}$ is a Gaussian distributed variable. Having this fact we only need to find the mean and the covariance of the distribution over $\mathbf{f}$. Having in mind that the probability distribution over $\mathbf{w}$ given by the Eq. (2.5) induces probability distribution over $\mathbf{f}$, defined as

$$\mathbb{E}[\mathbf{f}] = \boldsymbol{\Phi}\mathbb{E}[\mathbf{w}] \tag{2.7}$$

$$\operatorname{cov}[\mathbf{f}] = \mathbb{E}\left[\mathbf{f}\mathbf{f}^T\right] = \boldsymbol{\Phi}\mathbb{E}\left[\mathbf{w}\mathbf{w}^T\right]\boldsymbol{\Phi}^T = \alpha^{-1}\boldsymbol{\Phi}\boldsymbol{\Phi}^T = \mathbf{C} \tag{2.8}$$

where $\mathbf{C}$ is the Gram matrix with elements

$$C_{nn'} = k(\mathbf{x}_n, \mathbf{x}_{n'}) = \alpha^{-1}\phi(\mathbf{x}_n)^T \phi(\mathbf{x}_n) \tag{2.9}$$



(a) Three sample functions from the prior **(b)** Three sample functions from the posterior

Figure 2.1.: Fig. (a) shows three functions drawn at random from a GP prior by joining a large number of evaluated points. Fig. (b) shows three random functions drawn from the posterior, i.e. the prior conditioned on the five noise free observations indicated. In both plots the shaded area represents the point-wise mean plus and minus two times the standard deviation for each input value (corresponding to the 95% confidence region), for the prior and posterior respectively.

where $k(\mathbf{x}, \mathbf{x}')$ is a kernel function.

A major point in the Gaussian process model is that the join distribution over $f_1, \ldots, f_N$ is completely specified by the mean and the covariance. In most of the application we do not have any information about the mean of $f(\mathbf{x})$, so we will take it to be zero. This is the same as choosing the prior over the weight values to be zero in a basis function point of view. Then the specification of the Gaussian process model is completely given by the covariance of the $f(\mathbf{x})$, which is given by the kernel function

$$\mathbb{E}[f(\mathbf{x}_n) f(\mathbf{x}_{n'})] = k(\mathbf{x}_n, \mathbf{x}_{n'}). \tag{2.10}$$

Usually, we define the kernel function directly, instead of indirectly through choice of basis functions. In section 2.1 we can see how we can influence the process by the choice of the kernel function.

If we want to apply the Gaussian process model to regression problems practically, we have to consider the noise which is added to the target values, which is given by

$$y_n = f_n + \epsilon_n \tag{2.11}$$

where $f_n = f(\mathbf{x}_n)$ and ϵ_n is a random noise added independently to each target value. For the regression problem, we will consider a noise process that have Gaussian distribution defined by

$$p(y_n \mid f_n) = \mathcal{N}(y_n \mid f_n, \sigma) \tag{2.12}$$

where σ is the noise hyperparameter. Because the noise is independent for each data point, the joint distribution of the target values $\mathbf{y} = (y_1, \ldots, y_N)^T$ conditioned on the values of $\mathbf{f} = (f_1, \ldots, f_N)$ is given by

$$p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}(\mathbf{y} \mid \mathbf{f}, \sigma \mathbf{I}_N). \tag{2.13}$$

From the definition of Gaussian process, the marginal distribution of $p(\mathbf{f})$ is given by

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f} \mid \mathbf{0}, \mathbf{C}) \tag{2.14}$$

where the kernel function that calculates $\mathbf{C}$ is chosen to express the similarity of two input points. If the input points $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar, then the value of $k(\mathbf{x}_i, \mathbf{x}_j)$ should express stronger correlation than the one for dissimilar points. The notion of similarity between two points may vary from application to application.

For finding the marginal distribution $p(\mathbf{y})$ conditioned to the input points $\mathbf{x}_1, \ldots, \mathbf{x}_N$, we have to integrate over $\mathbf{f}$. Using Eq. (A.7) the marginal distribution of $\mathbf{y}$ is given by

$$p(\mathbf{y}) = \int p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f}) \, \mathrm{d}\mathbf{f} = \mathcal{N}(\mathbf{y} \mid \mathbf{0}, \mathbf{K}) \tag{2.15}$$

where the covariance matrix $\mathbf{K}$ has the elements

$$K(\mathbf{x}_n, \mathbf{x}_{n'}) = k(\mathbf{x}_n, \mathbf{x}_{n'}) + \sigma \delta_{nn'} \tag{2.16}$$

The main goal of the solution of a regression problem is to be able to predict a target variable for a new input point. So far we have only derived the joint distribution over set of points. Let us now consider the training set of target values $\mathbf{y}_N = (y_1, \ldots, y_N)$ and input points $\mathbf{x}_1, \ldots, \mathbf{x}_N$, our goal will be to predict y_{N+1} for given $\mathbf{x}_{N+1}$. In order to make the prediction, we have to evaluate the predictive distribution $p(y_{N+1} \mid \mathbf{y}_N, \mathbf{X}_{N+1})$, where $\mathbf{X}_{N+1}$ denotes the vector $(\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{x}_{N+1})$. To find this conditional distribution first we have to calculate the marginal distribution $p(\mathbf{y}_{N+1})$, where $\mathbf{y}_{N+1}$ denotes the vector $(y_1, \ldots, y_N, y_{N+1})$, then we apply Eq. (A.12) to calculate the conditional distribution. From 2.13, the joint distribution over $y_1, \ldots, y_{N+1}$ is given by

$$p(\mathbf{y}_{N+1}) = \mathcal{N}(\mathbf{y}_{N+1} \mid \mathbf{0}, \mathbf{K}_{N+1}) \tag{2.17}$$

where $\mathbf{K}_{N+1}$ is a covariance matrix with elements given by Eq. (2.16). Then we can apply the Eq. (A.12) to find the conditional distribution from the joint distribution. We partition the covariance matrix as follows

$$\mathbf{K}_{N+1} = \begin{pmatrix} \mathbf{K}_N & \mathbf{k} \\ \mathbf{k}^T & c \end{pmatrix} \tag{2.18}$$

where $\mathbf{K}_N$ is the $N \times N$ covariance matrix, the vector $\mathbf{k}$ has elements $k(\mathbf{x}_n, \mathbf{x}_{N+1})$ for $n = 1, \ldots, N$ and the scalar $c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \sigma$. Using the Eq. (A.12) the conditional distribution $p(y_{N+1} \mid \mathbf{y})$ which is also called *predictive distribution* is a Gaussian distribution given by

$$p(y_{N+1} \mid \mathbf{y}) = \mathcal{N}\left(m(\mathbf{x}_{N+1}), \sigma^2(\mathbf{x}_{N+1})\right) \tag{2.19}$$

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^T \mathbf{K}_N^{-1} \mathbf{y} \tag{2.20}$$

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^T \mathbf{K}_N^{-1} \mathbf{k} \tag{2.21}$$

From the results of the Gaussian process derivation we can see that the mean and the covariance of the process are functions of the new input point $\mathbf{x}_{N+1}$.

The main computational cost in the Gaussian process is the $N \times N$ matrix inversion, which requires $\mathcal{O}(N^3)$. This matrix inversion must be performed only once for given training data. For making prediction of new input point, we have vector by matrix multiplication, which has a computational cost of $\mathcal{O}(N^2)$. The matrix inversion operation can be a problem if we have large training set or training set with high dimensionality. There are many approximations of the Gaussian process model for regression that improve the computaional cost. In the section 2.3 we shell present one approximation, and in section 2.3.3 we extend this model by adding regularization term that controls the overfitting of the data. Gaussian processes are very flexible regression models, and the model is limited by the covariance function. Using GP makes modeling non-stationary process really hard, because of the construction of a non-stationary covariance function. Section 2.2 presents properties of the covariance functions, as well as examples of covariance functions.

2.1.3. Learning the Hyperparameters in Gaussian Process for Regression

The choice of the covariance function, as well as the parameters of that function, control the prediction of a Gaussian process. The parameters of the covariance function control the length scale of the correlation, as well as the precision of the noise. These parameters are called *hyperparameters*, and we will infer their values from the training data.

One technique of learning the hyperparameters for a Gaussian process is to maximize the log likelihood function of the process $p(\mathbf{y} \mid \boldsymbol{\theta})$, where with $\boldsymbol{\theta}$ we denote the hyperparameters of the Gaussian process, and it is a vector comprised of the parameters of the covariance function. Maximization of the log likelihood function can be done with gradient based optimization techniques, like conjugate gradients [FR64].

The log likelihood function of a Gaussian process has the following form

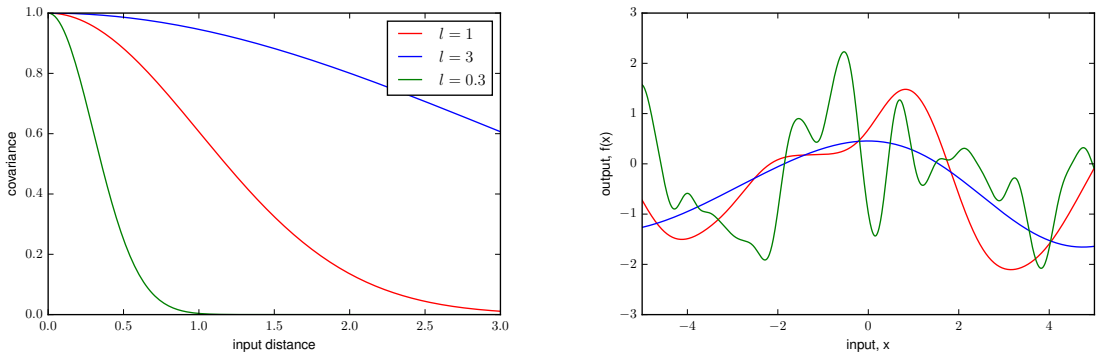
$$\ln p(\mathbf{y} \mid \boldsymbol{\theta}) = -\frac{1}{2}\ln|\mathbf{K}_N| - \frac{1}{2}\mathbf{y}^t\mathbf{K}_N^{-1}\mathbf{t} - \frac{N}{2}\ln(2\pi). \tag{2.22}$$

For the optimization we need the gradient of the log likelihood function with respect to the hyperparameters $\boldsymbol{\theta}$. Using Eq. (A.3) for $\mathbf{K}_N^{-1}$ and Eq. (A.4) for $\ln|\mathbf{C}_N|$, we obtain

$$\frac{\partial}{\partial\theta_i}\ln p(\mathbf{t} \mid \boldsymbol{\theta}) = -\frac{1}{2}\mathtt{Tr}\left(\mathbf{C}_N^{-1}\frac{\partial\mathbf{C}_N}{\partial\theta_i}\right) + \frac{1}{2}\mathbf{t}^T\mathbf{C}_N^{-1}\frac{\partial\mathbf{C}_N}{\partial\theta_i}\mathbf{C}_N^{-1}\mathbf{t}. \tag{2.23}$$

In fully Bayesian treatment we need to evaluate marginals over $\boldsymbol{\theta}$ weighted by the product of the prior $p(\boldsymbol{\theta})$ and the likelihood function $p(\mathbf{t} \mid \boldsymbol{\theta})$. In practice though, exact marginalization is intractable, therefore we need to approximate.

In this section we have derived the Gassian process model for regression from the linear basis function model. The covariance function in the Gaussian process model has an important role, changing the properties of the covariance function leads to changing the model. Therefore, in the next section we will explore different covariance functions.



(a) **(b)**

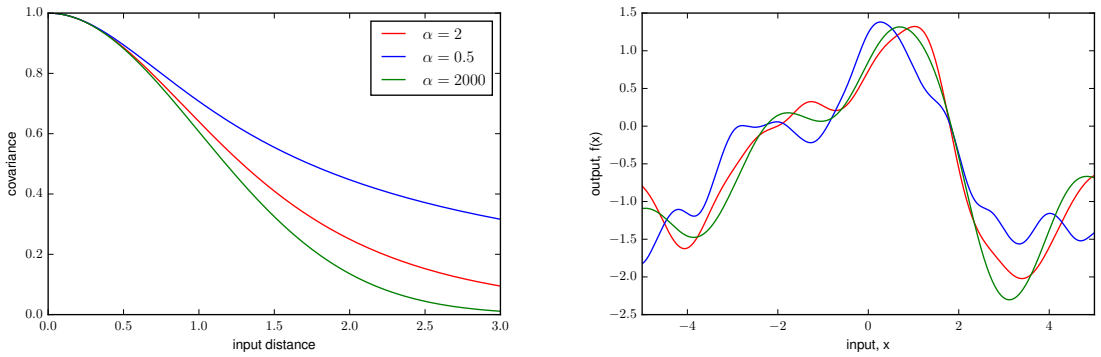
Figure 2.2.: (a): a square exponential covariance function; (b): three functions, randomly sampled from three Gaussian processes, defined by a square exponential covariance functions with different length scales.

2.2. Covariance Functions

The covariance function is of crucial importance in a GP regression model. The GP model defines distribution over functions, and the covariance function defines the properties of a sample function drawn from that distribution. Furthermore, the assumption we have about the possible properties of the target function is encoded into the covariance function itself. Also, the choice of the covariance function defines the notion of similarity between the data points. In supervised learning data points $\mathbf{x}$ that are close to each other are most likely to have similar y values. Therefore, training points that are close to a new test point should be more informative than the points further from the new test point. The covariance function then provides the possibility of interpretation of the model through its learned parameters. In section 2.2.1 are introduced some basic terms and properties of covariance functions. Further in section 2.2.2 are presented examples of covariance functions. More detailed analysis of different types and properties of covariance functions could be found in [Ras06].

2.2.1. Preliminaries

If a covariance function is a function of $\mathbf{x} - \mathbf{x}'$ than it is called *stationary*, and it is invariant to translation in the input space. If it is a function of $|\mathbf{x} - \mathbf{x}'|$ than it is called *isotropic*, and it is invariant to all motions. Covariance functions that depend only on $\mathbf{x}$ and $\mathbf{x}'$ through $\mathbf{x} \cdot \mathbf{x}'$ are called *dot product* covariance functions. An example of a such covariance function is $k(\mathbf{x}, \mathbf{x}') = \sigma_0^2 + \mathbf{x} \cdot \mathbf{x}'$. These functions are invariant to rotation of the coordinate system, but not to translation. Given a set of input points $\mathbf{X} = \{\mathbf{x}_i \mid i = 1, \ldots, N\}$ the *Gram matrix* $\mathbf{K}$ is calculated as $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, and if k is a covariance function then $\mathbf{K}$ is called *covariance matrix*.



(a) **(b)**

Figure 2.3.: (a): a rational quadratic covariance function; (b): three functions, randomly sampled from three Gaussian processes, defined by a rational quadratic covariance functions with $l = 1$ and different values for α.

2.2.2. Examples of Covariance Functions

In this section we shall present different types of stationary covariance functions over the input domain $\mathcal{X}$ that is a subset of the $\mathbb{R}^D$.

Squared Exponential Covariance Function

The first covariance function that we are presenting is the *squared exponential* (SE) covariance function defined as

$$k_{SE}(\mathbf{x}, \mathbf{x}') = c^2 \exp\left(\frac{(\mathbf{x} - \mathbf{x}')^2}{2l^2}\right) \tag{2.24}$$

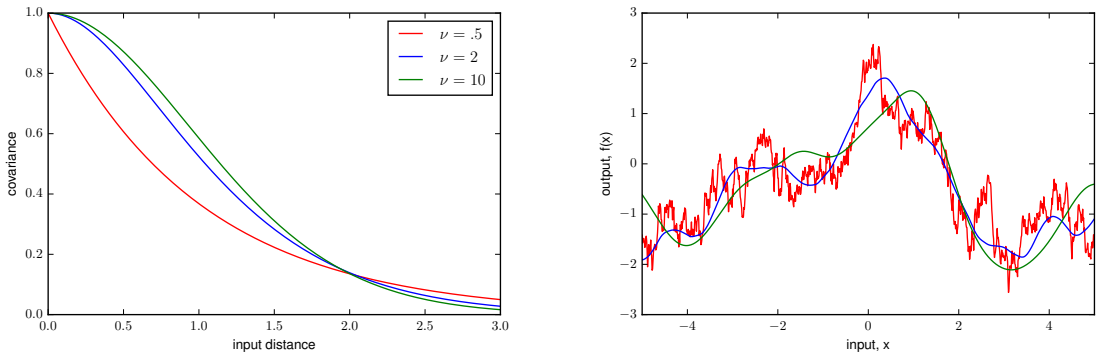
where c is the variance and l is the length-scale. The *length-scale* of a function can informally be understood as the distance within the input space, where one should move in order for the function to change significantly. Such covariance function is infinitely differentiable, which means that a GP using this function has mean square derivatives of all orders, therefore it is very smooth.

In the figure 2.2 one can see, that the function drawn from a process modeled with a SE covariance function with a long length-scale $l = 3$ is much smoother, then the function drawn form a process modeled with a short length-scale $l = 0.3$.

Rational Quadratic Covariance Function

The *rational quadratic* (RQ) covariance function is defined as

$$k_{RQ}(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{(\mathbf{x} - \mathbf{x}')^2}{2\alpha l^2}\right)^{-\alpha} \tag{2.25}$$



(a) **(b)**

Figure 2.4.: (a): a covariance functions from the Matérn class;(b): three functions, randomly sampled from three Gaussian processeswith Matérn covariance functions with different values of ν and $l = 1$.

with $\alpha, l > 0$. This covariance function can be seen as an infinite sum of squared exponentials with different length-scales. A sum of covariance functions is also a covariance function itself. The limit of a rational quadratic covariance function when $\alpha \to \infty$ is a SQ covariance function with a length-scale l, Eq. (2.24).

Figure 2.3 presents the behavior of a RQ covariance function for different values of α. A GP process using RQ covariance function is also an infinitely mean square differentiable.

Matérn Class of Covariance Functions

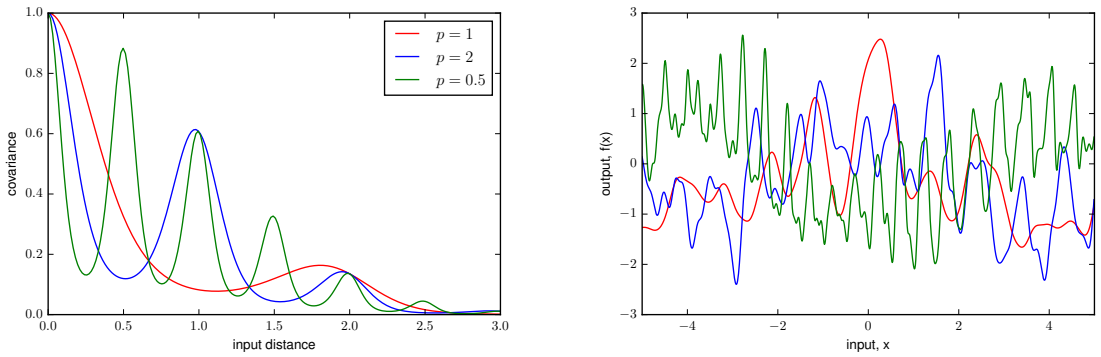
The *Matérn class* of covariance functions is given by

$$k_{Matern}\left(\mathbf{x}, \mathbf{x}'\right) = \frac{2^{1-\nu}}{\Gamma\left(\nu\right)} \left(\frac{\sqrt{2\nu \left(\mathbf{x} - \mathbf{x}'\right)^2}}{l} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu \left(\mathbf{x} - \mathbf{x}'\right)^2}}{l} \right) \tag{2.26}$$

where ν and l are positive, and K_{ν} is a modified Bassel function [AS64]. This class of functions has also a connection with the SE covariance function. For the case when $\nu \to \infty$ we obtain a SE covariance function. When setting ν to $\frac{1}{2}$, then a special case of a SE covariance function emerges

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \exp\left(-\frac{\left|\mathbf{x} - \mathbf{x}'\right|}{l} \right) \tag{2.27}$$

and the corresponding process is called Ornstein-Uhlenbeck process [UO30]. This process is mean square continuous but not mean square differentiable. The most interesting cases in Machine learning of Matérn class are when $\nu = \frac{3}{2}$ and $\nu = \frac{5}{2}$. In Figure 2.4 are presented covariance functions from the Matérn class with different values for ν.



(a) **(b)**

Figure 2.5.: (a): covariance functions that is product of SQ and periodic covariance function; (b): three random function sampled from a Gaussian processes with SQ/periodic covariance functions with different values for the period p, where $l_1 = 1$ and $l_2 = 1$.

Creating New Covariance Functions from Old

It is also possible to create a new covariance function with combinations of covariance functions already defined. This extends the flexibility of the GP and gives a possibility to model complex functions. The sum of two independent covariance functions is also a covariance function $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$. Constructions like these can also be used for summing covariance functions with different length-scales. The product of two independent covariance functions is also a covariance function, $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')\, k_2(\mathbf{x}, \mathbf{x}')$.

We shall introduce a new covariance function that is a result of a product combination between a SE covariance function and a periodic covariance function (SE-PER)

$$k_{se-per}(\mathbf{x}, \mathbf{x}') = c^2 \exp\left(-\frac{(\mathbf{x} - \mathbf{x}')^2}{2l_1^2}\right) \exp\left(-\frac{2\sin^2\left(\pi\frac{(\mathbf{x}-\mathbf{x}')}{p}\right)}{l_2^2}\right) \tag{2.28}$$

where c is the amplitude of the function, p is the period of the periodic component, l_1 is the *decay-time* of the periodic component and the l_2 is the smoothness of the periodic component. In Figure 2.5b are presented the SE-PER covariance functions with different values for the period p. One can see that in Figure 2.5a the amplitude of the covariance function is not constant, but it is decaying. This decay is controlled by the length-scale of the SE covariance function.

In Figure 2.6 are presented similarity matrices generated with SE-PER covariance functions. With the change of the length-scales of the covariance function the notion of similarity between two points changes. The length-scale of the periodic part is responsible for the width of each period, short periodic length-scale correspond to tide period. The length-scale of the exponential part correlates to the decay of the amplitude, short exponential length-scale forces the amplitude of the function to decay faster.

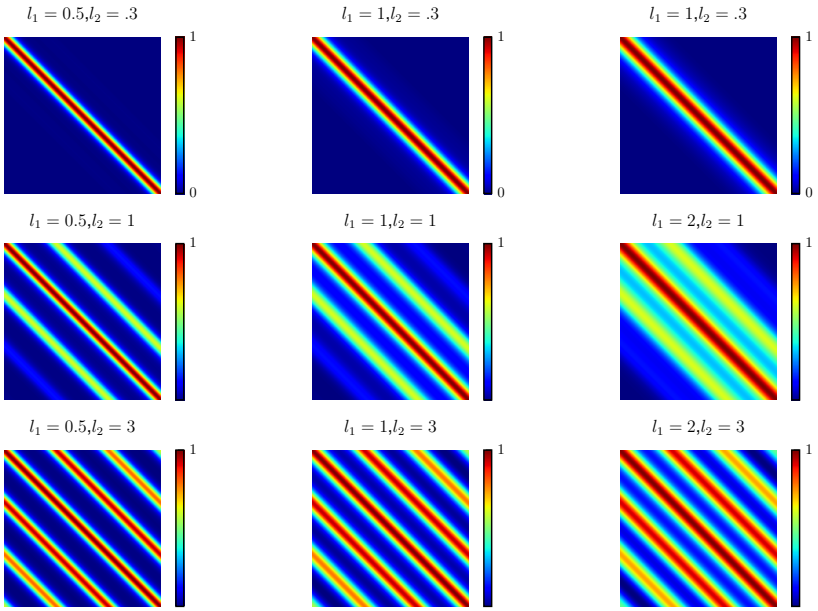


Figure 2.6.: Similarity matrix is obtained by applying the SE-PER covariance function with amplitude $c = 1$, period $p = 1$, and different values for the length-scales (l_1, l_2) on discretized x-axis of 200 equally spaced points between 0 and 3.

2.3. Sparse Approximation of Gaussian Process

The flexibility and the non-parametric nature of the Gaussian process model has also its disadvantages. Namely, an inversion of $N \times N$ matrix have to be done. This results with computational complexity of the model of $\mathcal{O}(N^3)$ operations, where N is the number of training points. Many approximation techniques try to reduce the computational cost and apply Gaussian processes to large problems in machine learning.

All proposed approximation techniques can be roughly divided into two classes. The first class contains methods that approximate the full posterior by introducing matrices of lower rank $M < N$, whereas the second class contains methods that try to approximate using matrix-vector multiplication conjugate methods. Detailed overview of the approximation methods is elaborated by [QW07].

In this section we present approximation methods that belong to the first class, and many of those are called *sparse* approximation [SB01; WS01; Csa02; CH; CO02; SWL03; See03]. All these methods have in common the modeling of the full posterior, trough taking only a subset of the latent variables exactly. For the latent variables, some kind of an approximation that has low computational cost is used. All of the methods however choose a subset M (active set) of the training data with size $M \ll N$, and try to lower the computational cost to $\mathcal{O}(M^2 N)$. The M points used for the approximation are chosen by an information criteria. For example [SWL03] uses very fast approximate information criterion, that greedily selects points to the active set. However, this leads to difficulties of learning the kernel hyper-parameters by maximizing the marginal likelihood of the GP

using gradient ascent. The reselecting of the active set causes non-smooth fluctuations of the gradients of the marginal likelihood, which results with difficulties of converging to a good local maxima.

In section 2.3.1 we present sparse pseudo-input Gaussian process [SG05] approximation method that overcomes the problem of learning the hyperparameters of the process. Next in section 2.3.2, we describe an extension to the model [SG12], which gives a flexibility to it, for modeling a data with input dependent noise. Finally in section 2.3.3 we examine the overfitting problem that occurs in the model for specific data sets, and we provide a solution to the problem.

2.3.1. Sparse Input Gaussian Process (SPGP)

[SG05] have designed an approximation model for the full GP, called sparse pseudo-input Gaussian process (SPGP) regression model. This model enables a search for the kernel hyperparameters and the active set in one smooth join optimization. This is possible due to the fact that they allow the active set (pseudo-inputs M) not to be only a subset of the training data. With parametrization of the covariance function of the GP by the pseudo-inputs, one has the possibility to learn the pseudo-inputs in gradient ascent. This is a major advantage since it allows an improvement of the fit of the model by fine tuning the location of the pseudo-inputs.

In order to derive this computationally tractable model, that still preserves the properties of the full GP, [SG05] have examined the *predictive distribution* of the GP with predictive distribution defined by Eq. (2.19). The mean and the covariance of the predictive distribution can be considered as functions that are parametrized by the location of the N training pairs. Every training pair has an input value $\mathbf{x}$ and target value $\mathbf{y}$.

The likelihood of the SPGP model is defined by the predictive distribution of the GP, and it is parametrized by a *pseudo data set* $\overline{\mathcal{D}}$. The active set is called pseudo data set because the points can take any position, and they are not restricted to the training points. The size of the pseudo data set is M, and it contains pseudo inputs $\overline{\mathbf{X}} = \{\overline{\mathbf{x}}_m\}_{m=1}^{M}$ and pseudo targets $\overline{\mathbf{f}} = \left\{\overline{\mathbf{f}}_m\right\}_{m=1}^{M}$. The sparsity of the model arise from the fact that $M \ll N$. The pseudo targets are denoted with $\overline{\mathbf{f}}$ instead of $\overline{\mathbf{y}}$ because they are not real observation, and therefore there is no need to add noise variance to them. This setup leads to single data point likelihood defined as

$$p\left(y \mid \mathbf{x}, \overline{\mathbf{X}}, \overline{\mathbf{f}}\right) = \mathcal{N}\left(y \mid {\mathbf{k}_{\mathbf{x}}}^T \mathbf{K}_M^{-1} \mathbf{y}, K_{\mathbf{xx}} - {\mathbf{k}_{\mathbf{x}}}^T \mathbf{K}_M^{-1} \mathbf{k}_{\mathbf{x}} + \sigma^2\right) \tag{2.29}$$

where $[K_M]_{mm'} = K\left(\overline{\mathbf{x}}_m, \overline{\mathbf{x}}_{m'}\right)$ and $[\mathbf{k}_{\mathbf{x}}]_m = K\left(\overline{\mathbf{x}}_m, \mathbf{x}\right)$, for $m = 1, \ldots, M$. The target data is generated i.i.d. given the input data, hence we have the complete data likelihood defined as

$$p\left(\mathbf{y} \mid \mathbf{x}, \overline{\mathbf{X}}, \overline{\mathbf{f}}\right) = \prod_{n=1}^{N} p\left(y_n \mid \mathbf{x}_n, \overline{\mathbf{X}}, \overline{\mathbf{f}}\right) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{K}_{NM} \mathbf{K}_M^{-1} \overline{\mathbf{f}}, \sigma^2 \mathbf{\Gamma}\right) \tag{2.30}$$

where $\sigma^2 \mathbf{\Gamma} = \mathbf{\Lambda} + \sigma^2 \mathbf{I}_N$, $\mathbf{\Lambda} = \operatorname{diag}\left(\mathbf{K}_N - \mathbf{Q}_N\right)$, $[\mathbf{K}_{NM}]_{nm} = K\left(\mathbf{x}_n, \overline{\mathbf{x}}_m\right)$, and $\mathbf{Q}_N = \mathbf{K}_{NM} \mathbf{K}_M^{-1} \mathbf{K}_{MN}$. Defining Gaussian prior distribution $p\left(\overline{\mathbf{f}} \mid \overline{\mathbf{X}}\right)$ over the pseudo inputs $\overline{\mathbf{f}}$

makes it possible to integrate out the pseudo targets. The prior distribution $p\left(\overline{\mathbf{f}} \mid \overline{\mathbf{X}}\right)$ is defined by

$$p\left(\overline{\mathbf{f}} \mid \overline{\mathbf{X}}\right) = \mathcal{N}\left(\overline{\mathbf{f}} \mid 0, \mathbf{K}_M\right). \tag{2.31}$$

The posterior distribution over the pseudo targets $\overline{\mathbf{f}}$ can be calculated using Bayesian rule on Eqs. (2.30) and (2.31), and it is given by

$$\begin{aligned} p\left(\overline{\mathbf{f}} \mid \mathcal{D}, \overline{\mathbf{X}}\right) &= p\left(\mathbf{y} \mid \mathbf{x}, \overline{\mathbf{X}}, \overline{\mathbf{f}}\right) p\left(\overline{\mathbf{f}} \mid \overline{\mathbf{X}}\right) \\ &= \mathcal{N}\left(\mathbf{K}_M \mathbf{M}^{-1} \mathbf{K}_{MN} \sigma^{-2} \mathbf{\Gamma}^{-1} \mathbf{y}, \mathbf{K}_M \mathbf{M}^{-1} \mathbf{K}_M\right) \end{aligned} \tag{2.32}$$

where $\mathbf{M} = \mathbf{K}_M + \mathbf{K}_{MN} \sigma^{-2} \mathbf{\Gamma}^{-1} \mathbf{K}_{NM}$. The predictive distribution for a given new point $\mathbf{x}_*$ is calculated by integrating the likelihood Eq. (2.29) and (2.32) and it is given by

$$p\left(y_* \mid \mathbf{x}_*, \mathcal{D}, \overline{\mathbf{X}}\right) = \int p\left(y_* \mid \mathbf{x}_*, \overline{\mathbf{X}}, \overline{\mathbf{f}}\right) p\left(\overline{\mathbf{f}} \mid \mathcal{D}, \overline{\mathbf{X}}\right) d\overline{\mathbf{f}} = \mathcal{N}\left(y_* \mid \mu_*, \sigma_*^2\right), \tag{2.33}$$

where

$$\begin{aligned} \mu_* &= \mathbf{k}_*^T \mathbf{M}^{-1} \mathbf{K}_{MN} \sigma^{-2} \mathbf{\Gamma}^{-1} \mathbf{y} \\ \sigma_*^2 &= K_{**} - \mathbf{k}_*^T \left(\mathbf{K}_M^{-1} - \mathbf{M}^{-1}\right) \mathbf{k}_* + \sigma^2. \end{aligned}$$

Learning the model involves learning the hyperparameters $\mathbf{\Theta} = \{\boldsymbol{\theta}, \sigma^2\}$ and the locations of the pseudo inputs $\overline{\mathbf{X}}$, which can actually be done in one join smooth optimization maximizing the marginal likelihood with respect to the parameters. This can be calculated using Eqs. (2.30) and (2.31), and the marginal likelihood is defined by

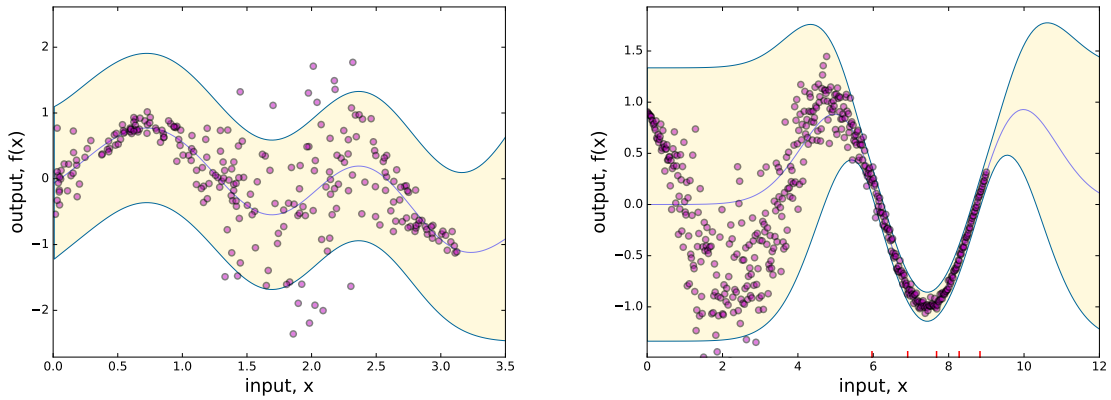
$$\begin{aligned} p\left(\mathbf{y} \mid \mathbf{x}, \overline{\mathbf{X}}\right) &= \int p\left(\mathbf{y} \mid \mathbf{x}, \overline{\mathbf{X}}, \overline{\mathbf{f}}\right) p\left(\overline{\mathbf{f}} \mid \overline{\mathbf{X}}\right) d\overline{\mathbf{f}} \\ &= \mathcal{N}\left(\mathbf{y} \mid \mathbf{0}, \mathbf{K}_{NM} \mathbf{K}_M^{-1} \mathbf{K}_{NM} + \mathbf{\Lambda} + \sigma^2 \mathbf{I}\right) \end{aligned} \tag{2.34}$$

The marginal likelihood can be maximized with respect to $\left\{\overline{\mathbf{X}}, \mathbf{\Theta}\right\}$ using gradient ascent. The details of the calculation of the gradients are presented in section B.1 using Cholesky factorization. The exact form of the gradients will of course depend on the functional form of the covariance function. The dominant computational cost in this model is the matrix multiplication $\mathbf{K}_{MN} \sigma^{-2} \mathbf{\Gamma}^{-1} \mathbf{K}_{NM}$ in $\mathbf{M}$ which is $\mathcal{O}\left(M^2 N\right)$. The inversion of $\mathbf{\Lambda} + \sigma^2 \mathbf{I}$ is not expensive since this matrix is diagonal.

Although the SPGP is not specifically designed to model non-stationary functions, parameterizing the model on the location of the pseudo input points provides some kind of non-stationarity. This non-stationarity is of course a side effect or an artifact to the approximation of the full GP. This provides the possibility to model an input dependent noise using GP. An input dependent noise

$$y_n = f_n + \epsilon_n, \epsilon_n \sim \mathcal{N}\left(0, \sigma_y^2\right) \tag{2.35}$$

follows it is own distribution, and the intensity of the noise is varies in different regions of the input space.



(a) full Gaussian process **(b)** SPGP

Figure 2.7.: Synthetic heteroscedastic data set learned by full Gaussian Process model Fig. (a); synthetic heteroscedastic data set learned by sparse pseudo-input Gaussian process (SPGP) Fig.(b). In both plots the shaded area represents the point-wise mean plus and minus two times the standard deviation for each input value (corresponding to the 95% confidence region). The red lines at the bottom in the Fig. (b) represent the locations of the pseudo-input points.

Two different models, that model synthetic data sets with input depend noise are presented in Figure 2.7. This kind of data sets are also called *heteroscedastic data sets*. We can see that the SPGP model fits the data much better by moving some of the pseudo-input points to the right. The origin of this non-stationary behavior of the SPGP model comes from the fact that the noise of the model on the locations where the pseudo-inputs are is σ^2. As we move away from the pseudo input points the noise grows to $K_{nn} + \sigma^2$. This behavior of the SPGP is a subject of a further research in [SG12].

2.3.2. Sparse Input Gaussian Process with Variable Noise (SPGP+HS)

In section 2.3.1 we have presented the sparse pseudo-input [SG05] approximation of the GP. The main goal of designing this model is to improve the computational complexity of the GP model from $\mathcal{O}(N^3)$ to $\mathcal{O}(M^2N)$, where the M is the size of the pseudo data set, and N is the size of the training data set. A non-stationary behavior emerges by allowing the pseudo-input points to take any position in the input space. The possibility of modeling heteroscedastic data set with the SPGP model is very powerful. Although, the model shows a non-stationary behavior, the fact that this behavior is a side effect and not a property, does not enable the user to influence it.

The limitation of the SPGP to model variable noise can be best seen in Figure 2.7b. Although the SPGP has a single global noise level σ^2, the predictive variance will only drop to this level in regions close to the pseudo-input points. The predictive covariance away from the pseudo-input points will rise to $c + \sigma^2$, because the correlation in this

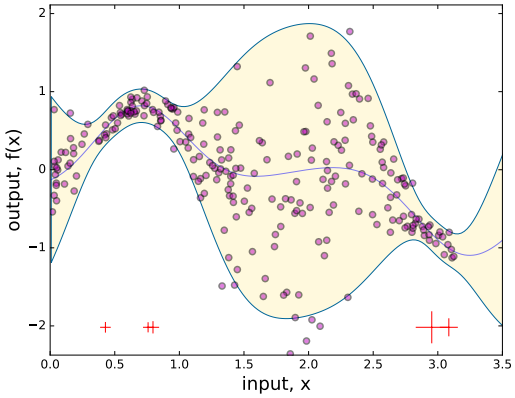


Figure 2.8.: Synthetic heteroscedastic data set learned by sparse pseudo-input Gaussian process with heteroscedastic extension (SPGP+HS) model. The red pluses are representing the locations of the pseudo-input points, and the size of the pluses is proportional to the magnitude of the influence of a pseudo-input point to the prediction. Pseudo-input points that have large pluses influence the prediction more, hence the uncertainty associated with that point is smaller. The shaded area represents the point-wise mean plus and minus two times the standard deviation for each input value (corresponding to the 95% confidence region).

region cannot be modeled. The SPGP adjusts the location of the pseudo-input points during training, in order to take an advantage of the non-stationarity behavior of the sparse covariance function. The influence of the pseudo input points on the predictive distribution is all or nothing at the position where they are. Occasionally though this can cause a problem for some of the data sets, because it forces all the pseudo-input points to be shifted to one place, and the predictive distribution at that place is modeled well. At the other parts of the function however, the correlation is modeled poorly, because there are no pseudo-inputs.

For solving this problem [SG12] introduce extra uncertainty parameter to every pseudo-input point. The altered covariance of the pseudo-input points have the following form

$$\mathbf{K}_M \rightarrow \mathbf{K}_M + \mathrm{diag}\,(\mathbf{h}) \tag{2.36}$$

where $\mathbf{h}$ is a positive vector of uncertainties that needs to be learned. If $h_m = 0$, then the particular pseudo-input point behaves exactly like SPGP. As h_m grows, the associated pseudo-input point has less influence on the predictive distribution. This means that the pseudo-input points does not have the same role like in SPGP all or nothing, but they can be gradually turned off. If $h_m \rightarrow \infty$, then the pseudo-input point associated with the uncertainty is ignored. Defining an uncertainty vector like this, allows defining different noise variance in the prediction in different regions. The heteroscedastic extension of the SPGP model is referred to as SPGP+HS. The extra parameters $\mathbf{h}$ in the SPGP+HS model are learned by the gradient based maximum likelihood. In Figure 2.9 one can see the power of the SPGP+HS model, where the pseudo-input points have moved left and

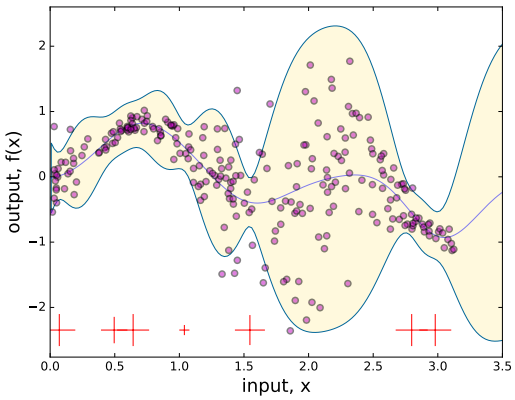


Figure 2.9.: Synthetic heteroscedastic data set learned by sparse pseudo-input Gaussian process with functional heteroscedastic extension (SPGP+RBFSIN-HS) model. The red pluses are representing the locations of the pseudo-inputs. The size of the plus is proportional to the influence of this pseudo-input to the prediction. Pseudo-inputs that have large pluses influence the prediction more, hence the uncertainty associated with that point is smaller.

right from the middle. This is possible because the uncertainty associated with every pseudo-input point can gradually reduce the influence of the point to the predictive distribution.

In this section we have presented the extension of the SPGP model by associating uncertainty to every pseudo-input point, which makes SPGP more powerful and makes applying this model to heteroscedastic data sets possible. However, this flexibility also can lead to problems of overfitting for certain data sets. The authors of the SPGP+HS model do not provide a method of how to deal with this problem. In the next section, we introduce an extension to the SPGP+HS model, which provides a solution to avoid overfitting and make the SPGP+HS more robust.

2.3.3. Sparse Input Gaussian Process with Functional Variable Noise (SPGP+FUNC-HS)

In section 2.3.1, we have introduced an approximation of the full GP model, the sparse pseudo-input Gaussian process. Parameterizing the covariance function of the SPGP model by the pseudo-input points resulted with a non-stationary behavior of the model, and provided a possibility of modeling heteroscedastic data sets, and showed a behavior over which we have no control of. Next, in section 2.3.2 we presented an heteroscedastic extension to the SPGP model. This extension provides the possibility to control the degree of the influence of every pseudo-input point on the predictive distribution, and makes the model more flexible, which on the other hand raises the possibility of overfitting certain data sets. In this section we examine different solutions to the problem of overfitting.

Everything that was done until now is done within the Bayesian framework, therefore it would be natural to continue to work in this framework and be consistent. The first step we should do is to define prior distribution over the vector $\mathbf{h}$ of uncertainties. Once we have done that we need to derive the predictive distribution of the process, including the prior distribution over the uncertainty vector. However, taking this approach leads to calculating computationally intractable integral. We can overcome this problem by using some sampling algorithm and approximate the computation of this integral. This makes the algorithm more complex for implementation, and also increases the computational complexity. Even though, continuing to work in the Bayesian framework would be the natural and most compact way to solve the problem of overfitting, we end up with a complex algorithm.

The fact that we work with data sets with input depend noise, informs us that the noise is generated by some unknown function. Therefore, the simplest approach of solving this problem would be to define an uncertainty function of the pseudo-input points. The covariance function of the SPGP+HS with functional noise is defined as

$$\mathbf{K}_M \rightarrow \mathbf{K}_M + \operatorname{diag}\left(f_h\left(\overline{\mathbf{X}}\right)\right) \tag{2.37}$$

where f_h is the uncertainty function and $\overline{\mathbf{X}}$ are the pseudo-inputs. The exact form of the uncertainty function depends on the data set we want to model. Having a prior knowledge about the nature of the noise, would help us in choosing the right uncertainty function. Defining the heteroscedastic extension in this way makes the learning of the parameters of the uncertainty function by the gradient based maximum likelihood possible. In this way we are able to interpret the parameters of the heteroscedastic noise function as parameters that govern the noise of the model. Another advantage of having a heteroscedastic function is that it is restricting the parameter search space when learning the model. This restriction can be beneficial when training, because it removes the sub optimal local maximas. This results with much faster convergence when learning the model and also the chance of overfitting is reduced. We refer to our new heteroscedastic function model as SPGP+FUNC-HS.

Here we introduce two types of heteroscedastic noise functions, that we think are suitable for modeling user behavior time series. In general any function that describes best the noise of the given data set can be used. The first heteroscedastic noise function that we introduce is the simple sine function

$$f_h\left(\overline{\mathbf{x}}_m\right) = c\sin\left(2\pi\omega\overline{\mathbf{x}}_m + \varphi\right), \tag{2.38}$$

where c is the amplitude, ω is the frequency and φ is the phase. We refer to this model as SPGP+SIN-HS. The second heteroscedastic noise function we introduce is a product of the sine function and the RBF kernel and is defined by

$$f_h\left(\overline{\mathbf{x}}_m, \mathbf{h}_m\right) = c^2 e^{-\frac{\left(\overline{\mathbf{x}}_m - \mathbf{h}_m\right)^2}{2l^2}} \sin\left(2\pi\omega\overline{\mathbf{x}}_m + \varphi\right), \tag{2.39}$$

where c is the variance, $\mathbf{h}_m$ is a mean associated with every pseudo-input $\overline{\mathbf{x}}_m$ point in the RBF kernel, l is the length scale of the RBF kernel. The mean in the RBF kernel can

be initialized by random or set by the user, if the user has a prior knowledge. Setting a mean for every pseudo-input point divides the whole input space into regions, and in each region we have a function which governs the uncertainty associated with every pseudo input. The uncertainty function defined in this way is behaving like a mixture of experts. We refer to this model as SPGP+RBFSIN-HS model. We have also tried to use a polynomial with different degrees as heteroscedastic function, but it showed a poor performance.

2.4. Poisson Processes

Finding a common behavior between the users requires to model individually every user. However, the data we have for every user is spares, since we are working with data generated from human behavior i.e. user have limit how much can work. Modeling this behavior with GP models is not possible, however the recent development of a Scalable Nonparametric Bayesian Inference on Point Processes with Gaussian Processes [SR14] makes possible of modeling the behavior of a user with the intensity function of the Poisson process. Modeling a user behavior using Poisson point process is done for explanation for heavy tails in e-mail communication [Mal+08].

In this section, we describe the Poisson process, therefor we refer to the book "Stochastic Processes, Theory for Applications" [Gal13] for a more detailed description. A Poisson process is a simple and widely used stochastic process, for modeling the times at which arrivals enter a system. Arrivals may occur at arbitrary positive times, and its probability at any particular instant is 0. Hence, it is convenient to define a Poisson process in terms of the sequence of interarrival times $X_1, X_2, \ldots,$, which are defined to be IID.

An *arrival process* is a sequence of increasing random variables $0 < S_1 < S_2 < \cdots$, where $S_i < S_{i+1}$ means that $S_{i+1} - S_i$ is a positive random variable. The random variables $S_1, S_2, \ldots$ are called arrival epochs, and represent the times at which some repeating phenomenon occurs. In order to fully specify the process by the sequence $S_1, S_2, \ldots$ of random variables, it is necessary to specify the joint distribution of the subsequences $S_1, S_2, \ldots S_n$ for all $n > 1$. Any arrival process also can be specified by an alternative stochastic process. The sequence of interarrival times $X_1, X_2, \ldots,$ are positive random variables, defined in terms of arrival epochs by $X_1 = S_1$ and $X_i = S_i - S_{i-1}$ for $i > 1$. Similarly, given the X_i, the arrival epochs S_i are specified as

$$S_n = \sum_{i=1}^{n} X_i. \tag{2.40}$$

The joint distribution of $X_1, \ldots, X_n$ for all $n > 1$ is sufficient to specify the arrival process.

Other alternative for specifying an arrival process is the counting process $N(t)$, where for each $t > 0$, the random variable $N(t)$ is the number of arrivals up to t, and including time t. A counting process $\{N(t); t > 0\}$ has a *stationary increment property*, if every

$t' > t > 0, N(t') - N(t)$ has a same distribution function as $N(t'-t)$. We define $\tilde{N}(t,t') = N(t') - N(t)$ as the number of arrivals in the interval $(t,t']$ for any given $t' \geq t$. A counting process $\{N(t); t > 0\}$ has a *independent increment property* if for every integer $k > 0$, and every k-tuple of times $0 < t_1 < t_2 < \cdots < t_k$, the k-tuple of random variables $N(t_1), \tilde{N}(t_1, t_2), \ldots, \tilde{N}(t_{k-1}, t_k)$.

One example of an arrival process is a Poisson process. This process is best described by the the interarrival times. They provide the most convenient description of a Poisson process, because the interarrival times are defined to be IID. An arrival process for witch the sequence of interarrival times is a sequence of IID random variables is called *renewal process*. A Poisson process is a renewal process in which the interarrival intervals have exponential distribution i.e. for some real $\lambda > 0$, each X_i has the density

$$\mathrm{f}_X(x) = \lambda \exp(-\lambda x) \tag{2.41}$$

where $x \geq 0$, and λ is called the rate of the process, because for any interval of size t, λt is the expected number of arrivals in that interval. An alternative definition of the Poisson process is that, it is a counting process that satisfies Eq. (2.42) and has the stationary and independent increment properties.

$$\begin{aligned} \Pr\{\tilde{N}(t, t+\delta) = 0\} &= e^{-\lambda\delta} \approx 1 - \lambda\delta + o(\delta) \\ \Pr\{\tilde{N}(t, t+\delta) = 1\} &= \lambda\delta e^{-\lambda\delta} \approx \lambda\delta + o(\delta) \\ \Pr\{\tilde{N}(t, t+\delta) \geq 2\} &\approx o(\delta). \end{aligned} \tag{2.42}$$

The currently specified Poisson process is characterized by a constant arrival rate λ, however it is often useful to consider a more general type of process, in which the arrival rate varies as a function of time. A *non-homogeneous Poisson process* with time varying arrival rate $\lambda(t)$ is defined as a counting process $\{N(t); t > 0\}$ which has the independent increment property and, for all $t \geq 0, \delta > 0$, also satisfies

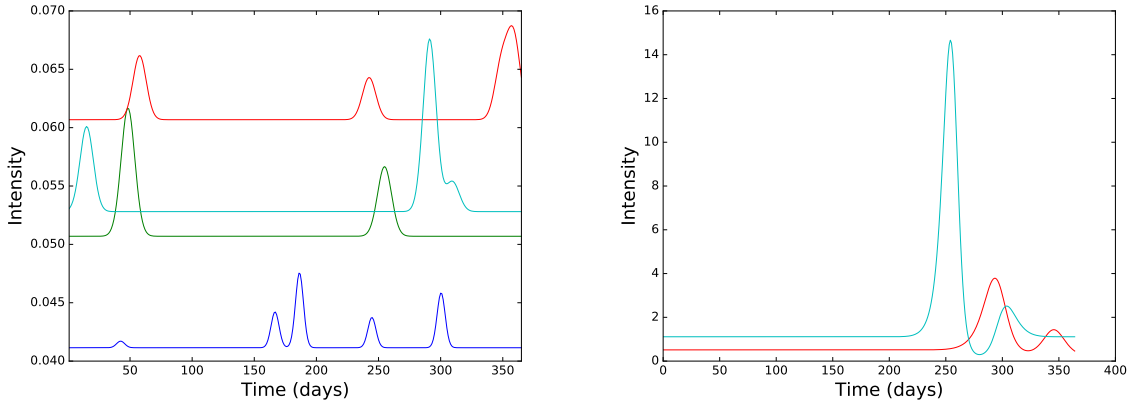
$$\begin{aligned} \Pr\{\tilde{N}(t, t+\delta) = 0\} &= 1 - \lambda(t)\delta + o(\delta) \\ \Pr\{\tilde{N}(t, t+\delta) = 1\} &= \lambda(t)\delta + o(\delta) \\ \Pr\{\tilde{N}(t, t+\delta) \geq 2\} &= o(\delta) \end{aligned} \tag{2.43}$$

where $\tilde{N}(t, t+\delta) = N(t+\delta) - N(\delta)$.

2.5. Clustering Time Series of User Behavior

The problem of finding common user behavior we map to a clustering of the intensity functions of the users. In this section we define a problem of clustering time series, and we propose the Dynamic Piecewise Time similarity measure and K-PSC clustering algorithm as a solution for this problem. Our solution can be considered as a generalization of the K-SC clustering algorithm proposed by [YL11].

First of all, we assume we have time series of some kind of user behavior. It can be for example, the user dynamics of watching videos on Youtube, postings on Tweeter, or



(a) **(b)**

Figure 2.10.: User intensity functions generated from a Poisson point process.

the postings of a question or an answer to some Q&A web site. Then we try to find common behavior patterns between the users. Finding a common behavior pattern can be considered as clustering time series. The centroids are representatives of the common behaviors between the users.

2.5.1. Problem Definition

We have N sets of arrivals, where each set corresponds to a particular user. An *arrival* is a time stamp when a particular action was taken by the user, for example posting a question, watching a video on Youtube, etc. Then we create a time series for every user by taking the number of arrivals at time t, where t is measured in some time unit e.g days. Most of the cases when we do not have enough arrivals to create time series for a particular user. The reason for this is because we work with data generated by a human that have limit in how much can work. In that case we have to look for alternatives, for example create a Poisson point process for every user and cluster the intensity function of the process as time series. The intensity function from a Poisson point process does not provide the exact number of actions at the time step t. Therefore one criterion in designing a similarity measure would be to not take into account the intensity of the intensity function. Thus, we would like to group the time series in groups depending on the similarity of their shape and position on the time axis, and not taking into account the intensity of the time series.

2.5.2. Dynamic Piecewise Time Series Similarity Measure

In order to cluster time series based on their shape and location on the time axis, first of all we have to discuss, how we can measure similarity between two time series based on the above mentioned criteria. In Figure 2.10 are presented samples of intensities from Poisson point processes. We can see that there are intensities with similar shape

but different volume, therefore we would like our similarity measure still to consider these two time series as similar. Additionally, we would like our similarity measure to consider two time series as similar, when they have similar shape, but one of those time series is shifted by some delta. The delta should be user defined and it should be time unit e.g days. This can be seen in Figure 2.10b, where the red time series is shifted by approximately 30 days or one month.

Even though there have been proposed many time series distance measures and clustering algorithms, there is still a no exact solution to the problem we try to solve i.e. the solution is problem dependent. The commonly used metric Dynamic Time Warping [Mül07] can detect that two time series are similar even if they are shifted by some delta. However, if the intensities of the time series are different then this measure fails to consider them as similar. [YL11] propose a similarity measure and clustering algorithm that is invariant to scaling and translation. This similarity measure gives solution to the to criterion of scaling invariant, but not to the problem when the time series are shifted by some delta. Influenced from the distance measure designed by [YL11], we propose a similarity measure that is invariant of scaling, but can still detect time series with similar shapes shifted by some delta.

Consider the two time series $\mathbf{x} = \{x_i\}_{i=1}^{D}$ and $\mathbf{y} = \{y_i\}_{i=1}^{D}$, where $D \in \mathbb{N}^0$ is the dimensionality of the time series and $x_i, y_i \in \mathbb{N}^0$. Then, we divide the two time series into R equal parts, where $0 < R \leq D$, then we obtain the following transformed time series, $\mathbf{p} = \{\mathbf{p}_s\}_{s=1}^{R}$ and $\mathbf{q} = \{\mathbf{q}_t\}_{t=1}^{R}$.

We propose the Dynamic Piecewise Time similarity measure

$$d^2(\mathbf{p}, \mathbf{q}) = \frac{1}{2R} \min_{\alpha} \sum_{s=1}^{R} \sum_{t=1}^{R} k_{[0,1]}(s,t)\, \tilde{d}_{\alpha}(\mathbf{p}_s, \mathbf{q}_t) \tag{2.44}$$

The Dynamic Piecewise Time similarity measure Eq. (2.44) offers the freedom of choosing the shape similarity measure $\tilde{d}_{\alpha}$ between any two pieces of the time series, and the temporal kernel $k_{[0,1]}$ that defines the way in which the parts of two time series are compared. The particular choice of the kernels depends on the application. In our case of finding common user behavior patterns for shape similarity measure, we propose

$$\tilde{d}_{\alpha}(\mathbf{p}_s, \mathbf{q}_t) = \frac{\|\mathbf{p}_s - \alpha \mathbf{q}_t\|^2}{\|\mathbf{p}\|^2} \tag{2.45}$$

where α is a scaling factor. For the choice of the temporal kernel we propose using a RBF kernel

$$k_{[0,1]}(s,t) = \exp\left(-\frac{(s-t)^2}{2\sigma^2}\right). \tag{2.46}$$

Other possible option for temporal kernel could be the *uniform kernel* $k_{[0,1]}(s,t) = \mathbb{I}_{\{|s-t|<\delta\}}$. This kernel compares every piece from the first time series with parts of the second time series that are close in time, and the δ defines how close two pieces have to be.

For finding the minimum of $d^2(\mathbf{p},\mathbf{q})$ w.r.t α we have to calculate the derivative $\frac{\partial d^2(\mathbf{p},\mathbf{q})}{\partial \alpha}$, and then set to 0 and find the α that minimizes the distance.

$$\begin{aligned}\frac{\partial d^2(\mathbf{p},\mathbf{q})}{\partial \alpha} &= \frac{\partial}{\partial \alpha}\frac{1}{2R}\min_{\alpha}\sum_{s=1}^{R}\sum_{t=1}^{R}k_{[0,1]}(s,t)\frac{\|\mathbf{p}_s-\alpha\mathbf{q}_t\|^2}{\|\mathbf{p}\|^2} \\ &= -\frac{1}{R\|\mathbf{p}\|}\sum_{s=1}^{R}\sum_{t=1}^{R}k_{[0,1]}(s,t)\,\mathbf{p}_s^T\mathbf{q}_t+\frac{\alpha}{R\|\mathbf{p}\|}\sum_{s=1}^{R}\sum_{t=1}^{R}k_{[0,1]}(s,t)\,\|\mathbf{q}_t\|^2\end{aligned}$$

Setting the derivative to 0, we have for α

$$\alpha=\frac{\sum_{s=1}^{R}\sum_{t=1}^{R}k_{[0,1]}(s,t)\,\mathbf{p}_s^T\mathbf{q}_t}{\sum_{s=1}^{R}\sum_{t=1}^{R}k_{[0,1]}(s,t)\,\|\mathbf{q}_t\|^2} \tag{2.47}$$

For calculation and implementation convenience we shall replace the double sums with matrix operations. In order for this to be done, we define three matrices. The first matrix is

$$\mathbf{K} = \begin{bmatrix}k(0,0) & \cdots & k(0,R)\\ \vdots & \ddots & \vdots \\ k(R,0) & \cdots & k(R,R)\end{bmatrix} \tag{2.48}$$

is $R\times R$ matrix where $k_{ij}=k_{[0,1]}$ for $i,j=1,\ldots,R$. The second matrix $\tilde{\mathbf{K}}=\mathbf{K}\otimes\mathbf{I}_G$ is of size $GR\times GR$ where $G=\text{ceil}\left(\frac{D}{R}\right)$, $\otimes$ is the Kronecker product and $\mathbf{I}_G$ is the identity matrix of size G. The third matrix

$$L=\text{diag}\left(\left[\underbrace{k(0,0),\ldots,k(0,0)}_{G},\ldots,\underbrace{k(R,R),\ldots,k(R,R)}_{G}\right]\right) \tag{2.49}$$

is diagonal matrix of size $GR\times GR$ where every element of K is repeated G times along the diagonal. Finally, the α that minimizes the distance between two time series

$$\alpha=\frac{\mathbf{x}^T\tilde{\mathbf{K}}\mathbf{y}}{\beta} \tag{2.50}$$

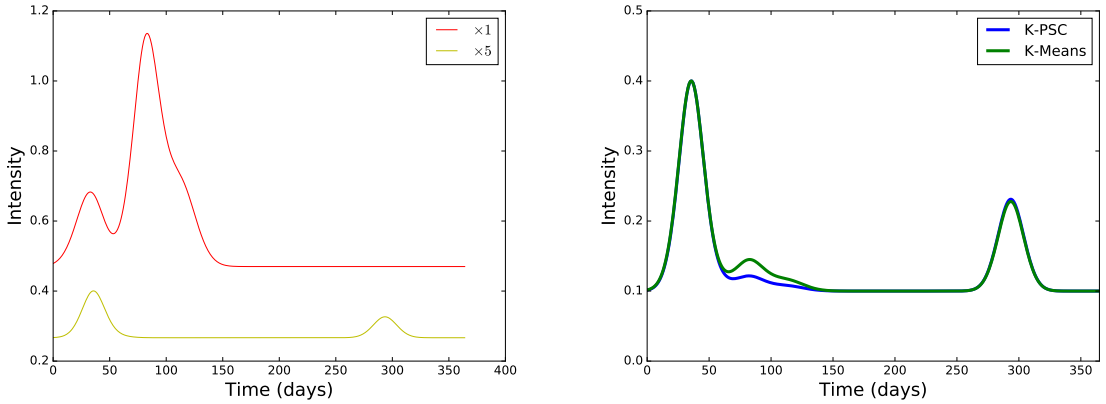
where $\beta=\mathbf{y}^T\mathbf{L}\mathbf{y}$.

Using the matrix notation we rewrite the distance measure in the following form

$$d^2(x,y)=\frac{1}{2R\|\mathbf{x}\|^2}\left(\mathbf{x}^T\mathbf{L}\mathbf{x}-2\alpha\mathbf{x}^T\tilde{\mathbf{K}}\mathbf{y}+\alpha^2\mathbf{y}^T\mathbf{L}\mathbf{y}\right). \tag{2.51}$$

2.5.3. K-Piece Wise Spectral Centroid

In this section we will present an extended version of the K-SC algorithm [YL11], that clusters time series based on the Dynamic Piecewise Time similarity measure, called



(a) Time series **(b)** Cluster centroid

Figure 2.11.: (a) six time series, five of them have the same shape (two picks) and one time series that is considered as outlier; (b) cluster centroids, one centroid is found by K-means, the other by K-PSC. The centroid found by the K-PSC algorithm is much more descriptive and resistant to outliers then the centroid found by K-means.

K-Piece Wise Spectral Centroid (K-PSC). K-PSC is iterative algorithm similar to the K-means algorithm [J A79], but finds clusters under the DPT similarity measure Eq. (2.44) more efficiently. The K-means algorithm iterates over two steps, the assignment and refinement step. At the assignment step, the algorithm assigns every item to the closes cluster based on the Euclidean distance. In the refinement step the clusters centroids are recalculated. By iterating through those two steps the algorithm minimizes inter cluster distance. The K-PSC algorithm works similarly, but uses only the DPT similarity measure. In the refinement step, the K-means algorithm for calculating the new centroids simply averages over the items belonging to the same class. Our similarity measure is scale invariant, therefore not all of the items belonging to the same class contribute the same in averaging for calculating the new centroid. The simple averaging used in K-means is not suitable in our scenario, therefore we have to scale every item in order to find a cluster centroid differently. This can be seen in Figure 2.11. The time series shown in Figure 2.11a are clustered with the K-Means and the K-PSC algorithms, and the results are shown in Figure 2.11b. The centroid found by the K-Means is more sensitive to outliers, whereas, the K-PSC scales every time-series differently in order to find the cluster centroid, and this scaling decreases the influence of outliers.

Next, we define the K-PSC algorithm. We are given a data set of time series $\mathbf{x}_i$, a number of clusters K and a number of pieces R where we divide the time series. The goal is to find for each cluster k the assignment C_k of time-series, and the centroid μ_k of the cluster that minimizes a function F

$$F = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} d^2 \left(\boldsymbol{\mu}_k, \mathbf{x}_i \right). \tag{2.52}$$

The K-PSC algorithm has the same assignment step as the K-means and the K-SC

Algorithm 1: K-PSC clustering algorithm: K-PSC($\mathbf{X}, K, R, \sigma$)

Data: Time series $\mathbf{x}_i$, $i = 1, \ldots, N$, The number of clusters K, The number of pieces R, σ is the length scale of a RBF kernel

Initial cluster assignments $C = \{C_1, \ldots, C_K\}$;

$G = \text{ceil}\left(\frac{D}{R}\right)$;

$\mathbf{K} \leftarrow k_{[0,1]}(i, j)$, where $i, j = 1, \ldots, R$;

$\tilde{\mathbf{K}} \leftarrow \mathbf{K} \otimes \mathbf{I}_G$;

$$\mathbf{L} \leftarrow \text{diag}\left(\left[\underbrace{k(0,0), \ldots, k(0,0)}_{G}, \ldots, \underbrace{k(R,R), \ldots, k(R,R)}_{G}\right]\right);$$

repeat

  $\tilde{C} \leftarrow C$;

  for $j = 1$ **to** K **do**

    $\mathbf{M} \leftarrow \sum_{\mathbf{x}_i \in C_k} \left(\mathbf{L} - 2\mathbf{A}_i^T \tilde{\mathbf{K}} + \mathbf{A}_i^T \mathbf{L} \mathbf{A}_i\right)$;

    $\boldsymbol{\mu}_j \leftarrow$ The smallest eigenvector of $\mathbf{M}$;

    $C_j \leftarrow \emptyset$;

  end

  for $j = 1$ **to** N **do**

    $j^* \leftarrow \arg\min_{j=1,\ldots,K} d(\mathbf{x}_i, \boldsymbol{\mu}_j)$;

    $C_{j^*} \leftarrow C_{j^*} \cup \{i\}$

  end

until $\tilde{C} = C$;

return $C, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$

algorithms. The refinement step, on the other hand is different in all three algorithms. Finding the centroids μ_k of the cluster C_k leads to solving the following minimization problem

$$\boldsymbol{\mu}_k^* = \arg\min_{\boldsymbol{\mu}} \sum_{\mathbf{x}_i \in C_k} d^2(\mu, \mathbf{x}_i). \tag{2.53}$$

The solution of Eq. (2.53) should be computationally efficient, because the K-PSC recalculates the cluster centroids many times until it converges.

Next, we present the closed form solution of the Eq. (2.53). We combine Eqs. (2.51) and (2.53)

$$\boldsymbol{\mu}_k^* = \arg\min_{\boldsymbol{\mu}} \frac{1}{\|\boldsymbol{\mu}\|^2} \sum_{\mathbf{x}_i \in C_k} \left(\boldsymbol{\mu}^T \mathbf{L} \boldsymbol{\mu} - 2\alpha_i \mathbf{x}_i^T \tilde{\mathbf{K}} \mathbf{x}_i + \alpha_i^2 \mathbf{x}_i^T \mathbf{L} \mathbf{x}_i\right). \tag{2.54}$$

Then we replace α_i with Eq. (2.50)

$$\boldsymbol{\mu}_k^* = \arg\min_{\boldsymbol{\mu}} \frac{1}{\|\boldsymbol{\mu}\|^2} \sum_{\mathbf{x}_i \in C_k} \left(\boldsymbol{\mu}^T \mathbf{L} \boldsymbol{\mu} - 2\left(\frac{\boldsymbol{\mu}^T \tilde{\mathbf{K}} \mathbf{x}_i}{\beta}\right)^T \boldsymbol{\mu}^T \tilde{\mathbf{K}} \mathbf{x}_i + \left(\frac{\boldsymbol{\mu}^T \tilde{\mathbf{K}} \mathbf{x}_i}{\beta} \mathbf{x}_i\right)^T \mathbf{L} \left(\frac{\boldsymbol{\mu}^T \tilde{\mathbf{K}} \mathbf{x}_i}{\beta} \mathbf{x}_i\right)\right)$$

We flip the order of $\mathbf{x}_i^T \boldsymbol{\mu}^T \tilde{\mathbf{K}} \mathbf{x}_i$ and we have the simplified expression

$$\boldsymbol{\mu}_k^* = \arg\min_{\boldsymbol{\mu}} \frac{1}{\|\boldsymbol{\mu}\|^2} \sum_{\mathbf{x}_i \in C_k} \boldsymbol{\mu}^T \left(\mathbf{L} - 2\mathbf{A}_i^T \tilde{\mathbf{K}} + \mathbf{A}_i^T \mathbf{L} \mathbf{A}_i \right) \boldsymbol{\mu}$$

where $\mathbf{A}_i = \frac{\tilde{\mathbf{K}} \mathbf{x}_i^T \otimes \mathbf{x}_i}{\beta}$ and $\otimes$ is the outer product. Substituting $\sum_{\mathbf{x}_i \in C_k} \left(\mathbf{L} - 2\mathbf{A}_i^T \tilde{\mathbf{K}} + \mathbf{A}_i^T \mathbf{L} \mathbf{A}_i \right)$ by a matrix $\mathbf{M}$ we arrive at the following minimization problem

$$\boldsymbol{\mu}_k^* = \arg\min_{\boldsymbol{\mu}} \boldsymbol{\mu}^T \frac{\mathbf{M}}{2R\|\boldsymbol{\mu}\|^2} \boldsymbol{\mu}. \tag{2.55}$$

The solution to this minimization problem is the eigenvector $\mathbf{u}_m$ corresponding to the smallest eigenvalue λ_m of matrix $\mathbf{M}$ [GV12].

The proposed K-Piece Wise Spectral Centroid clustering (Algorithm 1) extends the K-SC algorithm [YL11] and can be consider as generalization.

3. Coarse Grained Analysis of Population

In section 2.3 are presented approximations method of the full GP model. The SPGP+HS model [SG12] reduces the computational complexity of the full GP model, and as a side effect provides the possibility to model heteroscedastic data sets. Our proposed model RSPGP+HS extends the SPGP+HS model by defining an uncertainty function of the pseudo-input points, and is more robust to overfitting. Having an uncertainty function RSPGP+HS also provides the option to interpret the parameters of the function of the noise.

In this chapter, we present and discuss the results of the forecasting from the daily postings behavior of the users of the Stackoverflow Q&A web platform. First we show an analysis of the periodicity of the time series of tag activity as apparent from the Stackoverflow data set. Next, we compare our prediction results with those of other models and discuss the advantages of introducing functional dependencies on noise terms.

3.1. Experimental Setup

The performances of our models are tested using publicly available data-dumps of Stackoverflow[1]. The data set is one dimensional, and it contains the number of questions and answers of postings classified by tag for every business day. The models are trained on a data sets containing information about the daily postings in the time between 01.02.2014 and 31.08.2014. The evaluation of the models is done by a validation set that has postings for the first 21 working days in September 2014.

3.2. Results

The performance of the presented models in section 2.3 highly depends on the choice of the kernels used for calculating the covariance matrix. When working with GP models, an additional analysis is required, to select the proper kernels for the covariance matrix. Because we work with a data set that contains some kind of user behavior, we suppose that it may have some periodicity in the behavior of the users. Thus, we have done *spectral density estimation* [MIK05; Ham94; OSB16] analysis of the time series using a *periodogram* analysis [MIK05; Ham94; OSB16]. This analysis shows the power (ampli-

[1]Downloadable URL: www.archive.org/details/stackexchange

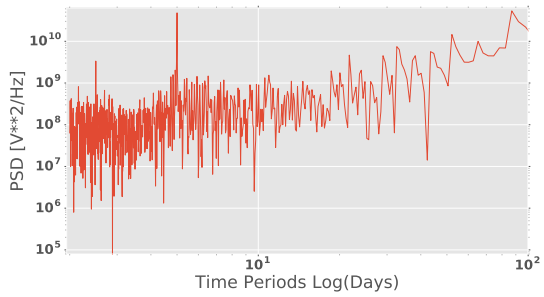


Figure 3.1.: Spectral Density Estimation of the Stackoverflow dataset using periodogram. We observe two peaks, one at two and a half days and the other at five days, where the latter peak is double the period of the former peak period.

	MSE				NLPD				NLML			
	GP	RBFSIN	HS	SIN	GP	RBFSIN	HS	SIN	GP	RBFSIN	HS	SIN
android	960.88	**692.03**	887.45	720.75	4.65	**4.49**	4.61	**4.49**	-1076.37	**-948.40**	-1149.22	-993.23
c#	1029.06	**881.11**	950.64	894.61	4.70	**4.62**	4.64	**4.62**	-1003.23	**-949.54**	-962.43	-961.62
c++	1216.94	**533.68**	5068.20	675.84	4.84	**4.45**	6.02	4.66	-717.14	**-698.50**	-756.95	-716.58
html	681.57	**678.19**	774.17	754.95	4.47	**4.45**	4.51	4.50	-841.93	**-784.78**	-798.28	-820.60
ios	2598.35	**1474.72**	3064.63	1660.90	5.82	**4.81**	5.53	4.86	-757.36	**-737.24**	-750.69	-740.49
java	1917.86	**1431.70**	3446.30	1782.17	5.12	**4.90**	5.79	4.95	-1098.13	**-1034.83**	-1087.29	-1068.30
javascript	2992.30	**1869.61**	2396.68	2102.05	6.09	**4.97**	5.52	5.22	-1493.42	**-883.31**	-1054.49	-1044.76
jquery	**808.26**	825.28	989.07	1163.88	**4.57**	4.77	4.69	4.73	-957.31	-932.99	**-866.17**	-862.45
php	5892.26	**907.07**	5379.89	2745.40	6.83	**4.60**	6.13	5.15	-1042.95	-883.65	-945.85	**-853.21**
python	**604.89**	702.25	744.28	881.65	**4.44**	4.58	4.53	4.62	**-782.68**	-842.76	-787.24	-788.14

Table 3.1.: Results showing the MSE and NLPD (smaller better) on 2014 question test set and NLML (larger is better) on 2014 question training set. GP indicates pure Gaussian process, HS indicates sparse pseudo-input Gaussian process with heteroscedastic noise, SIN-HS sparse pseudo-input Gaussian process with sine functional noise and RBFSIN-HS sparse pseudo-input Gaussian process with sine in combination with RBF kernel functional noise.

tude) of the time series as a function of frequency. In this way we are able to discover if there are some periodicities, and at what frequency they occur.

We present a periodogram of the time series data that we analyzed in Figure 3.1. Since all of our tag related time series have almost the same periodogram, we show only one of them. For the purpose of better interpretability of the periodogram, we have converted the frequencies into periods, in order to observe how many days a period lasts. There are two apparent peaks, where the first peak is at two and a half days, and the second peak is at five days. In this case the period of five days is appearing as an echo of the two and a half days period, therefore we dismiss the second period and we take into account only the first period. Additional characteristics of this data set are some irregularities as well as a long term rising trend in the overall time-series.

Having this information at hand, the models that have the best performance are using

	MSE				NLPD				NLML			
	GP	RBFSIN	HS	SIN	GP	RBFSIN	HS	SIN	GP	RBFSIN	HS	SIN
android	1097.05	1098.29	1041.58	**1031.10**	4.80	4.79	4.81	**4.78**	**-903.82**	-919.78	-913.40	-927.61
c#	2889.76	**2723.95**	2998.26	2878.46	5.24	**5.18**	5.24	5.22	-1007.62	**-983.75**	-989.81	-995.13
c++	1602.27	**1436.71**	3491.81	3010.85	4.89	**4.85**	6.21	5.15	-825.76	-805.98	-886.82	**-775.62**
html	**1856.82**	2016.96	2162.96	1904.25	4.98	4.99	5.02	**4.96**	-1082.09	-957.67	**-907.46**	-954.44
ios	3944.90	**1541.55**	5156.82	5017.53	5.74	**4.87**	5.48	5.41	-831.93	-839.15	-778.98	**-777.68**
java	3207.22	**2987.25**	4085.50	3090.13	5.38	**5.19**	5.35	5.20	-1283.56	**-1016.72**	-1024.00	-1047.48
javascript	5360.20	**4869.97**	5434.37	5374.24	5.61	**5.50**	5.68	5.51	-1141.66	**-1110.28**	-1139.14	-1131.77
jquery	1817.16	1728.42	1749.74	**1725.81**	5.12	5.03	5.07	**5.00**	**-976.82**	-1021.99	-1009.31	-1023.81
php	2950.13	**2948.65**	3076.88	2982.74	**5.16**	**5.16**	5.20	5.17	-1011.84	-1015.56	-995.81	**-994.36**
python	911.70	606.00	1660.13	**605.22**	**4.64**	**4.64**	4.90	**4.64**	-867.67	-820.73	**-792.96**	-813.29

Table 3.2.: Results showing the MSE and NLPD (smaller better) on 2014 answers test set and NLML (larger is better) on 2014 answers training set. GP indicates pure Gaussian process, HS indicates sparse pseudo-input Gaussian process with heteroscedastic noise, SIN-HS sparse pseudo-input Gaussian process with sine functional noise and RBFSIN-HS sparse pseudo-input Gaussian process with sine in combination with RBF kernel functional noise.

a covariance function that is a sum of four kernels

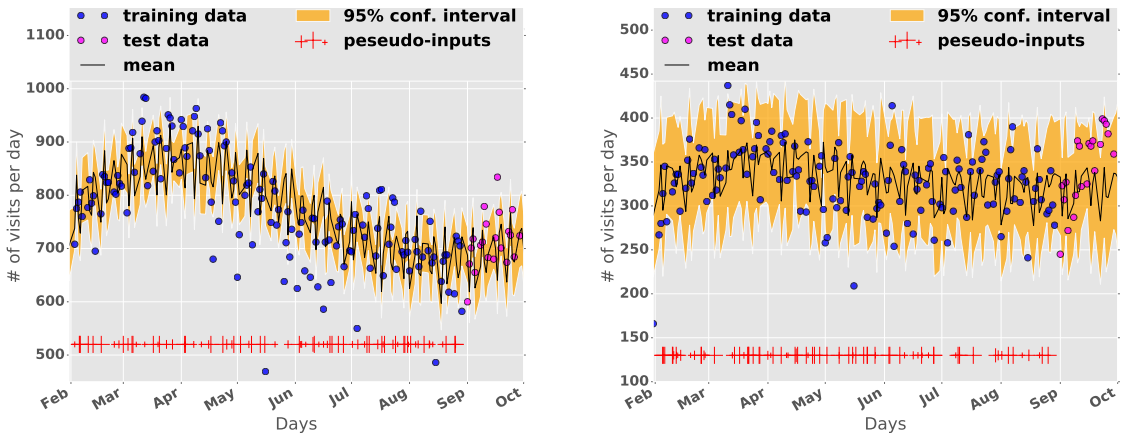
$$k(x, x\prime) = k_1(x, x\prime) + k_2(x, x\prime) + k_3(x, x\prime) + k_4(x, x\prime). \tag{3.1}$$

The topics of how to choose the kernels and the particular role of each kernel in the learned model is a subject of discussion in the next section.

We present the result achieved on the top ten tags according to the number of posted questions and answers in the 2014 Stackoverflow data set. In Table 3.1 are presented the results from the models modeling the posted questions time series, and in Table 3.2 are presented the results from the models modeling the posted answers. In order to compare the prediction models we have used the following measures:

- **Mean Square Error (MSE):** Accounts for the accuracy of the prediction for an unseen data
- **Negative Log Predictive Distribution (NLPD):** Accounts for the confidence of the predicted values on an unseen data
- **Negative Log Marginal Likelihood (NLML):** Accounts for how well the model fits the training data.

For the MSE and the NLPD measures, the smaller values are better, and for the NLML the larger values are better. The best model of each tag has been chosen using the Akaike information criterion (AIC) [Bis06]. Models with functional noise perform better in nine of the ten tags for the answer time series, and eight of the ten tags for the question time series. The superior performance of the SPGP+FUNC-HS over the full GP is due to the fact that the data set contains variable noise. Note that for this data set, SPGP+FUNC-HS performs better, because of the sparsity of the model and the additional functional



(a) "Java" **(b)** "iOS"

Figure 3.2.: Models learned with SPGP+SIN-HS for the "Java" and "iOS" tags for 2014 data set.

noise that is added to the pseudo-input points. The SPGP+HS performs worse than the best models, because associating only a positive vector of uncertainty to the pseudo-input points, increases the flexibility of the covariance function. This can lead during search for an optimal hyperparameter values using gradient ascent, to overfitting and converging to a sub optimal maxima. Using a functional noise constrains the optimization space and removes the unwanted local maximas, resulting with much faster converge of the gradient ascent into a good local maxima. The drawback is that the function of the noise should follow the distribution of the noise in the data set, otherwise the model will perform poorly. This is probably the case why the SPGP+FUNC-HS performs worse on the one tag for the answers and the two tags for the questions. In Figure 3.2 we present two learned models, one for the "Java" tag (Figure 3.2a) and one for the "iOS" tag (Figure 3.2b). One can see from the figures, that the model that models the "Java" tag, strives to predict the test point with the mean. In contrast to the "Java" tag model, the model of the "iOS" tag predicts the test points as a noise.

3.3. Analysis of the Learned Kernels Parameters

The different kernels trained in Eq. (3.1) allows us to dissect the dynamical behavior of the population in different scales and patterns. In order to portrait these behaviors, we calculated the mean function and variance Eq. (2.19) by generating vector $\mathbf{k}_*^\top$ with the independent kernels. We present the values of each kernel in the "android" question data set in Figure 3.3

- **Mean trends** (Figure 3.3a): are characterizing the behavior of the population of the users over the length scales in months. They represent the global mean behavior of the population. We hypothesize that it is driven by the shear size of

the user base. The more people interested in the tag visiting the site, the more average number of questions posted per month. Further, this overall trend might represent the changes in the dominance of this particular tag of questions in the data set. Because the tags represent programming languages, these trends indicate changes in the dominance of a language. This is modeled by the use of the rational quadratic kernel that we define as

$$k_1(x, x') = \theta_6^2 \Big(1 + \frac{(x - x')^2}{2\theta_8 \theta_7^2}\Big)^{-\theta_8} \tag{3.2}$$

- **Seasonal trends** (Figure 3.3b): these arise in a time scale smaller than the major trends and posses both a periodical and a stochastic nature. They represent changes in the population behavior trough the different months of the year. These trends are shown with the Ornstein-Ulenbeck kernel which is defined as:

$$k_2(x, x') = \theta_1 \exp(-\frac{|x - x'|}{\theta_2}). \tag{3.3}$$

- **Weekly periods** (Figure 3.3c): as obtained from the periodogram, this kernel guarantees the weekly patterns of the users, and represent the fine grained periods of our data set. We hypothesize that this behavior is related to the natural patterns of a fatigue during the working week. The analytical form of the kernel is presented in Eq. (3.4).

$$k_3(x, x') = \theta_3^2 \exp\Big(L_1 + L_2\Big) \tag{3.4}$$

where we define L_1 and L_2 as

$$L_1 = -\frac{(x - x')^2}{2\theta_4^2} \tag{3.5}$$

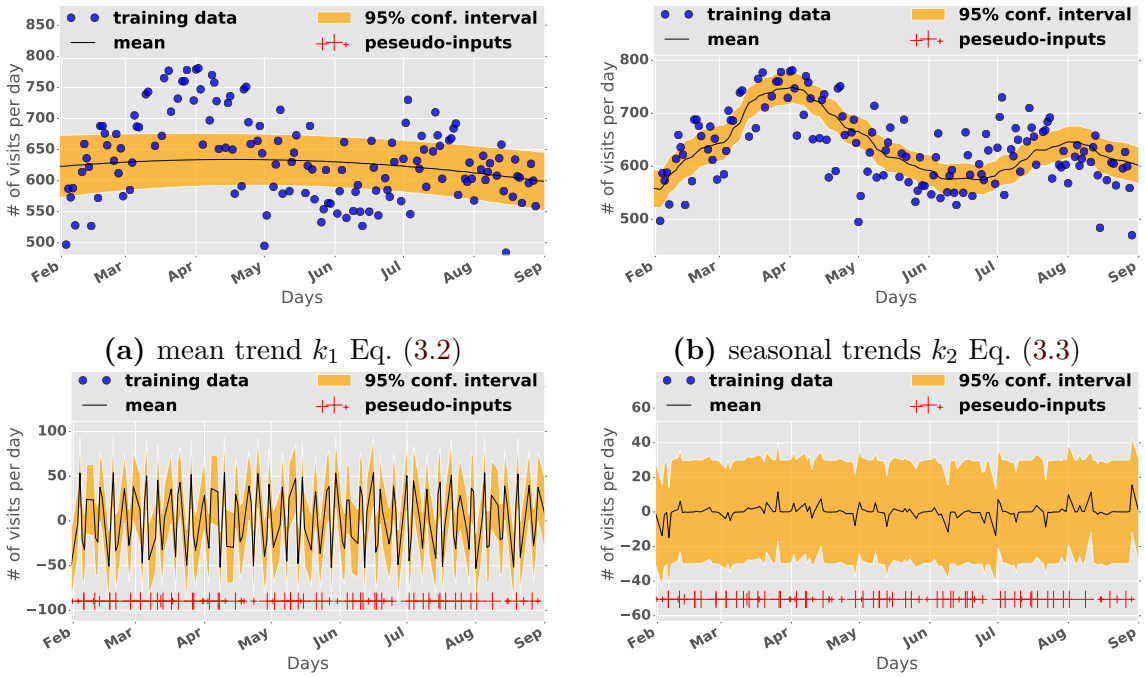
and

$$L_2 = -\frac{2\sin^2[\pi(x - x')/P]}{\theta_5^2}. \tag{3.6}$$

- **Weekly noise** (Figure 3.3d): These are fluctuations appearing in the weakly behavior, which are natural to expect, due to the statistical nature of our data set. Randomness in the behavioral pattern of each user might give rise to such fluctuations. The analytical form of the kernel is defined as:

$$k_4(x, x') = \theta_9^2 \Big(1 + \frac{(x - x')^2}{2\theta_{11}\theta_{10}^2}\Big)^{-\theta_{11}} \tag{3.7}$$

The models learned are modeling the collective behavior of all users for a given tag. The model presented in Figure 3.3 answers the question of the collective dynamics of the system, however each of the sub figure also poses at leas one additional question. For example, in Figure 3.3a we do not know if the mean emerges from the noise users



(a) mean trend k_1 Eq. (3.2)

(b) seasonal trends k_2 Eq. (3.3)

(c) weekly periods k_3 Eq. (3.4)

(d) weekly noise k_4 Eq. (3.7)

Figure 3.3.: Decomposition of the SPGP+SIN-HS model for the "android" tags in the different kernels. We observe four main behaviors: mean trends, seasonal trends, weekly periods and weekly noise.

(users that have posted only once in the system) or from the regular users. Also, the seasonality of the system presented in Figure 3.3d does not give us information of its origin. Is the seasonality emergent, or there are users that work seasonal?

Answering these, and similar to these questions, is not possible with the currently used model, because it does not provide us with enough detailed description. We must analyze the behavior of the system at much finer level. In the next chapter, we present results and analysis of the system on user level, i.e. we model the behavior of each individual user.

4. Fine Grained Analysis of Population

Modeling the collective users behavior, in the Stackoverflow web platform, provides an insight into the general tendencies and the dynamics of the system. The results analysis, described in the previous chapter, provided us with answers to some questions related to the collective behavior of the system. However, various additional questions have arisen, and answering them have required an additional, very meticulous analysis of the whole system. In order for us to do that, we have done a finer system analysis, through constructing a behavioral model for every single user.

In this chapter we analyze the behavior of the users in the Stackoverflow web platform. In section 4.1 are presented learned models of the behavior of each individual user, and in section 4.2 are presented the common behavioral patterns between the users.

4.1. User Behavior Models Results

Using GPs for modeling the behavior of each user require having enough data for creating time series. Around 6% of the users of the top ten tags in Stackoverflow dataset have more than ten arrivals. Thus, it is not possible to create appropriate time series and then model them with a GP model. However, the Poisson Point process for modeling the behavior of each user, already presented in section 2.4, offers a solution to this problem. More specifically we use the scalable nonparametric Bayesian inference on Poisson process with Gaussian process [SR14]. This is an inhomogeneous Poisson process with a GP modeling the arrival rate λ. We use the SPGP+FUNC-HS model, and the hyperparameters are learned using Monte Carlo. The scalability of the model as well as for the flexibility that offers the GP makes this method attractive. The end result of the user behavior model depends on the choice of a covariance function of the GP process inside the Poisson point process. For modeling the behavior of all users we use the square exponential kernel Eq. (2.24).

In Fig. 4.1 are presented four models of users behavior, modeled with a Poisson point process. For every single user its arrivals, and the intensity function of the Poisson process are present. The number, and the distribution of the arrivals, highly are results of a process defined with the intensity function. Namely, if we integrate the intensity function of a Poisson point process, it yields a value that is equal to the number of the arrivals of the user.

The shape of an intensity function depends on the distribution of the arrivals, and the distribution of the arrivals we assume is defined by two competing processes. The first process defines the dynamics of a user, namely how often a user decides to post i.e. how often the user works. The second process defines the intensity of the postings

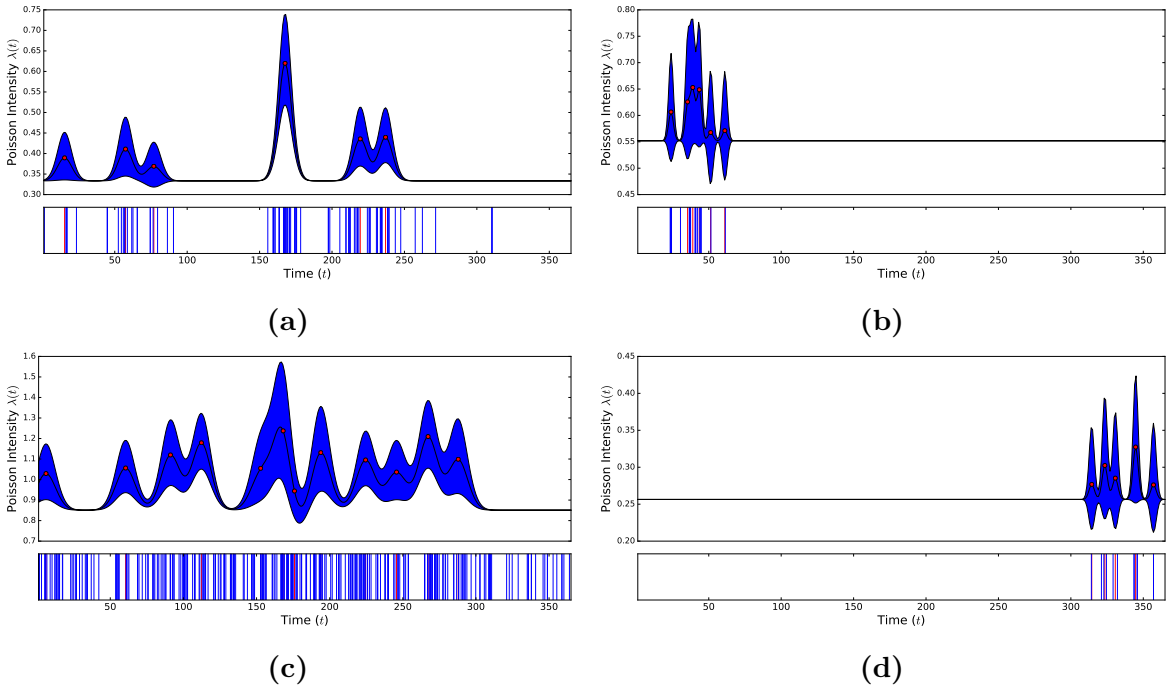


Figure 4.1.: Intensities of Poisson point process models and arrivals of four user from the Stackoverflow dataset. Read points are called induced points, and are used for approximating the full Poisson point process. Bayesian optimization method is used for finding the location of the induced points. The shaded area represents the point-wise mean plus and minus two times the standard deviation for each input value (corresponding to the 95% confidence region)

once a user decided to post. The intensity function models both process simultaneously, without excluding one or the other. However, which process dominates and is shown in more details by the intensity function, highly depends on the number of arrivals, and the delta between the posts. The intensity of the picks in the intensity function corresponds to the delta between the arrivals, and the smaller the delta the larger the pick. Having too few arrivals forces the Poisson point process method to model the process on the basis of the intensity of the postings, once a user decided to post. The users presented in Figures 4.1a and 4.1c have larger number of arrivals, so one can see, that the intensity function models more the process of how often a user decided to post instead of the process of the intensity of the users postings. In contrast to it however, the users presented in Figures 4.1b and 4.1d have quite fewer arrivals in comparison to the other two Figures, so the intensity function models more the process of the intensity of the users postings.

Creating intensity functions for every user provide the possibility of analyzing the behavior of the user. However, modeling the behavior of the user without splitting the two processes, does not provide an absolutely clear picture of the behavior of the user. Mixing the two processes in the model of the users behavior also influences the common users behavior picture, because we would have to compare between models of users that

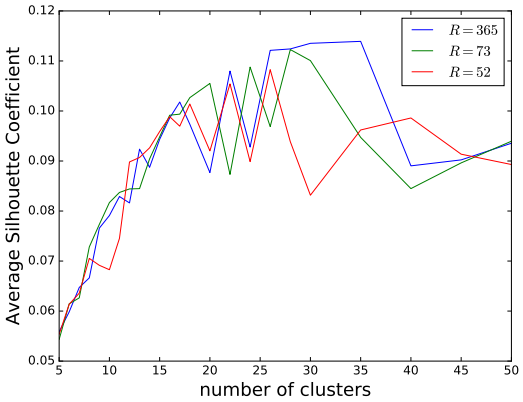


Figure 4.2.: Average Silhouette Coefficient using DPT similarity measure for different values of R (number of pieces).

model two different process.

4.2. Common Patterns in the Users Behavior

Finding common behavior patterns between the users is related to clustering the number of postings in time. In our case however, the data is too sparse, so we cluster the intensity functions of the Poisson point process for every user, and as a clustering algorithm we use the K-PSC algorithm already presented in section 2.5.3. The K-PSC algorithm, like all the other variants of the K-means algorithm, requires the number of clusters to be specified beforehand. But because there is no a unique solution of how to choose the number of clusters beforehand, we first measure the quality of the clustering by running the K-PSC algorithm multiple times with different number of clusters, and we measure the Average Silhouette [KR09]. In Figure 4.2 are presented three Average Silhouette Coefficient as a function of the number of clusters, for different values of R. The higher value of the Average Silhouette Coefficient is the better. All three functions are similar, however the function of the Average Silhouette Coefficient for the DPT similarity with value of $R = 365$ shows best results. The Average Silhouette Coefficient for the number of clusters between twenty five and thirty five has the highest value. Although, the highest value of the Average Silhouette Coefficient is achieved in this range, we choose the number of clusters to be $K = 16$, instead of the number of clusters between twenty five and thirty five, because they are just variants of the sixteen clusters.

In Figure 4.3 are presented the centroids of the clusters obtained using the K-PSC clustering algorithm for $K = 16$. One can see from Figure 4.3 that all the clusters have a single peak except the cluster presented in Figure 4.3a. The picks differ from each other by their width, the time when they appear, and the time when they peak. Almost all of the clusters have equal size, on average of 5% of the total number of users.

Although, a seasonal trend of the collective users behavior, in the coarse grained analysis presented in chapter 3 is observed, here in the fine grained analysis of the

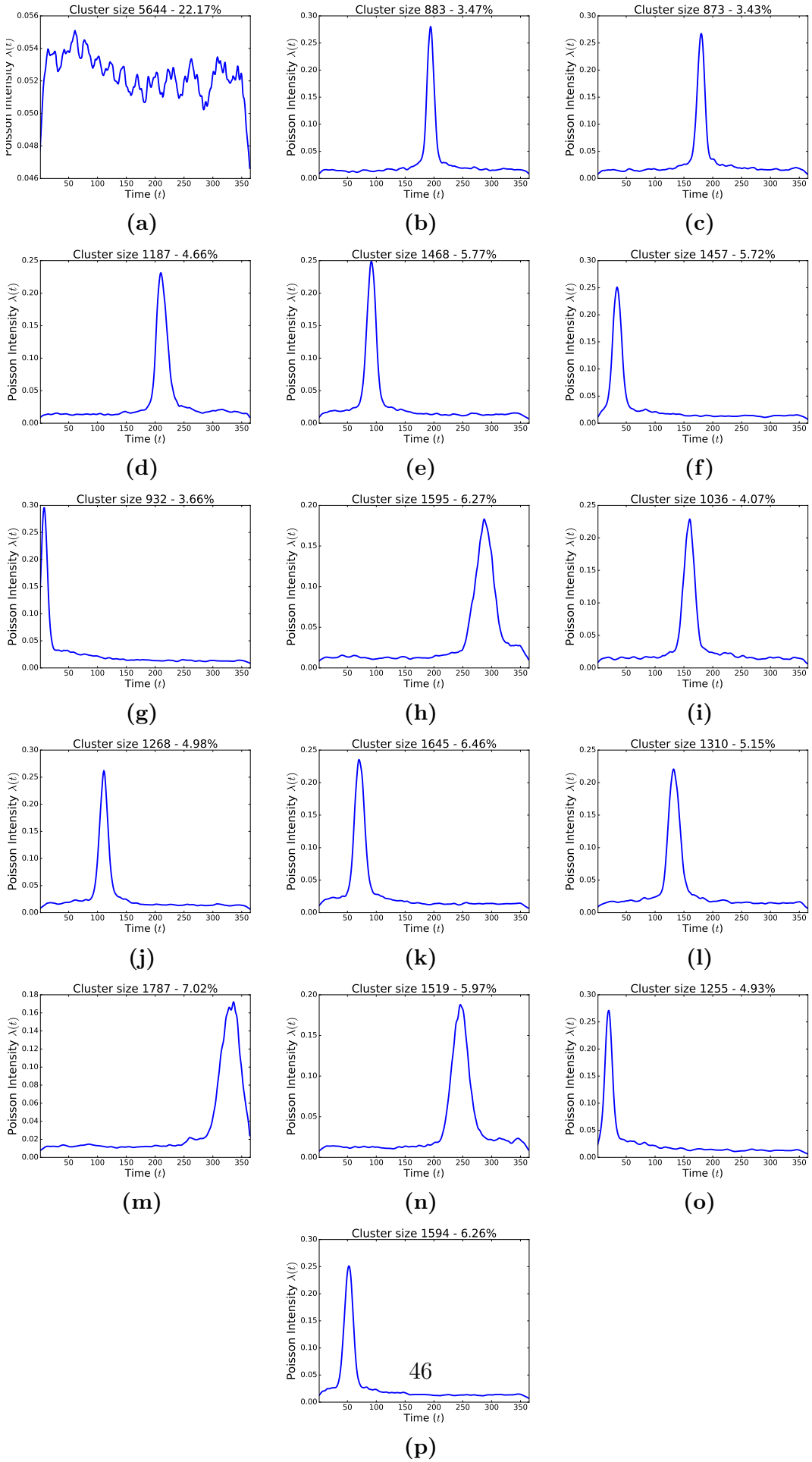

Cluster size 5644 - 22.17%
Cluster size 883 - 3.47%
Cluster size 873 - 3.43%
Cluster size 1187 - 4.66%
Cluster size 1468 - 5.77%
Cluster size 1457 - 5.72%
Cluster size 932 - 3.66%
Cluster size 1595 - 6.27%
Cluster size 1036 - 4.07%
Cluster size 1268 - 4.98%
Cluster size 1645 - 6.46%
Cluster size 1310 - 5.15%
Cluster size 1787 - 7.02%
Cluster size 1519 - 5.97%
Cluster size 1255 - 4.93%
Cluster size 1594 - 6.26%
Poisson Intensity $\lambda(t)$
Time (t)
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)
(k)
(l)
(m)
(n)
(o)
(p)

Figure 4.3.

common users patterns, a seasonal behavior of a single user is not shown.

What also catches an eye by analyzing the shown results, is that most of the users work for a short period of time in the year, usually between one to three months. This could be also related to a well known phenomenon that users are the most active in the months when searching for a job. Since we cluster the users in a relatively small number of clusters, and most of the users are active for a short period of time in the year, the users who are continuously active are averaged out by the behavior of the most users and their behavior is not strongly presented by the shown cluster centroids.

The size of the cluster presented by the cluster centroid in Figure 4.3a has almost a quarter of the total number of users, and the shape of the centroid drastically differ than the rest of the cluster centroids. The uneven size of this cluster compared to the other clusters is resulting from the large number of users, who are active for only a couple of days or for a week at most. Hence, the users corresponding intensity function has a single sharp peak, and different users have function peaks at a different point of time. These intensities can not be clustered in one of the other fifteen clusters. We have tried to reduce the size of this cluster by increasing the number of clusters. However, no matter how big the number of the clusters is, one cluster is always much larger than the others. In appendix C we present the cluster centroids of clustering $K = 150$.

5. Conclusion and Feature Work

In this study we have analyzed the temporal dynamics of the Stackoverflow users. For achieving that, we have taken an approach of analyzing the time-series of posted questions and answers related to a specific tag. Our analysis is divided into two parts: coarse grained analysis and fine grained analysis.

In the coarse grained analysis part we study the collective behavior of users that have posted questions or answers related to a specific tag. In order to accomplish this task, we have extended the variable noise pseudo inputs Gaussian Process model by introducing a functional noise variant. The idea of using functional descriptions of noise has offered us the possibility to study periodic patterns in collective attention shifts, and was also found to act as a regularizer in model training. Our extended Gaussian Process framework, with functional representations of various kinds of noise, provides an additional advantage of increased interpretability of the results, as the different kernels, which are defined solely for this purpose, uncover different kinds of dynamics. In particular, our kernels revealed major distinct characteristics of the question - answering behavior of the users. Firstly, there are major trends shown on time scales of about six months, showing an increase as well as a decrease of interest in particular topics or corresponding tags. Secondly, these major trends are perturbed by seasonal behavior, for instance, overall activities usually drop during the summer season. Thirdly, on a fine grained scale, there are weekly patterns characterized by periods of 2.5 days. Finally, there are noisy fluctuations in activities on daily scales.

In the fine grained analysis part we study the temporal dynamics of the system on more detailed level, namely we search for the common patterns of the daily posting behavior of the users. The sparsity of the data on user level force us to define a Poisson point process for each user, and the intensity function of this process models the behavior of each individual user. Through clustering the intensity function of each user we obtain the common patterns of the daily posting behavior of the users. For clustering the users intensity functions we propose the scale invariant Piecewise dynamic similarity measure and the K-PSC clustering algorithm, which provide us with much more descriptive clusters than the K-Means algorithm. From the obtained clusters we can see that most of the users work only for a short period of time during the year, mainly one to three months. This could be related to a well known trend that users are the most active in the months when searching for a job. Since we cluster the users in a relatively small number of clusters, and most of the users are active for a short period of time in the year, the users who are continuously active are averaged out by the behavior of the majority of the users and their behavior is not strongly presented by the shown cluster centroids. Also, we did not observe any seasonality to appear when analyzed on a user level, as we observed by the analysis of the collective user behavior. Therefore, we can conclude

that the seasonality that appears in the collective behavior is emergent.

Given the models and results presented in this study, we propose couple various directions for future work. First goal is dividing the assumed two processes defining how often a user is active and with what intensity. Making a distinction between these two processes, when we model the users behavior using the Poisson point process will make clustering the intensities more precise, and will presumably show similarity in the sizes of the clusters. This will be tested by modeling the whole set of Stackoverflow users. Second goal is implementing a distributed Gaussian Process framework, in order to extend our approach towards massive amounts of behavioral data (use of tags, comments, and likes) that can be retrieved from similar social media platforms such as Twitter or Facebook.

A. Mathematical Background

A.1. Matrix Properties

The Woodbury matrix identity is

$$\left(A + UBU^T\right)^{-1} = A^{-1} - A^{-1}U\left(B^{-1} + U^T A^{-1}U\right)^{-1} U^T A^{-1} \tag{A.1}$$

where A, U, C and V all denote matrices of the correct (conformable) sizes.

Determinant of sum of matrices can be expressed as:

$$|A + UBU^T| = |A||B||B^{-1} + U^T A^{-1}U| \tag{A.2}$$

The derivative of the inverse of a matrix can be expressed

$$\frac{\partial}{\partial x}\left(\mathbf{A}^{-1}\right) = -\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\mathbf{A}^{-1}. \tag{A.3}$$

The derivative of the log determinant of a matrix can be expressed

$$\frac{\partial}{\partial x}\ln|\mathbf{A}| = \mathtt{Tr}\left(\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\right). \tag{A.4}$$

A.2. Gaussian Distribution

Given a marginal Gaussian distribution for x and conditional Gaussian distribution for y given x in the form

$$p\left(\mathrm{x}\right) = \mathcal{N}\left(\mathrm{x} \mid \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right) \tag{A.5}$$

$$p\left(\mathrm{y} \mid \mathrm{x}\right) = \mathcal{N}\left(\mathrm{y} \mid \mathbf{A}\mathrm{x} + \mathrm{b}, \mathbf{L}^{-1}\right) \tag{A.6}$$

the marginal distribution of y and the conditional distribution of y are given by

$$p\left(\mathrm{y}\right) = \mathcal{N}\left(\mathrm{y} \mid \mathbf{A}\boldsymbol{\mu} + \mathrm{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T\right) \tag{A.7}$$

$$p\left(\mathrm{x} \mid \mathrm{y}\right) = \mathcal{N}\left(\mathrm{x} \mid \boldsymbol{\Sigma}\left\{\mathbf{A}^T\mathbf{L}\left(\mathrm{y} - \mathrm{b}\right) + \boldsymbol{\Lambda}\boldsymbol{\mu}\right\}, \boldsymbol{\Sigma}\right) \tag{A.8}$$

where

$$\boldsymbol{\Sigma} = \left(\boldsymbol{\Lambda} + \mathbf{A}^T\mathbf{L}\mathbf{A}\right)^{-1} \tag{A.9}$$

If we have joint Gaussian distribution $\mathcal{N}(\mathrm{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\Lambda} \equiv \boldsymbol{\Sigma}^{-1}$ and we define the following partitions

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{pmatrix}, \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix} \tag{A.10}$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}, \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix} \tag{A.11}$$

then the conditional distribution $p(\mathbf{x}_a \mid \mathbf{x}_b)$ is given by

$$p(\mathbf{x}_a \mid \mathbf{x}_b) = \mathcal{N}\left(\mathbf{x} \mid \boldsymbol{\mu}_{a|b}, \boldsymbol{\Lambda}_{aa}^{-1}\right) \tag{A.12}$$

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1} \boldsymbol{\Lambda}_{ab} (\mathbf{x}_b - \boldsymbol{\mu}_b) \tag{A.13}$$

and the marginal distribution $p(\mathbf{x}_a)$

$$p(\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_a \mid \boldsymbol{\mu}_a, \boldsymbol{\Sigma}_{aa}). \tag{A.14}$$

B. Gaussian Process Derivations

B.1. Derivation of the Sparse Input Gaussian Process with Functional Variable Noise

In this section we will present the detailed derivation of the predictive distribution, marginal distribution, as well as calculation of gradients of the hyperparameters. Because, we are also implementing this model, we shall use the Cholesky factorization in the derivation for better performance in the implementation.

Lets consider the data set $\mathcal{D}$ consisting of N input vectors $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ of dimension D and corresponding real valued targets $\mathbf{y} = \{y_n\}_{n=1}^N$. The distribution of the target value at the new point is:

$$p\left(y \mid \mathbf{x}, \mathcal{D}, \theta\right) = \mathcal{N}\left(y \mid \mathbf{k_x}^T \left(\mathbf{K}_N + \sigma^2\mathbf{I}\right)^{-1} \mathbf{y}, K_{\mathbf{xx}} - \mathbf{k_x}^T \left(\mathbf{K}_N + \sigma^2\mathbf{I}\right)^{-1} \mathbf{k_x} + \sigma^2\right) \quad \text{(B.1)}$$

Consider a pseudo data set $\overline{\mathcal{D}}$ of size $M < N$: pseudo inputs $\overline{X} = \{\overline{\mathbf{x}}_m\}_{m=1}^M$ and pseudo targets $\overline{\mathbf{f}} = \{\overline{f}_m\}_{m=1}^M$. The single data point likelihood has the following form:

$$p\left(y \mid \mathbf{x}, \overline{\mathbf{X}}, \overline{\mathbf{f}}\right) = \mathcal{N}\left(y \mid \mathbf{k_x}^T \mathbf{K}_M^{-1} \overline{\mathbf{f}}, K_{\mathbf{xx}} - \mathbf{k_x}^T \mathbf{K}_M^{-1} \mathbf{k_x} + \sigma^2\right) \quad \text{(B.2)}$$

where $[K_M]_{mm'} = K\left(\overline{\mathbf{x}}_m, \overline{\mathbf{x}}_{m'}\right)$ and $[\mathbf{k_x}]_m = K\left(\overline{\mathbf{x}}_m, \mathbf{x}\right)$, for $m = 1, \ldots, M$.

The target data are generated i.i.d given the inputs, then the complete data likelihood has the following form:

$$p\left(\mathbf{y} \mid \mathbf{x}, \overline{\mathbf{X}}, \overline{\mathbf{f}}\right) = \prod_{n=1}^{N} p\left(y_n \mid \mathbf{x}_n, \overline{\mathbf{X}}, \overline{\mathbf{f}}\right) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{K}_{NM}\mathbf{K}_M^{-1}\overline{\mathbf{f}}, \sigma^2\mathbf{\Gamma}\right) \quad \text{(B.3)}$$

where $\sigma^2\mathbf{\Gamma} = \mathbf{\Lambda} + \sigma^2\mathbf{I}_N$, $\mathbf{\Lambda} = \operatorname{diag}\left(\mathbf{K}_N - \mathbf{Q}_N\right)$, $[\mathbf{K}_{NM}]_{nm} = K\left(\mathbf{x}_n, \overline{\mathbf{x}}_m\right)$ and $\mathbf{Q}_N = \mathbf{K}_{NM}\mathbf{K}_M^{-1}\mathbf{K}_{MN}$. We define Gaussian prior on the pseudo targets:

$$p\left(\overline{\mathbf{f}} \mid \overline{\mathbf{X}}\right) = \mathcal{N}\left(\overline{\mathbf{f}} \mid 0, \mathbf{H}\right) \quad \text{(B.4)}$$

where $\mathbf{H} = \mathbf{K}_M + \operatorname{diag}\left(\mathbf{f}_h\left(\overline{\mathbf{x}}_m\right)\right)$, and $m = 1, \ldots, M$.

To find the posterior distribution over pseudo targets $\overline{\mathbf{f}}$ we use Bayes rule on Eq. (B.3) and (B.4):

$$p\left(\overline{\mathbf{f}} \mid \mathcal{D}, \overline{\mathbf{X}}\right) = p\left(\mathbf{y} \mid \mathbf{x}, \overline{\mathbf{X}}, \overline{\mathbf{f}}\right) p\left(\overline{\mathbf{f}} \mid \overline{\mathbf{X}}\right) = \mathcal{N}\left(\overline{\mathbf{f}} \mid \mu_{\overline{\mathbf{f}}}, \mathbf{\Sigma}_{\overline{\mathbf{f}}}\right). \quad \text{(B.5)}$$

The covariance of the posterior distribution of pseudo targets is:

$$\begin{aligned}
\boldsymbol{\Sigma}_{\overline{\mathbf{f}}}^{-\mathbf{1}} &= \boldsymbol{\Lambda} + \mathbf{A}^T\mathbf{L}\mathbf{A} \\
&= \mathbf{H}^{-1} + \mathbf{K}_M^{-1}\mathbf{K}_{MN}\sigma^{-2}\boldsymbol{\Gamma}^{-1}\mathbf{K}_{NM}\mathbf{K}_M^{-1} \\
&= \mathbf{H}^{-1} + L^{-T}L^{-1}\mathbf{K}_{MN}\sigma^{-2}\boldsymbol{\Gamma}^{-1}\mathbf{K}_{NM}L^{-T}L^{-1} \\
&= \sigma^{-2}\mathbf{L}^{-T}\mathbf{M}\mathbf{L}^{-1} \qquad \text{(B.6)} \\
\boldsymbol{\Sigma}_{\overline{\mathbf{f}}} &= \sigma^{-2}\mathbf{L}\mathbf{M}^{-1}\mathbf{L}^T \qquad \text{(B.7)}
\end{aligned}$$

where $\mathbf{M} = \sigma^2\mathbf{L}^T\mathbf{H}^{-1}\mathbf{L} + \mathbf{V}\boldsymbol{\Gamma}^{-1}\mathbf{V}^T$, $\mathbf{K}_M = \mathbf{L}\mathbf{L}^T$ and $\mathbf{V} = \mathbf{L}^{-1}\mathbf{K}_{MN}$.

The mean of the posterior distribution of pseudo targets is:

$$\begin{aligned}
\mu_{\overline{\mathbf{f}}} &= \boldsymbol{\Sigma}\left(\mathbf{A}^T\mathbf{L}\left(\mathbf{y}-\mathbf{b}\right)+\boldsymbol{\Lambda}\mu\right) \\
&= \sigma^2\mathbf{L}\mathbf{M}^{-1}\mathbf{L}^T\mathbf{K}_M^{-1}\mathbf{K}_{MN}\sigma^{-2}\Gamma^{-1}\mathbf{y} \\
&= \mathbf{L}\mathbf{M}^{-1}\mathbf{V}\boldsymbol{\Gamma}^{-1}\mathbf{y} \qquad \text{(B.8)}
\end{aligned}$$

Given a new input point $\mathbf{x}_*$, the predictive distribution is then obtained by integrating the likelihood Eq. (B.2) with the posterior Eq. (B.5):

$$p\left(y_* \mid \mathbf{x}_*, \mathcal{D}, \overline{\mathbf{X}}\right) = \int p\left(y_* \mid \mathbf{x}_*, \overline{\mathbf{X}}, \overline{\mathbf{f}}\right)p\left(\overline{\mathbf{f}} \mid \mathcal{D}, \overline{\mathbf{X}}\right)d\overline{\mathbf{f}} = \mathcal{N}\left(y_* \mid \mu_*, \sigma_*^2\right). \qquad \text{(B.9)}$$

The mean of the predictive distribution is defined as:

$$\begin{aligned}
\mu_* &= \mathbf{A}\mu + \mathbf{b} \\
&= \mathbf{K}_{*M}\mathbf{K}_M^{-1}\mathbf{L}\mathbf{M}^{-1}\mathbf{V}\Gamma^{-1}\mathbf{y} \\
&= \boldsymbol{\ell}_{M*}^T\beta, \qquad \text{(B.10)}
\end{aligned}$$

and the covariance of the predictive distribution is defined as:

$$\begin{aligned}
\sigma_*^2 &= \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T \\
&= k_{**} - \mathbf{K}_{*M}\mathbf{K}_M^{-1}\mathbf{K}_{M*} + \sigma^2 + \mathbf{K}_{*M}\mathbf{K}_M^{-1}\sigma^2\mathbf{L}\mathbf{M}^{-1}\mathbf{L}^T\mathbf{K}_M^{-1}\mathbf{K}_{M*} \\
&= k_{**} - \boldsymbol{\ell}_*^T\boldsymbol{\ell}_* + \sigma^2 + \sigma^2\boldsymbol{\ell}_*^T\mathbf{L}_M^{-T}\mathbf{L}_M^{-1}\boldsymbol{\ell}_* \\
&= k_{**} - \|\boldsymbol{\ell}_*\|^2 + \|\boldsymbol{\ell}_{M*}\|^2 + \sigma^2 \qquad \text{(B.11)}
\end{aligned}$$

where $\mathbf{M} = \mathbf{L}_M\mathbf{L}_M^T$, $\boldsymbol{\ell}_* = \mathbf{L}^{-1}\mathbf{K}_{M*}$, $\boldsymbol{\ell}_{M*} = \mathbf{L}_M^{-1}\boldsymbol{\ell}_*$ and $\boldsymbol{\beta} = \mathbf{L}_M^{-1}\mathbf{V}\boldsymbol{\Gamma}^{-1}\mathbf{y}$

The marginal likelihood of the model is obtained by integrating the complete data likelihood Eq. (B.3) with the prior distribution over the pseudo targets Eq. (B.4):

$$p\left(\mathbf{y} \mid \mathbf{x}, \overline{\mathbf{X}}\right) = \int p\left(\mathbf{y} \mid \mathbf{x}, \overline{\mathbf{X}}, \overline{\mathbf{f}}\right)p\left(\overline{\mathbf{f}} \mid \overline{\mathbf{X}}\right)d\overline{\mathbf{f}} = \mathcal{N}\left(\mathbf{y} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \qquad \text{(B.12)}$$

where the mean and the covariance of the distribution are defined as:

$$\begin{aligned}
\boldsymbol{\mu} &= \mathbf{0} \qquad \text{(B.13)} \\
\boldsymbol{\Sigma} &= \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T \\
&= \sigma^2\boldsymbol{\Gamma}\mathbf{K}_{NM}\mathbf{K}_M^{-1}\mathbf{H}\mathbf{K}_M^{-1}\mathbf{K}_{NM} \qquad \text{(B.14)}
\end{aligned}$$

Calculation of the gradients of the marginal likelihood with respect to hyperparameters is much easier if we take the negative log of the marginal likelihood and minimize the marginal likelihood. The negative log marginal likelihood has the following form:

$$p\left(\mathbf{y} \mid \mathbf{x}, \overline{\mathbf{X}}\right) = \underbrace{-\frac{1}{2}\ln|\mathbf{\Sigma}|}_{\mathcal{L}_1}\underbrace{-\frac{1}{2}\mathbf{y}^T\mathbf{\Sigma}\mathbf{y}}_{\mathcal{L}_2} - \frac{N}{2}\ln(2\pi) \tag{B.15}$$

Transforming the $\mathcal{L}_1$ using the Eq. (A.2) we have the following form:

$$\begin{aligned} 2\mathcal{L}_1 &= \ln|\sigma^2\mathbf{\Gamma} + \mathbf{K}_{NM}\mathbf{K}_M^{-1}\mathbf{H}\mathbf{K}_M^{-1}\mathbf{K}_{MN}| \\ &= \ln|\sigma^2\mathbf{\Gamma}||\mathbf{H}||\mathbf{H}^{-1} + \mathbf{K}_M^{-1}\mathbf{K}_{MN}\sigma^{-2}\mathbf{\Gamma}^{-1}\mathbf{K}_{NM}\mathbf{K}_M^{-1}| \\ &= \ln|\sigma^2\mathbf{\Gamma}| + \ln|\mathbf{H}| + \ln|\sigma^{-2}\mathbf{K}_M^{-1}\mathbf{A}\mathbf{K}_M^{-1}| \\ &= \ln|\mathbf{\Gamma}| + \ln|\mathbf{H}| + \ln|\mathbf{K}_M^{-1}\mathbf{A}\mathbf{K}_M^{-1}| + \ln\sigma^{2(N-M)} \\ &= \ln|\mathbf{\Gamma}| + \ln|\mathbf{H}| - 2\ln|\mathbf{K}_M| + \ln|\mathbf{A}| + (N-M)\ln\left(\sigma^2\right), \end{aligned} \tag{B.16}$$

using the Woodbury matrix identity Eq. (A.1) the second part of negative log marginal likelihood $\mathcal{L}_2$ we have transformed to:

$$\begin{aligned} 2\mathcal{L}_2 &= \mathbf{y}^T\left(\sigma^2\mathbf{\Gamma} + \mathbf{K}_{NM}\mathbf{K}_M^{-1}\mathbf{H}\mathbf{K}_M^{-1}\mathbf{K}_{MN}\right)\mathbf{y} \\ &= \mathbf{y}^T\left(\sigma^{-2}\mathbf{\Gamma}^{-1} - \sigma^{-2}\mathbf{\Gamma}^{-1}\mathbf{K}_{NM}\mathbf{K}_M^{-1}\left(\mathbf{H}^{-1} + \mathbf{K}_M\mathbf{K}_{MN}\sigma^{-2}\mathbf{\Gamma}^{-1}\mathbf{K}_{NM}\mathbf{K}_M^{-1}\right)\mathbf{K}_M^{-1}\mathbf{K}_{MN}\sigma^{-2}\mathbf{\Gamma}^{-1}\right)\mathbf{y} \\ &= \sigma^{-2}\left(\mathbf{y}^T\mathbf{\Gamma}^{-1}\mathbf{y} - \mathbf{y}^T\mathbf{\Gamma}^{-1}\mathbf{K}_{NM}\mathbf{A}^{-1}\mathbf{K}_{MN}\mathbf{\Gamma}^{-1}\mathbf{y}\right) \\ &= \sigma^{-2}\left(\|\underline{\mathbf{y}}\|^2 - \|\mathbf{L}_A^{-1}\underline{\mathbf{K}}_{MN}\underline{\mathbf{y}}\|^2\right), \end{aligned} \tag{B.17}$$

where $\underline{\mathbf{y}} = \mathbf{L}_{\mathbf{\Gamma}}^{-1}\mathbf{y}$, $\underline{\mathbf{K}}_{NM} = \mathbf{L}_{\mathbf{\Gamma}}^{-1}\mathbf{K}_{NM}$, $\underline{\mathbf{K}}_M = \mathbf{L}_{\mathbf{H}}^{-1}\mathbf{K}_M$, $\mathbf{\Gamma} = \mathbf{L}_{\mathbf{\Gamma}}\mathbf{L}_{\mathbf{\Gamma}}^T$, $\mathbf{H} = \mathbf{L}_{\mathbf{H}}\mathbf{L}_{\mathbf{H}}^T$ and $\mathbf{A} = \sigma^2\mathbf{K}_M\mathbf{H}^{-1}\mathbf{K}_M + \mathbf{K}_{MN}\mathbf{\Gamma}^{-1}\mathbf{K}_{NM} = \sigma^2\underline{\mathbf{K}}_M^T\underline{\mathbf{K}}_M + \underline{\mathbf{K}}_{MN}\underline{\mathbf{K}}_{NM}$.

B.2. Gradient Calculation of the Negative Log Marginal Likelihood of the Sparse Input Gaussian Process with Functional Variable Noise

In this section we will present the calculations of the gradients of the negative log marginal likelihood with respect to the hyperparameters. The calculations of the gradients are in general form and the exact form of the gradients will depend on the choice of the covariance function and the function modeling the variable noise. The gradient of the first part $\mathcal{L}_1$ of the log marginal likelihood has the following form:

$$\begin{aligned} \dot{\mathcal{L}}_1 &= \frac{1}{2}\ln|\mathbf{\Gamma}| + \frac{1}{2}\ln|\mathbf{H}| + \frac{1}{2}\ln|\mathbf{K}_M| + \frac{1}{2}\ln|\mathbf{A}| \\ &= \frac{1}{2}\mathrm{Tr}\left(\mathbf{\Gamma}^{-1}\dot{\mathbf{\Gamma}}\right) + \frac{1}{2}\mathrm{Tr}\left(\mathbf{H}^{-1}\dot{\mathbf{H}}\right) + \frac{1}{2}\mathrm{Tr}\left(\mathbf{K}_M^{-1}\dot{\mathbf{K}}_M\right) + \frac{1}{2}\mathrm{Tr}\left(\mathbf{A}^{-1}\dot{\mathbf{A}}\right) \end{aligned} \tag{B.18}$$

where,

$$\begin{aligned}\dot{\mathbf{\Gamma}} &= \mathbf{I}_N + \sigma^{-2}\mathbf{\Lambda} \\ &= \mathbf{I}_N + \sigma^{-2}\text{diag}\left(\mathbf{K}_N - \mathbf{K}_{NM}\mathbf{K}_M^{-1}\mathbf{K}_{MN}\right) \\ &= \sigma^{-2}\text{diag}\left(\dot{\mathbf{K}}_N - 2\dot{\mathbf{K}}_{NM}\mathbf{K}_M^{-1}\mathbf{K}_{MN} + \mathbf{K}_{NM}\mathbf{K}_M^{-1}\dot{\mathbf{K}}_M\mathbf{K}_M^{-1}\mathbf{K}_{MN}\right), \end{aligned} \tag{B.19}$$

$$\dot{\underline{\underline{\mathbf{\Gamma}}}} = \sigma^{-2}\text{diag}\left(\mathbf{\Gamma}^{-\frac{1}{2}}\dot{\mathbf{K}}_N\mathbf{\Gamma}^{-\frac{1}{2}} - 2\dot{\underline{\mathbf{K}}}_{NM}\mathbf{K}_M^{-1}\underline{\mathbf{K}}_{MN} + \underline{\mathbf{K}}_{NM}\mathbf{K}_M^{-1}\dot{\mathbf{K}}_M\mathbf{K}_M^{-1}\underline{\mathbf{K}}_{MN}\right), \tag{B.20}$$

$$\begin{aligned}\dot{\mathbf{A}} &= \sigma^2\dot{\mathbf{K}}_M\mathbf{H}^{-1}\mathbf{K}_M - \sigma^{-2}\mathbf{K}_M\mathbf{H}^{-1}\dot{\mathbf{H}}\mathbf{H}^{-1}\mathbf{K}_M + \sigma^2\mathbf{K}_M\mathbf{H}^{-1}\dot{\mathbf{K}}_M \\ &+ \dot{\mathbf{K}}_{MN}\mathbf{\Gamma}^{-1}\mathbf{K}_{NM} - \mathbf{K}_{NM}\mathbf{\Gamma}^{-1}\dot{\mathbf{\Gamma}}\mathbf{\Gamma}^{-1}\mathbf{K}_{NM} + \mathbf{K}_{MN}\mathbf{\Gamma}^{-1}\dot{\mathbf{K}}_{NM} \\ &= \sigma^2\dot{\underline{\mathbf{K}}}_M\underline{\mathbf{K}}_M - \sigma^{-2}\underline{\mathbf{K}}_M\dot{\underline{\underline{\mathbf{H}}}}\underline{\mathbf{K}}_M + 2\dot{\underline{\mathbf{K}}}_{MN}\underline{\mathbf{K}}_{NM} - \underline{\mathbf{K}}_{MN}\dot{\underline{\underline{\mathbf{\Gamma}}}}\underline{\mathbf{K}}_{NM}, \end{aligned} \tag{B.21}$$

$$\dot{\mathbf{H}} = \dot{\mathbf{K}}_M + \text{diag}\left(\dot{\mathbf{f}}_h\left(\overline{\mathbf{x}}_m\right)\right) \tag{B.22}$$

$$\dot{\underline{\underline{\mathbf{H}}}} = \mathbf{L}_{\mathbf{H}}^{-1}\dot{\mathbf{H}}\mathbf{L}_{\mathbf{H}}^{-T} \tag{B.23}$$

$$\dot{\underline{\mathbf{K}}}_M = \mathbf{L}_{\mathbf{H}}^{-1}\dot{\mathbf{K}}_M \tag{B.24}$$

$$\dot{\underline{\mathbf{K}}}_{NM} = \mathbf{\Gamma}^{-\frac{1}{2}}\dot{\mathbf{K}}_{NM} \tag{B.25}$$

The gradients of the second part $\mathcal{L}_1$ of the negative log marginal likelihood has the following form:

$$\begin{aligned}\dot{\mathcal{L}}_2 &= \frac{\sigma^{-2}}{2}\left(\mathbf{y}^T\mathbf{\Gamma}^{-1}\mathbf{y} - \mathbf{y}^T\mathbf{\Gamma}^{-1}\mathbf{K}_{NM}\mathbf{A}^{-1}\mathbf{K}_{MN}\mathbf{\Gamma}^{-1}\mathbf{y}\right) \\ &= \sigma^{-2}\Bigg(- \frac{1}{2}\underline{\mathbf{y}}^T\dot{\underline{\underline{\mathbf{\Gamma}}}}\underline{\mathbf{y}} + \underline{\mathbf{y}}^T\dot{\underline{\underline{\mathbf{\Gamma}}}}\underline{\mathbf{K}}_{NM}\mathbf{A}^{-1}\underline{\mathbf{K}}_{MN}\underline{\mathbf{y}} - \underline{\mathbf{y}}^T\dot{\underline{\mathbf{K}}}_{NM}\mathbf{A}^{-1}\underline{\mathbf{K}}_{MN}\underline{\mathbf{y}} \\ &+ \frac{1}{2}\underline{\mathbf{y}}^T\underline{\mathbf{K}}_{NM}\mathbf{A}^{-1}\dot{\mathbf{A}}\mathbf{A}^{-1}\underline{\mathbf{K}}_{MN}\underline{\mathbf{y}}\Bigg) \\ &= \sigma^{-2}\Bigg[- \frac{1}{2}\underline{\mathbf{y}}^T\dot{\underline{\underline{\mathbf{\Gamma}}}}\underline{\mathbf{y}} + \left(\mathbf{L}_A^{-1}\underline{\mathbf{K}}_{MN}\underline{\mathbf{y}}\right)^T\Bigg(\frac{1}{2}\mathbf{L}_A^{-1}\dot{\mathbf{A}}\mathbf{L}_A^{-T}\left(\mathbf{L}_A^{-1}\underline{\mathbf{K}}_{MN}\underline{\mathbf{y}}\right) \\ &+ \mathbf{L}_A^{-1}\underline{\mathbf{K}}_{MN}\dot{\underline{\underline{\mathbf{\Gamma}}}}\underline{\mathbf{y}} - \mathbf{L}_A^{-1}\dot{\underline{\mathbf{K}}}_{MN}\underline{\mathbf{y}}\Bigg)\Bigg] \end{aligned} \tag{B.26}$$

B.3. Kernels Derivatives

In this section we will present the calculations of the kernel derivatives with respect to hyperparameters $\boldsymbol{\theta}$ and the pseudo-inputs $\overline{\mathbf{X}}$. The exact form of the kernel derivatives depends on the exact form of the covariance function we have chosen. In the work we have done we have used four different kernels. The derivatives of the squared exponential kernel Eq. (2.24) has the following forms:

- with respect to the length scale l:

$$\frac{\partial k_{sq}(x,y)}{\partial l} = \frac{c^2}{l^3}(x-y)^2 e^{-\frac{(x-y)^2}{2l^2}} \tag{B.27}$$

- with respect to the amplitude c:

$$\frac{\partial k_{sq}(x,y)}{\partial c} = 2ce^{-\frac{(x-y)^2}{2l^2}} \tag{B.28}$$

- with respect to x:

$$\frac{\partial k_{sq}(x,y)}{\partial x} = -\frac{c^2}{2l^2}(2x-2y) e^{-\frac{(x-y)^2}{2l^2}} \tag{B.29}$$

- with respect to y:

$$\frac{\partial k_{sq}(x,y)}{\partial y} = -\frac{c^2}{2l^2}(-2x+2y) e^{-\frac{(x-y)^2}{2l^2}}. \tag{B.30}$$

The derivatives of the rational quadratic kernel Eq. (2.25) has the following forms:

- with respect to the length scale l:

$$\frac{\partial k_{rq}(x,y)}{\partial l} = \frac{c^2\left(1+\frac{(x-y)^2}{2\alpha l^2}\right)^{-\alpha}(x-y)^2}{l^3\left(1+\frac{(x-y)^2}{2\alpha l^2}\right)} \tag{B.31}$$

- with respect to the amplitude c:

$$\frac{\partial k_{rq}(x,y)}{\partial c} = 2c\left(1+\frac{(x-y)^2}{2\alpha l^2}\right)^{-\alpha} \tag{B.32}$$

- with respect to α:

$$\frac{\partial k_{rq}(x,y)}{\partial \alpha} = c^2\left(1+\frac{(x-y)^2}{2\alpha l^2}\right)^{-\alpha}\left(-\log\left(1+\frac{(x-y)^2}{2\alpha l^2}\right)+\frac{(x-y)^2}{2\alpha l^2\left(1+\frac{(x-y)^2}{2\alpha l^2}\right)}\right) \tag{B.33}$$

- with respect to x:

$$\frac{\partial k_{rq}(x,y)}{\partial x} = -\frac{c^2\left(1+\frac{(x-y)^2}{2\alpha l^2}\right)^{-\alpha}(2x-2y)}{2l^2\left(1+\frac{(x-y)^2}{2\alpha l^2}\right)} \tag{B.34}$$

- with respect to y:

$$\frac{\partial k_{rq}(x,y)}{\partial y} = -\frac{c^2\left(1+\frac{(x-y)^2}{2\alpha l^2}\right)^{-\alpha}(-2x+2y)}{2l^2\left(1+\frac{(x-y)^2}{2\alpha l^2}\right)}. \tag{B.35}$$

The derivatives of the Ornstein-Ulenbeck kernel Eq. (2.27) has the following forms:

- with respect to the length scale l:

$$\frac{\partial k_{ou}(x,y)}{\partial l} = \frac{c}{l^2}e^{-\frac{1}{l}|x-y|}|x-y| \tag{B.36}$$

- with respect to the amplitude c:

$$\frac{\partial k_{ou}(x,y)}{\partial c} = e^{-\frac{1}{l}|x-y|} \tag{B.37}$$

- with respect to x:

$$\frac{\partial k_{ou}(x,y)}{\partial x} = -\frac{ce^{-\frac{1}{l}|x-y|}}{l|x-y|}\left((\Re x-\Re y)\frac{d}{dx}\Re x+(\Im x-\Im y)\frac{d}{dx}\Im x\right) \tag{B.38}$$

- with respect to y:

$$\frac{\partial k_{ou}(x,y)}{\partial y} = -\frac{ce^{-\frac{1}{l}|x-y|}}{l|x-y|}\left(-(\Re x-\Re y)\frac{d}{dy}\Re y-(\Im x-\Im y)\frac{d}{dy}\Im y\right). \tag{B.39}$$

The derivatives of the exponential periodic kernel Eq. (2.28) defined in section has the following forms:

- with respect to the length scale l:

$$\frac{\partial k_{per}(x,y)}{\partial l} = \frac{4c^2}{l^3}e^{-\frac{2}{l^2}\sin^2\left(\frac{\pi}{p}(x-y)\right)}\sin^2\left(\frac{\pi}{p}(x-y)\right) \tag{B.40}$$

- with respect to the amplitude c:

$$\frac{\partial k_{per}(x,y)}{\partial c} = 2ce^{-\frac{2}{l^2}\sin^2\left(\frac{\pi}{p}(x-y)\right)} \tag{B.41}$$

- with respect to x:

$$\frac{\partial k_{per}(x,y)}{\partial x} = -\frac{4\pi c^2}{l^2 p}e^{-\frac{2}{l^2}\sin^2\left(\frac{\pi}{p}(x-y)\right)}\sin\left(\frac{\pi}{p}(x-y)\right)\cos\left(\frac{\pi}{p}(x-y)\right) \tag{B.42}$$

- with respect to y:

$$\frac{\partial k_{per}(x,y)}{\partial y} = \frac{4\pi c^2}{l^2 p}e^{-\frac{2}{l^2}\sin^2\left(\frac{\pi}{p}(x-y)\right)}\sin\left(\frac{\pi}{p}(x-y)\right)\cos\left(\frac{\pi}{p}(x-y)\right). \tag{B.43}$$

C. Fine Grained Analysis Clusters

We present here the cluster centroids obtained through clustering the users of the Stackoverflow web platform, who have posted an answer to a question that is related to one of the top ten tags in 2014, and amount to around 29 000 users. They are clustered in 150 clusters. Alongside the centroids, 10 users who are randomly chosen from the cluster, and correspond to each centroid, are also presented. Same as the clustering presented in chapter 4, here a cluster with a size much bigger than the sizes of the other clusters also appears. These clusters however are much more descriptive than the clusters presented in the chapter 4, because the size of the clusters is much smaller, which on the other hand corresponds to a smaller averaging out. By fifteen of the one hundred and fifty centroids two picks have occurred. In Figure **??** are presented the centroids obtained by the clustering.

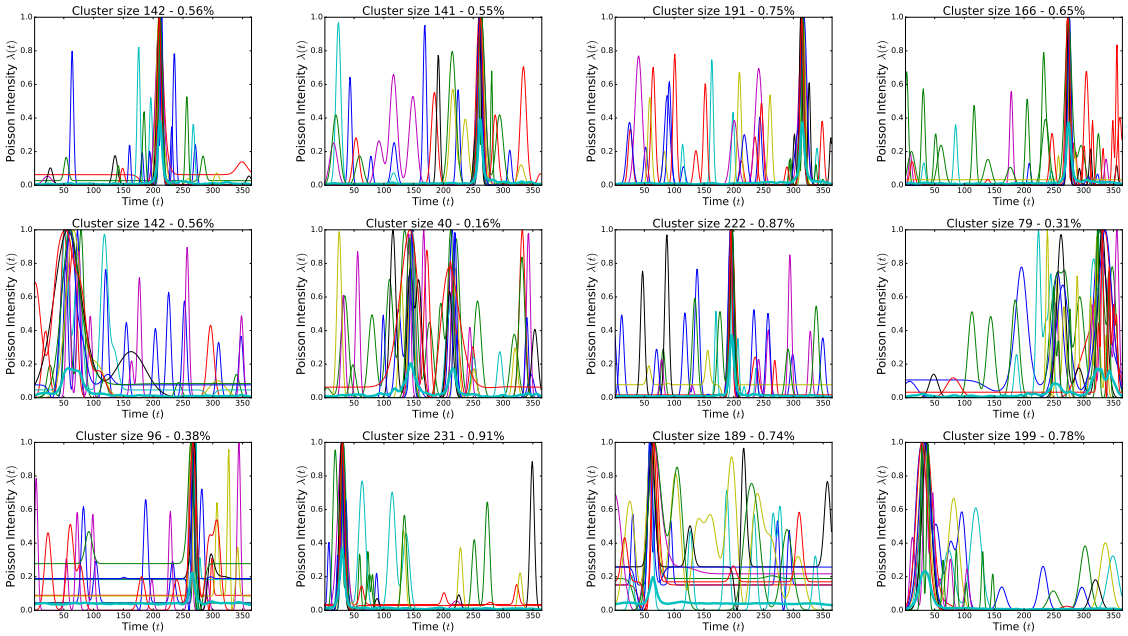


Figure C.1.: Centroids resulted from clustering ($K = 150$) the Stackoverflow users who posted an answer to a question related to the top ten tags in 2014. With every centroid, 10 users randomly chosen from the corresponding cluster are also presented.

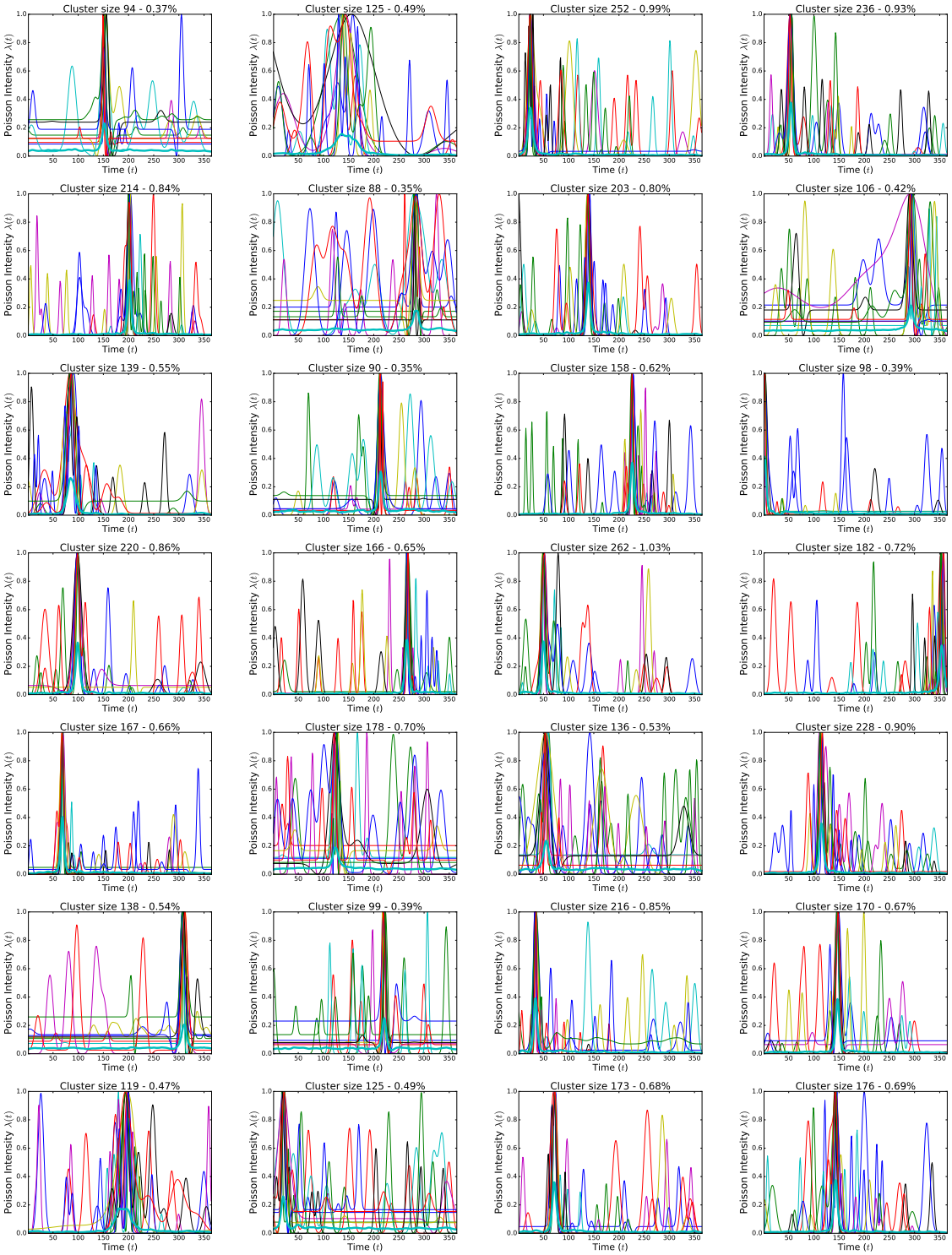


Figure C.1.: Centroids resulted from clustering ($K = 150$) the Stackoverflow users who posted an answer to a question related to the top ten tags in 2014. With every centroid, 10 users randomly chosen from the corresponding cluster are also presented.

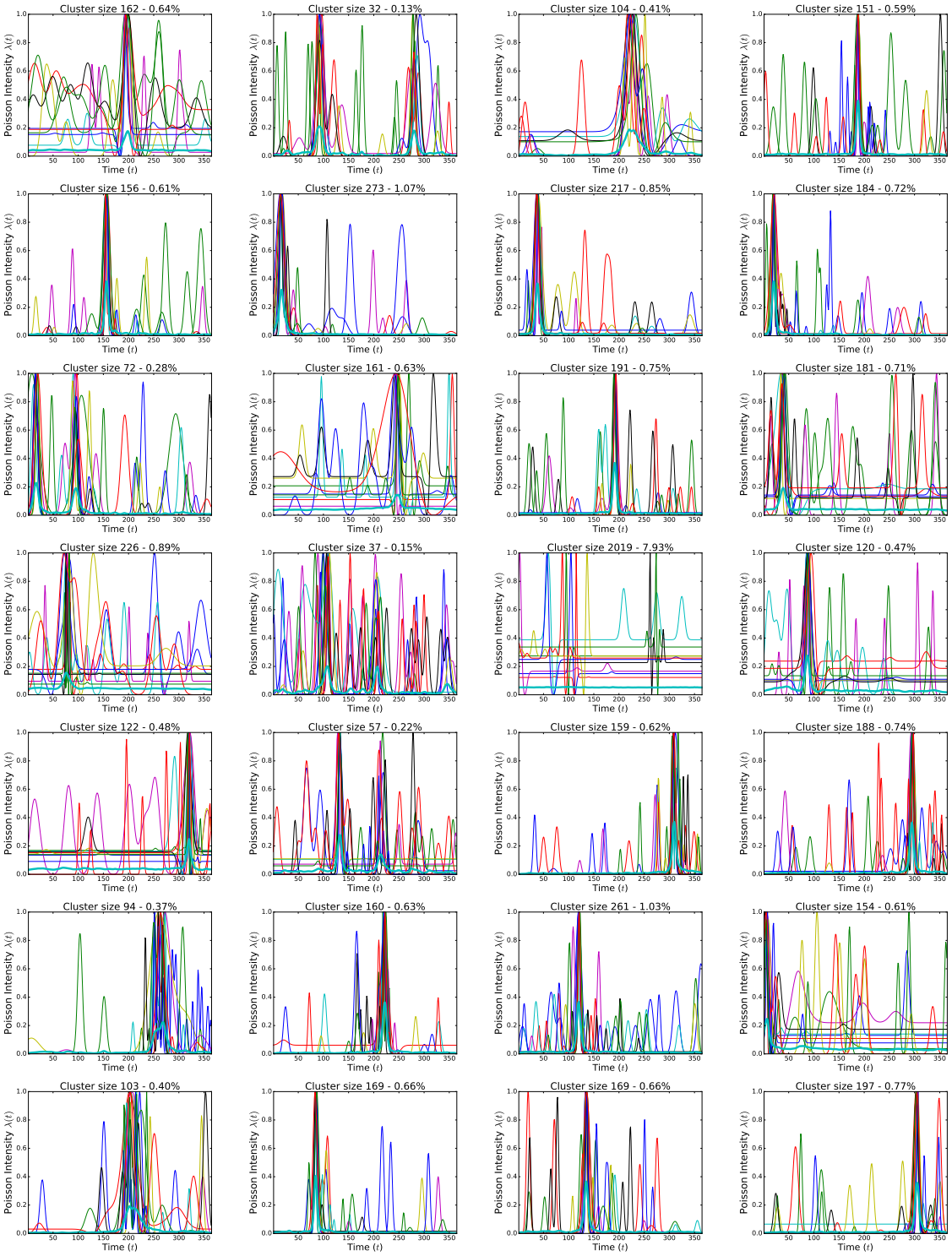


Figure C.1.: Centroids resulted from clustering ($K = 150$) the Stackoverflow users who posted an answer to a question related to the top ten tags in 2014. With every centroid, 10 users randomly chosen from the corresponding cluster are also presented.

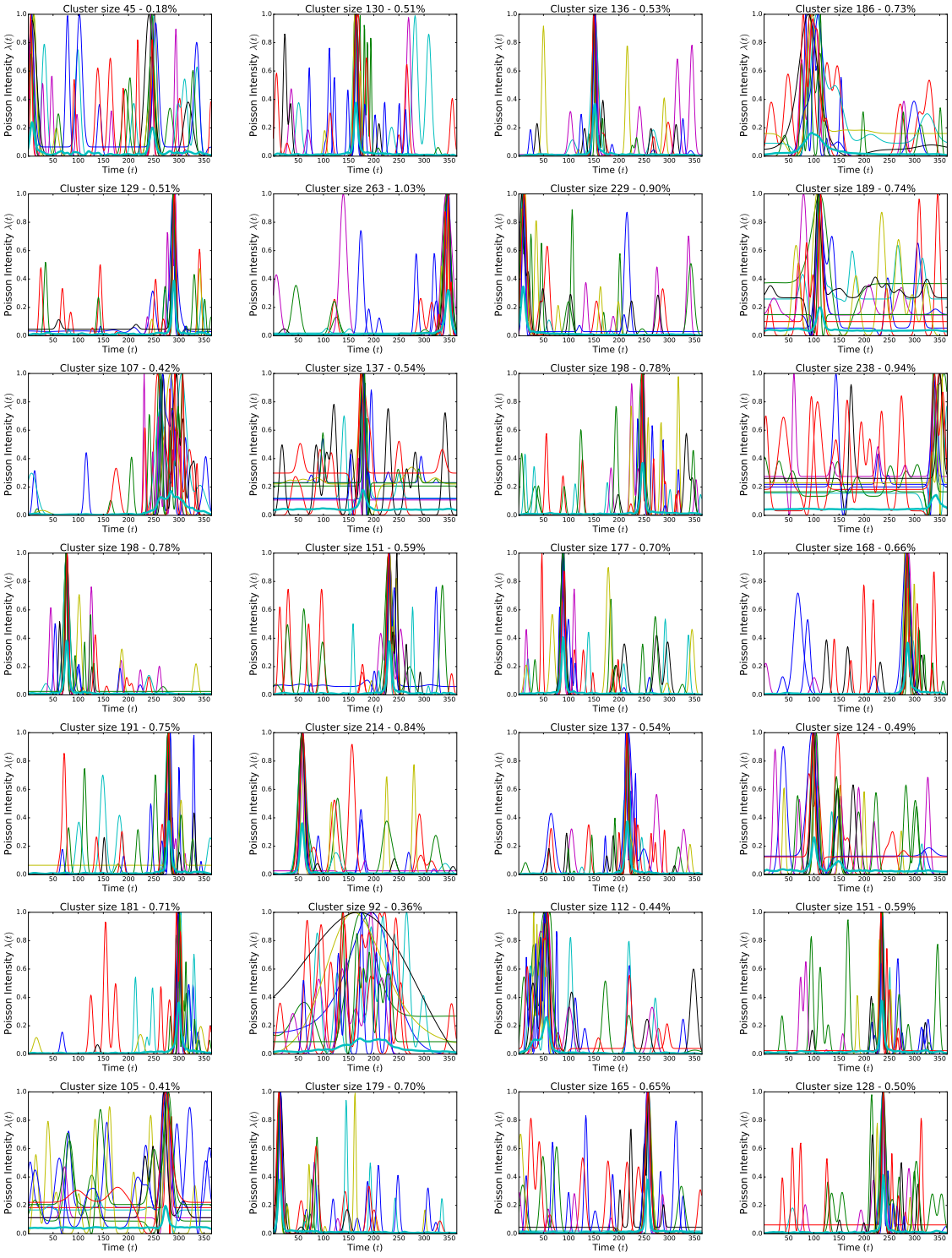


Figure C.1.: Centroids resulted from clustering ($K = 150$) the Stackoverflow users who posted an answer to a question related to the top ten tags in 2014. With every centroid, 10 users randomly chosen from the corresponding cluster are also presented.

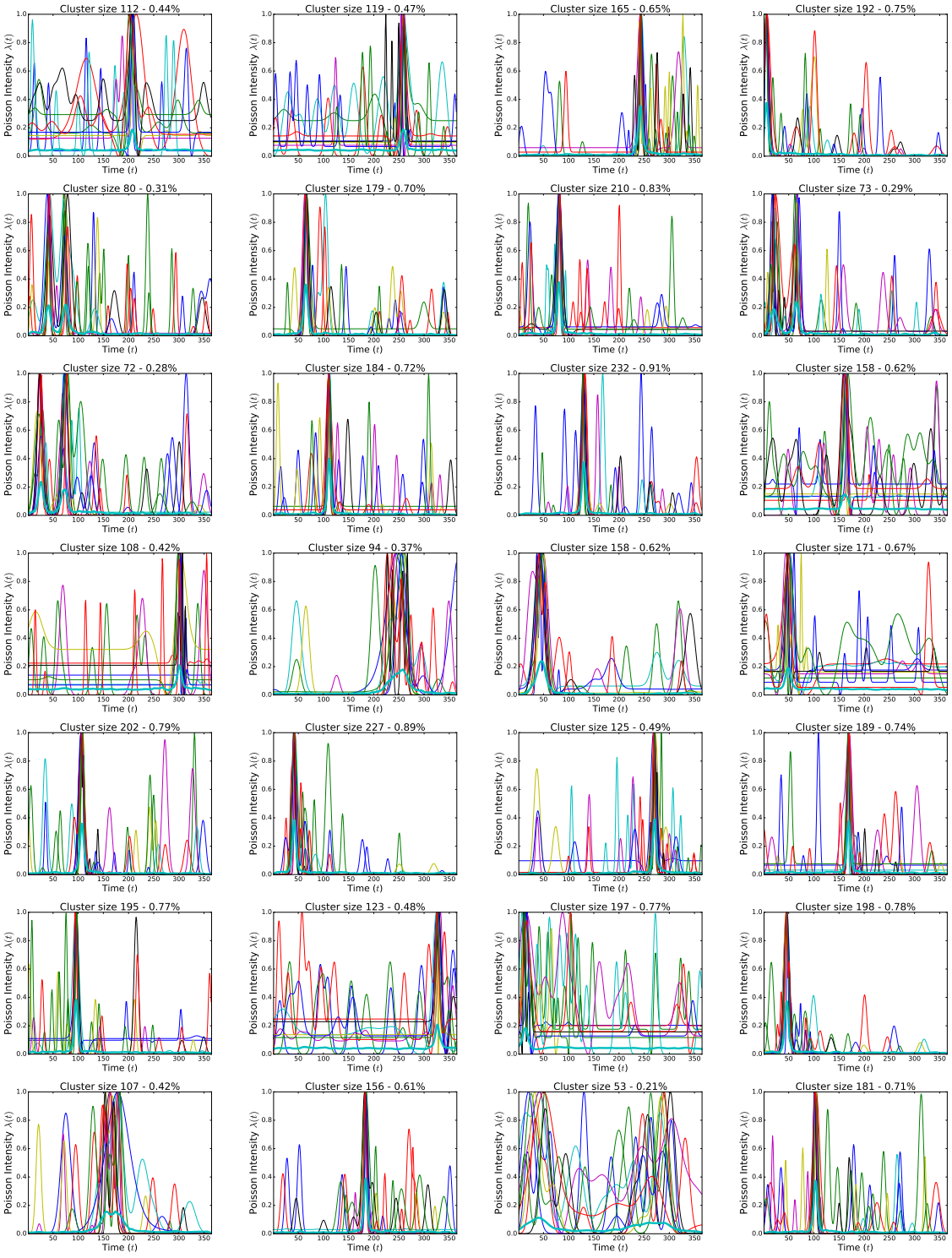


Figure C.1.: Centroids resulted from clustering ($K = 150$) the Stackoverflow users who posted an answer to a question related to the top ten tags in 2014. With every centroid, 10 users randomly chosen from the corresponding cluster are also presented.

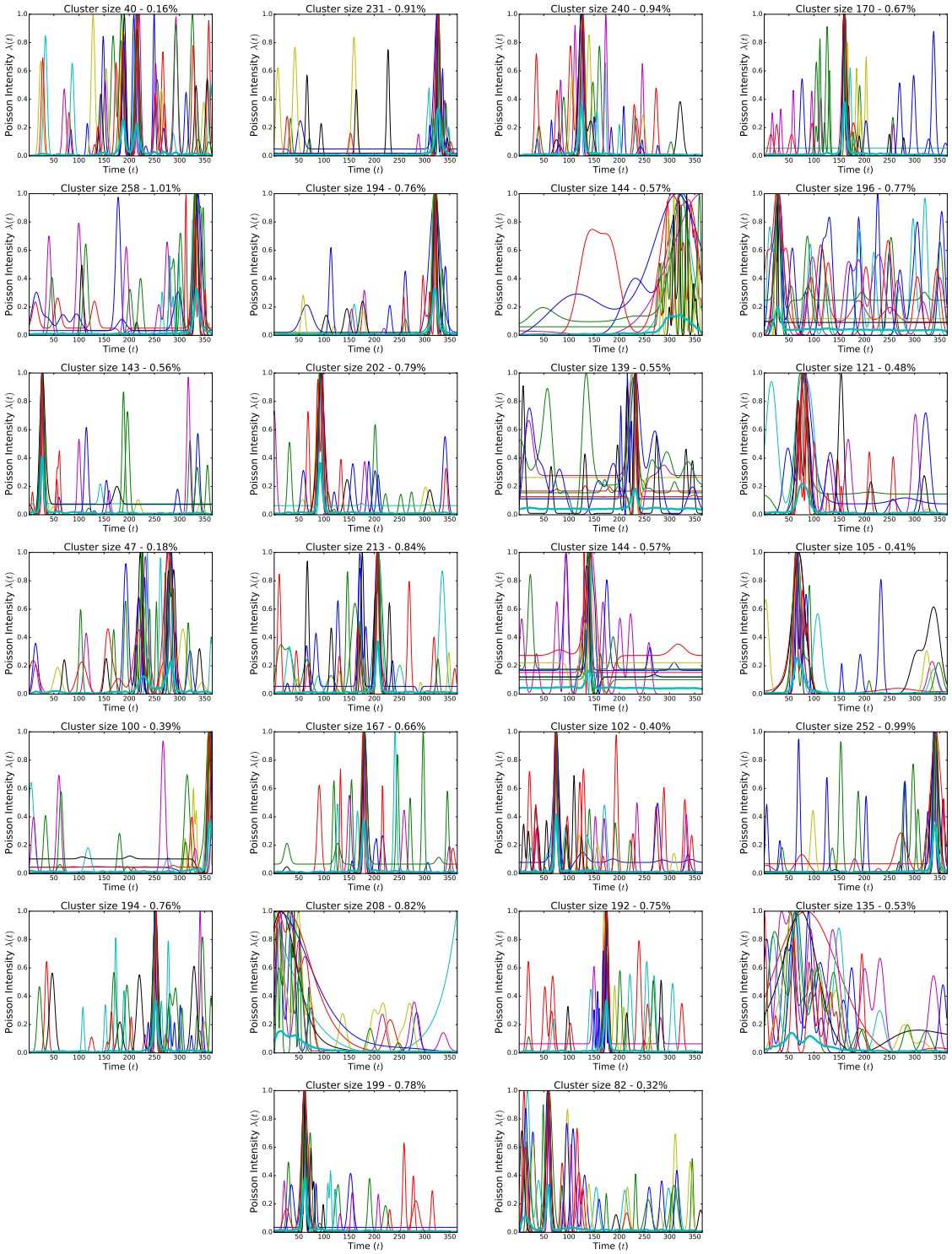


Figure C.1.: Centroids resulted from clustering ($K = 150$) the Stackoverflow users who posted an answer to a question related to the top ten tags in 2014. With every centroid, 10 users randomly chosen from the corresponding cluster are also presented.

Bibliography

[Ada+08] L. A. Adamic, J. Zhang, E. Bakshy, and M. S. Ackerman. "Knowledge sharing and yahoo answers: everyone knows something". In: *Proceedings of the 17th international conference on World Wide Web*. ACM. 2008, pp. 665–674 (cit. on p. 11).

[OTJ10] H. Oktay, B. J. Taylor, and D. D. Jensen. "Causal discovery in social media using quasi-experimental designs". In: *Proceedings of the First Workshop on Social Media Analytics*. ACM. 2010, pp. 1–9 (cit. on p. 11).

[PAT14] J. S. Pudipeddi, L. Akoglu, and H. Tong. "User churn in focused question answering sites: characterizations and prediction". In: *Proceedings of the companion publication of the 23rd international conference on World wide web companion*. International World Wide Web Conferences Steering Committee. 2014, pp. 469–474 (cit. on p. 11).

[And+12] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. "Discovering value from community activity on focused question answering sites: a case study of stack overflow". In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2012, pp. 850–858 (cit. on p. 11).

[Ras06] C. E. Rasmussen. "Gaussian processes for machine learning". In: (2006) (cit. on pp. 11, 13, 18).

[SG12] E. Snelson and Z. Ghahramani. "Variable noise and dimensionality reduction for sparse Gaussian processes". In: *arXiv preprint arXiv:1206.6873* (2012) (cit. on pp. 11, 23, 25, 26, 37).

[SG05] E. Snelson and Z. Ghahramani. "Sparse Gaussian processes using pseudo-inputs". In: *Advances in neural information processing systems*. 2005, pp. 1257–1264 (cit. on pp. 11, 23, 25).

[Mal+08] R. D. Malmgren, D. B. Stouffer, A. E. Motter, and L. A. Amaral. "A Poissonian explanation for heavy tails in e-mail communication". In: *Proceedings of the National Academy of Sciences* 105.47 (2008), pp. 18153–18158 (cit. on pp. 12, 29).

[SR14] Y.-L. K. Samo and S. Roberts. "Scalable nonparametric Bayesian inference on point processes with Gaussian processes". In: *arXiv preprint arXiv:1410.6834* (2014) (cit. on pp. 12, 29, 43).

[YL11] J. Yang and J. Leskovec. "Patterns of temporal variation in online media". In: *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM. 2011, pp. 177–186 (cit. on pp. 12, 30, 32, 33, 36).

[Bis06] C. M. Bishop. "Pattern Recognition". In: *Machine Learning* (2006) (cit. on pp. 13, 39).

[FR64] R. Fletcher and C. M. Reeves. "Function minimization by conjugate gradients". In: *The computer journal* 7.2 (1964), pp. 149–154 (cit. on p. 17).

[AS64] M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Vol. 55. Courier Corporation, 1964 (cit. on p. 20).

[UO30] G. E. Uhlenbeck and L. S. Ornstein. "On the theory of the Brownian motion". In: *Physical review* 36.5 (1930), p. 823 (cit. on p. 20).

[QW07] J. Quinonero-Candela and C. K. Williams. "Approximation methods for gaussian process regression". In: (2007) (cit. on p. 22).

[SB01] A. J. Smola and P. Bartlett. "Sparse greedy Gaussian process regression". In: *Advances in Neural Information Processing Systems 13*. Citeseer. 2001 (cit. on p. 22).

[WS01] C. Williams and M. Seeger. "Using the Nyström method to speed up kernel machines". In: *Proceedings of the 14th Annual Conference on Neural Information Processing Systems*. EPFL-CONF-161322. 2001, pp. 682–688 (cit. on p. 22).

[Csa02] L. Csató. "Gaussian processes: iterative sparse approximations". PhD thesis. Aston University, 2002 (cit. on p. 22).

[CH] J. Q. Candela and L. K. Hansen. "Learning with uncertainty-Gaussian processes and relevance vector machines". PhD thesis. unknown (cit. on p. 22).

[CO02] L. Csató and M. Opper. "Sparse On-Line Gaussian Processes". In: *Neural Computation* 14.3 (Mar. 2002), pp. 641–668. ISSN: 0899-7667. DOI: 10.1162/089976602317250933 (cit. on p. 22).

[SWL03] M. Seeger, C. Williams, and N. Lawrence. "Fast forward selection to speed up sparse Gaussian process regression". In: *Artificial Intelligence and Statistics 9*. EPFL-CONF-161318. 2003 (cit. on p. 22).

[See03] M. Seeger. "Pac-bayesian Generalisation Error Bounds for Gaussian Process Classification". In: *J. Mach. Learn. Res.* 3 (Mar. 2003), pp. 233–269. ISSN: 1532-4435. DOI: 10.1162/153244303765208386. URL: http://dx.doi.org/10.1162/153244303765208386 (cit. on p. 22).

[Gal13] R. G. Gallager. *Stochastic processes: theory for applications*. Cambridge University Press, 2013 (cit. on p. 29).

[Mül07] M. Müller. "Dynamic time warping". In: *Information retrieval for music and motion* (2007), pp. 69–84 (cit. on p. 32).

[J A79] M. A. W. J. A. Hartigan. "Algorithm AS 136: A K-Means Clustering Algorithm". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1 (1979), pp. 100–108. ISSN: 00359254, 14679876. URL: http://www.jstor.org/stable/2346830 (cit. on p. 34).

[GV12] G. H. Golub and C. F. Van Loan. *Matrix computations*. Vol. 3. JHU Press, 2012 (cit. on p. 36).

[MIK05] D. G. Manolakis, V. K. Ingle, and S. M. Kogon. *Statistical and adaptive signal processing: spectral estimation, signal modeling, adaptive filtering, and array processing*. Vol. 46. Artech House Norwood, 2005 (cit. on p. 37).

[Ham94] J. D. Hamilton. *Time series analysis*. Vol. 2. Princeton university press Princeton, 1994 (cit. on p. 37).

[OSB16] C. Ojeda, R. Sifa, and C. Bauckhage. "Investigating and Forcasting User Activities in Newsblogs: A Study of Seasonality, Volatility and Attention Burst". In: *Work On Progress* (2016) (cit. on p. 37).

[KR09] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Vol. 344. John Wiley & Sons, 2009 (cit. on p. 45).