Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Datateknik
202019 | LIU-IDA/LITH-EX-A--2019/001--SE

Trajectory-based Arrival Time Prediction using Gaussian Processes

–

Trajektoriebaserad ankomsttidsprediktion med Gaussiska Processer

Sebastian Callh

Supervisor : Mattias Tiger
Examiner : Fredrik Heintz

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

`Abstract.tex`

Acknowledgments

`Acknowledgments.tex`

Contents

List of Figures

List of Tables

1 Introduction

As cities grow, efficient public transport systems are becoming increasingly important. To offer a more efficient service, public transport providers use systems that predict arrival times of buses, trains and similar vehicles, and present this information to the general public. The accuracy and reliability of these predictions are paramount, since many people depend on them, and erroneous predictions reflect badly on the public transport provider.

Machine learning algorithms have been applied with great promise to predict arrival times [9, 15, 14], and research is still ongoing. Modern approaches have seen heavy use of Recurrent Neural Networks (RNNs) to directly model arrival times from the current state of public transport vehicles, an approach which has shown very good prediction accuracy but has major drawbacks:

1. It does not quantify prediction uncertainties.
2. It is severely lacking in explainability.

To make good decisions based of model predictions, it is important to know the prediction uncertainty. Being able to explain how a model makes its predictions is also extremely desirable, since it allows reasoning about why a model made a certain prediction, and about how a model would act in previously unseen scenarios. To address the first problem, the RNN model can be replaced with a statistical model. However, solving the second problem and achieving a satisfying level of explainability requires a much richer model of the data altogether. One promising approach to create such a model is *motion pattern learning* (or *trajectory learning*), where the goal is to learn a *motion pattern* (also known as *activity pattern* or *activity model*) which represents a cluster of individually observed trajectories. A trajectory would in this setting be observed positions and velocities of a public transport vehicle in service. Motion pattern learning is a problem which spans several fields, such as computer vision [13, 32, 10, 3], pattern recognition [22], autonomous vehicles [6], and health informatics [16], and is like arrival time prediction an active area of research.

By combining motion pattern learning and statistical modeling, it is possible to make sophisticated predictions, and gain a lot of insight on how the system performs. For instance, it would be possible to present the probability of arriving on time, in real-time, to passengers planning trips with several transits. The public transport provider could see routes where predictions are particularly inaccurate, and analyse the motion patterns on those routes, to help explain their poor performance. The motion patterns could be automatically analysed for detecting events of interest, such as “drastic velocity change”, “stand-still”, “congestion”, “emergency break”, to help explain the bad predictions. They could also be processed for anomaly detection, to automatically detect if stations are missed or if public transport vehicles stray from the designated route, or stand still for a long time. Based on the information extracted, the public transport provider could make efforts to improve the routes by, moving routes, moving stops or changing time schedules, to mention a few things. Motion patterns created after the changes have been made could then be compared to previous ones, to see what effects the changes had. By iterating this process, continuous improvements could be made.

1.1 Aim

The aim of this thesis project is to model motion patterns of public transport vehicles using Gaussian Processes (GPs), and use these models to make arrival time predictions with an accuracy that is competitive to current state-of-the-art models [19, 8, 15]. In addition, the project aims to construct *transition models* which model distributions when transitioning from one motion pattern to a consecutive one. Furthermore, the project aims to detect specific characteristics from motion patterns, such as “the driver had to emergency break”, or “the vehicles speed was very slow”, to see if certain characteristics are more common in motion patterns where the vehicle is too early or too late, compared to publicly available time schedules. Finally, this project aims to investigate how outlier motion patterns can be detected.

1.2 Research questions

1. How can GPs be used to capture trajectory motion patterns of public transport vehicles?
2. How can GPs be used to predict arrival times of motion pattern models with quantifiable uncertainty, while minimising MAE?
3. How can consecutive GP motion pattern models be predicted from the current, minimising misclassification rate?
4. How can user-specified events be automatically detected in a motion pattern using composite kernel search?
5. How can outlier motion patterns be detected?

1.3 Delimitations

The domain of the thesis project is specifically buses in Linköping. Data on trains is available, but making the system work for trains is out of scope. The thesis project is limited further to motion pattern models and arrival time predictions on consecutive but stops. Predictions for bus stops several steps ahead will consequently make use of both motion pattern models and the transition model.

1.4 Report Outline

The report begins with a background chapter, which explains the structure of trajectory data, and gives a brief summary of the relevant machine learning techniques and other theory. After the background chapter the report acknowledges related work done on trajectory data and arrival time predictions in a theory chapter. Following that is a brief data chapter which goes into more detail on the structure of tha data used in the project. The main contributions of the thesis project is adressed in the following methods chapter. In it, a formal derivation of the model is presented, and the implementation is described. After the methods chapter, the research questions are adressed in the results chapter. Finally, the report discusses the implications of its contributions before concluding.

2 Background

This chapter describes theoretic results that this thesis is based on. It covers trajectory data, motion pattern learning, supervised machine learning from a Bayesian perspective, GPs, and arrival time prediction using machine learning.

2.1 Trajectory Data

A *trajectory* T_k of can be seen as an ordered collection of observations $(x_1^{(k)}, x_2^{(k)}, \dots, x_N^{(k)})$. The thesis project focuses on *spatio-temporal* trajectories, which mean that each observation in a trajectory has a position both in time and in space. A trajectory has a *length* and a *duration*. The length $\text{len}(T_k) = N$ is the number of observations, and the duration $\text{dur}(T_k) = \text{time}(x_N^{(k)}) - \text{time}(x_1^{(k)})$ is the time from the first observation to the last. These properties are in general not the same for different trajectories, which makes it hard to compare them in a meaningful way. Having different lengths makes a comparison particularly difficult. If a set of trajectories all have the same length, they could be viewed as vectors, with one observation per dimension, and compared using Euclidian distance. But trajectories of different lengths do not exist in the same vector space, so Euclidian distance fails and more advanced similarity metrics have to be used. An illustration of trajectories with different length can be seen in Figure 2.1.

Even trajectories generated from the same underlying process do not in general have the same length and duration, even though they are typically more similar. For instance, a bus

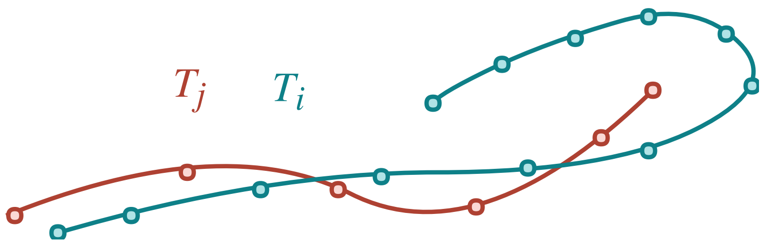


Figure 2.1: An illustration of two trajectories T_i and T_j with different lengths and duration. There is no natural way of measuring the distance between them.

driving the same route several times would not produce trajectories with identical duration and length. If two trajectories from the same underlying process have the same amount of observations at the same time points, they are said to be *synchronised*. Otherwise they are said to be *unsynchronised*.

2.2 Supervised Machine Learning

Supervised machine learning is the process of training computers programs to recognise patterns in data [2]. It is incredibly flexible, and can find patterns such as "What temperature is expected for this time of year on this geographical position?" or "What subject is this text about?". Finding the first type of pattern of continuous temperature is called *regression*, and finding the second pattern with a discrete number of subjects is called *classification*. Both types of problems have two inputs: the *observations*

$$X = \begin{pmatrix} x_1^1 & x_2^1 & \cdots & x_D^1 \\ x_1^2 & x_2^2 & \cdots & x_D^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^N & x_2^N & \cdots & x_D^N \end{pmatrix},$$

containing N observations of dimension D, and the *target vector* $Y = (y_1, y_2, \ldots, y_N)$. In the first example of predicting temperatures, X could contain rows of latitude, longitude and time of year, and Y the corresponding recorded temperatures. The goal is to learn how the input and output is related by finding a model which approximates a D-ary function $y_n = f(x_n)$ for $1 \leqslant n \leqslant N$. Such a model could then predict y for a previously unseen input $\hat{x}$. The way a model approximates f depends entirely on the model, of which there are many. Common for all models however, is that they are all *trained* on data. It is during this process they learn the patterns in the data which they then use to make predictions. While there are many approaches to this, the remainder of this section describes the process of selecting and training models, and using them to make predictions from a Bayesian point of view.

Model Training

The first step is to pick a model for the data, which in a Bayesian framework is a probability distribution $p(x|\theta)$, where θ is a hyper-parameter vector to the distribution. The probability of all observations is then $p(X|\theta) = p(x_1, x_2, \ldots, x_N|\theta)$. For mathematical convenience it is often assumed that the observed data come from the same distribution and that each observation is independent of all other. This means that the probability of a pair of observations x_i, x_j is $p(x_i, x_j|\theta) = p(x_i|\theta)p(x_j|\theta)$ and consequently, that the probability of observing all the data is $p(X|\theta) = \prod_{i=1}^{N} p(x_i|\theta)$. This is known as the *likelihood*, which describes how likely the data is given a certain θ. One way of training a model is by finding $\hat{\theta} = \arg\max_\theta p(X|\theta)$, which is called *maximum likelihood*-estimation (ML-estimation), since it picks the θ that maximises the likelihood of the data. However, this picks a single "best value", highly sensitive to the choice of training data, which can lead to a problem called *over-fitting*.

The goal when training a model is to learn patterns that *generalise* to unseen data. If θ is estimated from data using ML-estimation, it is possible to find a $\hat{\theta}$ that makes for an increadibly flexible model which *perfectly* captures the patterns in the data. However, it is very common for data to contain noise and outliers, which do not represent a pattern that generalise well. A model that learns these is said to have *over-fit*, which is highly undesirable. An illustration of this can be seen in Figure 2.2, together with Figure 2.3.

One way of avoiding over-fitting is to use *Bayesian inference*, in which Bayes theorem

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)} \tag{2.1}$$

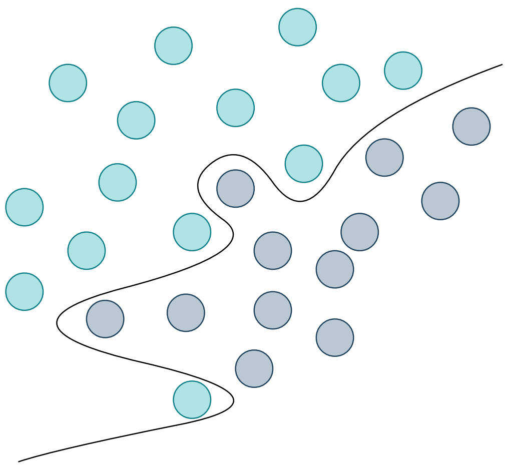


Figure 2.2: Example of an over-fit in a classification problem. The function separating the two classes is too flexible, and captures observations which should be consider noise. This model will not generalise well.

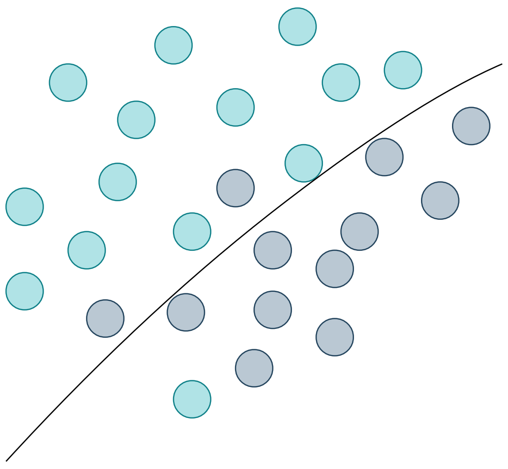


Figure 2.3: Example of a good fit in a classification problem. The function separating the two classes captures the generale structure of the data. Even though this misclassifies some observations, this model stands a much better chance at generalising.

is used to estimate θ from the *posterior* distribution $p(\theta|X)$. To use Bayes theorem a *prior* distribution $p(\theta)$ is required, which formalises a prior belief about θ before observing any data. The prior is subjective, and different people may pick different priors, representing their personal belief about the data. By picking a prior corresponding to a not-too-flexible model, over-fitting can be avoided. In the case of GPs, this corresponds to a distribution which implies places most probability mass on parameters that give a high correlation between observations further apart, which in turn implies a slowly-varying function. GPs are explained in more detail in Section 2.6.

The posterior quantifies the uncertainty in θ, but can be computationally expensive to work with. Because of this it is sometimes desirable to estimate θ as $\theta_{MAP} = \arg\max_\theta p(\theta|X)$; a process called *maximum a posteriori*-estimation (MAP-estimation). In summary, the process of training a model is equivalent to computing the posterior, from which the most probable model parameters can be extracted.

Training a model using ML- or MAP-estimation both require that the likelihood can be optimised. This is not always possible to do in closed form, in which case iterative methods, such as Stochastic Gradient Descent (SGD) can be used. These types of methods are prone to to finding local optimas, so random restarts need to be used to increase the chances of finding a good parametrisation.

If a prior is specified, it is possible to estimate the parameters of model $\mathcal{M}$ *without* fitting it to data, avoiding the problem of over-fitting entirely. This is done by maximising the *marginal likelihood*

$$p(x|\mathcal{M}) = \int p(x|\theta, \mathcal{M})p(\theta|\mathcal{M})d\theta \tag{2.2}$$

which considers both the uncertainty in x and θ. However, the marginal likelihood is very sensitive to the choice of prior, so if the prior is not selected carefully this way of estimating θ will produce a bad model. The integral can also be difficult to compute.

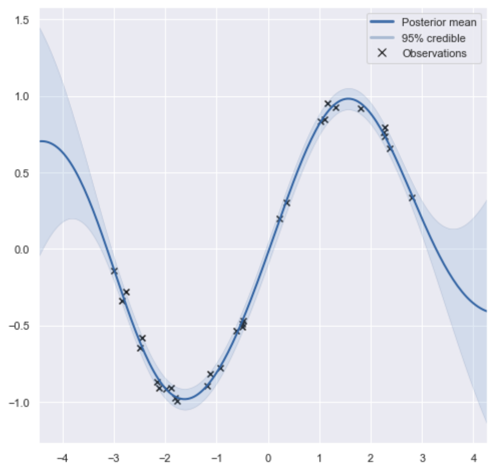


Figure 2.4: Illustration of the 95% credible intervals for the posterior predictive distribution for a regression model. The predictive distribution is narrow, giving tighter confidence bands.

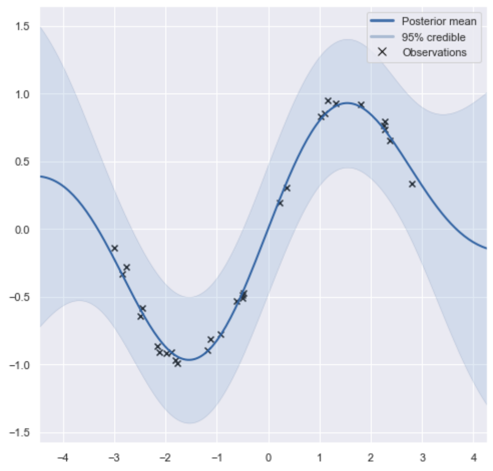


Figure 2.5: Illustration of the 95% credible intervals for the posterior predictive distribution for a regression model. In this case the predictive distribution is very wide, giving wider confidence bands.

Making Predictions

When the model is trained, it can be used to make predictions about new observations, via the *posterior predictive distribution*

$$p(y|X) = \int p(y|X,\theta)p(\theta|X)d\theta, \tag{2.3}$$

which accounts for the parameter uncertainty through the marginalisation of θ. From the predictive distribution, very sophisticated predictions can be made. The single "best guess" of y is represented by $\mathrm{E}[y|X]$, but thanks to having an entire distribution it is possible to compute exactly how certain it is. This can be done by computing the *credible intervals* [12] for the distribution, which is the span in which a certain amount of probability mass lies. For instance, the 95%-credible interval would contain 95% of the probability mass, which can be interpreted as a 95% chance of future observations falling inside this span. An illustration of credible intervals can be seen in Figure 2.4 and Figure 2.5.

Bayesian Model Selection

It is quite possible to have several models for the same data. In such a situation it is interesting to select the model that *best* explains the data. This is known as a *model selection problem*, and from a Bayesian point of view it can be solved by selecting the model with the highest posterior probability

$$\mathcal{M} = \arg\max_k p(\mathcal{M}_k|x) \propto \arg\max_k p(x|\mathcal{M}_k)p(\mathcal{M}_k). \tag{2.4}$$

Computing this quantity requires computing the model marginal likelihood given by equation 2.2, which is often refered to as the *model evidence* in the context of model selection.

2.3 Arrival Time Prediction

Arrival time prediction is in the context of machine learning on trajectory data the problem of predicting when a trajectory ends. That is, the goal is to learn a function $t = f(\bar{x})$ for arrival time t and observation $\bar{x}$, which contains information on the current trajectory state. For instance, it could contain the position, velocity and the time of day.

Prediction Evaluation

Two complementary ways of evaluating how good a prediction is are the metrics *Mean Absolute Error* (MAE) and *Mean Absolute Percentage Error* (MAPE), defined for the true value $\hat{x}$ and predicted value x as

$$MAE(\hat{x}, x) = \frac{1}{n}\sum_{i=1}^{n} |\hat{x} - x|, \tag{2.5}$$

and

$$MAPE(\hat{x}, x) = \frac{1}{n}\sum_{i=1}^{n} |\frac{\hat{x} - x}{\hat{x}}| \tag{2.6}$$

respectively. MAE provides a natural interpretation as "the average of the total error" which is a relevant and easy-to-understand quantity [31], while MAPE calculates a relative error [1], which is unaffected by the magnitude.

2.4 Data Clustering

There are cases where no labels y exist, and the task is to learn the structure of the data. These types of problem fall under *unsupervised learning* (as opposed to supervised learning when y is known), and one common problem of this type is *clustering* [2]. The goal of clustering is to identify groups of observations that are in some sense similar. These groups are known as clusters, and can be created in many ways. Intuitively, observations that are "close" with respect to some distance metric can be considered similar, and should be clustered together. However, without a distance metric there is not way to measure closeness and consequently no way of clustering. The concept of clustering is illustrated in Figure 2.6 and Figure 2.7.

2.5 Motion Pattern Learning

Motion pattern learning is the problem of learning motion patterns from a set of trajectories, such that each pattern captures a different characteristic of the trajectories. An example with synthetic data can be seen in Figure 2.8. The term *trajectory learning* is often used in the literature for the same problem. However, in the context of this thesis the term "trajectory" refers to data with certain structure, as described in Section 2.1, so the name motion pattern learning will be used instead.

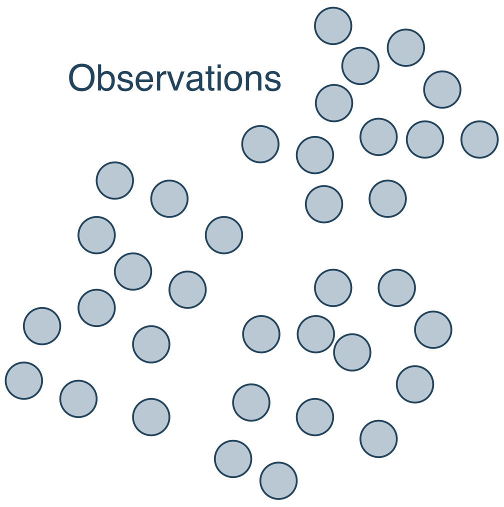


Figure 2.6: Observations without labels. The goal is to place similar data points into the same clusters.

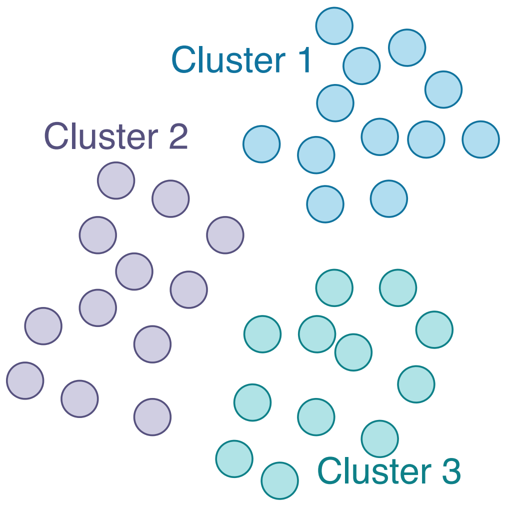


Figure 2.7: The observations have been clustered based on distance into three distinct clusters.

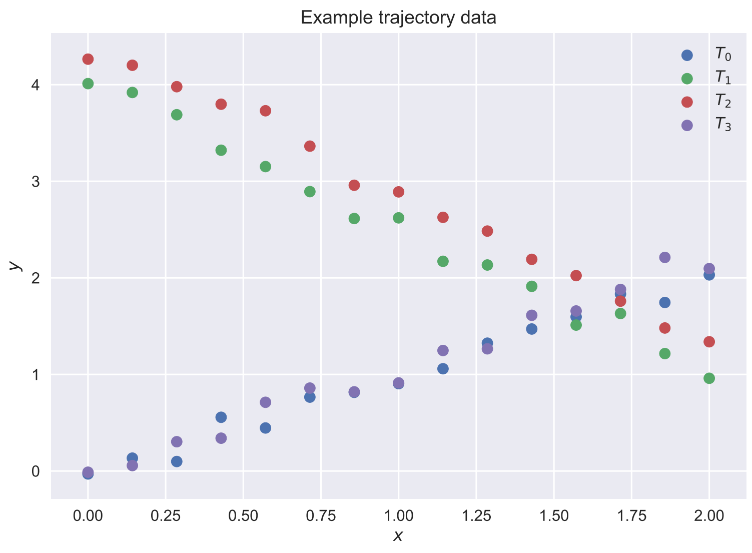


Figure 2.8: Synthetic data showing two motion patterns with two trajectories in each. T_1 and T_2 belong to one motion pattern and T_0 and T_3 belong to a second motion pattern.

Motion pattern learning has a natural interpretation as a clustering problem, but clustering trajectories is difficult, since it is hard to define a similarity metric for trajectories for reasons described in Section 2.1.

2.6 Gaussian Processes

A GP generalises a multivariate normal distribution, and can be seen as a distribution over functions, completely defined by its mean function $m(x)$ and covariance function $k(x, x')$ [18], where x and x' define elements in the domain of modeled functions. In addition, the covariance function k typically depend on a vector of hyperparameters θ, which are omitted for notational clarity. A GP is a suitable prior when modeling a continuous function $y = f(x)$. Assuming that observations are $y_{obs} \sim \mathcal{N}(f, \sigma_n)$, the posterior is also a GP for

$$y = f(x) \sim \mathcal{N}(\mu(x), \Sigma(x)) \tag{2.7}$$

where

$$\mu(x) = m(x) + K(x, \mathbf{x})\mathbf{V}^{-1}(y - m(x))^T, \tag{2.8}$$

$$\Sigma(x) = K(x, x) + \sigma_n^2 \mathbf{I} - \mathbf{K}(x, \mathbf{x})\mathbf{V}^{-1}\mathbf{K}(x, \mathbf{x})^T, \tag{2.9}$$

and $\mathbf{K}$ is the gram matrix with elements $K_{ij} = k(x_i, x_j)$ and $\mathbf{V} = K(x, x) + \sigma_n^2 I$. When using a *stationary* kernel (a kernel that only depends on the relative difference of inputs), the mean function $m(x)$ can be assumed to be $m(x) = 0$ without loss of generality, making the covariance function $k(x, x')$ the only free parameter. Picking a specific function represents a prior belief on how values close in y are related, expressed in x. This concept is explored in more detail in Section 2.6. Training a GP is done by learning the covariance function parameters θ. This can be done using done ML/MAP-estimation, which unfortunately is a non-convex optimisation problem, so iterative methods have to be used. To avoid local minimas, random restarts are typically used. How good a GP explains data can be computed through its data likelihood, which is conveniently computed in log scale as

$$\begin{aligned} \log P(y|x) &= -\frac{1}{2}(y - \mu(x))^T \Sigma(x)^{-1}(y - \mu(x)) \\ &= -\frac{1}{2}\log|\Sigma(x)| + C, \end{aligned} \tag{2.10}$$

where C is constant term, and $\mu(x)$, $\Sigma(x)$ are given by equations 2.8 and 2.9 respectively.

Kernels as Covariance Functions

Covariance function formalises a prior belief on the shape of the target function, by specifying how correlated function values y are by evaluation the kernel function pair-wise on the observarions. While in practice any binary function can be plugged into a GP, a class of functions known as *kernels* are typically used, since they have useful properties for expressing covariance. In particular, kernels are positive-definite functions, which in turn gives a positive-definite covariance matrix. This is required for it to be invertable, which in turn is required to compute the GP posterior. The requirement of positive-definiteness makes intuitive sense, since covariance can not be negative. Kernels are increadibly flexible, and can be defined for other entities than continuous funtions, such as graphs, strings and images [4]. However, for this thesis project only kernels on continuous functions are considered. Kernels on continuous functions are able to express a wide range of prior beliefs, from linearity to symmetry and periodicity. Figure 2.9 illustrates several kernels found in the literature on continuous functions together with samples from their priors.

This thesis project uses RBF, Matérn32 and Linear kernels, and so they deserve som special attention. Both the RBF and the Matérn32 are special cases of the more general Matérn kernel, defined by

$$\mathrm{Mat}_\nu(x, x') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{d}{\ell}\right)^\nu K_\nu \left(\sqrt{2\nu}\frac{d}{\ell}\right). \tag{2.11}$$

The Matérn32 is simply given by equation 2.11 parameterised by $\nu = 3/2$ and RBF is recovered through

$$\lim_{\nu \to \infty} \mathrm{Mat}_\nu(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right). \tag{2.12}$$

The parameter ν determines how smooth the function is, where a larger ν gives a smoother function. The Matérn32 consequently models functions that varies more quickly than those

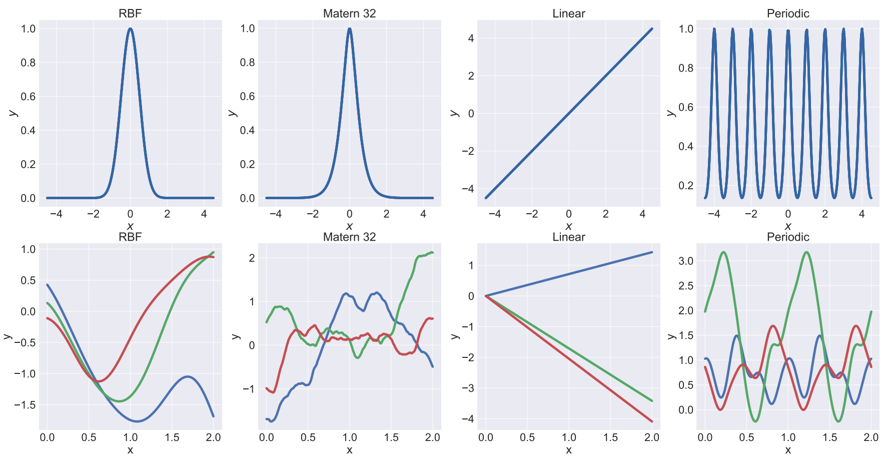


Figure 2.9: Illustration of different kernels and samples from their priors. From left to right the kernel functions are RBF (Radial Basis Fuction), Matern 32, linear kernel, and periodic kernel.

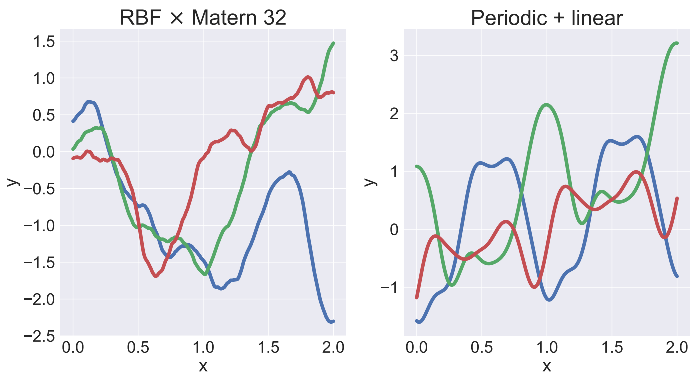


Figure 2.10: Illustration of compound kernels. In the top row RBF and Matern 32 are combined by multiplication, and in the bottom row a linear and periodic kernel are combined by addition.

modeled by the RBF, as illustrated in Figure 2.9. The final kernel used is the linear one, which is given by

$$k_{Lin}(x, x') = \sigma_b^2 + \sigma_v^2 (x - c)(x' - c), \tag{2.13}$$

where σ_b^2 controls the bias, or where the function intersects the y-axis and c defines a point in x which all functions intersect.

Even though a single kernel captures a lot of priors, it is sometimes desireable to express more complicated one. Fortunately, the set of kernel functions are closed under multiplication and addition, which creates a principled way of combining them, creating *compound kernels* [4]. Kernels can be combined as required to capture prior beliefs about periodicity *and* linearity, for instance. Figure 2.10 illustrates the concept of compound kernels.

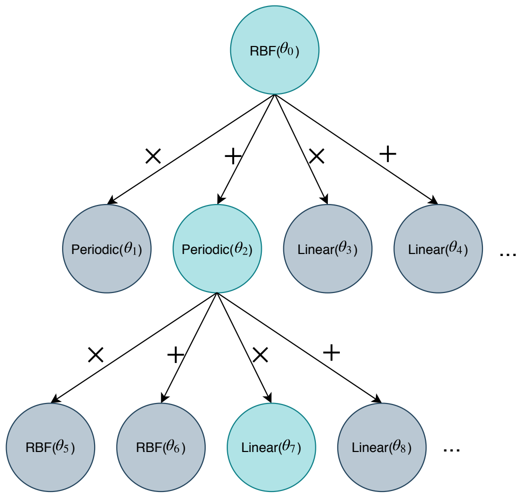


Figure 2.11: Illustration of greedy search over kernel structures under multiplication and addition, where the expanded nodes are highlighted. Each kernels hyperparameters θ_i are learned from data, and the kernel that minimises BIC the most is selected.

Structure Discovery Using Gaussian Processes

Structure discovery is the problem of automatically learning the structure of data. By creating kernels for specific characteristics and combining them in a way that maximises the likelihood of the data, it is possible to detect said characteristics using GPs, based on the combination of kernels [4].

There are infintely many ways to construct compound kernels, and some kernel combinations explain a specific data set X better than other. How well a kernel k explains X can be approximated by the Bayesian information criterion (BIC)

$$BIC(k) = \log p(X|k) - \frac{1}{2}|k| \log N, \tag{2.14}$$

where $|k|$ denotes the number of kernel parameters and N the number of observations. BIC assumes independence between observations conditioned on parameters, but this is in general not the case in a GP model. However, it has been shown to be a good enough approximation [4].

With a way of comparing kernels established, it is interesting to find *the best* compound kernel for X. That is, the kernel composition that maximises BIC. One way to do so is to greedily search over a set of pre-defined kernel compositions, fitting them to data as the search goes, and composing the ones that minimise the BIS into the compound kernel. This process is illustrated in Figure 2.11. The structure of the resulting kernels reveals structure in the data. For instance, if the resulting kernel contains a RBF component and a periodic component, the data is locally correlated, and follow a periodic trend. This process can be used to automatically detect structure in data from a cleaverly crafted set of kernels that correspond to known characteristics.

2.7 Sparse Gaussian Processes

Computing the posterior of a GP requires inverting the matrix $\mathbf{V}$ (see equations 2.9, 2.8), which has time complexity $\mathcal{O}(n^3)$ for matrix dimensions $n \times n$. This quickly makes naive implementations of GPs intractable for larger data sets. One way to get around this is to use a *Sparse Gaussian Process* (SGP) [26], which approximates the true posterior in favour of performance. This is done by finding a set of $m < n$ synthetic observations, called *inducing* *inputs*, such that the Kullback-Leibler divergence

$$D_{KL}(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{Q(x)}{P(x)} \right) \tag{2.15}$$

is minimised between the true posterior P and the posterior induced by the inducing inputs Q. This improves the time complexity of computing the posterior to $\mathcal{O}(nm^2)$.

2.8 The Equirectangular Map Projection

Working with data in latitude-longitude space is tricky, since the two dimensinos are not scaled 1:1. To make it easier to work with such data, it can be projected onto a Euclidian space, where it can be described using cartesian coordiantes. However, since latitude and longitude describe points on an ellipsis, and cartesian coordiantes are orthogonal, a perfect projection does not exist [21]. Because of this, several different projections have been invented, each making a different trade-off. One of these is the *Equirectangular Projection*, which maps latitude ϕ and longitude λ onto cartesian coordinates x and y through

$$\begin{aligned} x &= (\lambda - \lambda_0) \cos \phi_0 \\ y &= (\phi - \phi_0), \end{aligned} \tag{2.16}$$

where λ_0 and ϕ_0 define the central meridian and the standard parallels of the latitude-longitude space respectively. The projection is correct for $\phi = \phi_0$, but the further away points are from ϕ_0 the larger the residuals will be. The y- and x–axis of the resulting cartesian coordinate system will be λ_0 and ϕ_0 respectively.

3 Related Work

This section covers related work on arrival time prediction and motion pattern modeling using spatio-temporal data. It also covers related work relevant to clustering of trajectory data and event detection in motion patterns.

3.1 Arrival Time Prediction

In recent years, machine learning techniques have been very successful at predicting the arrival time with small errors, and Long Short-Term Memory Networks (LSTMs) in particular have proven extremely effective. J. Pang et al. predicted bus arrival times to the next station in a continuous setting using LSTMs given the current position of a bus and static domain knowledge about its last stop [15]. D. Nguyen et al. also used LSTMs, but with entire trajectories [14]. They predicted arrival times of ships by training a model to generate continuations of trajectories, and when the model was presented with a new unfinished trajectory, it generated a probable continuation until it arrived at a port. This was then used to prediction both the destination and the arrival time.

While these approaches do perform admirably with respect to accuracy, they lack explainability and a way to quantify prediction uncertainty.

Trajectory Similarity Metrics

There are several ways to construct a similarity metric for trajectories. Some widely-used metrics are Euclidan distance, Dynamic Time Warping (DTW), Longest Common Sub-Sequence (LCSS), and Edit Distance (ED) [29]. DTW and LCSS have seen use in motion pattern learning [22, 28], but only work with discrete time-series data. Furthermore, they also suffer from high time-complexity, making them unsuitable for real time processing [32] An alternative to previously mentioned similarity metrics is to create a probabilistic model for the data, and use data likelihood as a similarity metric [10, 27, 25]. One advantage of a probabilistic model is that it is not limited by purely spatial information, since any information about the trajectories can be modeled. For instance, spatial position could be combined with velocity and acceleration. Two popular probabilistic models for modeling motion pattern are GPs and hierarchical Bayesian models.

Gaussian Processes

GPs have been used in several domains for modeling trajectories. M. Pimentel et al. used GPs to model the vital signs of patients [16], and were successfully able to cluster the data using hierarchical clustering and the GP model likelihoods. They then modeled the motion pattern for a cluster as a GP with the average mean and variance for all GPs in it. K. Kim. et al. used GPs to model the motion patterns of cars from surveillance camera [10]. They introduce the concept of a *frame*, in which the trajectories are discretisised before fitting GPs to them. Having discrete trajectories meant that the local likelihood for observed data x_t, y_t in time step t could be computed as $p(y_t|x_t, M_k)$ for model M_k, which were aggregated to compute a global similarity metric L_k, which in turn was used to cluster the trajectories. To compute the motion patterns of clusters, a sampling scheme was used. In each time point, three GPs were drawn uniformly without replacement from the cluster. The mean value of all drawn GPs were then used as data points to fit a GP for the clusters motion pattern.

Q. Tran and J. Firl used data with both spatial position and velocity, and used GPs to model a vector field for each observed trajectory [27]. Before GPs were trained, the trajectories were normalised using spline interpolation and a sampling scheme. They did not perform any clustering. Instead, they constructed their motion patterns by driving their own test vehicle. When presented with a new trajectory, the model could compute the most likely motion pattern, and use a particle filter for the vector field to predict continuations of motion patterns.

The work of M. Tiger and F. Heintz aimed to improve upon the approach of K. Kim et al. who had implicitly assumed that all trajectories were generated from one underlying processes [25]. Doing so causes the model to underestimate its variance, which in turn causes the models likelihood to be too small for data that should be considered likely. To avoid this, sets of trajectories were modeled as a mixture-of-Gaussian-processes (MoGP). They then considered "slices" of trajectories, orthogonal to progression, which were approximated as Gaussian distributions. The posterior of these approximations was then assumed to come from a single GP, which was approximated on synthetic data, a process they call *inverse Gaussian process regression*. M. Tiger and F. Heintz also suggested an improvement on the vector field approach proposed by Q. Tran and J. Firl [24], in which they use GPs for the functions $(x, y) \mapsto \tau \mapsto x, y, x', y'$ for $\tau = [0, 1]$, compared to $(x, y) \mapsto x', y'$ used by Q. Tran and J. Firl. Their approach was shown to perform better in benchmarks.

C. Leysen et al. also aim to improve on the work of K. Kim et al [11]. Instead of fitting a GP to re-sampled trajectories, they simply select the trajectory in a cluster which maximises the overall likelihood of the data points in the cluster. Their approach still assumed a single underlying function for all trajectory data, and consequently still underestimated model variance.

Hierarchical Bayesian Models

The idea behind hierarchical Bayesian models used for trajectory learning is borrowed from the natural language processing field, where they are known as topic modeling. Topic modeling are unsupervised techniques for clustering documents into different topics, and by considering spatio-temporal trajectories as documents, their observations as words, and the motion patterns they belong to as topics, the same techniques can be used to model motion pattern. These models are usually based on Latent Dirichlet Allocation (LDA) or a generalisation thereof known as Hierarchical Dirichlet Process (HDP) introduced by Teh et al. [23], which are both so called *bag- of-words*-models. A bag-of-words-model assumes independence between words in a document, which in the domain of trajectories translates to the assumption that observed data points are independently drawn.

Both LDA and HDP require a set amount of clusters, which is a great weakness. Wang et al. proposed a model called *Dual Hierarchical Dirichlet Process* (Dual-HDP) [30], which improves upon the HDP model by allowing the model to learn the number of topics and documents from data. Zhou et al. further improved upon HDP and LDA by using Markov random fields to encode prior beliefs in a model called *Random Field Topic* (RFT) [33].

3.2 Event Detection

In the context of motion patterns and trajectory data, event detection can be seen as the problem of finding patterns in time series. For this, convolution can be used with specific convolution kernels corresponding to the event of interest [20].

Duvenaud et al. explored a different approach, where they modeled time series as GPs with a composite kernel [4]. To construct the kernel they greedily searched over different kernel structures, and composed kernel functions to maximise the marginal likelihood. The structure of the resulting kernel could then be analysed to reveal characteristics of the time series.

4 Data

The data is provided by Östgötatrafiken, and is collected from buses and trains in service in the public transport system. A single observation comprises a timestamp, GPS position, and a specific event type. Depending on the event type, additional fields are present. There are 22 different event types, but only four of them are relevant for the thesis project. The observation format is illustrated in Figure 4.1, and event specific fields are illustrated in Figure 4.1.

All data is sent from either a bus or a train. However since the thesis project is limited to buses, observations sent from trains are discarded. A summary of the different even types, including when they are sent can be seen in Table 4.1. A more thorough exploration of the data can be found in <referera Linus Kortesalmi, men det finns ej på DIVA>.

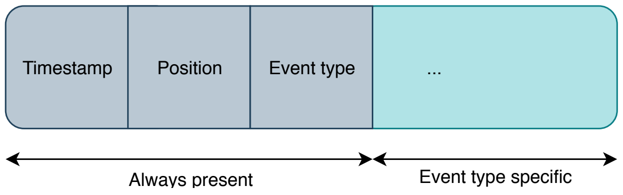


Figure 4.1: Format of an observation. Includes only the fields relevant to this thesis project.

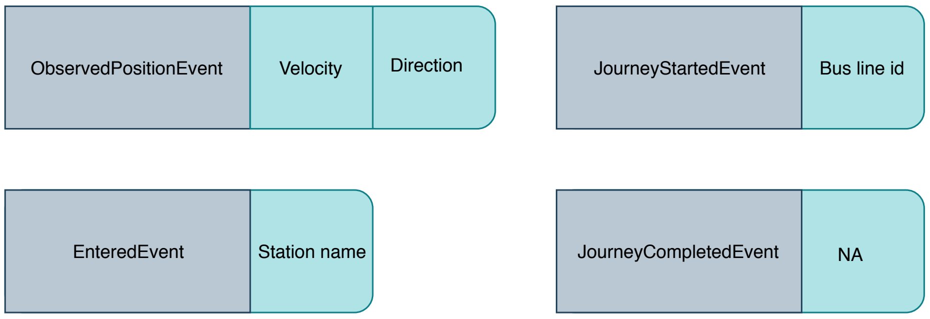


Figure 4.2: Format of fields specific to event types used in this thesis project.

Table 4.1: Summary of the event types used in the thesis project.

Event type	Additional data	Sent
ObservedPositionEvent	Velocity, Direction	Approximately every second.
EnteredEvent	Station name	When driving sufficiently close to a bus stop.
JourneyStartedEvent	Bus line id	When a bus is assigned a journey, before it leaves the station.
JourneyCompletedEvent		When a bus enters the final bus stop on a bus line.

While there are many bus lines in the data set, the thesis project focuses on the first eleven segments of bus line three. The reason for this is that the data was collected during a period of major road-building, and to avoid needing to sift through outliers a subset of the data which was known to be collected during ordinary circumstances was selected. The chosen trajectories are illustrated in Figure 4.3.

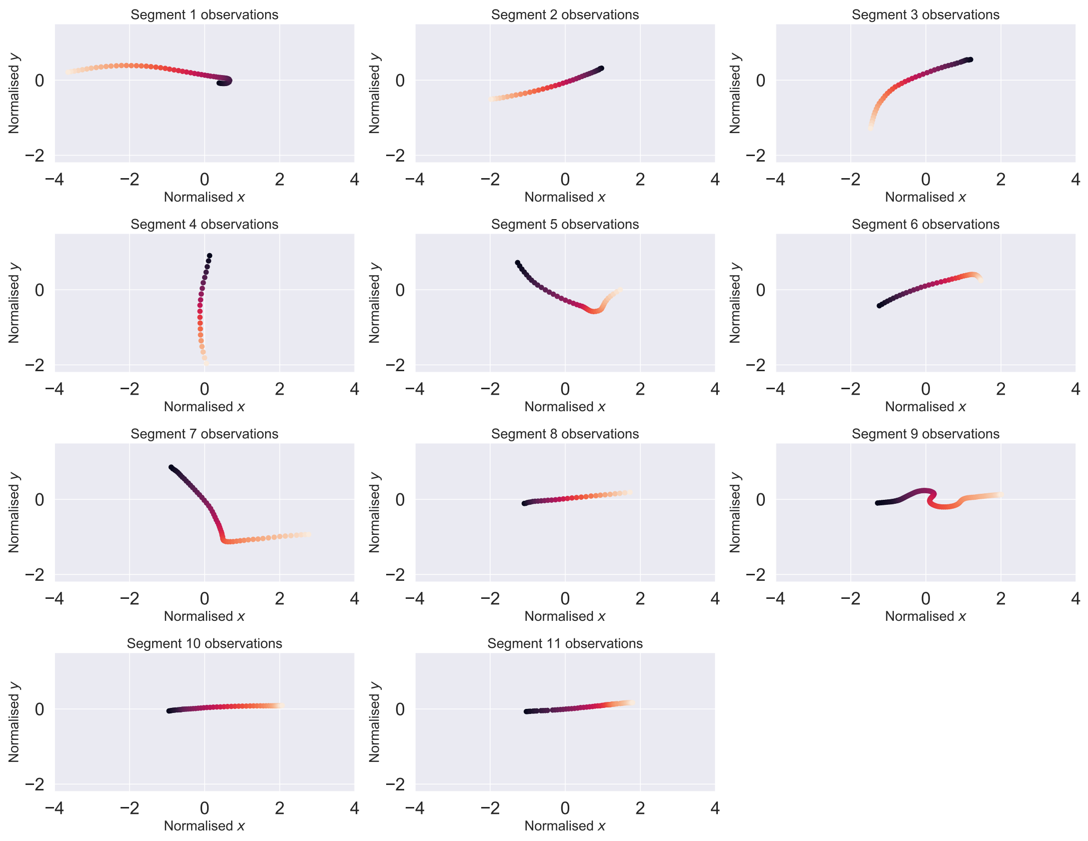


Figure 4.3: Spatial plot of all trajectories used in the thesis project. The colour indicate progression along the segment, from start in bright to end in dark. The observations have been scaled to have zero mean and unit variance.

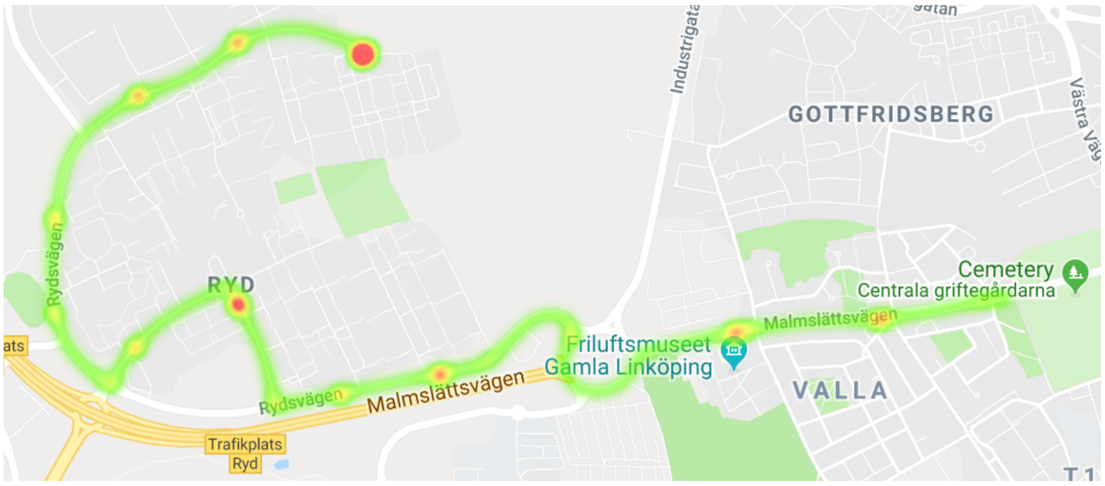


Figure 4.4: Heat map of observations collected in the beginning of bus line 3. Red indicates areas with high frequency of observations and green indicates areas with low frequency. The start of the route is in the top left, where the frequency of observations is very high.

Since the ObservedPositionEvent is sent every second it causes observations to spatially cluster when a bus is standing still. This happens frequently at bus stops, when a bus stops to pick up or drop off passengers, but also at the start of the journey. This is because a bus is assigned a journey several minutes before actually leaving. A heat map of observations can be seen in Figure 4.4 which shows this phenomenon.

5 Method

This chapter motivates the proposed system and derives a formal description of the proposed trajectory model from the problem of predicting arrival times by comparing trajectories. It then describes the implementation details of the model, and what approximations and simplifications that have been made.

5.1 Model derivation

Conceptually, the model assumes that similar trajectories, with respect to their motion patterns, should arrive at approximately the same time. When presented with a new trajectory, the system finds previously observed trajectories with similar motion patterns, and predict arrival times based on the most similar ones.

To find similar trajectories, a trajectory similarity metric is needed. Additionally, it need to be well defined for trajectories of different temporal lengths, and unevenly distributed observations. One way to define a local similarity metric from an observed trajectory T_i to trajectory T_j is as the sum of distances of orthogonal projections onto T_j, as illustrated in Figure 5.1. This approach would eliminate the need to align observations before comparison. However, it requires a continuous trajectory representation, even though trajectories are observed as discrete samples. Seeing the observations as samples from a continuous function $T_k = \tilde{f}_k(t)$ gives a description the trajectory as a function of time, which is indeed continuous. However,

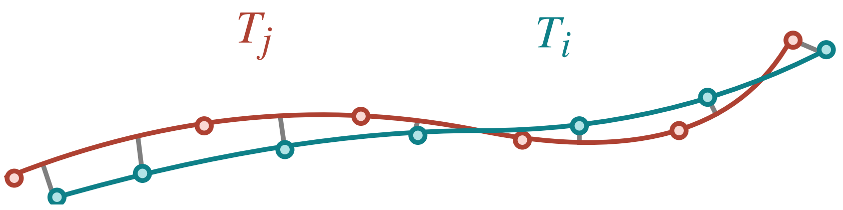


Figure 5.1: An illustration of local distances from T_i to T_j as point-wise orthogonal projections.

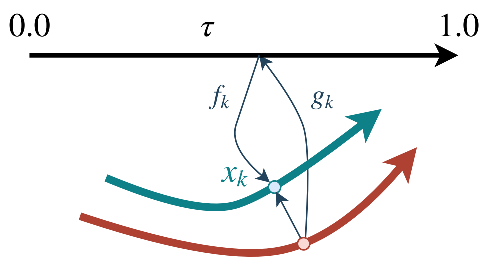


Figure 5.2: Conceptual illustration of finding the orthogogonal projection x_k onto trajectory T_k from observation x through the composition of $f_k \circ g_k$.

this description is based on specific time points, but the *exact points in time* of observations are not relevant, since trajectories for a single segment are generated at many different occasions. Instead, observations are modeled as samples from a function $T_k = \bar{f}_k(\tau)$, where $\tau = [0, 1]$ is the *progress* from start ($\tau = 0.0$) to finish ($\tau = 1.0$) of a trajectory. This function models observations based on progression for any trajectory, regardless of the exact observation time points.

The function $\bar{f}_k$ maps values in τ to a state vector of observed positions and velocities p_x, p_v, v_x, v_y. However, modeling a function which takes a scalar to a vector as a GP would require dependent outputs, which complicates computations considerably. Instead $\bar{f}_k$ is assumed to be a family of four functions $f_k = (f^{(k)}_{p_x}, f^{(k)}_{p_y}, f^{(k)}_{v_x}, f^{(k)}_{v_y})$ which maps the corresponding dimensions of the state vector. Independent GP priors are assumed for each function, which gives

$$\begin{aligned} p_x|\tau \sim &\mathcal{N}(\mu_{p_x}, \Sigma_{p_x}), \\ p_y|\tau \sim &\mathcal{N}(\mu_{p_y}, \Sigma_{p_y}), \\ v_x|\tau \sim &\mathcal{N}(\mu_{v_x}, \Sigma_{v_x}), \\ v_y|\tau \sim &\mathcal{N}(\mu_{v_y}, \Sigma_{v_y}). \end{aligned} \tag{5.1}$$

Finding a projection for an observation $\bar{x}$ onto trajectory T_k can now be seen as finding how far along the trajectory $\bar{x}$ would have traveled. More precisely, finding the τ that $\bar{x}$ corresponds to, which can then be taken through f_k to get the projection. To find the corresponding τ to $\bar{x}$ the inverse of a mapping from state to τ g_k is required. However, when considering progress along a segment, the current velocity does not matter, and so g_k only depends on (p_x, p_y). A GP prior is assumed for g_k which gives $\tau|\bar{x} \sim \mathcal{N}(\mu_\tau, \Sigma_\tau)$. The projection onto T_k can now be computed through the composition of the estimated functions $X_k = (f_k \circ g_k)(x)$, illustrated in Figure 5.2. While this approach serves as good intuition, it does not consider the uncertainty in the estimated functions or observations. To do this, a probabilistic approach is needed, where the similarity metric of a trajectory is seen as the probability of the corresponding synchronisation model. The probability of a model is given by Bayes theorem

$$p(\tilde{\mathcal{M}}_k|X_k = \bar{x}, X_{obs} = \bar{x}) \propto p(X_k = \bar{x}|X_{obs} = \bar{x}, \tilde{\mathcal{M}}_k)p(\tilde{\mathcal{M}}_k), \tag{5.2}$$

for a uniform prior, where X_{obs} is a stochastic variable for observations, X_k is a stochastic variable for the projection. The likelihood is given by

$$p(X_k = \bar{x}|X_{obs} = \bar{x}, \tilde{\mathcal{M}}_k) = \int p(X_k = \bar{x}|\tau, \tilde{\mathcal{M}}_k)p(\tau|X_{obs} = \bar{x}, \tilde{\mathcal{M}}_k)d\tau, \tag{5.3}$$

where $p(X_k = \bar{x}|\tau, \tilde{\mathcal{M}}_k)$ and $p(\tau|X_{obs} = \bar{x}, \tilde{\mathcal{M}}_k)$ are given by equations 2.7, 2.8, and 2.9 for the GPs modeling f_k and g_k respectively. This properly expresses the probability of a model given an observation.

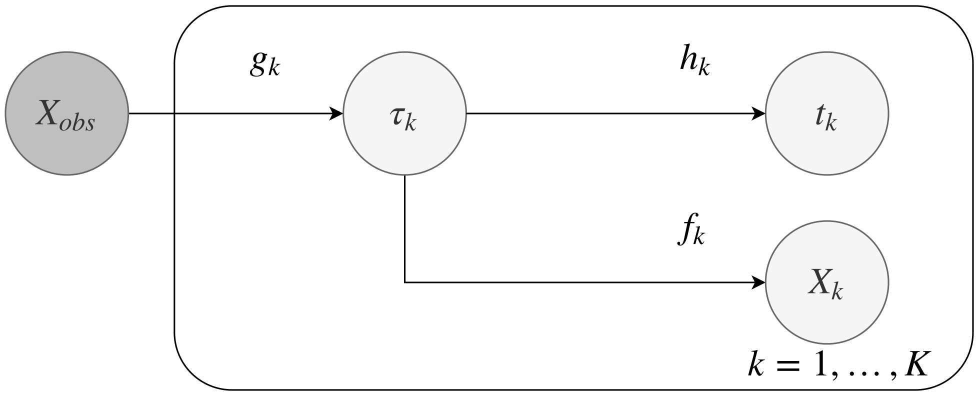


Figure 5.3: The model in plate notation.

The final piece is to make arrival time predictions. Predicting the arrival time to the next stop can be seen as learning the remaining time of a segment t_k given a point in progression τ. Since time is naturally smooth, a GP prior is assumed for the function $t_k = h_k(\tau)$. This gives

$$p(t_k|X_{obs} = \bar{x}, \mathcal{M}_k) = \int p(t_k|\tau, \mathcal{M}_k) p(\tau|X_{obs} = \bar{x}, \mathcal{M}_k) d\tau, \tag{5.4}$$

where $p(t_k|X_{obs} = \bar{x}, \mathcal{M}_k)$ and $p(\tau|X_{obs} = \bar{x}, \tilde{\mathcal{M}}_k)$ are given by equations 2.7, 2.8, and 2.9 for the GPs modeling h_k and g_k respectively. The trajectory model $\mathcal{M}_k$ for trajectory T_k is given by the triple $\mathcal{M}_k = (f_k, g_k, h_k)$, and the final model $\mathcal{M}$ is given by the set of trajectory models for all K previously observed trajectories, as illustrated in Figure 5.3.

Approximating the mapping onto τ

The integrals in equations 5.4 and 5.3 have no closed form solutions. Although they can be approximated by using sampling algorithms, those are computationally expensive. To avoid this, the likelihood is approximated as δ_{μ_τ}, that is, a Dirac delta function at the mean prediction. Intuitively, this approximation discards the variance of the distribution, treating it as a point-estimated value. This simplifies equation 5.4 to

$$\begin{aligned} p(X_k = \bar{x}|X_{obs} = \bar{x}, \tilde{\mathcal{M}}_k) &= \int p(X_k = \bar{x}|\tau, \tilde{\mathcal{M}}_k) p(\tau|X_{obs} = \bar{x}, \tilde{\mathcal{M}}_k) d\tau \\ &= \int p(X_k = \bar{x}|\tau, \tilde{\mathcal{M}}_k) \delta_{\mu_\tau} d\tau \\ &= \int p(X_k = \bar{x}|\tau, \tilde{\mathcal{M}}_k) d\tau \times \int \delta_{\mu_\tau} d\tau = \\ &= \int p(X_k = \bar{x}|\tau, \tilde{\mathcal{M}}_k) d\tau \times 1 = \\ &= \int p(X_k = \bar{x}|\tau, \tilde{\mathcal{M}}_k) d\tau, \end{aligned} \tag{5.5}$$

through the linearity of the integral operator and the property

$$\int_{-\infty}^{\infty} \delta_{\mu_\tau} = 1, \tag{5.6}$$

of the Dirac delta function. The resulting integrand is Gaussian so the integral has a closed form solution, and can be computed efficiently. Equation 5.3 is in the same way simplified to

$$p(t|X_k = \bar{x}, \mathcal{M}_k) = \int p(t_k|\tau, \tilde{\mathcal{M}}_k) \delta_{\mu_\tau} d\tau \tag{5.7}$$

5.2 Training the Model

Before models are trained the data must be pre-processed. In particular, it must be projected onto an Euclidian space and normalised. Training a model $\mathcal{M}_k$ on the processed data is then done by learning the function family $f_k = (f_{p_x}, f_{p_y}, f_{v_x}, f_{v_y})$, g_k and h_k by fitting GPs using MAP-estimation. However, there are more constraints on the functions than can be formulated in priors; in particular for the synchronisation function g. In addition, training GPs for f_k so that their likelihoods can be used as a similarity metric is not completely straightforward.

Projecting onto Euclidian Space

A GP assumes that the data lives in an Euclidian space, which is not the case for the spatial properties of the data set (it lives in latitude/longitude-space). Consequently, before any GPs can be trained, the data has to be be mapped onto a proper Euclidian space.To do so, the equirectangular map projection is used. All trajectories in a certain segment is projected onto their own space. This gives a smaller error in the equirectangular projection, compared to projecting the entire data set at a time, and still places all trajectories in the same space, allowing meaningful comparison.

Normalising the Data

The GPs trained on the data use a constant mean function $m(x, x') = 0$, which requires normalising the data to have zero mean. It is also scaled to be centered around 0, which helps prevent numerical instabilities. It is important that the scaling does not deform the Euclidian space of the data, which would violate assumptions made by GPs. Because of this, the scaling is done through

$$z_{p_x} = \frac{x_{p_x} - \bar{x_{p_x}}}{p_{max}}$$
$$z_{p_y} = \frac{x_{p_y} - \bar{x_{p_y}}}{p_{max}}$$
$$z_{v_x} = \frac{x_{v_x} - \bar{x_{v_x}}}{v_{max}}$$
$$z_{v_y} = \frac{x_{v_y} - \bar{x_{v_y}}}{v_{max}}$$

where $p_{max} = \max\left(\max_k x_{p_x}^{(k)}, \max_k x_{p_y}^{(k)}\right)$ and $v_{max} \max\left(\max_k x_{v_x}^{(k)}, \max_k x_{v_y}^{(k)}\right)$. That is, the observations are scaled in position by the largest position observed, and likewise in velocity. To avoid introducing additional notation, normalised observations will also be referred to as x. However, the normalising transformation is bijective, and the underlying mathematics is sound for both normalised and unnormalised observations.

An alternative to normalising the observations is to let the GPs have a constant mean function which value is learned from the data. However, this would add another parameter to the training process, so the described method of normalisation is preferred.

Kernel Selection

The choice of kernels greatly impact the models capabilities, and are very important parts of the model specification. The kernels chosen for the different models are described in Table 5.1. Since velocities change quickly, Matérn32 is used instead of RBF to model f_{v_x} and f_{v_y}. The Linear additions for g and h are motivated by the fact that time is linear, and by the assumption that a bus line should show a spatial linear trend.

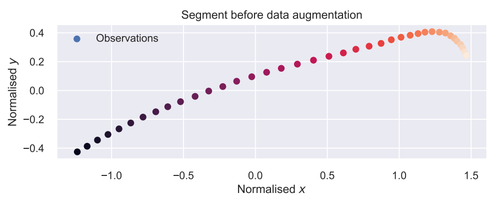


Figure 5.4: Spatial progression of a trajectory before data augmentation. Color indicate τ from dark ($\tau = 0.0$) to bright ($\tau = 1.0$).

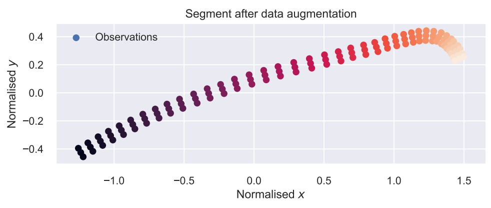


Figure 5.5: Spatial progression of a trajectory after data augmentation. Color indicate τ from dark ($\tau = 0.0$) to bright ($\tau = 1.0$).

Table 5.1: Kernels for the different models.

Model	Kernel
f_{p_x}	RBF
f_{p_y}	RBF
f_{v_x}	Matèrn32
f_{v_y}	Matèrn32
g	RBF + Linear
h	RBF + Linear

Learning the Synchronisation Model

The synchronisation model g_k have some critical properties that need to be enforced in training. One such property is that, it should be monotonically increasing in the direction of progression, and stationary orthogonal to it. This is intuitively explained by the fact that a vehicle is no closer to its destination should it drive more to the left or right on a road; only the progression *along* the road matters. To enforce this property, data augmentation is used.

When training GPs using a stationary kernel function, it is assumed that the data is uniformly distributed, which is not the case for the data set used. In the case of function $g_k : S' \mapsto \tau$, the data is assumed to be uniformly distributed in S', but as described in Chapter 4, all data is collected approximately uniform in *time*. This causes many observations to be generated in close proximity during stand-stills, which skews the learning of the kernel lengthscale parameters to small values. To make data approximately uniformly distributed spatially instead and avoid this problem, a technique called *stop compression* is used.

g_k is bivariate and evaluating its posterior is expensive. To make it more efficient, g_k is learned using a SPG with $0.3N$ inducing inputs, where N is the number of trajectory observations. Event though it is approximated, g_k can model complex trajectories as seen in Figure 5.6 and Figure 5.7.

Enforcing Orthogonality

For reasons previously described, g_k should be monotonically increasing in the direction of progression and stationary orthogonal to it. To enforce learning such a function, synthetic data is generated by duplicating the observations orthogonally to progression. This process is illustrated in Figure 5.4 and Figure 5.5.

Handling Stand-stills

Stand-stills causes an uneven spatial distribution of data. This is prevented using stop compression, which aggregates observations in close proximity into a single observation through

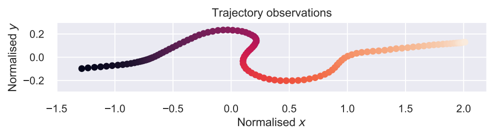


Figure 5.6: Spatial observations of a segment. Colour indicate values of τ, from brightdark ($\tau = 0.0$) to bright ($\tau = 1.0$).

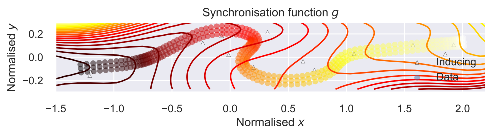


Figure 5.7: The learned synchronisation model g. Colour indicate values of τ, from brightdark ($\tau = 0.0$) to bright ($\tau = 1.0$).

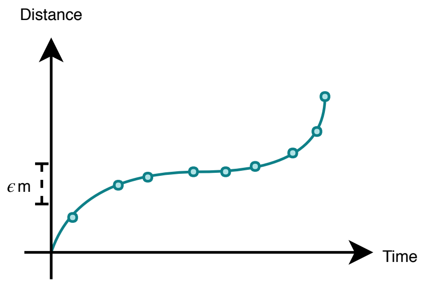


Figure 5.8: Trajectory before stop compression. Observations are approximately uniformly distributed temporally, but form a cluster much denser than the desired ϵ spatialy.

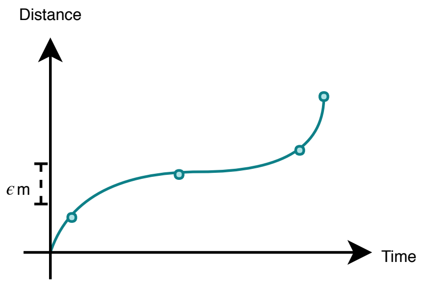


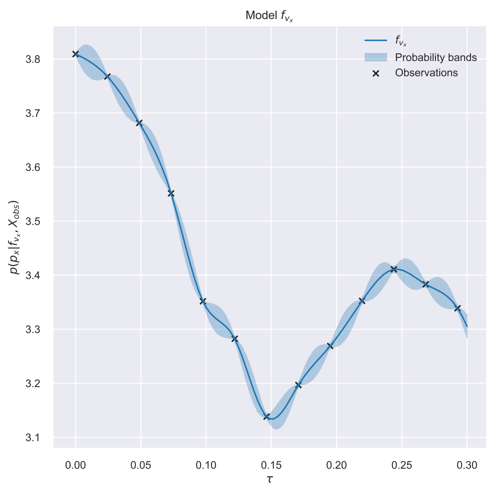
Figure 5.9: Spatial progression of a trajectory after stop compression. Data which within a radius of ϵ has been compressed so the resulting data is approximately uniformly distributed spatially.

averaging. Observations within a radius of ϵ are clustered into a single observation with the mean value of the clustered observations. An example of this is seen Figure 5.8 and Figure 5.9. During evaluation $\epsilon = 4$ metres was used.

Learning the Motion Pattern Model

The motion pattern model of a single trajectory is defined by the family of function $(f_{p_x}, f_{p_y}, f_{v_x}, f_{v_y})$ that map from τ to S. The uncertainty of these models depend on how the buses move and how accurate sensors they are equipped with. Since traffic moves smoothly and modern sensors can measure position and velocity quite accurately, there is very little noise in these models, which leads to an undesirably low variance. This causes a motion pattern model to consider observations that are just a few meters away to be highly unlikely, which means that it will never be able to confidently predict a likely motion pattern for observed data. This issue is addressed by defining *pseudo-clusters* models, which have desirable variance. A pseudo-cluster f' is created for each GP in the motion pattern model. They are all created in the same way, and unless specifically stated, f' can belong to any GP in a motion pattern model. I.e. f'_{p_x} would belong to f_{p_x}. f' is modeled pont-wise as $f'(\tau^*) \sim \mathcal{N}(\mu(\tau^*), \sigma^2(\tau^*))$ for any point τ^* and for each state model $f_{p_x}, f_{p_y}, f_{v_x}, f_{v_y}$. The pseudo-cluster parameters $\mu(\tau^*), \sigma^2(\tau^*)$ are given by using the *combining formula* [25]

$$\mu(\tau^*) = \frac{\sum_{j=1}^{J} N_j \mu_j(\tau^*)}{\sum_{j=1}^{J} N_j} \tag{5.8}$$



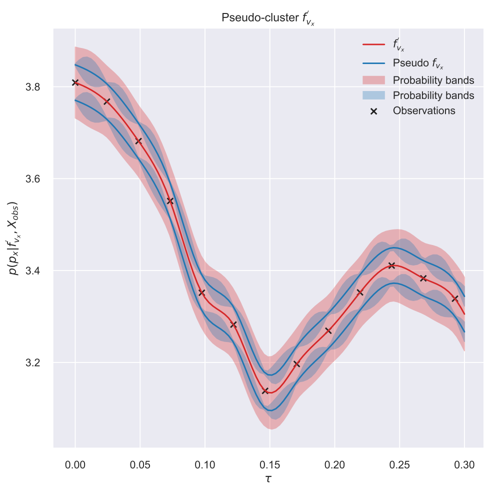


Figure 5.10: A state model fit on observation data. Trajectory data close-by will appear highly unlikely.

Figure 5.11: A state model fit on a pseudo-cluster. Trajectory data close-by will appear likely.

$$\sigma^2(\tau^*) = \frac{\sum_{j=1}^{J} N_j(\sigma_j^2(\tau^*) + \mu_j^2(\tau^*))}{\sum_{j=1}^{J} N_j} \tag{5.9}$$

with uniform weights N over two copies of f, offset by δ orthogonal to τ. δ is a user-specified parameter which controls what distance the model should consider close when comparing trajectories. During evaluation $\delta =< TODO >$ was used. As is illustrated in Figure 5.10 and Figure 5.11, this creates a larger variance, which enables meaningful comparison of models.

5.3 Querying the model

There are two queries that can be asked of the model. The first one is arrival time prediction and the second is event detection. This section addresses how both these queries are performed.

Arrival Time Prediction

The predicted arrival time for observation $\bar{x}$ is given by equation 5.4 for the most similar, or probable, model $\hat{\mathcal{M}}$. $\hat{\mathcal{M}}$ is given by Bayesian model selection with a uniform prior over all learned models

$$\hat{\mathcal{M}} = \arg\max_k p(\mathcal{M}_k|X = \bar{x}) = p(X = \bar{x}|\theta_k, \mathcal{M}_k)p(\mathcal{M}_k) \propto p(X = \bar{x}|\hat{\theta}_k, \mathcal{M}_k), \tag{5.10}$$

where $\hat{\theta}_k$ are the ML-estimated hyper-parameters of corresponding model $\mathcal{M}_k$. Given the uniform prior, the model which maximises the likelihood is selected, which is equivalent with the model that maximises the log likelihood, since the log function is monotonically increasing. The model log likelihood is given by the sum of the log likelihood of the pseudo-

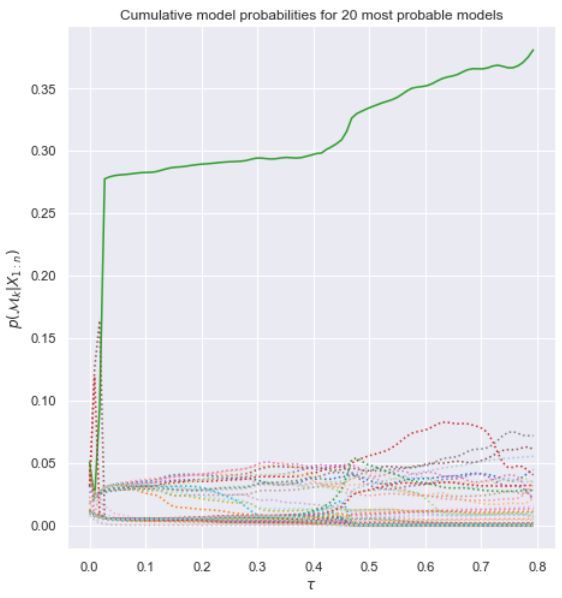


Figure 5.12: Plots of trajectory model probabilities from observed data. The model quickly identifies the true model, indicated by the solid line.

cluster of individual GP models

$$\begin{aligned}\log p(X_k = x|\hat{\theta}_k, \mathcal{M}_k) =& \log p(X_k = x|f'_{p_x}, X_{obs} = x) \\ &+ \log p(X_k = x|f'_{p_y}, X_{obs} = x) \\ &+ \log p(X_k = x|f'_{v_x}, X_{obs} = x) \\ &+ \log p(X_k = x|f'_{v_y}, X_{obs} = x).\end{aligned} \tag{5.11}$$

where the log likelihoods of pseudo-clusters are given by equation 2.10. An illustration of model probabilities computed in this way can be seen in Figure 5.12. Figure 5.13 shows trajectory models that are presented with new observations, together with the corresponding model probabilities in Figure 5.14. While predicting arrival time through equation 5.4 gives the correct mean, the variance in the predictive distribution $t_k \sim \mathcal{N}(\mu_{h_k}, \sigma_{h_k})$ only considers the uncertainty in h_k. This is very optimistic, since the uncertainty in arrival time depend on variance in velocity in the motion patterns to a great extent. The uncertainty is more accurately modeled as $\sigma_{t_k} = \sigma_{h_k} + \sigma_{mp}$ where the addition σ_{mp} is the motion pattern uncertainty given by integrating over the probability bands of f_{v_x} and f_{v_y}

$$\sigma_{mp} = \int_\tau 2\sigma_{v_x}(\tau) + 2\sigma_{v_x}(\tau) d\tau \tag{5.12}$$

where $\sigma_{v_x}(\tau)$ and $\sigma_{v_y}(\tau)$ are given by the square root of the pseudo cluster variances from equation 5.9 for velocity models f'_{v_x} and f'_{v_y}.

$$\int_{\tau_0}^{\tau_1} 2\sigma(\tau) d\tau = \int_{\tau_0}^{\tau_1} 2\sigma d\tau = 2\sigma \int_{\tau_0}^{\tau_1} d\tau = 2\sigma(\tau_1 - \tau_0)$$

Event Detection

The problem of detecting events have not yet been investigated.

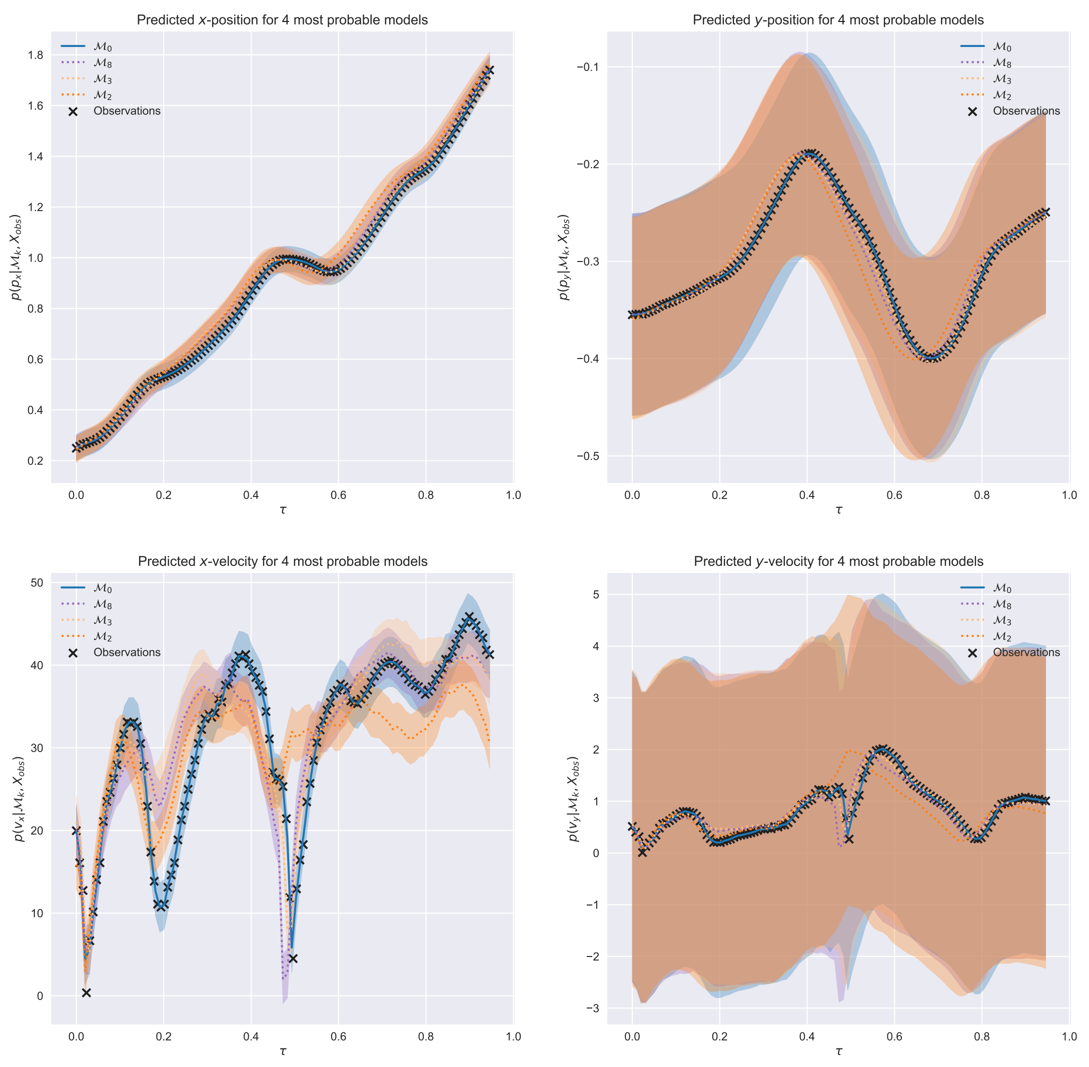


Figure 5.13: Plots of state models mean functions and probability bands for different $\mathcal{M}_k$.

5.4 System Implementation Details

This section describes the implementation details of the system and how data was processed. The GP implementations were done in Python, using the GPy framework [7], which uses the BFGS algorithm [5] for optimising θ. The database PostgreSQL [17] was used for storing trained GPs.

The integrals in equation 5.12 were computed by approximating them by <inte klar här, osäker>

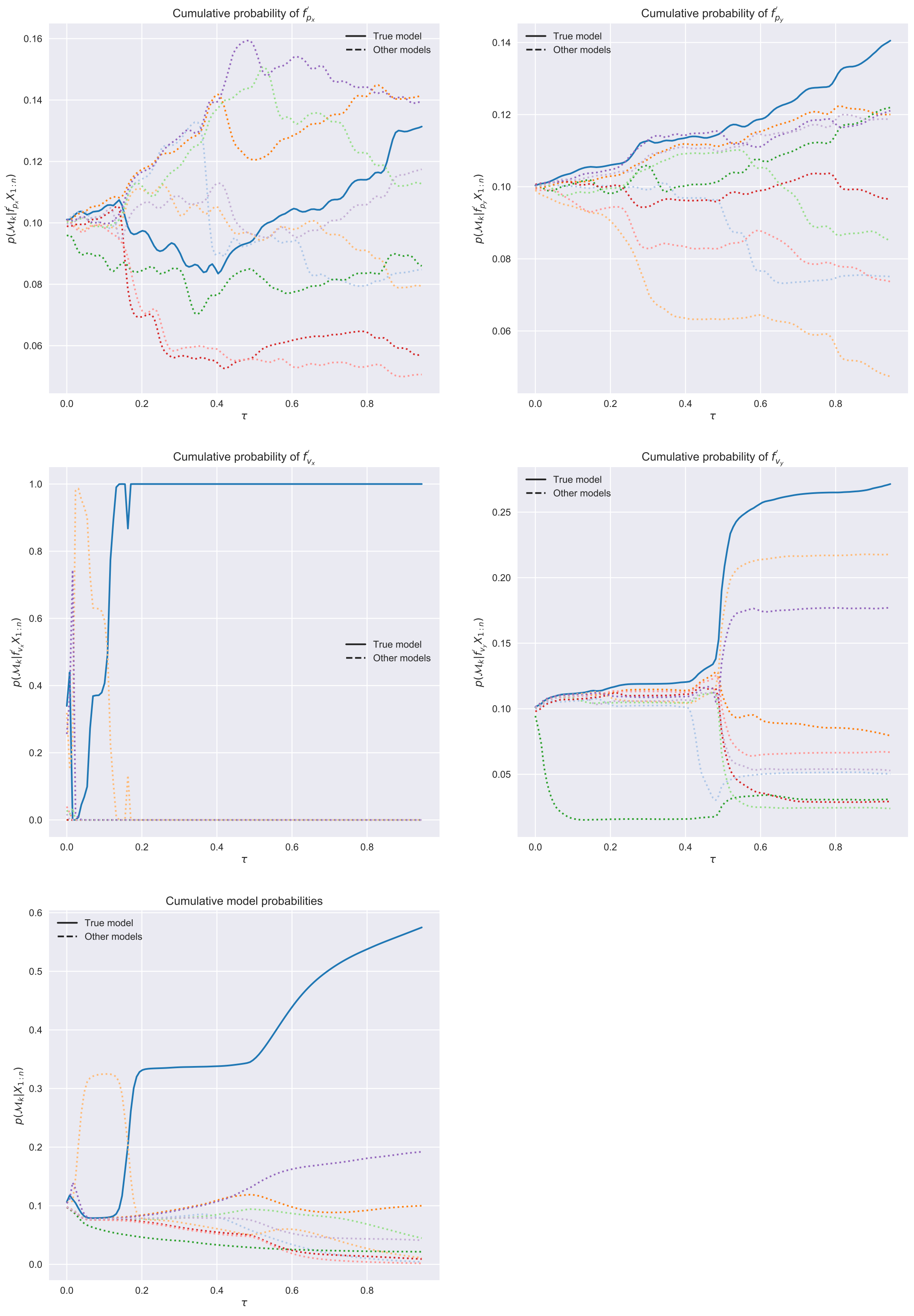


Figure 5.14: Plots of the model probabilities for each state model. The model places high probability on the true model. It is able to identify the true model solely from velocity. Since all motions patterns are recorded on the same road road, they are incredibly similar and so the spatial information does not convey much information, as indicated by the noisy plots.

6 Results

7 Discussion

klustering av rörelsemönster mängden data som används, inducing inputs

7.1 Method

This section addresses the method used and its flaws.

Maximum Likelihood Estimation

ML-estimated models are typically prone to overfitting, and attempts were made to MAP-estimate them. The kernel parameters were modeled as $\ell \sim \Gamma(\alpha_\ell, \beta_\ell)$, $\sigma^2 \sim \Gamma(\alpha_\sigma^2, \beta_\sigma^2)$, $\sigma_b^2 \sim \Gamma(\alpha_{\sigma_b^2}, \beta_{\sigma_b^2})$, and were estimated by fitting distributions to populations of ML-estimated parameters of 100 trajectories set aside solely for this purpose; the models were never presented with these trajectories during training or testing. An illustration of the parameters estimation can be seen in Figure 7.1. Using MAP-estimation did however have a much greater regularising effect than desired. Since the project considered single-trajectory motion patterns the characteristics of a single trajectory was paramount to be able to identify it. Paradoxically, this means that over-fitting was *good* for the model; regularisation caused the differences in models to be "smoothed out", making it hard for the model to tell them apart.

Low Sample Sizes

The project did use a fairly low amount of trajectories for its evaluation. The simple reason for this is time constraints; evaluating the model performance on only 50 trained models took over two hours. Attempts were made to use SGPs for all functions, which would have made computing model posteriors much faster and enabled training and evaluating on a larger data set, but this had a large regularising effect akin to MAP-estimation, as illustrated in Figure 7.2.

Unrepresentative Samples

Using only data from a single bus line is of course hardly representative of the greater population of bus lines. <inte klar. kanske mät lite olika saker för olika buslinjer för att jämföra>

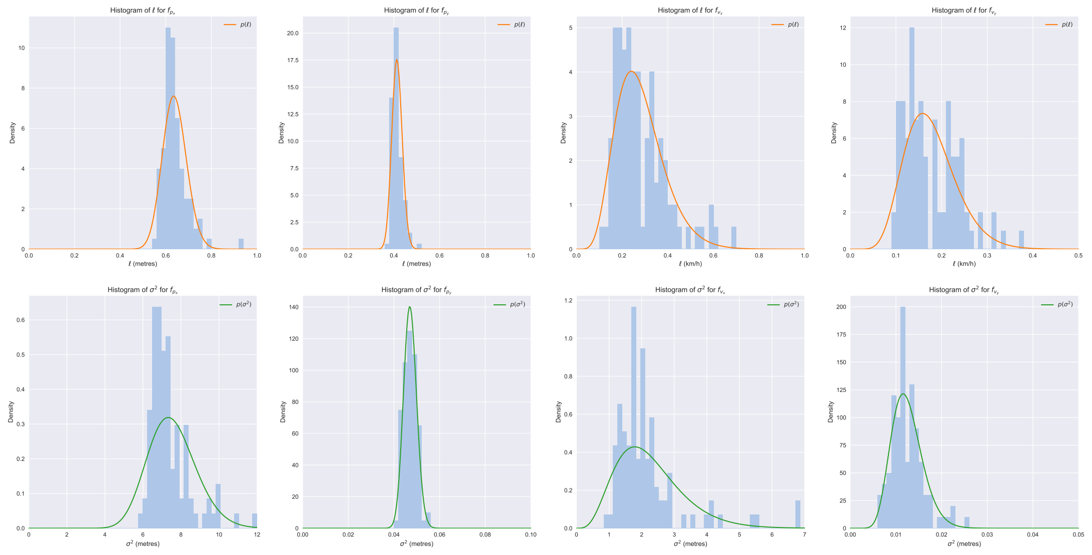


Figure 7.1: Plots of estimated prior parameters from state models for segmeent one (Parameters of g and h are not shown). The histograms show the parameter distributions from the learned trajectory models, and the Gamma distributions show the estimated priors.

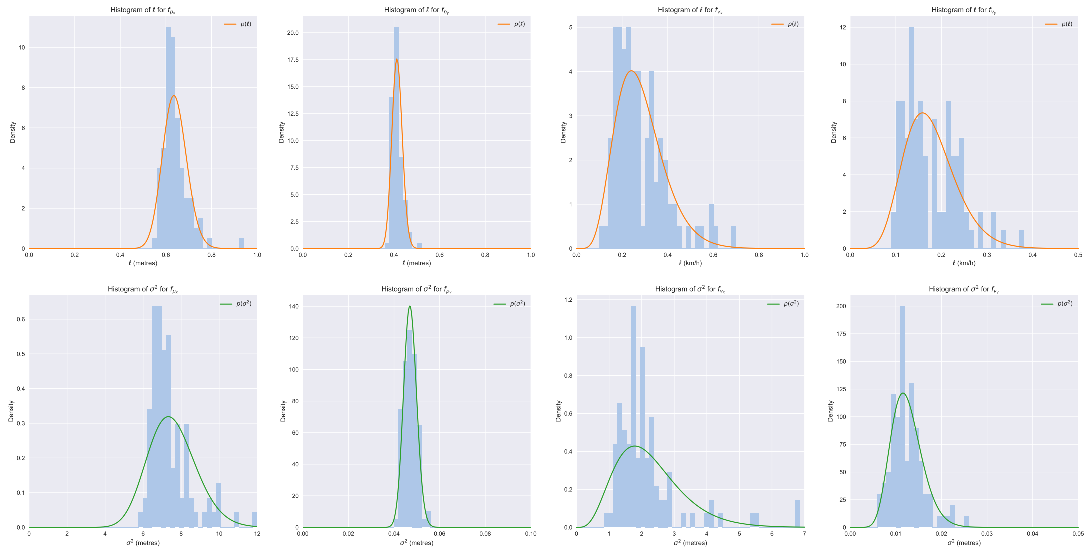


Figure 7.2: PLACEHOLDER - will describe regularisation induced by inducing inputs.

8 Conclusion

Bibliography

[1] J. Scott Armstrong and Fred Collopy. "Error measures for generalizing about forecasting methods: Empirical comparisons". In: *International Journal of Forecasting* 8.1 (June 1992), pp. 69–80. ISSN: 0169-2070. DOI: 10.1016/0169-2070(92)90008-W.

[2] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] Damian Campo, Mohamad Baydoun, Lucio Marcenaro, Andrea Cavallaro, and Carlo S. Regazzoni. "Modeling and classification of trajectories based on a Gaussian process decomposition into discrete components". In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* (Aug. 2017), pp. 1–6. DOI: 10.1109/AVSS.2017.8078495.

[4] David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B Tenenbaum, and Zoubin Ghahramani. "Structure discovery in nonparametric regression through compositional kernel search". In: *arXiv preprint arXiv:1302.4922* (2013).

[5] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[6] Sepideh Afkhami Goli, Behrouz H. Far, and Abraham O. Fapojuwo. "Vehicle Trajectory Prediction with Gaussian Process Regression in Connected Vehicle Environment⋆". In: *2018 IEEE Intelligent Vehicles Symposium (IV)* (June 2018), pp. 550–555. ISSN: 1931-0587. DOI: 10.1109/IVS.2018.8500614.

[7] GPy. *GPy: A Gaussian process framework in python*. http://github.com/SheffieldML/GPy. since 2012.

[8] Zegeye Kebede Gurmu and Wei (David) Fan. "Artificial Neural Network Travel Time Prediction Model for Buses Using Only GPS Data". In: *Scholar Commons* 17.2 (2014), p. 3. DOI: 10.5038/2375-0901.17.2.3.

[9] ByeoungDo Kim, Chang Mook Kang, Seung Hi Lee, Hyunmin Chae, Jaekyum Kim, Chung Choo Chung, and Jun Won Choi. "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network". In: *arXiv preprint arXiv:1704.07049* (2017).

[10] Kihwan Kim, Dongryeol Lee, and Irfan Essa. "Gaussian process regression flow for analysis of motion trajectories". In: *2011 International Conference on Computer Vision* (Nov. 2011), pp. 1164–1171. ISSN: 2380-7504. DOI: 10.1109/ICCV.2011.6126365.

[11] Christiaan Leysen, Mathias Verbeke, Pierre Dagnely, and Wannes Meert. "Energy consumption profiling using Gaussian processes". In: *2016 IEEE 8th International Conference on Intelligent Systems (IS)* (Sept. 2016), pp. 470–477. DOI: `10.1109/IS.2016.7737463`.

[12] Richard D. Morey, Rink Hoekstra, Jeffrey N. Rouder, Michael D. Lee, and Eric-Jan Wagenmakers. "The fallacy of placing confidence in confidence intervals". In: *Psychon. Bull. Rev.* 23.1 (Feb. 2016), pp. 103–123. ISSN: 1069-9384. DOI: `10.3758/s13423-015-0947-8`.

[13] Brendan T. Morris and Mohan M. Trivedi. "Learning and Classification of Trajectories in Dynamic Scenes: A General Framework for Live Video Analysis". In: *2008 IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance* (Sept. 2008), pp. 154–161. DOI: `10.1109/AVSS.2008.65`.

[14] Duc-Duy Nguyen, Chan Le Van, and Muhammad Intizar Ali. *Vessel Destination and Arrival Time Prediction with Sequence-to-Sequence Models over Spatial Grid*. ACM, June 2018. ISBN: 978-1-4503-5782-1. DOI: `10.1145/3210284.3220507`.

[15] Junbiao Pang, Jing Huang, Yong Du, Haitao Yu, Qingming Huang, and Baocai Yin. "Learning to Predict Bus Arrival Time From Heterogeneous Measurements via Recurrent Neural Network". In: *IEEE Transactions on Intelligent Transportation Systems* (2018).

[16] Marco A. F. Pimentel, David A. Clifton, and Lionel Tarassenko. *Gaussian process clustering for the functional characterisation of vital-sign trajectories*. IEEE, Sept. 2013. DOI: `10.1109/MLSP.2013.6661947`.

[17] *PostgreSQL: The world's most advanced open source database*. [Online; accessed 21. Feb. 2019]. Feb. 2019. URL: `https://www.postgresql.org`.

[18] Carl Rasmussen and Christopher Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[19] Mathieu Sinn, Ji Won Yoon, Francesco Calabrese, and Eric Bouillet. "Predicting arrival times of buses using real-time GPS measurements". In: *2012 15th International IEEE Conference on Intelligent Transportation Systems* (Sept. 2012), pp. 1227–1232. ISSN: 2153-0017. DOI: `10.1109/ITSC.2012.6338767`.

[20] Steven W Smith et al. "The scientist and engineer's guide to digital signal processing". In: (1997).

[21] John Parr Snyder and Philip M Voxland. *An album of map projections*. 1453. US Government Printing Office, 1989.

[22] Jingren Tang, Hong Cheng, Yang Zhao, and Hongliang Guo. "Structured dynamic time warping for continuous hand trajectory gesture recognition". In: *Pattern Recognit.* 80 (Aug. 2018), pp. 21–31. ISSN: 0031-3203. DOI: `10.1016/j.patcog.2018.02.011`.

[23] Yee W Teh, Michael I Jordan, Matthew J Beal, and David M Blei. "Sharing clusters among related groups: Hierarchical Dirichlet processes". In: *Advances in neural information processing systems*. 2005, pp. 1385–1392.

[24] Mattias Tiger and Fredrik Heintz. "Gaussian Process Based Motion Pattern Recognition with Sequential Local Models". In: *2018 IEEE Intelligent Vehicles Symposium (IV)* (June 2018), pp. 1143–1149. ISSN: 1931-0587. DOI: `10.1109/IVS.2018.8500676`.

[25] Mattias Tiger and Fredrik Heintz. "Online sparse Gaussian process regression for trajectory modeling". In: *2015 18th International Conference on Information Fusion (Fusion)* (July 2015), pp. 782–791. URL: `https://ieeexplore.ieee.org/abstract/document/7266640/citations?tabFilter=papers#citations`.

[26] Michalis Titsias. "Variational learning of inducing variables in sparse Gaussian processes". In: *Artificial Intelligence and Statistics*. 2009, pp. 567–574.

[27] Quan Tran and Jonas Firl. *Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression*. IEEE, June 2014. DOI: 10.1109/IVS.2014.6856480.

[28] M. Vlachos, G. Kollios, and D. Gunopulos. *Discovering similar multidimensional trajectories*. IEEE, Feb. 2002. DOI: 10.1109/ICDE.2002.994784.

[29] Haozhou Wang, Han Su, Kai Zheng, Shazia Sadiq, and Xiaofang Zhou. *An effectiveness study on trajectory similarity measures*. Australian Computer Society, Inc., Jan. 2013. ISBN: 978-1-921770-22-7. URL: http://dl.acm.org/citation.cfm?id=2525416.2525418.

[30] Xiaogang Wang, Keng Teck Ma, Gee-Wah Ng, and W. Eric L. Grimson. “Trajectory analysis and semantic region modeling using a nonparametric Bayesian model”. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition* (June 2008), pp. 1–8. ISSN: 1063-6919. DOI: 10.1109/CVPR.2008.4587718.

[31] Cort J Willmott and Kenji Matsuura. “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance”. In: *Climate research* 30.1 (2005), pp. 79–82.

[32] Zhang Zhang, Kaiqi Huang, and Tieniu Tan. *Comparison of Similarity Measures for Trajectory Clustering in Outdoor Surveillance Scenes*. Vol. 3. Aug. 2006. DOI: 10.1109/ICPR.2006.392.

[33] Bolei Zhou, Xiaogang Wang, and Xiaoou Tang. “Random field topic model for semantic region analysis in crowded scenes from tracklets”. In: *CVPR 2011* (June 2011), pp. 3441–3448. ISSN: 1063-6919. DOI: 10.1109/CVPR.2011.5995459.