Linköping University | Department of Computer and Information Science
Master thesis, 30 ECTS | Computer Science
202018 | LIU-IDA/LITH-EX-A--2018/XXX--SE

Gaussian Process Regression based GPS Variance Estimation and Trajectory Forecasting

Regression med Gaussiska Processer för Estimering av GPS Varians och Trajektorie Prognostisering

Linus Kortesalmi

Supervisor : Mattias Tiger
Examiner : Fredrik Heintz

External supervisor : Simon Johansson

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

Abstract

Abstract.tex

Acknowledgments

Acknowledgments.tex

Contents

List of Figures

Todo list

1 Introduction

1.1 Motivation

1.2 Aim

The aim of this thesis project is to explore a novel dataset and formulate problems and solution suggestions which could be interesting to not only the dataset provider but also in a broader context. The dataset is currently used to provide time forecasts on the public transportation available in Östergötland county.

1.3 Research Questions

The specific research questions treated in this thesis project is presented in this section. They are divided into three groups: *Metadata Questions*, *Higher-Level Problem Questions*, and *Problem Specific Questions*. The *Metadata Questions* investigate and explore the provided dataset in depth. They cover issues such as pre-processing, data noise and outliers. The questions can be seen as a pre-study to Machine Learning. The insights from these questions are the basis for the *Higher-Level Problem Questions*. The *Higher-Level Problem Questions* formulate various high-level problems by using the information available both internally in the dataset and externally from sources outside the provided dataset. High-level problems are problems which are not inherent in the dataset but rather problems which can be solved by using the information available in the dataset. They also aim to suggest solutions for these formulated problems. The *Problem Specific Questions* are questions related to a specific problem and solution described by the *Higher-Level Problem Questions*.

Metadata Questions:

1. *What kind of information is available in the dataset provided by Östgötatrafiken AB?*
   This question explores what kind of data the dataset contains. The dataset is a novel dataset which has not been worked on before. The provided documentation is minimal, which makes this question non-trivial and the insights from the question even more valuable. It analyses the features of the dataset, e.g. different event types, event parameters, and event structure.

2. *What pre-processing needs to be done in order to solve higher-level problems?*
The higher-level problems are here defined as problems which are not inherent inside the dataset but rather problems which can be solved by using the dataset. The pre-processing focuses instead of solving problems inherent in the dataset, such as processing events and extracting relevant information from them or building a knowledge base by looking at the order of events.
3. *How can noisy measurements be detected?*
Noisy measurements can affect the solutions for higher-level problems negatively. Various anomaly detection algorithms can be applied to detect such cases, but they could also be solved using dataset-tailored algorithms.
4. *How is the provided dataset related to external data sources and how can they be combined?*
There are external data sources available which could complement the data in the provided dataset. This question explores if these sources could be combined in order to solve problems on a higher level.

Higher-Level Problem Questions:

1. *What are some of the higher-level problems which can be formulated using the available data?*
These problems can utilise both the data available in the provided dataset and any complementary external data. The answer to this question will not be a complete list of all possible problems, but rather a short list of a few interesting examples. Each formulated problem will have its core problem explained.

Fråga: Ska jag lista de problem jag har redan här? Annars blir det svårt att fortsätta med Problem Specific Question.

2. *What are some of the solutions to the formulated problems?*
The list of solutions for each formulated problem will not be a complete list of all possible solutions. The solutions will be tackling the core of each formulated problem. Each solution will explicitly state if there is a baseline available for comparison or if one could be easily created.
3. *Who could benefit from the solutions?*
This question analyses the solutions in a broader context, e.g. from a societal or ethical point of view. For example, a solution could yield great results for the industry at the cost of consumer privacy.

Problem Specific Questions

1. *How can GPS variance estimation be solved with combined Gaussian Processes Regression using a local trajectory model?*

This question is very explicit and stems from the results from asking the questions in the two previous groups. Is this okay, or do I need to add more information after each question in the previous groups? In one way this question kind of pops up from nowhere.

2. *How can the GPS variance estimation model be evaluated?*
The method employed makes certain assumptions regarding the inherent noise of the data. The kernels applied to the model need to evaluated. The local trajectory model shall be compared with a global trajectory model.
3. *How can Trajectory Forecasting be realised with Gaussian Processes Regression using a local trajectory model?*
4. *How can the Trajectory Forecasting model be evaluated in the context of information gain?*
The Trajectory Forecasting model could, for example, be evaluated by comparing the new forecasts with the existing forecasts from the baseline created by the internal system of Östgötatrafiken AB. This evaluation leads to information regarding which model to use to get more precise forecasts. The model could also be evaluated by looking at what kind of information and insights are available from the output of the model. A model which returns a probability distribution

of arrival times, which are updated in real-time, should be evaluated in a broader context than precision of a single forecast.

Det här kanske inte hör hemma här. Det är väldigt klumpigt skrivet iaf.

1.4 Delimitations

The dataset is provided by Östgötatrafiken AB and is not available for public use.

2 Theory

3 Data

This chapter describes the spatiotemporal dataset used in the thesis project. The data provider is briefly mentioned, followed by the structure of the data. This chapter also mentions issues encountered in the data set.

3.1 Background

The dataset was provided by Östgötatrafiken AB and contains GB of data. Östgötatrafiken AB is owned by Östergötland County and is responsible for the public transportation in the county. This thesis project only analysed the bus data available in the given data set. The dataset is a collection of documents, where one document represents a full day of data. A typical day has a document size of around GB.

300?

2.5?

Data Gathering

The process of gathering the data used in this thesis project can be generally described by the following simplified procedure:

1. Each Östgötatrafiken AB bus is running a system collecting data from sensors installed inside the bus.
2. The system collects the sensor data and transmits it to a central server or database.
3. A log containing all events for a full day is created and stored as a document in a collection.
4. The central server processes and analyses the data. The results from the data analysis is stored in the log.

Figure 3.1 illustrates the procedure. The collection of logs is the dataset used in this thesis project. The logs contain the GPS data from the buses and also events created by the "Internal Analysis" component in the system. The "Internal Analysis" component is simplified and is beyond the scope of this thesis project.

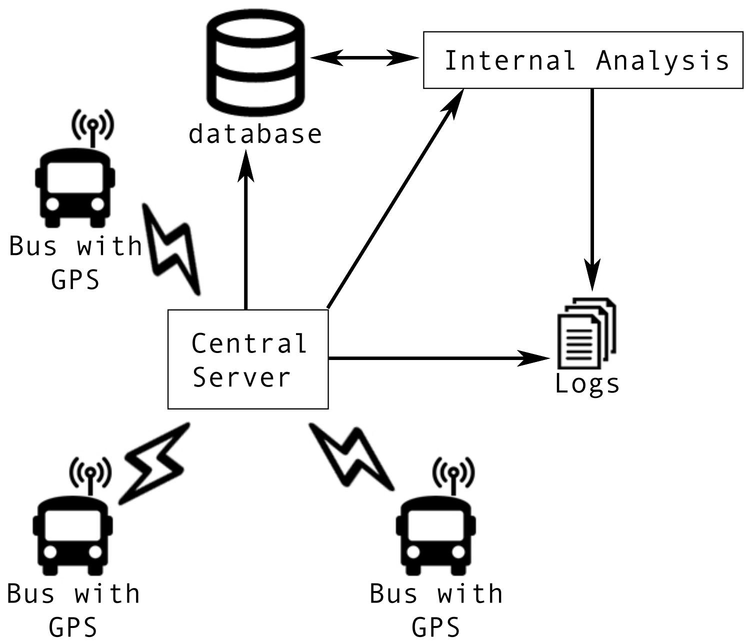

Figure 3.1: Simplified graph illustrating the data gathering process. Each bus is equipped with a GPS sensor and transmits its position to a central server/database. The dataset used in this thesis project is the log, which is a collection of documents. Each document contains the GPS data sent from all buses during a single day, together with data from the "Internal Analysis" component of the server.

3.2 Structure

A document in the dataset is made up of a large number of events representing a single day. A single day typically contains roughly 21 million events. Each event is represented by a single line of text. An event can be split into two groups: a header and a body. There are different types of events reported during the span of a single day. Each type has its own header and body structure.

Is it interesting to plot this exact number and its variance?

Event Example

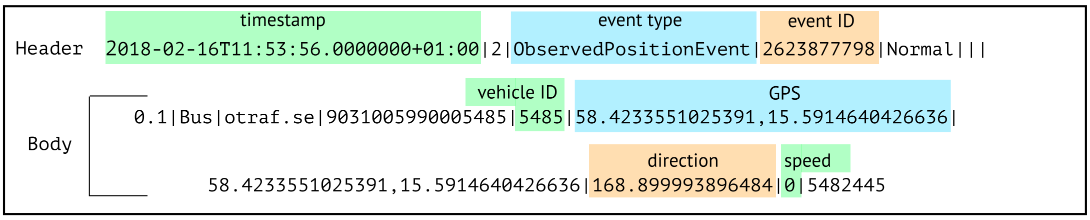

Figure 3.2: Example of a raw `ObservedPositionEvent` entry. The header and the body is separated by `|||`. Each parameter in the header and body is separated by a single `|`. Key parameters for the `ObservedPositionEvent` event type is highlighted.

Figure 3.2 illustrates an event with the event type `ObservedPositionEvent`. The header is defined as all the parameters before the `|||` separator. All the parameters after the separator is defined as the body of the event. In this example the header and body contain seven key parameters:

- *Timestamp:* A timestamp (`2018-02-16T11:53:56.0000000+01:00`), which is the timestamp from the system running on the bus.
- *Event Type:* The event type (`ObservedPositionEvent`).
- *Event ID:* The event id (`2623877798`). This is a number set by the system responsible for collecting the data from all buses. It is incremented for every event added to the log by either the database system or the "Internal Analysis" component in Figure 3.1.
- *Vehicle ID:* Unique ID for the bus transmitting its position.
- *GPS:* The GPS position of the bus in latitude and longitude.
- *Direction:* The direction of the bus.
- *Speed:* The current speed of the bus.

Event Types

The dataset contains 20 unique event types. Figure 3.3 visualises the distribution of event types for a random day in the dataset. The figure only gives an indication of what the true distribution could be, as it is computationally expensive to calculate the true distribution for the given dataset, due to its size. Knowledge about the true distribution is not required in order to reason about the event types. As the figure shows, the majority of events that occur are of the type `ObservedPositionEvent`, which is the event containing an updated GPS position for a vehicle.

Should we instead do a boxplot of the whole dataset? Will take a lot of time to create!

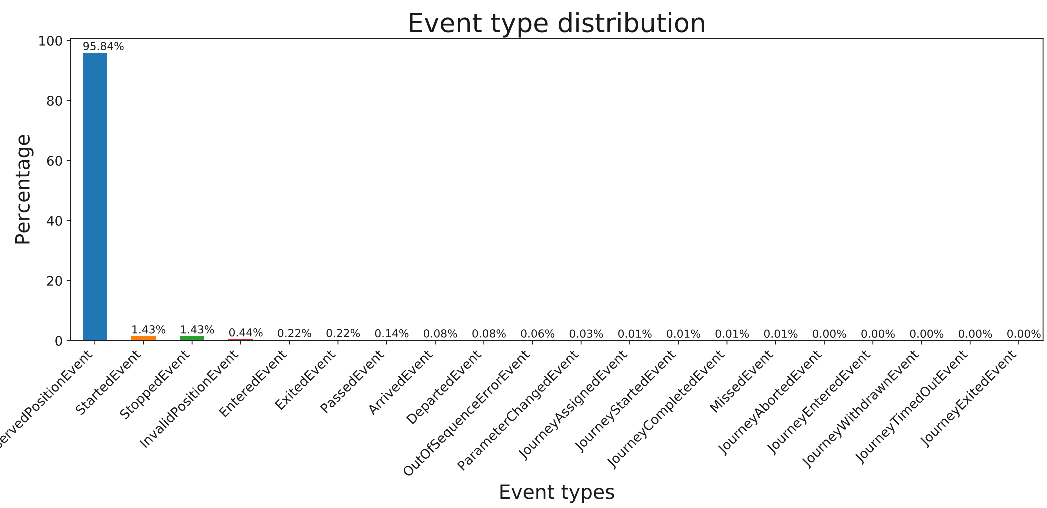

Figure 3.3: The distribution of event types for a random day in the dataset.

Of the 20 event types available in the dataset, this thesis project only used 12 of them. The events to use were chosen by analysing the log for a single day in great detail. Event types which occurred rarely and seemingly random were discarded, as no pattern could be determined for them. The `InvalidPositionEvent` type only contains the GPS position of the vehicle. The valid `ObservedPositionEvent` type also contains the *Speed* and *Direction* parameters. The GPS position of the `InvalidPositionEvent` events were always the same coordinates. In this thesis project the `InvalidPositionEvent` type was discarded, due to the missing parameters and static GPS position.

The 12 event types used in this thesis project were:

- *ObservedPositionEvent*: This event type contains the information highlighted in Figure 3.2. It is the most prevalent event type in the provided dataset. This event type is contextless, as it contains no information about which public transportation line the vehicle is currently serving, if any.
- *StartedEvent* and *StoppedEvent*: These two event types provide context to a sequence of observed position events. They denote when the vehicle has started or stopped moving, respectively. For example, they can be used to identify road intersections, bus stops, traffic or driver breaks.
- *EnteredEvent* and *ExitedEvent*: These two event types are used by the "Internal Analysis" component to identify bus stops. The *EnteredEvent* is produced by the system when the vehicle is within a certain predefined distance to a bus stop. The *ExitedEvent* is similarly produced when the vehicle leaves the predefined distance to the bus stop. These event types could, for example, be used in an algorithm which improves the bus stop detection.

Wasnt it something like this Östgötatrafiken did? @Mattias

- *PassedEvent*, *ArrivedEvent*, and *DepartedEvent*: These three event types are used to provide information regarding which bus stop a particular bus is at. The *PassedEvent* type denotes when a particular bus, serving a specific public transportation line, passed a bus line stop. It contains information about the predefined position of the stop, the public transportation line the particular bus is currently serving and the time of the passing. Similarly, the *ArrivedEvent* and *DepartedEvent* types denote when a particular bus arrived at or departed from a particular bus stop. These event types provide context to the observed positions. In this thesis project they are used to group a sequence of observed positions into a segment between two stops for a particular bus line.
- *ParameterChangedEvent*: The *ParameterChangedEvent* type is the most dynamic event type in the data set, e.g. it can be used to inform the system when the doors on a particular bus open or close or when a bus changes journeys. In this thesis project it is only used to identify bus lines and give context to observed positions.
- *JourneyStartedEvent* and *JourneyCompletedEvent*: The *JourneyStartedEvent* type is produced by the "Internal Analysis" component when a bus has reached the starting bus stop for a bus line. The *JourneyCompletedEvent* event type is produced when the bus has reached the final bus stop for a bus line.
- *JourneyAssignedEvent*: This event type is accompanied by a *ParameterChangedEvent* to denote when a bus changes its journey.

3.3 Pre-processing Events

The first step deemed necessary in order to use the data in the provided dataset was to transform it from strings to object with attributes. Figure 3.2 visualises which attributes an *ObservedPositionEvent* object contains. Similar structures were created for each of the mentioned event types.

After this parsing step the idea was to provide context to *ObservedPositionEvent*s. Only using this one contextless event type greatly reduces the potential problems one could solve with the provided dataset. Context was provided by constructing a finite-state machine.

Context-Providing Finite-State Machine

Finite-state machines are well-defined and can be modelled with simple algorithmic struc-

what is the formal definition?

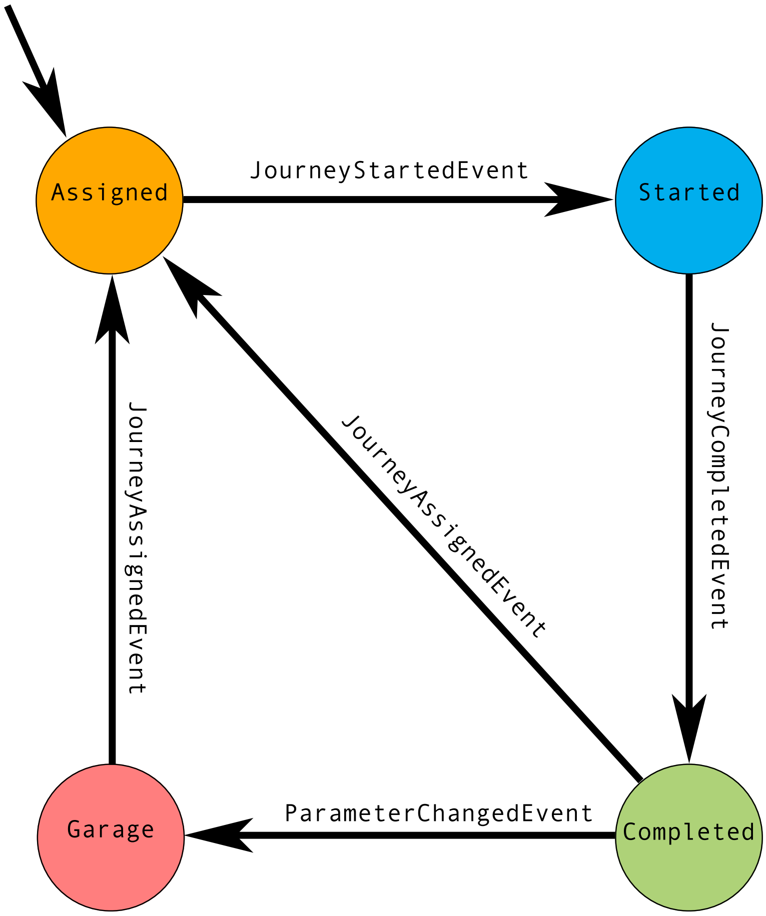

Figure 3.4: Finite-state machine providing context to *ObservedPositionEvent*s. The constructed finite-state machine is simplified to illustrate the best-case scenario. The "Assigned" state is the starting state. *ObservedPositionEvent*s are assigned to the current state the state machine is in.

tures . The context-providing finite-state machine constructed in this thesis project is shown in Figure 3.4. The shown state machine is illustrating the best-case scenario, when the actual order of events is equal to the logical ordering of events. This is not always the case when working with real world data. Occasionally the timing of events get mixed up, e.g. a bus in the "Started" state receives a *JourneyAssignedEvent* before it receives a *JourneyCompletedEvent*. This ordering breaks the logical ordering of events: a journey needs to be completed before a new one can assigned.

source?

The problem is solved by partly changing the ordering of events from timestamps to event IDs. This is a feasible approach when the data is batched into separate files, where one file is a full day of events. When processing data in real time the approach would have to be slightly altered. The *ObservedPositionEvent*s would have to be placed in a temporary buffer once an anomaly is detected in the event ordering, e.g. *JourneyAssignedEvent* is received before *JourneyCompletedEvent* and the system is in the "Started" state. Once the *JourneyCompletedEvent* type is finally received, the data in the buffer could be added to the "Started" state. The "Started" state would then correctly contain all *ObservedPositionEvent*s received from the starting bus stop to the final bus stop.

Unfortunately, it is not always the case that the *JourneyCompletedEvent* type is in an erroneous ordering. Occasionally the type is missing from the dataset due to human error.

Add visual examples of this. Available in the "Problem" doc.

Human Error: Early Stopping

Figure 3.5 and 3.6 illustrate two scenarios when the *JourneyCompletedEvent* type for a started journey would be missing. In Figure 3.5, the bus driver is supposed to visit the three markers in order to complete the journey for a particular line. In Figure 3.6, the dashed line highlights the route of the bus line, while the blue line is the actual route the bus drove. In both these real-world examples, the bus drivers ignore the final stops of the journeys.

Figure 3.5: Example of early stopping in a journey. The three markers are the final three stops of a particular bus line. Instead of following the pre-determined route of the bus line, the final three stops are skipped. This results in the journey never being deemed completed.

The "Internal Analysis" component never deems the journey as completed in these scenarios, which results in the *JourneyCompletedEvent* type never being produced. This scenario is easy to detect in historical data, but in real-time certain assumptions need to be made. For example, if the journey is compared with an average journey, the anomaly could be detected early. However, it would be uncertain if the anomaly is due to a journey being stopped early or if the bus driver took a wrong turn on the highway. A time constraint threshold would have to be introduced to the system in order to separate these two cases. In the case of a wrong turn, the bus would eventually converge to the average journey, while in the early-stopping scenario the journey would most likely never converge.

Figure 3.6: Another example of early stopping in a journey. The red dashed line is the planned route of the bus line. The blue line at "Nya Ledbergsvägen" is the actual route the bus drove. This results in the journey never being deemed completed, creating a erroneous ordering of contextual events.

4 Experiments

5 Discussion

6 Conclusion