Master Thesis

Utrecht University

TNO innovation for life

Pattern Recognition for Scenario Detection in Real-World Traffic Data

---

Petar Henricus Johannes Markotić

July 2018

Master Thesis

Submitted in partial fulfillment of the requirements for the degree Master of Science in the Subject of Artificial Intelligence

Pattern Recognition for Scenario Detection in Real-World Traffic Data

Petar Henricus Johannes Markotić

July 2018

1. Examiner prof. dr. Arno P.J.M. Siebes

2. Examiner dr. Ad J. Feelders
Department of Information and Computing Sciences
Utrecht University

Supervisor dr. Hala B.H. Elrofai
Department of Integrated Vehicle Safety
TNO

Petar Henricus Johannes Markotić
Pattern Recognition for Scenario Detection in Real-World Traffic Data
July 2018
Reviewers: prof. dr. Arno P.J.M. Siebes and dr. Ad J. Feelders
Supervisor: dr. Hala B.H. Elrofai

Utrecht University
Algorithmic Data Analysis Group
Faculty of Science
Department of Information and Computing Sciences
Princetonplein 5
3584 CC Utrecht

TNO, Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek
Department of Integrated Vehicle Safety
Automotive Campus 30
5708 JZ Helmond

Abstract

The continuous development and integration of Automated Driving Systems (ADS) leads to complex systems. Such systems need to be tested and validated thoroughly for all situations these systems may encounter on the road, to assure their safety and reliability. Test drives with ADS require millions of operational hours which is infeasible. TNO proposes to acquire micro-traffic data (data collected on the level of the individual vehicles) to generate real-world scenarios (situations) for testing and validating ADS. Such scenarios are resembled by typical patterns in the data. To achieve extraction and classification of scenarios from micro-traffic traffic data, TNO already developed knowledge-driven, rule-based techniques. However, a drawback of these techniques is the loss of generalization. This work proposes an unsupervised data mining approach to mine events from real-world traffic data to overcome this limitation. After decimation and discretization of the data, we combine Frequent Itemset Mining and Frequent Sequence Mining over multiple sensor outputs in the form of data streams for recognizing patterns that represent real-world traffic scenarios. The method shows that different configurations can result in different generalizations to satisfy the need of the expert, as different levels of abstraction can be desired by the user. Finally by conducting experiments we conclude which configuration provides the most desirable result in finding events for longitudinal movement.

Za Anđa

Acknowledgement

In this section I would like to show my gratitude to the people that have contributed in helping me finalizing my thesis.

First of all, I want to thank Arno Siebes for not only guiding and supervising me during this project. But also for all that I have learned from him during this journey, as well as during the courses Big Data and Pattern Set Mining, which were a major contribution in me choosing to pursue the path of a data scientist.

Second, I want to thank Hala Elrofai for giving me the opportunity of writing my thesis at TNO and guiding me during my project. For all the pleasant trips to the station and support even when she was on holiday. I also show my gratitude to the other colleagues at TNO who have aided me, in particular Jan-Pieter Paardekooper and Erwin de Gelder for their extensive feedback.

And last but not least, I want to thank all my friends and family, who have always been there for me showing unconditional support and heartwarming encouragement.

Contents

1 Introduction

> *The idea is to go from numbers to information to understanding.*
>
> **— Hans Rosling**

Over the past few years the development and integration of Advanced Driver Assistance Systems (ADAS) has seen an increase as they provide safety and comfort to the driver [18]. Yet, before these systems can be taken into production, their safety and reliability need to be guaranteed. Thus, an important aspect in developing ADAS is the verification and validation of these systems. For legal and public acceptance, it is important that there is a clear definition of system performance and quantitative measures for the level of performance of the system [9].

Early adoptions of ADAS as Lane Keep Assistance (LKA) or Adaptive Cruise Control (ACC) only utilize single dedicated sensors such as RADAR or video based detection. However, the development of ADAS is seeing an increase in autonomy resulting in more complex Automated Driving Systems (ADS) as start-and-go assist and predictive ACC, which often require a combination of sensors as input [19]. Due to the increase in complexity of ADS, testing these systems becomes more challenging. Estimates show one has to conduct at least one billion operational hours of test drives to validate the failure rates [5]. Such a large number of hours for testing ADS is infeasible, therefore there is an essential need for constructing and collecting relevant traffic events and situations (scenarios) for testing and validation of ADS functionalities. The collection of these scenarios should in principal represent and cover the entire range of real-world traffic situations that might be encountered by the ADS under test.

At TNO, research is being conducted in developing scenario-based assessment methods. At the department of Integrated Vehicle Safety (IVS) the 'StreetWise' project has been set up. The aim of StreetWise is to construct a real-world scenario database for testing and validating ADS. Scenario-based safety validation of automated driving is broadly supported by the automotive community. This is reflected in a draft standard of NHTSA and the ISO 26262 working group on SOTIF. Related projects in Germany (Pegasus [17]) and EU (ENABLE-S3 [10]) strongly support this approach. Using

scenario based assessment methods assures quantitative measures for the level of performance of the system [5]. However, identifying and extracting these scenarios from real-world traffic data introduces many challenges.

1.1 Problem Definition

Scenarios are defined as sequences of events (or activities). A proper definition of scenarios, events and activities will be provided in Chapter 2. The core goal of this project is to mine these so-called events (or activities), as they serve as building blocks for generating scenarios. At TNO several methods for event and activity detection have been developed which all are rule-based. Development of these algorithms rely on experts defining these rules. A drawback from these rule-based approaches is the possibility of not finding the optimal rules. Therefore, there is a need in the development of a data-driven method, which translates to the following research question:

> *What is a suitable data-driven approach to mine events and activities from real-world traffic data?*

TNO has established research to detect events and activities from real-world traffic data [9]. The current developed algorithms are based on deterministic models and rule-based approaches. Even though the number of events and activities are finite, it is still very challenging to detect and mine events/activities from terabytes of data collected from many sensors. The rule-based method is a top-down approach that has many limitations. Experts need to define and characterize events and scenarios a priori. This might result in subjective non-optimal definitions of the events and activities, i.e. being too loose or too tight. Also there is a risk of missing events/activities that are not defined a priori or meeting the defined characterizations. A suggested possible solution to this problem by TNO is by making use of data mining/machine learning techniques to mine events/activities from the data. These techniques adopt a bottom-up approach which has the benefit that there is no need for defining an event or activity a priori.

The remaining part of the thesis proceeds as follows. The next Chapter is explaining StreetWise, the scenario based approach developed at TNO and the corresponding definitions. The next chapter is a literature study looking into the state-of-the art progress from TNO. Chapter 3 discusses the developed data mining methods for detecting events and activities. Chapter 4 describes characteristics of the data and the experimental-setup. Chapter 5 provides the results and the discussion of the

results. Chapter 6 ends with a conclusion and some recommendations to TNO and improvements and limitations of the developed methods.

Literature Study

2

In this section we discuss the scenario-based safety validation method developed at TNO to improve our understanding of this concept. At first we state the formal definitions of a scenario, event and activity. Next we discuss the limitations of the current developed techniques at TNO to detect events and activities. Finally we present and argue the proposed unsupervised data-driven technique for event/activity detection.

2.1 Background on a Scenario based approach: Streetwise

TNO has been working on defining scenarios and events, developing an ontology utilizing traffic scenarios for safety validation of ADS. Before we introduce the notions of scenarios and events, we discuss some terminology. We present the notions of ego vehicle, Most important vehicle(s), static environment and dynamic environment as defined in [7].

Ego vehicle The ego vehicle is the perspective from which the world is seen. It is common to refer to the vehicle perceiving the world through its sensors as the ego vehicle.

Most important vehicle(s) MIV The most important vehicle is somewhat of a controversial notion. Currently the MIV is seen as the vehicle closest to the ego vehicle or the vehicle in front of the ego vehicle. However, in some cases, the MIV can be associated with multiple vehicles. In the latter case, it is conventional to consider the vehicles in front and behind the ego vehicle as well as the vehicles next to the ego vehicle to be the MIV.

Static environment The static environment refers to all the parts of a scenario that do not change during a scenario. These are for instance geo-spatial objects, but also traffic lights and weather conditions. One could argue that the latter are not part of the static environment as they can change, but since the change

is not significant in the timespan of a scenario, it is considered to be part of the static environment.

Dynamic environment The dynamic environment refers to all the parts of a scenario that do change during the timespan of a scenario. Usually the parts that belong to the dynamic environment are moving actors (road users) other than the ego vehicle.

2.1.1 Scenarios

As mentioned earlier, scenarios are the atoms of a car's trajectory. In essence, a car's trajectory can be interpreted as a sequence of scenarios. A scenario in the automotive setting is formally defined in [7] as follows:

Definition 1 (Scenario) *A scenario is a quantitative description of the ego vehicle, its activities and/or goals, its static environment, and its dynamic environment. From the perspective of the ego vehicle, a scenario contains all relevant events.*

In Figure 2.1, a schematic overview of a scenario is given. It clearly shows the components of a scenario namely the ego vehicle, the dynamic environment and the static environment. Let us look at an example of a scenario for clarification of the ontology. An example of a scenario could be the ego vehicle (e.g. a sedan) driving on a 2 lane highway (road type) with sunny weather (weather type) and another vehicle (e.g. a car) behind the ego vehicle. In this scenario the other vehicle will overtake the ego vehicle by changing lane, accelerating and finally changing back to the original lane. During the scenario the ego vehicle will maintain its speed (cruising).

By breaking down this scenario we can observe some events/activities taking place. The ego vehicle performs 2 activities: maintaining its lane and speed (lane following and cruising). The other vehicle performs 3 activities: changing lane, accelerating and then changing back to the original lane. This represents a combination of events/activities that describe an instance of an 'overtaking' scenario. It is evident that events and activities are vital in constructing scenarios as specific combinations of events/activities define a scenario. Next we will see the possible events that can be used to construct scenarios.

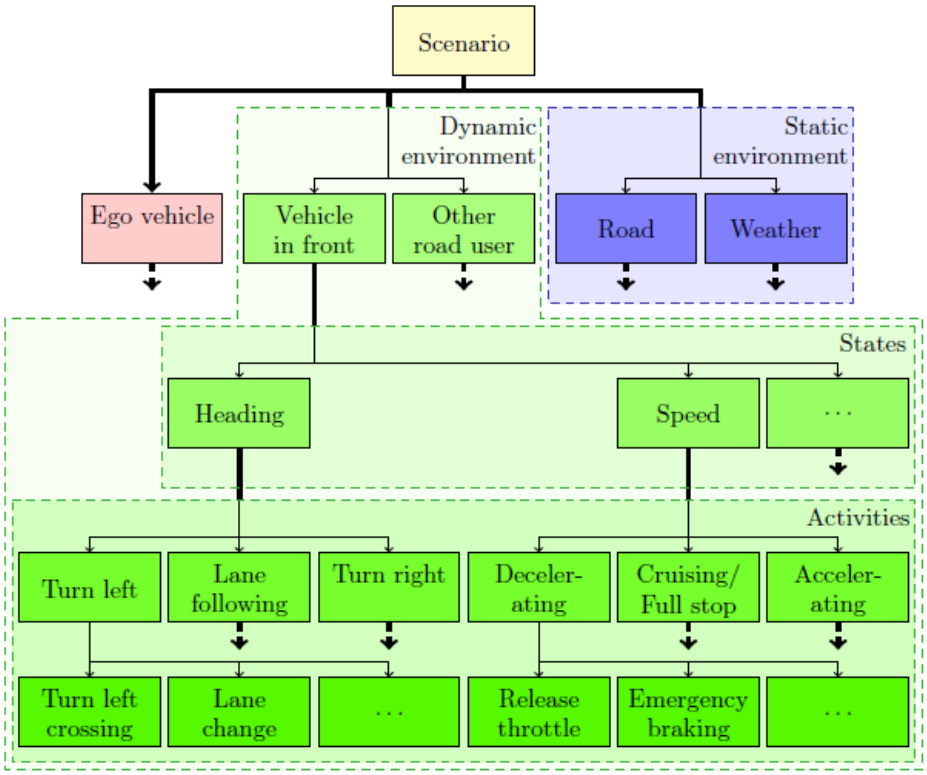

Fig. 2.1: Schematic overview of a scenario. Source: [7].

2.1.2 Events and Activities

Events combined in a sequence form a scenario as just defined. These events can be seen as the building blocks of a scenario. In the literature the notion of an event has various definitions for different disciplines. In automotive, we speak of an event as a time instant in which a transition of a state takes place. The formal definition of an event according to [8] is as follows:

Definition 2 (Event) *An event marks the time instant at which a transition of state occurs, such that before and after an event, the state corresponds to two different activities.*

Thus, since before and after an event, the state corresponds to two different activities, we can conclude that events mark the start and end of activities. In [8] an activity is defined as:

Definition 3 (Activity) *An activity is a time evolution of state variables such as speed and heading to describe for instance a lane change, or a braking-to-standstill. The end*

of an activity marks the start of the next activity. The moment in time that marks the beginning or ending of an activity is called event.

As a consequence we can see that events can be derived from activities and vice versa. They are proportionate to each other, meaning that detecting one of both can suffice in finding the other. As a result, it can be the case that we use these terms interchangeably, as mining events is equivalent to mining activities.

To illustrate the definition of event and activity we consider an example presented in Figure 2.2. Take note, acceleration is consciously not defined as gassing, since gravitational forces can cause the vehicle to accelerate, e.g. in the case of a downward slope, where no gassing is involved. The sate referred to in this case is the longitudinal movement (or 'Speed' as referred to in Figure 2.1), which has the possibility of being in either the activity: decelerating $(a < 0)$, cruising $(a = 0)$ or accelerating $(a > 0)$. The transition from either of these activities to another is what we call an event. In our example we can identify 2 events. The first event is the acceleration going from positive to negative, indicating the start of a braking activity. The second event is the acceleration going from negative to zero, indicating the start of a cruising activity.

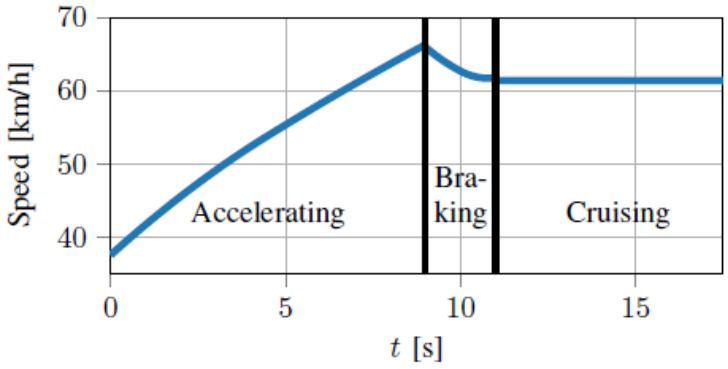

Fig. 2.2: Activities of the ego vehicle considering its speed. The black vertical lines indicate the events. the three different activities that can be qualitatively described as 'accelerating', 'braking' and 'cruising', respectively. Source: [8].

Not only longitudinal acceleration (speed) can be taken into account, but also lateral movement (heading). As the vehicle changes heading, there is the case of an event as well. Now combining sequences of events for different states in parallel, will ultimately lead to a scenario. Events as change in longitudinal and lateral movement are difficult to detect. For instance, lateral movement can be seen as a lane change, but also as making a turn. These events can only be found through analyzing the data and finding patterns that define a specific event.

2.2 State of the art

TNO has already developed an algorithm to detect events from data. The algorithm is written in python and can currently detect events regarding the speed of the ego vehicle (acceleration, deceleration and cruising) and events regarding the heading of other vehicles and the ego vehicle (cut-in, cut-out and lane change). The algorithm works as follows: the data is read into memory and preprocessed (applying smoothing filters). Next it checks which events/activities the user has declared he/she wants to detect. For all of the events the user desires to detect, the algorithm checks whether certain conditions hold for a certain time period. If these criteria hold, the start and end (events) of the activities will be registered. The events are stored with their corresponding time instances and saved. Finally it plots a graph of the data, with an overlay of the events at their corresponding point in time.

Algorithm 1: TNO: Acceleration Event Detection

Input: Vector acceleration signal D of size $N \times 1$; Convolution mask $conv\_mask$; Acceleration threshold θ; Convolution ratio r
Output: Acceleration event vector v

foreach $d \in D$ **do**
  if $d \geq \theta$ **then**
    $d \leftarrow 1$
  else
    $d \leftarrow 0$
  end
end
Insert $0.5 \times length(conv\_mask)$ 0's to the beginning and end of D for the convolution
for $(i = 0;\ i < length(D);\ i++)$ **do**
  $Y[i] = 0$
  for $(j = 0;\ j < length(conv\_mask);\ j++)$ **do**
    $Y[i] += D[i - j] \times conv\_mask[j]$
  end
end
foreach $y \in Y$ **do**
  if $y \geq length(conv\_mask) \times r$ **then**
    $v \leftarrow 1$
  else
    $v \leftarrow 0$
  end
end
return v

As an example we describe the algorithm for detecting acceleration events developed by TNO in Algorithm 1. The algorithms takes 4 inputs: the data vector D containing the acceleration signal, a convolution mask $conv\_mask$ being a vector of 1's of length corresponding to the preferred time (in this case 0.5 seconds which is 25 points), an Acceleration threshold θ which is set to 0.15 and a Convolution ratio r which is set to 0.75. First the algorithm maps all the values in D to 1 if the value is greater than θ or to 0 otherwise (1-7). Next it slides a convolution mask over the data to determine the number of neighbouring points having a value greater than θ (8-14). And finally it checks for every cell in Y if the value is greater than the length of the convolution mask multiplied by the ratio r(15-21), meaning that at least $r \times 100\%$ of the sequential points should be accelerating.

2.2.1 Conclusion

As becomes clear, the algorithm by TNO adopts a top-down approach. Currently, the signals or features derived from signals used to detect events and activities are chosen by experts. However, it could be the case that other signals can improve the performance of the algorithm in detecting events and activities. Or, it could even be such that certain features (or signals) have no significant effect on the quality of the algorithm. Opposed to knowledge-driven features, data-driven features can be exploited in optimizing the required features in detecting events and activities. This takes care of on the one hand not missing any important features and on the other hand not using superfluous features. Surely it could be the case that the expert is actually aware of the optimal features needed in detecting an event or activity. In such cases, applying machine learning techniques for data-driven feature generation could be considered superfluous. Although, the expert knowing the optimal characterization of an event or activity is highly unlikely. First, the expert knowing the optimal signals and corresponding threshold values is not apparent, due to the inconsiderable number of possibilities. Second, without applying machine learning techniques there is no way of telling if the knowledge-driven features are sound. Thus, we can conclude that using machine learning/data mining techniques can be beneficial in defining events and activities in terms of microscopic traffic-data. In the following chapter we will argue how events and activities are represented in data and how to extrapolate them from the data.

3 Methodology

To tackle a data mining or machine learning problem it is always wise to classify what type of learning the problem it is. It becomes immediately clear that this problem is an unsupervised learning problem. There is no ground truth that confirms whether the mined events are correct. The only reference one could consider is the video data collected during the test drives. A possibility would be to annotate/label video data of the test drives manually, so that one could use these as labels in a supervised learning paradigm. However, this approach of annotating the video data and using these annotations as labels has two drawbacks. At first, annotating all this data is very time consuming. And since the amount of video data collected, or even needed, ranges from hours to hundreds of hours, it is considered infeasible. Next to this, annotating the data manually will result in very subjective labels. Starting and ending points of the activities annotated in the data could differ depending on the point of view of the expert. A possible solution to this problem could be letting multiple experts annotate the same data and somehow take the average of their findings. However, this solution only decreases the bias instead of getting rid of it. And also, the more experts needed to annotate the data, the more time it will take as already stated in the first counter-argument. And it would also mean that the same experts have to annotate the same data to keep the bias consistent, which can result in old data/labels being unusable.

Thus due to the fact that labeling the data is impractical, we will limit ourselves to unsupervised methods to address the problem. Unsupervised learning has many approaches in finding patterns in data. One approach is clustering the data. With clustering, features and characteristics of the data are used to define datapoints that should be addressed as the same category based on similar characteristics. An example of clustering is clustering coins based on their weight and diameter. After clustering, clusters will emerge that represent the same coin. However, up front it would not be known what coin will be represented by which cluster. Afterwards it is possible to label the clusters with their coin accordingly. The same applies to finding events in the sensor data. There is no label in terms of time when an activity starts of ends/an event occurs. Thus the only way to find these events is by looking for patterns in the data.

The goal of this thesis is to develop a method to detect the occurrence of events/activities without knowing when the events occur a priori or what features would influence detection of an event. In the state-of-the-art algorithm by TNO a deterministic model was created to detect events. However, as argued in Chapter 2, the drawback of this approach was that the rule-based event/activity detection algorithm is using predefined knowledge-driven features to detect events/activities. These features where chosen based on expert knowledge and validated on the data gathered by TNO. The aim of this research/work is to improve on the current algorithm and develop an unsupervised machine learning/data mining technique(s) to determine the optimal features for detecting events/activities.

From the just stated arguments, we can justify the use of unsupervised data mining techniques since labeling is impractical, and even if labeling the data would not be time consuming, the labels would be subjective with respect to the expert, which limits the performance. This chapter presents the developed unsupervised data-driven method to detect events and activities from real-world (microscopic) traffic data. First, we explain how events are represented as patterns in the collected data. Next, we examine and argue for the steps needed to detect these patterns. And finally, we present and discuss the developed methods for detecting/mining these patterns for the data.

3.1 How are events represented in the data?

Intuitively, for an event to occur a change in sensor values is a necessary condition. The start or end of an activity would be described by similar changes in combinations of sensor values. Thus, the idea is to search for combinations of change in different sensor values. In the most simple case an event is defined by change in only one sensor, but for the events in heading and speed this is not the case, as these activities can not be described by one sensor. An objection could be that the acceleration signal would be sufficient in detecting events for speed, but due to noise in the signal this is not possible. Finding the same pattern in change in multiple sensor values on multiple occasions would suggest the same activity starting or ending, thus the same event occurring at those time instances. It is not known what activity would actually start or end at that specific time instance. Therefore, there is a need to label these patterns afterwards, to determine what event is actually represented by this pattern. This differs greatly from manually annotating all the events/activities in video-streams to later determine what is happening in the data at those time instances. This clearly shows the distinction between the implemented top-down approach

from TNO and our proposed bottom-up approach. Our bottom-up approach does not suffer from subjective noise introduced by biased labeling.

It could be the case that multiple different patterns represent the same event. In a rule-based algorithm this would correspond to a disjunction of multiple rules. There are a few causes for this to happen. Firstly, it is possible that some of the patterns represent a more explicit form of an event. The scenario decomposition in Figure 2.1 shows such an example, where deceleration can be broken down into emergency braking or releasing the throttle. In this case both of the patterns are an event of changing from cruising to decelerating, however they are distinct at a lower level. Secondly, it could be the case that features taken into account are redundant, meaning that they do not contribute to the occurrence of the event. Thus finding an event where one signals shows the same pattern, while another signals shows different patterns, meaning that the patterns is not dependent on the signal. Lastly, it could be due to stochastic noise in the data.

However, only looking at sensor values at a single time instance would not be sufficient. Especially if the subsequent measured values are occurring after each other on a low time interval. It is not possible to assume that events occur in the order of milliseconds. Looking at human driving behaviour, it is not rational to assume that one can determine if the driver changes from acceleration to cruising in a time interval of a few milliseconds. As it is never the case that state transitions are entirely smooth. Thus considering this naturalistic behaviour, it is required to look for time intervals instead of time instances in the form of sequences. These sequences represent small time intervals differing from tens of milliseconds to a few seconds. The optimal length for these sequences is not known upfront, but should be determined through experiments. Essentially the sequence length should correspond to the time needed by the driver to perform a state change.

3.2 Algorithm

This section will give an overview of the developed algorithm to detect events/activities in traffic data. The data collected by TNO is multivariate data, as it are data streams for multiple sensors over time. Since TNO has the desire of knowing when an event occurs, the location of the patterns in the data needs to be determined and saved during the mining process.

The proposed solution, to mine combinations and sequences of sensor values over multiple sensors, is an instance of frequent itemset/sequence mining. To mine

frequent sequences over multivariate data we adopt a two step approach. First, frequent itemsets are mined over multiple streams. Second, the frequent itemsets are used as input for mining sequences over multiple streams. Next we will explain Frequent Itemset Mining and Frequent Sequence Mining (techniques) in general. Thereafter we will discuss how the combination of both techniques is applied in our context with the needed adaptations.

3.2.1 Frequent Itemset Mining

This section will convey a brief description of Frequent Itemset Mining (FIM). Frequent Itemset Mining was originally introduced in [1] for finding association rules in market basket analysis. The idea behind FIM is to mine all frequent sets of items in a database of customer transactions. Given a certain threshold, find all sets of items that occur more often than this predefined threshold. Formally FIM is defined as follows. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ be a set of literals, called items. Let D be a database, being a set of transactions, where each transaction is a pair (tid, t), with $t \subseteq \mathcal{I}$ and $tid \in \mathbb{N}$ being a unique transaction id. We then say that an itemset $I \subseteq \mathcal{I}$ *occurs* in a transaction t if $I \subseteq t$. The *support* of an itemset is the number of transactions the itemset occurs in, given a transaction database D:

$$supp_D(I) = |\{(tid, t) \in D \mid I \subseteq t\}| \tag{3.1}$$

The relative support, called *frequency*, is the number of transactions the itemset occurs in divided by the total number of transactions in D, defined as:

$$freq_D(I) = \frac{|\{(tid, t) \in D \mid I \subseteq t\}|}{|D|} \tag{3.2}$$

An itemset I is called *frequent* if its support (or frequency) is greater or equal than a predefined threshold min_supp. The goal of FIM is then defined as: *Given a database D over items $\mathcal{I}$, find all itemsets I that are frequent in D given min_supp.*

Apriori principle

Finding all frequent itemsets in a database can be difficult. The naive way, considering all subsets of $\mathcal{I}$ will require us to check $|\mathcal{P}(\mathcal{I})| - 1$ itemsets (the empty set is always frequent). The number of sets to be considered grows exponentially

since $|\mathcal{P}(\mathcal{I})| = 2^{|\mathcal{I}|}$. To overcome this problem, the authors of [2] proposed the Apriori algorithm which capitalizes on the Apriori (or anti-monotonicity) property. According to this property, if we have two itemsets X and Y, if $X \subseteq Y$ then $supp(Y) \leq supp(X)$. From this we can conclude two things utilized during the mining process. If an itemset X is infrequent, then none of its supersets can be frequent. And also, for an itemset X to be frequent, all its subsets must be frequent as well. To exploit these properties, the Apriori algorithm does a level-wise search. At each level k, only frequent itemsets of length k are combined to generate candidate itemsets of length $k+1$. Then for all candidate itemsets it must hold that all of its subsets are frequent or else they are still pruned. By pruning, the search-space gets reduced immensely, which is a huge improvement on the naive approach. The worst case complexity of the Apriori algorithm is still $\mathcal{O}(2^{|\mathcal{I}|})$, as all itemsets could be frequent, but this is unlikely in practice. Many other algorithms have been introduced since, with two very popular algorithms being Eclat [21] and FP-Growth [12] utilizing different properties to speed up the mining process. In Section 3.2.3 we will argue which algorithm suits best and why.

3.2.2 Frequent Sequence Mining

Frequent Sequence Mining (FSM) is an adaptation of Frequent Itemset Mining introducing a temporal (or some other e.g. lexicographical) order over the items in transactions in D. Due to this order on the items, the transactions in D become sequences over $\mathcal{I}$ instead of sets. The analogy to market basket analysis is that we are now also interested in the order in which the items are bought. FSM was originally proposed, and formally defined in [3] as follows. Let us have a set of literals $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ and each transaction is now a sequence in the form of $\mathbf{s} = \langle s_1 s_2 \ldots s_n \rangle$. The goal is to find all frequent subsequences over all sequences in D. A sequence $\langle a_1 a_2 \ldots a_n \rangle$ *occurs* in, or is a subsequence of, another sequence $\langle b_1 b_2 \ldots b_n \rangle$ if there exist integers $i_1 < i_2 < \ldots < i_n$ such that $a_1 \subseteq b_{i1},\ a_2 \subseteq b_{i2},\ \ldots,\ a_n \subseteq b_{in}$. The goal of frequent sequence mining is to find all frequent sequences for a given database D. The support of a sequence $\mathbf{r}$ is the number of sequences it occurs in, given a database D.

$$supp_D(\mathbf{r}) = |\{\mathbf{s} \in D \mid \mathbf{r} \subseteq \mathbf{s}\}| \tag{3.3}$$

Also the relative support, referred to as the frequency, is defined as:

$$freq_D(\mathbf{r}) = \frac{|\{\mathbf{s} \in D \mid \mathbf{r} \subseteq \mathbf{s}\}|}{|D|} \tag{3.4}$$

Similar to itemsets, a sequence is called *frequent* if its support (or frequency) is greater or equal than a predefined threshold min_supp. The goal of FSM is then defined as: *Given a database D of sequences $\{\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_n}\}$, find all sequences that are frequent in D given min_supp.*

3.2.3 Applying FIM and FSM to data streams

In our case we are dealing with real-valued data streams. However, mining itemsets over such data is impossible as the groundset I is too big depending on the accuracy of the sensors. After discretization the data steams are streams of categorical data, which can be used as input for frequent itemset and sequence mining. In Section 3.4 we will elaborate on discretization of the data.

Each column in the set of streams represents a time instance. The data is sampled synchronously, thus data stream measures occur at the same time instances. Translating our data to itemset mining, the data streams can be seen as if each column represents a transaction in a transaction database D. Thus finding itemsets over different streams is actually an instance of Itemset Mining, but with multivariate data as streams can adopt multiple values. Next we will formalize these notions.

Preliminaries and Notation

The input data is a set of m streams denoted as $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$. Each stream S_i is a sequence of categorical values taken from the set $\mathcal{V}_i$. We assume that every stream is of the same length and that they were recorded synchronously. To transform our setting to frequent itemset mining, we consider each column over streams as a transaction (tid, t) with tid being the time-point and t a set of tuples (s, v) where s denotes the sensor and v the value of the sensor.

There are various algorithms to mine frequent itemsets from a transactional database. We have already examined the Apriori algorithm and mentioned the Eclat and FP-Growth algorithm. Since one of TNO's requirements is to know when the events actually occur, we need to determine and save the time instance at which at itemset occurs. The Eclat algorithm makes use of a vertical database instead of a horizontal database which will serve as a solution for the requirement of saving the time instances along with the frequent itemsets.

The Eclat algorithm makes use of a vertical database instead of a horizontal database. Till now we have seen that a database D is a set of transactions, with each transaction having a transaction id. However, with the vertical database layout this will be reversed. We now consider a database of itemsets followed by the set of transactions ids they occur in, also known as their cover (or support set).

To mine all the frequent itemsets, the first step is to generate all itemsets of length 1 followed by their cover. Then we check which itemsets are frequent by checking if their support is greater than min_supp, thus $|cover| \geq min\_supp$. All the itemsets that are not frequent are pruned and all frequent itemsets are used to generate the itemsets for length 2. To generate the itemsets for length $l + 1$ we join the frequent itemsets of length l with itself, with the following two conditions. 1) for each generated itemset of length $l + 1$, all its subsets must be frequent as well. This is due to the monotonicity principle we have introduced earlier, saying that if an itemsets is frequent, all its subsets must be frequent as well. 2) we are dealing with itemsets of tuples in the form of (s, v). Thus when we join all the itemsets, we only want to join the ones that result to valid itemsets. To ensure this, and also that we do not generate itemsets of length $l + 1$ multiple times we order the itemsets ascending on their sensor value. Then when we combine 2 itemsets for length l, we only combine the ones that share the first $l - 1$ items **and** differ in s at index l. This prevents generation of itemsets of length $l + 1$ where an itemset would contain two tuples for one sensor, which is impossible and would yield an empty cover in the end anyway, but the check saves the effort of intersecting their covers.

Then for each generated candidate, the cover will be equaled to the union of the covers of the itemsets that generated it. Then again we check which itemsets are frequent by looking at the cardinality of the covers of the itemsets and we prune the infrequent itemsets. This keeps on going until we have reached itemsets that have the same length as the number of sensors, or if no more itemsets are frequent. In

the end we return all frequent itemsets of sensor/value (s, v) pairs, including their occurrences.

Mining Frequent Sequences in Multivariate Stream Data

As we have argued before, we do not only want to look at time instances, as these are sensitive to various forms of noise. Thus, we want to find frequent sequences of itemsets in the data. However, since we are dealing with streams of data, simply interpreting a data stream as a sequence in the database D is not the desired approach as the subsequences to be found can become arbitrarily large. There are various approaches in the literature that make use of techniques as a sliding window [6], [15] and [20] or pattern growth without candidate generation [16]. However, these approaches lack either in the fact that sequences could be split up due to the placing of the sliding window, the location of the sequences not being preserved, which is a requirement in our context. And in any case, we can use the mined frequent itemsets in mining sequences to skip the first level which is beneficial. Next we will see one more very important limitation that we can exploit in the mining process.

Due to our application on event mining, we can limit ourselves to a specific case of frequent sequence mining. We have just defined a sequence a being a subsequence of b if each position in a is mapped to a position in b with the same label, and the order of the labels is preserved. However, this gives the possibility of gaps in the sequence, meaning that the elements of a do not occur subsequently in b. Allowing for gaps in sequences would result in patterns of which it is not clear when the pattern actually starts or ends.

The definition of a subsequence without gaps is given by: a sequence $\langle a_1 a_2 \dots a_n \rangle$ *occurs* in, or is a subsequence of, another sequence $\langle b_1 b_2 \dots b_n \rangle$, if there exist integers $i_1 < i_2 < \ldots < i_n$ such that $\mathbf{i_n} - \mathbf{i_1} < \mathbf{n}$ and $a_1 \subseteq b_{i1},\ a_2 \subseteq b_{i2},\ \ldots,\ a_n \subseteq b_{in}$.

Our goal is to find all frequent subsequences in the data streams without gaps with a maximal length of l. To find these sequences, we can exploit the approach used to mine the frequent itemsets over the data. We consider the frequent itemsets with their occurrences as a vertical database, being our frequent sequences for $l = 1$. Now to find all frequent sequences of $l = 2$, we apply the same method of joining as we did in mining the frequent itemsets. We only join frequent sequences for $l = 1$, however all sequences are frequent since the frequent itemsets are frequent sequences for $l = 1$. However, instead of joining the itemsets, we now append them to a list to preserve the temporal order. Then by exploiting the fact that we do not

allow for gaps, we can join the cover sets of the itemsets in $\langle I_1, I_2 \rangle$ by decrementing each tid in the cover set of I_2 by 1, and then joining the sets. By this we assume that a sequence of length l its starting points is I_1 in the sequence. From here we prune the infrequent sequences based on the cardinality of their cover. Then to extend, going from any l to $l+1$, we join all frequent sequences $a = \langle a_1 a_2 \dots a_l \rangle$ and $b = \langle b_1 b_2 \dots b_l \rangle$, where $a[1:] = b[:-1]$, thus all combinations of sequences that are equal; where for one sequence the first itemset is not considered and for the other sequence the last itemset is not considered. Then before joining their cover sets, we first check if all their subsequences are frequent. If this is the case, we join their cover sets the same way as we did for the $l = 1$ case. We keep on going until there are no more frequent sequences, or if we arrive at a predefined maximal sequence length. We then return all the frequent sequences, accompanied by their occurrences.

3.3 Downsampling

An interesting trade-off emerges from mining patterns in highly frequent sampled data. The sampling frequency states the number of datapoints needed for one second of data. Thus the higher the frequency, the more datapoints needed to cover one second in time. Since activities could continue for a few seconds, it is needed to look at patterns that cover at least such an amount of time. However, this results in mining longer patterns for higher frequencies. Consequentially, the patterns can become so big making it impractical or even infeasible to mine these patterns. Especially if the number of signals increase, computational effort will grow exponentially.

Decreasing the length of these patterns could be achieved through decreasing the sampling frequency of the sensors used to measure the data. If this is not possible, another option would be downsampling the data at hand. Downsampling the data can be done in multiple ways. The most straightforward approach would be taking a sample out of the data every k step where $k = original frequency \ / \ desired frequency$. For example, if the original sampling frequency equals $50Hz$ and the desired frequency is $5Hz$, one has to take a sample every $50/5 = 10$ steps. However, there is a major drawback of this approach namely neglecting a lot of data points and therefore neglecting a lot of information.

To solve this, an alternative method is to take the average over k (in our example 10) datapoints. Averaging over the data will prevent disregarding any data, while

still achieving decimation. This form of downsampling can be seen as applying a moving average filter followed by classic downsampling. To determine what sampling frequency is optimal, conducting experiments is needed.

3.4 Discretization

Discretization is needed because we can not mine itemsets over real-valued data. The sensor outputs are continuous signals, meaning they can adopt infinitively many values (depending on the resolution and quality of the sensors and measurements). Thus, the ground set $\mathcal{I}$ can become infinite when we are dealing with continuous signals, making mining infeasible. Clearly, we have to limit $\mathcal{I}$, to still be able to mine frequent patterns. This can be done by discretizing the values to so called bins, which segments the data in ranges. Another reason to consider discretizing the data is because, to detect certain events the accuracy of the sensors can be too high. Abstracting the data such that the events we want to find can still be represented reduced the number of patterns we will find, while the needed information is still available.

In [13] the authors present a survey of different discretization techniques. Here a distinction is made depending on whether or not methods make use of class information. Since there is no class information available in our case, we are limited to methods not exploiting class labels. The survey suggests two straightforward techniques, namely equal-width and equal-frequency binning. Both these binning methods require the user to predefine the desired number of bins. Next we will compare the two binning methods to argue which methods works best for our problem.

3.4.1 Equal-width binning

The equal-width discretization algorithm divides the data in a user defined number of bins. Equal-width refers to the bins having equal width in terms of the ranges of the bins being the same. The bins are defined by taking the maximum and minimum of the range of values, subtracting those, and dividing it by the desired number of bins:

$$W = \frac{V_{max} - V_{min}}{k} \tag{3.5}$$

Where W is the width (or range) of each bin and k is the desired number of bins. An advantage of this method is that it ensures similar values to be in the same bin. The disadvantage of this method is that it is also very sensitive to outliers. It could be the case that outliers introduce a bin for just itself.

3.4.2 Equal-frequency binning

Opposed to equal-width binning there is also equal-frequency binning. With equal-frequency binning the intent is to uniformly distribute the points over all bins. This is achieved by dividing the total number of datapoints by the desired number of bins. After sorting the data, the data is divided such that each bin contains (roughly) the same number of data points. In the case where the data can not be divided over all bins evenly, some bins have one data point extra arbitrarily. The drawback of this method is that it could be the case that equal values end up in different bins. In this case the discretization would introduce an unwanted implication in the data, namely that same valued data would have different meanings.

3.4.3 Discretization of specific signals

Signals provided by TNO can contain specific attributes that can be utilized to improve the discretization of the continuous data. In this subsection we will discuss, as an example, few signals that could be easily discretized by the nature of the signal.

Acceleration signal

The nature of the acceleration signal shows that not all distributions of the signal are equally beneficial/informative. Positive values represent an increase in speed, negative values represent a decrease in speed and else when the value equals to zero, speed is constant. For the acceleration, noise is unavoidable due to sensor drift and the driver not being able to maintain constant speed. Here we speak about constant speed in an abstract sense, usually called cruising, which occurs when the acceleration is approximately equal to zero. By using equal-width or equal-frequency binning on the acceleration signal, it could be the case that the resulting bins do not represent the characteristics of the acceleration signal. An example would be a case where all values up and until some value just above zero would fall into the

same bin. In this case cruising and decelerating will be represented by the same bin, which makes them inseparable and thus not able to mine.

A more suitable discretization of the acceleration signal is achieved by utilizing the fact that when the value equals (approximately) zero, the speed is constant (or the driver is cruising). This would result in three bins, one for accelerating, one for cruising and one for decelerating. The margins for the cruising bin need to be defined as noise of the sensor. Through experiments we will examine what the best ranges are for this bin. It is even possible to extend this discretization to also differentiate between the intensity of acceleration or deceleration. It could be beneficial to be able to distinguish between emergency braking or releasing the throttle before a traffic light, as we have seen in the scenario decomposition. The more bins one would define, the more specific, low level events one could find in the data.

Speed signal

The speed signal works a little different. The signal can not adopt negative values, thus we need not consider those cases. Also, it is almost always the case, that speed limits are set to tens. Yet again we have to deal with sensor drift and human behaviour resulting in the velocity fluctuating in this case around the tens, as drivers never exactly follow the speed limit. One is thus only interested if people are approximately driving around a certain tenfold. A possible approach is thus creating bins, ranging around every tenfold, such that we can determine if someone is speeding or not. However, for detecting other events/activities it could be interesting to bin the speed signal differently.

3.4.4 Conclusion

We see that equal-width and equal-frequency binning are respected binning methods in discretizing real-valued data. However, for the acceleration signal neither of them suffices for what we are trying to achieve. Fortunately we can use the characteristics and distribution of the data to determine suitable ranges for the bins. In some cases equal-width binning can be of use. For instance if we want to determine if the driver is speeding, it suffices to have bins that summarize all values between speed limits. If however, the speed signal is used to detect other events, other binning might be needed as it can heavily depend on the event to be detected.

4 Experimental Setup and Data

This chapter will discuss the setup of the conducted/implemented experiments to mine events/activities form naturalistic driving data. Also a description and analysis of the data provided by TNO will be given.

4.1 Real-World Microscopic Traffic Data

Data gathering has been done by TNO at the IVS department. Data has been collected by driving on public roads in the Netherlands, where a distinction is made between urban and highway data. Here we will describe and analyze the data and discuss preprocessing needed before we can start mining events/activities.

4.1.1 Data gathering

A data set of 60 hours of driving has been collected in 2015. A total of 20 participants contributed by driving a TNO lab car, a Toyota Prius. All participants had to conform to the following requirements defined by TNO in [11]:

- The participant is between the age of 25-60,
- The participant possesses his/her driving license for at least 5 years,
- The participant drives 5.000 km/year or more,
- The participant masters the Dutch language,
- The participant has a minimum educational degree of MBO 4.

The route focused mainly on urban and rural roads between Helmond and Eindhoven. In total the route amounted to 48.5 km, needing approximately 1 hour and 10 minutes to complete excluding delays. See figure 4.1 for an overview of the trajectory.

For each participant data has been recorded and sampled according to the following criteria in Table 4.1.

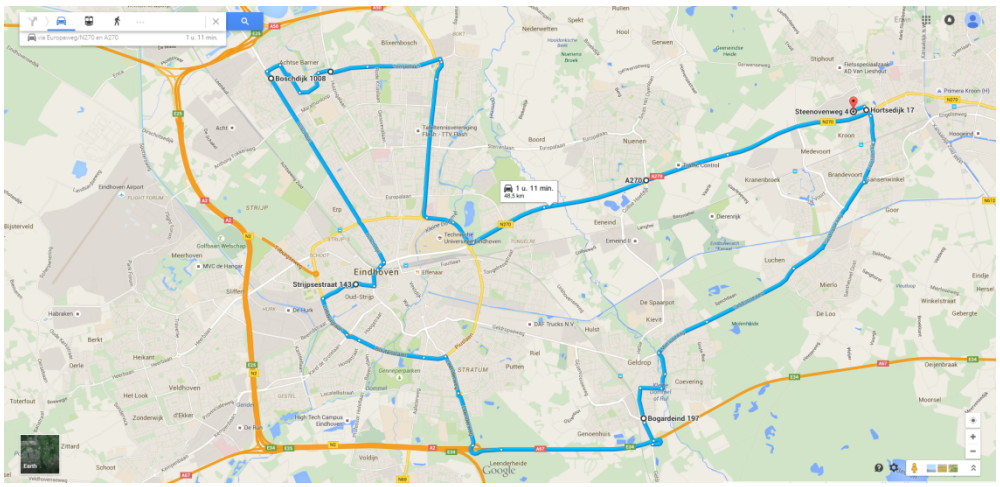

Fig. 4.1: Map showing the trajectory in blue for driving experiment. Source: [11].

Tab. 4.1: Sampling criteria of the data according to [11].

Measured signals	Unit	Frequency
Longitudinal velocity	m/s	50 Hz
Longitudinal acceleration	m/s^2	50 Hz
Lateral acceleration	m/s2	50 Hz
Steering wheel angle	deg	50 Hz
Yaw rate	deg/s	50 Hz
Object in front detected	-	50 Hz
Longitudinal distance to MIO (most important object)	m	50 Hz
Lateral distance to MIO	m	50 Hz
Relative velocity with MIO	m/s	50 Hz
Relative acceleration with MIO	m/s^2	50 Hz
Distance to left lane	m	50 Hz
Distance to right lane	m	50 Hz
GPS longitude	deg	10 Hz
GPS latitude	deg	10 Hz

4.2 Experimental design

In this work we will conduct experiments limited to events/activities concerned with longitudinal movement. The algorithm will be run on the data to mine patterns from which we can discover acceleration, cruising and deceleration.

As argued before, there is no ground truth to which we can test our findings against. Thus the conducted experiments will focus more toward a qualitative, exploratory study. Several parameters in the model that can be varied which could change the behaviour and outcome of the model. To observe the influence of these variations, the parameters will be varied to study the influence of each parameter on the behaviour and outcome of the model. Finally we can argue which configuration seems to fit best in detecting the desired events. The parameters possible to vary are the following.

4.2.1 Binning threshold

The first 'parameter' is the binning of the signal. In the case of acceleration, this comes down to the range of the cruising bin defining which values should be considered as cruising due to noise. The influence of varying this parameter on the data is quite evident, as increasing the value would just result in more points captured in the cruising bin. Therefore we do not vary this parameter, but we consider what seems to be the optimal range.

Determining a suitable value is done as follows. First a lower bound has been defined in terms of how much the bin should at least capture. If the acceleration signal seems to be about constant for a while, one could speak of constant speed. Even if the value does not fluctuate around zero, it is still to be considered constant speed. The acceleration signal not fluctuating around zero indicates a sufficient precision of the sensor, but a poor accuracy. If the signal is fluctuating around a value that is close to zero, it becomes apparent that the car is cruising. These signals should then at least be captured in the cruising bin. This is achieved by setting the absolute range for the cruising bin to be at least the maximum value that such a signal has. In practice, this comes down to a lower-bound of about 0.15 m/s^2. This value is also suggested by TNO as the noise of the acceleration signal. An upper-bound for the threshold is needed as well. Ideally no acceleration instances should be captured in the cruising bin. However, this is unachievable as the two are not linearly separable. Thus in order to find a threshold value in a more substantiated way we will use kernel density estimation on the data. From the generated density function we can search for local optima which represent cutting points with a statistical background.

4.2.2 Frequency Threshold

Second, the frequency threshold for the frequent itemsets and sequences. The frequency threshold will be varied from 1 (absolute) to 0.25%, 0.5%, 0.75% and 1% (relative). Setting the frequency threshold to 1 (absolute) seems trivial as every pattern will be frequent. However, in this way one can clearly see what the added value is of mining frequent patterns in the data. In essence, this is the added value of the proposed method, because if everything would be frequent it would suffice to apply a convolution mask as in the state-of-the-art algorithm by TNO.

4.2.3 Sequence Length

Third, we have the length of the sequences we take into consideration when interpreting the patterns. A longer sequence leads to a larger interval around a point used in labeling the activity at a given point. Just as for the frequency threshold, we consider the base case of the length equal to 1. Next we consider two other cases where the length equals 5 and 10. In this way we can study the influence of taking more points into consideration whilst labeling/intercepting a pattern.

4.2.4 Sampling Frequency

Lastly, we vary the sampling frequency of the data. This is the only parameter that could be left out in the model by taking the original sampling frequency. However, as argued before, having a sampling frequency which is too high can lead to finding patterns on a too low level. Thus the purpose of the application serves as an indicator for the sampling frequency. We will vary the sampling frequency between $1Hz$, $5Hz$ and $10Hz$. The original sampling frequency of $50Hz$ is not considered as we need sequences of length 50 to represent 1 second in such a case which is impractical.

4.2.5 Combining Acceleration and Velocity

To show the workings of the method on multiple signals we run the algorithm on a combination of two sensors. We will add the velocity signal to the data trying to improve the classification of the events/activities. To use the speed signal, we need to discretize the signal just as for the acceleration signal. We take the derivative of

the speed signal to create a derived acceleration, which is binned in the same way as the original acceleration signal.

Also, in interpreting the patterns we now need a new method since the patterns adopt a different form than before. First we make a distinction between three types of sequences that can be found in the data. By examples we show how these patterns would look in the data. Consider the base case seen below:

S_0	1	1	1	1	1
S_1	1	1	1	1	2

Then, the first type of sequences are those where both signals are taken into account at each time point over the entire sequence:

S_0	1	1	1	1	1
S_1	1	1	1	1	2

The second type are sequences where only one signal is taken into account at each time point:

S_0	1	1	1	1	1
S_1	1	1	1	1	2

And the last type are those where at each time point, either one or two signals are taken into account, excluding the previously mentioned cases:

S_0	1	1	1	1	1
S_1	1	1	1	1	2

To limit ourselves, we only consider the first type of pattern. In this pattern both sensors are taken into account at each time point for the entire sequence. By considering these patterns we take both signals into consideration evenly. Our new labeling will now take the majority class over all the 10 values in the two sequences. This differs from just combining the majority class from separate signals. Consider the following example:

S_0	3	3	2	2	3
S_1	1	1	2	2	1

Normally S_0 and S_1 would choose acceleration (3) and deceleration (1) respectively. Then combining them afterwards, would result in a tie. If we look at the majority class over the S_0 and S_1, cruising (2) would be chosen. In the experiments we will show how this translates in practice.

5 Results

In this chapter we present and discuss the results of the experiments. For the experiments, frequent sequences have been mined in the data for various configurations. We have limited ourselves to only mining patterns concerning longitudinal movement, thus events/activities in the 'Speed' state as defined in Figure 2.1. For the experiments we used the data described in 4.1. First, we have ran the algorithm on just the 'longitudinal acceleration' signal to detect acceleration. On this signal we have also tested the influence of different configurations for different parameter values as described in Section 4.2. Second, we have added the longitudinal velocity (actually speed, but named velocity in the data), and mined patterns over the two signals, to show the workings of the method on multiple signals.

5.1 Interpretation of the patterns

The patterns returned by the algorithm are frequent sequences of itemsets of $(sensor, value)$ pairs, followed by their corresponding time stamps. For the case of longitudinal movement with just the longitudinal acceleration signal, we have one sensor and three bins, making the total number of possible configurations 3^n for a sequence length of n. However, since our approach yields an unsupervised nature, these sequences on their own have no meaning. Thus as argued before, these patterns still need to be labeled before they can be interpreted as events/activities.

Mining different events/activities requires different interpretations of patterns. The most straightforward approach for the events in the 'Speed' state, is choosing the majority bin in every sequence. By taking the majority class, we define the activity at each time point by taking the neighboring points into account. For instance, assume a vehicle accelerating for some seconds, then cruising for a few milliseconds, followed by accelerating for a few seconds again. In such a case, cruising for a few milliseconds in a signal that is accelerating for several seconds should be seen as noise and thus neglected. However, taking the majority class makes ties possible. A way to reduce ties is to only consider sequences with odd lengths. Yet, since we

have 3 bins a tie could still occur[1] for sequences of odd length, but in such a case we pick one arbitrarily.

5.2 Optimizing bin ranges

The acceleration signal is discretized into 3 bins according to our reason earlier. As mentioned in Section 4.2, the ranges for the deceleration, cruising and acceleration bins are set to $(-\infty, -0.15)$, $[-0.15, 0.15]$ and $(0.15, +\infty)$ respectively. These ranges are based on the noise level of the accelerometer, which is equal to 0.15. Yet, we stated that we would justify this threshold through experiments as well. To do so, we have used Kernel Density Estimation to estimate the distribution (probability density function) of the data, with a Gaussian kernel and a bandwidth of 0.01. In Figure 5.1 we can examine a plot of the estimated density function. We have also plot the threshold values of ± 0.15 to compare them to the local minima. It becomes evident that the suggested threshold values are close to local minima in the density function. The actual extremes are located at -0.155 and 0.156.

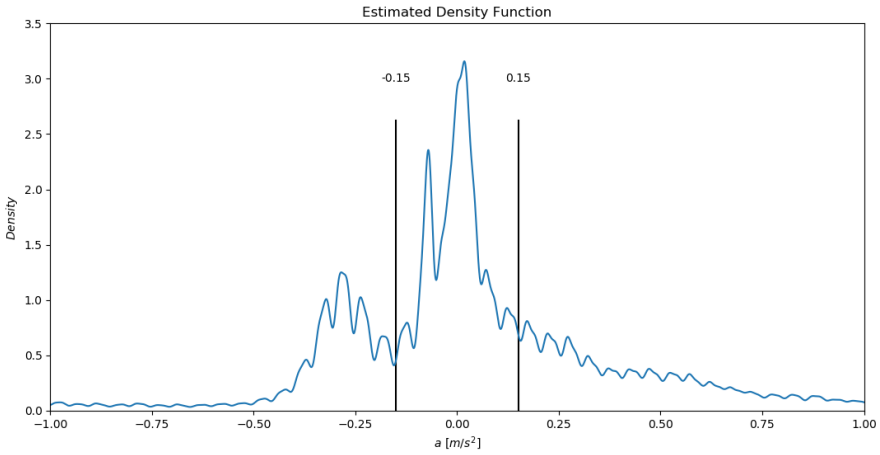

Fig. 5.1: Kernel Density Estimation of the acceleration signal. Kernel=Gaussian, bandwidth=0.01.

[1]However, a sequence with a tie that involves all 3 bins are unlikely to occur due to the nature of the data.

5.3 Mined Events and Activities

In this section we will present some figures and tables that clearly show the behaviour of the model when varying the different model parameters. In Chapter 4 we have seen the experimental setup where we mentioned all the configurations we were going to consider.

5.3.1 Varying the frequency threshold

We have varied the frequency threshold for 0.25%, 0.50%, 0.75% and 1.00%. Also we have examined the case where the absolute support is equal to 1, meaning that everything is frequent. Table 5.1 shows the number of instances found for each activity. From here we can see that the number of activities found does only decrease when the threshold increases, which shows that the frequent patterns are mined as intended. The total number of points in the data is 141404. Thus for sampling frequencies of $1Hz$, $5Hz$ and $10Hz$, the total number of points equals to respectively 2828, 14140 and 28280.

Tab. 5.1: Total number of occurrences for deceleration, cruising and acceleration activities for given Sampling frequency and Frequency Threshold with Sequence length being equal to 5.

Sampling Frequency	Frequency Threshold	Deceleration	Cruising	Acceleration	Total
1	0.25	687	1206	763	2656
	0.50	659	1162	738	2559
	0.75	608	1162	701	2471
	1.00	562	1066	602	2230
5	0.25	3477	6131	3711	13319
	0.50	3477	5733	3643	12853
	0.75	3305	5319	3369	11993
	1.00	3079	5208	3369	11656
10	0.25	7031	12173	7631	26835
	0.50	6746	11078	7171	24995
	0.75	6436	10328	6835	23599
	1.00	6436	10328	6835	23599

For considering the absolute support of 1, when all patterns are frequent, we have plotted graphs to compare the absolute support of 1 to the frequency of 0.5%. Figure 5.2 shows different sequence lengths for $5Hz$ with the support being 0.5%. The plots show the acceleration signal, the bin boundaries in black, and the blue, black and red dots for an instance of an acceleration, cruising or deceleration activity. In Figure 5.3 the configurations are the same, except for the support which is equal to 1. A nice observation can be seen in the bottom graphs in Figures 5.2 and 5.3. For the absolute support of 1, we can see a sequence of deceleration from $[155 - 158]$, while the signal clearly goes far above 0, even above the threshold value 0.15. This patterns occurs so little, that for the 0.5% support case it is already considered infrequent.

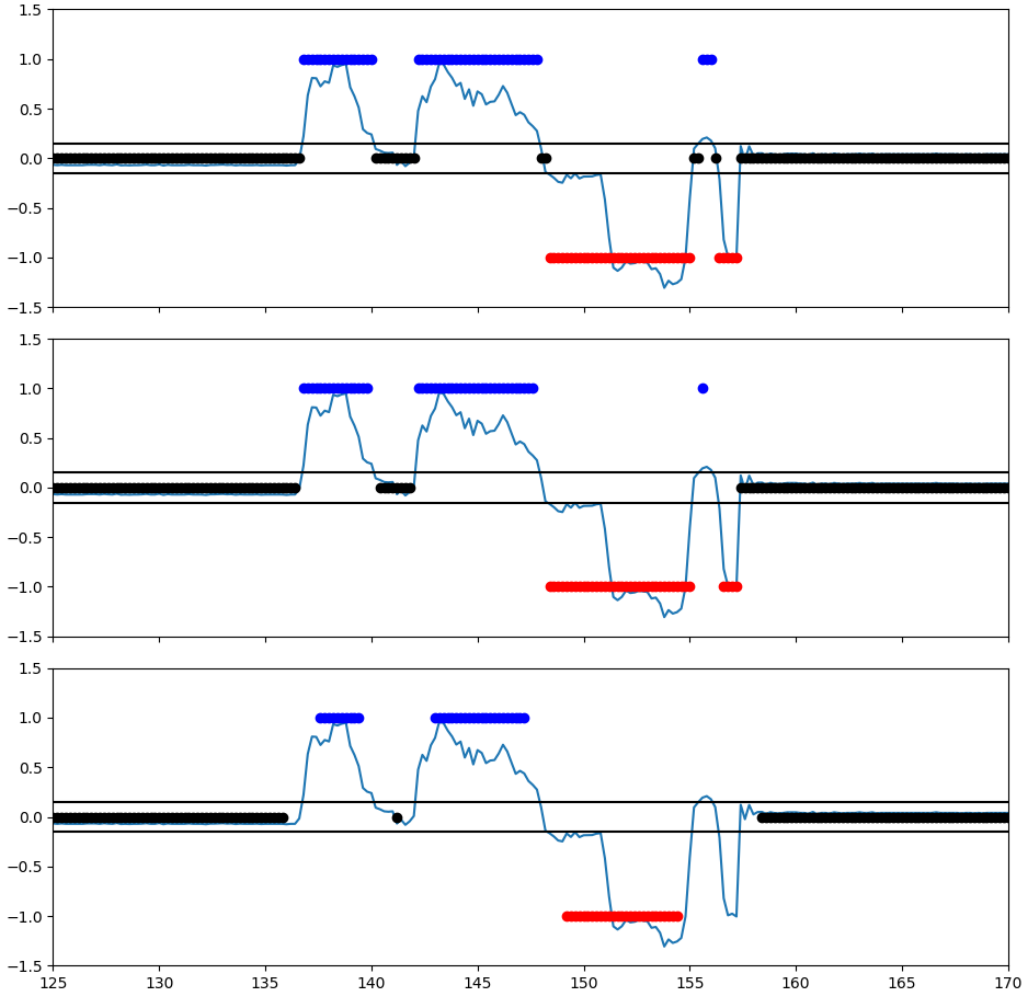

Fig. 5.2: Top: Sequence Length 1, Middle: Sequence Length 5, Bottom: Sequence Length 10. All have a sampling frequency of $5Hz$ and support of 0.5%.

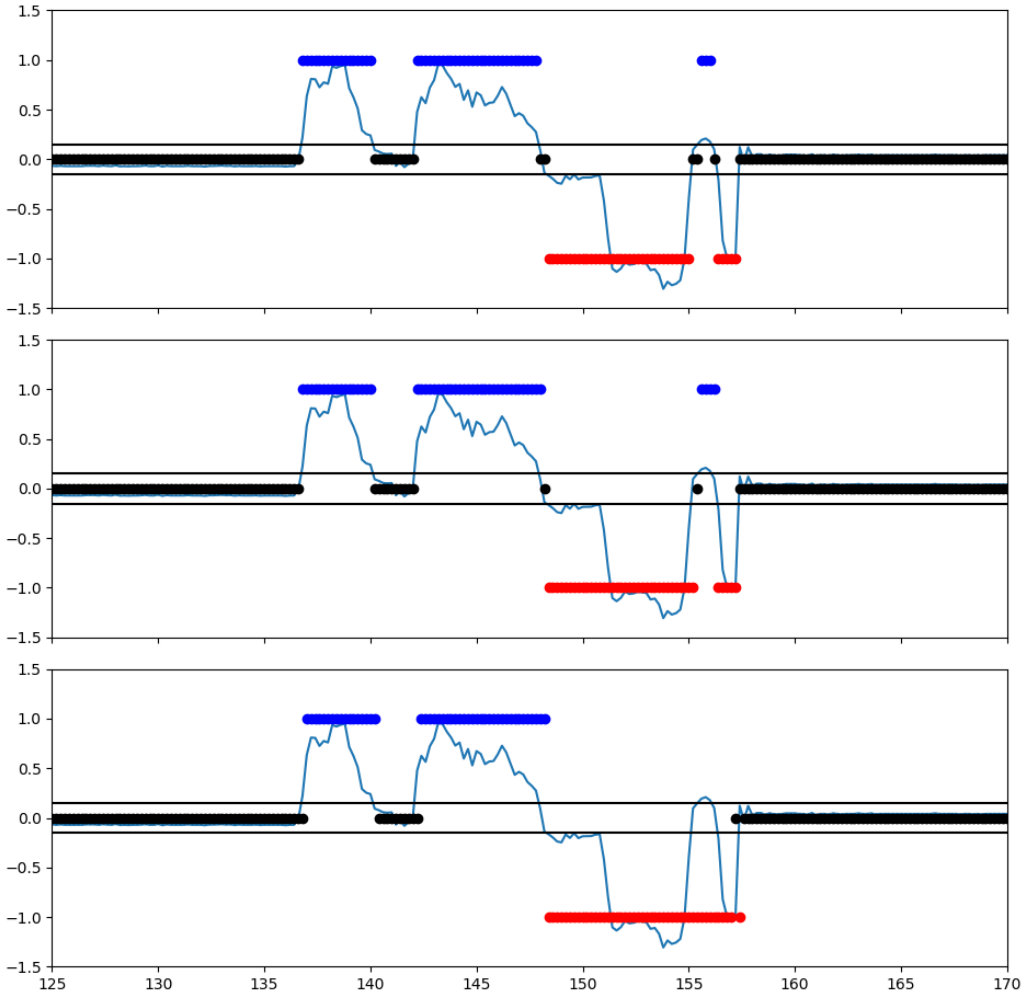

Fig. 5.3: Top: Sequence Length 1, Middle: Sequence Length 5, Bottom: Sequence Length 10. All have a sampling frequency of $5Hz$ and support of 1.

Influence of the frequency threshold

As to be expected from the frequency threshold, increasing the threshold will result in less sequences being frequent. The results show that by increasing the threshold, the first sequences (or patterns) that become infrequent are those where the sequential bins are opposite to each other. This is an expected result, as signals do not tend to fluctuate between very high and low values.

On the other hand, sequences having a high frequency are those with their signals falling into the same bin consistently. The sequences that are not frequent indicate patterns where there is a lot of fluctuation between acceleration and deceleration. In this case, when the patterns are labeled according to the majority class, the

sequences that are infrequent are not labeled, as they are not returned by the algorithm.

As can be seen in Figure 5.2, the number of gaps increase when the frequency is higher than 1. This is due to the fact that infrequent sequences are not taken into account when labeling the patterns. At these points anomalies are detected which can not be labeled with certainty with respect to the frequency threshold. As the threshold increases, less sequences will be frequent, resulting in less points being labeled. In this way the expert can set a frequency threshold to his/her liking, only labeling points where the algorithm is certain. The expert can manually label the infrequent points.

5.3.2 Varying the Sequence length

Second we have varied the sequence length by testing lengths 1, 5 and 10. Figure 5.2 already shows a plots which compare these variations. In the bottom graph the number of infrequent patterns is much bigger as for the sequences of length 5 or 1. Sequences of length 1 however, are always frequent for the 1 sensor setting. So compared to the sequence length being equal to 5, a higher sequence length generates too much infrequent patterns for the $5Hz$ case. When we look at Figure 5.4 we compare plots for $10Hz$. Here we can see that the sequences of length 5 are more noisy and provide less solid activity sequences. Also, for sequences of length 1 in the $10Hz$ case, the noise is much more evident, to which the labeling reacts.

Influence of the sequence length

Varying the sequence length results in looking at a bigger time interval to determine the label of a time point. The most extreme case is the sequence length equaling 1. In this case only the itemset at the point in time itself is considered. Since in the 1 length sequence (itemset) case all itemsets are frequent, every point will be labeled according to the discretization done by binning. The results show that it is very sensitive to noise when it is fluctuating around the bin threshold, resulting in alternating labels for sequential points or outliers in a range.

When increasing the sequence length from 1 to 5, the number of possible sequences increases to $3^5 = 243$. As the number of sequences increase, there are more sequences that can become infrequent. Since the total number of sequences in

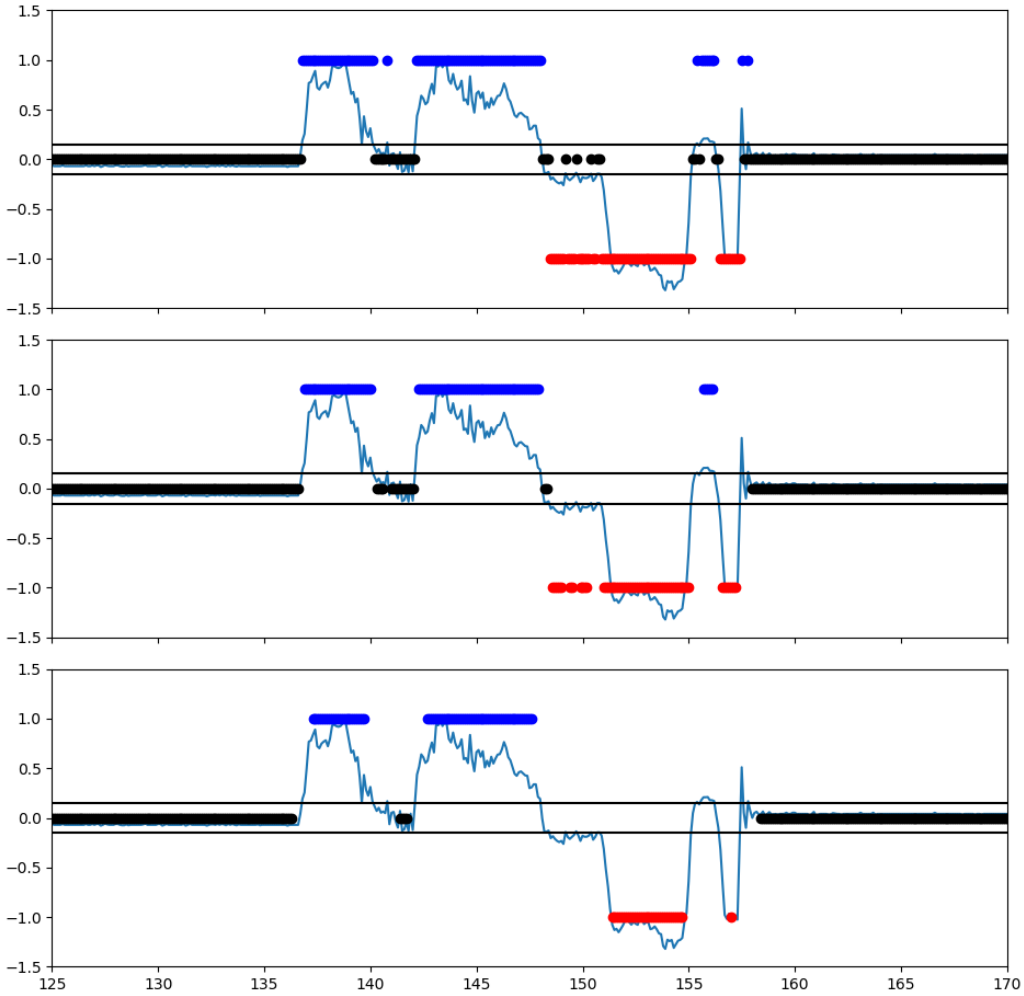

Fig. 5.4: Top: Sequence Length 1, Middle: Sequence Length 5, Bottom: Sequence Length 10. All have a sampling frequency of $10Hz$ and support of 0.5%.

the data remains the same[2], the increase in possible sequences results in more infrequent sequences. Theoretically speaking it is possible that the number of infrequent patterns doesn't change, but this won't happen in practice unless the binning is inaccurate.

More gaps can be observed in the results, since more infrequent sequences occur due to the possible sequences increasing, thus resulting in more unlabeled points. However, the increase in length also results in outliers not being labeled according to the bin they fall in. For instance in the case of a long sequence falling in the cruising bin, except for one point falling in the acceleration bin. When the sequence length equaled 1, the point would be labeled as acceleration. However, if the sequence

[2]Disregarding the 4 sequences lost due to the first and last 2 points not able to create a sequence.

length equals 5, the two points in front and the two points after the point will result in the point being labeled as cruising, given that this sequence $\langle aacaa \rangle$ is frequent with respect to the threshold. Thus longer sequences filter out noise well, but can become infrequent more easily as the number of bins increase.

5.3.3 Varying the Sampling Frequency

We have varied the sampling frequency for $1Hz$, $5Hz$ and $10Hz$. Figure 5.5 shows a nice overview for different sampling frequencies for a mutual sequence length of 5. The signals show that the shape of the curve is reasonably maintained, whereas it becomes worse for the $1Hz$ case. An observation in the data that is recurring, is a small spike before a cruising activity. This comes from the movements the car adopts before a full stop. However, when the sampling frequency is decreased, these patterns get faded out in the data. Next, we can see that for the $5Hz$ case, the labeling seems to represent naturalistic driving the best. For the $10Hz$ case the data signal is too precise, resulting in small interruptions in the activities found.

Influence of the sampling frequency

In the most extreme case the data is downsampled to $1Hz$, hence each point represents 1 second in time. The graph plotted in the $1Hz$ case still resembles the original data in terms of the shape reasonably well. It can clearly be seen that intervention as spikes are smoothed along the data. This results in small spikes phasing out in the data. Next to that, it averages the signal when fluctuating around the bin borders, such that the signal now falls into one bin at time instance (150). In general, the labeling of the patterns gives a good and generalized representation of the acceleration pattern of the car. The downside however, of the sampling frequency equaling $1Hz$, is the fact that as the sequence length increases by 1, the amount of original data taken into consideration increases by 1 second. Thus making a judgment based on long sequences results in making judgments based on a longer timespan. Next to this, since each point now represents 50 points (or 1 second) in the original data, the resolution of the labeling becomes bigger. Meaning, that if two consecutive points differ in label, the actual transition point could be anywhere $0.5s$ before or after the following point.

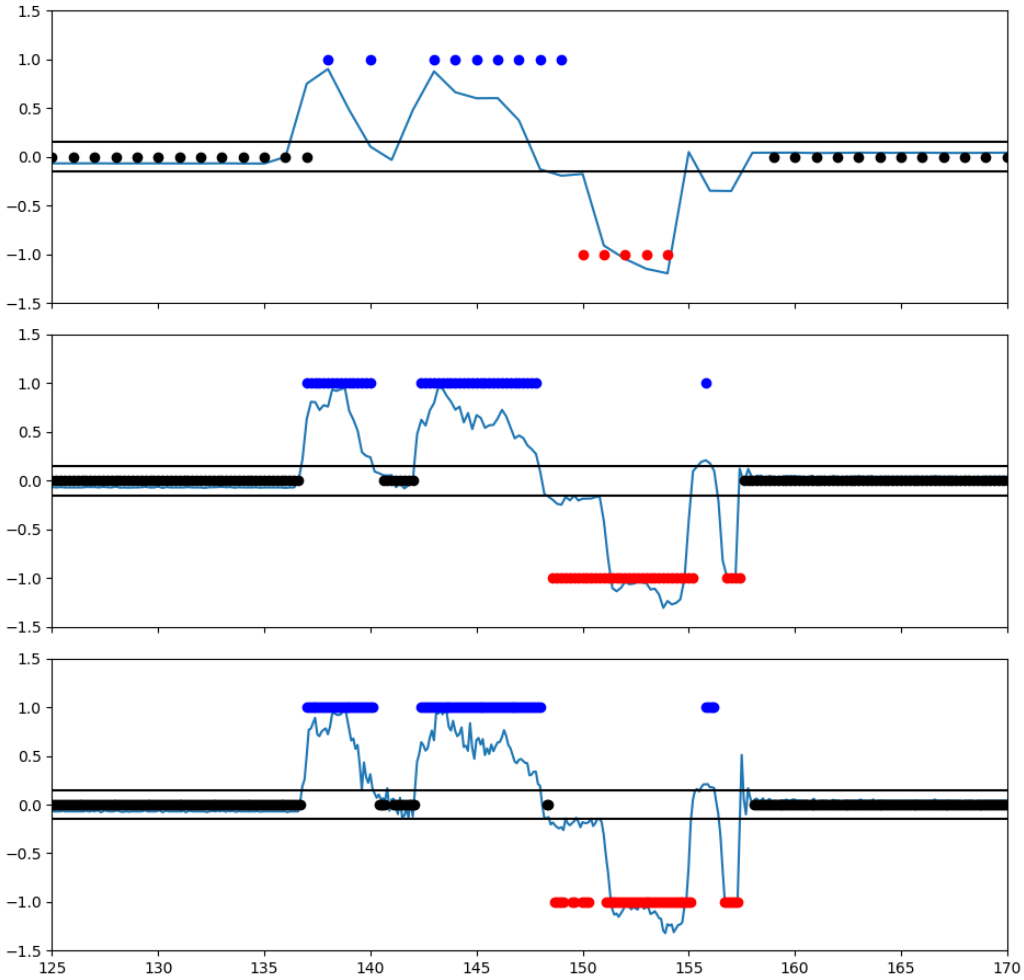

Fig. 5.5: Top: 1Hz, Middle: 5Hz, Bottom: 10Hz. All have a sequence length of 5 and support of 0.5%.

The biggest takeaway from the sampling frequency is the accuracy of the prediction of the start or end of an activity. As the sampling frequency becomes smaller, information is lost marking the accuracy of the prediction worse. For the most extreme case of $1Hz$, the prediction has an uncertainty of $0.5s$. As the sampling frequency grows bigger, the inaccuracy decreases inversely proportional to the frequency. Thus a lower sampling frequency generalizes well and is beneficial for computational reasons in terms of time and memory, but loses accuracy on the prediction of the start and end of an activity. It is up to the expert to make a consideration as to what accuracy is needed for predictions for events, as they could vary for different events.

5.3.4 Comparison of configurations

To make a fair comparison of different configurations, we should compare the configurations such, that for each judgment made the same amount of data is taken into account. For instance, if we compare the setting for 1, 5 and $10Hz$ with each other, each having a sequence length of 5, the amount of data used for each consideration differs. Respectively, we look at 5, 1 and 0.5 seconds for each judgment. Thus to look at the best configuration, we now look at 3 settings with each setting looking at 1 second of data in the original dataset (50 points) to make it's judgment. To do so, we only need to add one more image, the one for $1Hz$ and a sequence length of 1.

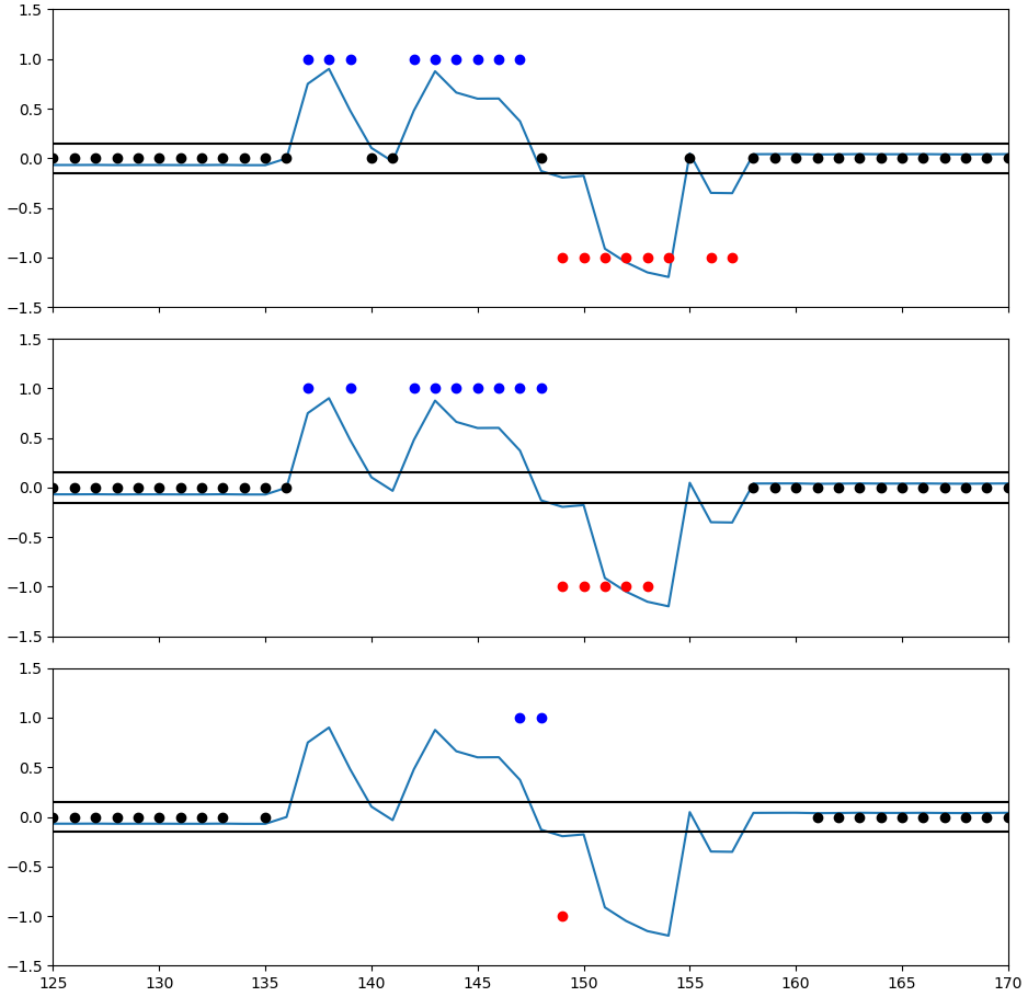

Fig. 5.6: Top: Sequence Length 1, Middle: Sequence Length 5, Bottom: Sequence Length 10. All have a sampling frequency of $1Hz$ and support of 0.5%.

When comparing the three plots, we need to look at the top plot of Figure 5.6, the middle plot of Figure 5.2 and the bottom plot of Figure 5.4. In all these plots, each point is labeled on 1 second of data. When comparing the plots, they look alike a lot, due to them all taking 1 second of data into account. The differences come from the smoothing done by the downsampling, where in the $1Hz$ case at time point 155, the data is smoothed such that what used to be above the 0.15 threshold is now below. Also we can see that longer sequences introduce more infrequent patterns.

5.4 Combing acceleration and velocity

In Chapter 4 we have explained how we would combine the acceleration and velocity signal to mine frequent sequences of itemsets. We have derived the acceleration by taking the derivative of the velocity signal. We have used a binning threshold of 0.15, a sampling frequency of $5Hz$ and a support threshold of 0.25%. The support threshold was lowered because by adding another signal the number of possible combinations grows, making more patterns infrequent. We have applied the same labeling method as for the single sensor class as defined in Chapter 4.

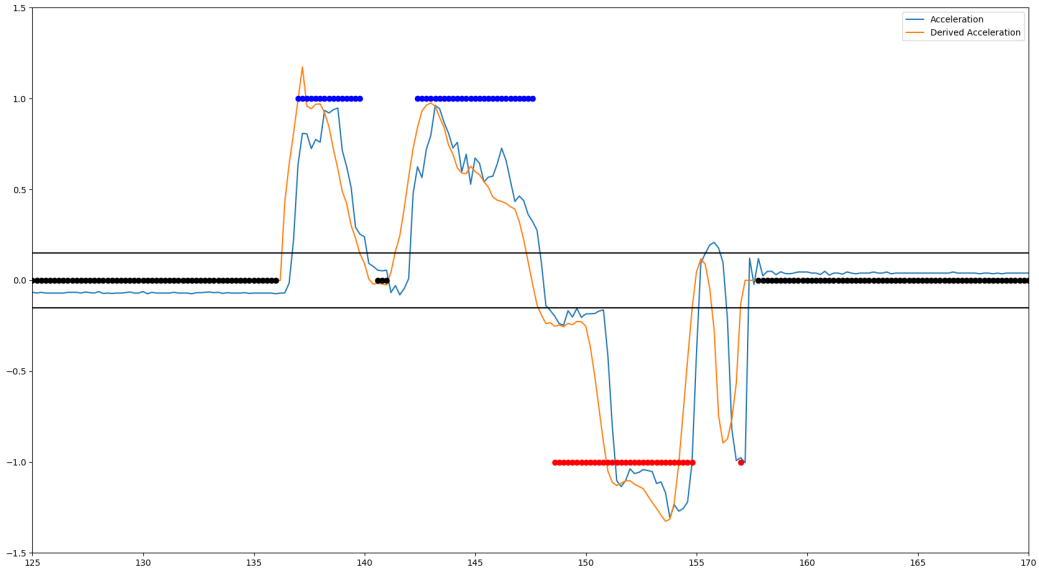

Fig. 5.7: Plot of the acceleration and derived acceleration, with the corresponding mined activities.

From the plot in Figure 5.7 we can see that the two signals are very close. Taking the majority class from the two signals benefits from the signals being so close, as this results in having similar values over an interval. The labeling of the activities is pretty decent, and it tends the most to the plot for $5Hz$ and a sequence length of 5,

as these values were the same in this case. However, a non-trivial difference is that the two signals combined are not labeling the peak at 155 as acceleration. Thus we can see that by using multiple signals we can filter noise out as well.

Conclusion

6

In this thesis we have examined whether unsupervised data mining techniques are adequate in mining events/activities from real-world microscopic traffic data. First a literature study has been conducted to formalize the notions of events and activities in the automotive context. Using these formalizations we have argued that frequent pattern mining over multiple data streams appeared to be a suitable approach in mining events/activities from the data, as the approach enables to mine events/activities on different abstraction levels, serving the subjective desire of the end user.

Through experiments we have analyzed how the different parameters in the model influence the resulting patterns from the algorithm. The parameters varied throughout the experiments were: the discretization threshold(s), frequency threshold, sequence length and sampling frequency. For each of these different configurations have been tested to examine their behaviour as well as to conclude which configuration suits best.

The discretization threshold was determined by a local minima in an estimated density function and was optimal for $0.15m/s^2$ (this was also addressed as the sensor noise by TNO). The frequency threshold influences the number of labeled datapoints. An increase in the frequency threshold will result in only labeling datapoints that occur more often in the data. As the sequence length, number of bins for a signal or the total number of signals increases, more combinations of patterns will be possible, thus when determining an optimal frequency threshold, one has to consider those factors. The sequence length will filter out noise or peaks in the data as it increases. However, if the sequences become too long, the algorithm becomes computationally expensive and the sequences tend to overgeneralize, missing small events in the data. And lastly, the sampling frequency. When it is too large, we need very long patterns to determine small time intervals in the data. However, as the sampling frequency gets smaller, we can see two consequences. One, the data will not resemble the original data well enough. And two, as the sampling frequency gets smaller, the point in the data will represent larger time intervals. Then when one wants to detect an event with a sampling rate of $1Hz$, the error margin will be

0.5 on both sides, which is pretty large, considering that an event should take place at an time instance.

When comparing our method to the state-of-the art method by TNO in terms of results, the activities found seem to be comparable. Since there is no objective ground truth, it is not possible to evaluate which method performs better. However, when comparing the methods theoretically, there is one evident difference which can be considered an enhancement. Our method only considers frequent patterns, neglecting infrequent instances of events/activities. Due to this, an interesting finding arises from the data. In the case of an infrequent activity the algorithm considers an activity as an transition event. We will elaborate on this in the recommendations.

The research question: '*What is a suitable data-driven approach to mine events and activities from real-world traffic data?*' can be answered accordingly. Via reasoning we have showed that frequent pattern mining is a suitable approach for mining discretized microscopic traffic data. To support our line of reason we have conducted empirical research. The results of this study show that the results are at least comparable to the state-of-the art method developed by TNO. From this we can conclude that using unsupervised data mining techniques is a suitable approach to mining events and activities from real-world (microscopic) traffic data. In particular frequent pattern (itemset/sequence) mining shows to perform well.

6.1 Recommendations

In our work we have some findings that can contribute to the work of TNO on StreetWise in detecting events/activities, and therefrom scenarios in real-world microscopic traffic data. These findings result in practical as well as theoretical recommendations.

First we have developed an unsupervised method in detecting events and activities from real-world microscopic traffic data. For this model we have ran experiments to discover the optimal configuration to detect events/activities. The approach is a broader approach that could tackle problems of which mining events from temporal data is an instance. TNO could generalize the approach to not only make it applicable for event mining, but to other similar problems as well. An example of such a problem TNO is dealing with that could also be tackled with this approach is finding errors in charging batteries log data.

Second, a result of our study shows that due to noise in the data it is difficult to detect activities in sequences over a timespan of seconds. Infrequent patterns in the data show sequences of activities at low levels which must not be averaged. These patterns represent actual, physical events as change in longitudinal movement. This suggests that events for change of activities in the sense of higher abstractions, think of intentions of the driver, should be addressed as a possible interval in time, instead of a time instance. In example, consider the following situation where a driver wants to enter the highway. When there are no other vehicles, the car would likely and ideally adopt the following sequence of activities (accelerate)->(cruising). However, it is not uncommon for drivers to speed over the limit to later decelerate again. Thus we argue that infrequent patterns in the data could indicate events in terms of time-intervals, when speaking of state transition on higher abstractions, e.g. the intentions of a driver.

Finally, and possibly the most valuable recommendation is using the method for mining. The method we have proposed is originally purposed to work with categorical data. It is true that we have shown that by adaptation, the algorithm is applicable to numerical data, however, due to the nature of the algorithm being meant for categorical data, it would presumably perform better when having discrete activities as input. Especially when having time-intervals as input, the method is suitable [4].

6.2 Future Work

The current implementation of our method has a drawback which many exploratory data mining algorithms have, namely the pattern explosion. As the number of sensors, bins or pattern lengths increase, the number of patterns show an exponentiation growth. Condensed representations of the data could be helpful in this case. However, we have to keep the requirement in mind of finding all the patterns as well as keeping the patterns humanly understandable. Thus condensed representations of the data which violate these requirements could not serve as an improvement of the algorithm. Yet, an improvement could be mining only non-derivable patterns presented in [14] for data streams. By mining only the non-derivable patterns, the number of patterns found decreases dramatically, while all patterns can be derived from the set of non-derivable patterns.

Also, the labeling of the patterns as it is implemented now is still very naive. There is much room for improvement, by exploiting the nature of the data or the nature of

the events/activities desired to detect. Also, the pattern explosion just mentioned serves as another reason to come up with improved labeling methods.

Bibliography

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. “Mining association rules between sets of items in large databases”. In: *Acm sigmod record*. Vol. 22. 2. ACM. 1993, pp. 207–216 (cit. on p. 14).

[2] Rakesh Agrawal, Ramakrishnan Srikant, et al. “Fast algorithms for mining association rules”. In: *Proc. 20th int. conf. very large data bases, VLDB*. Vol. 1215. 1994, pp. 487–499 (cit. on p. 15).

[3] Rakesh Agrawal and Ramakrishnan Srikant. “Mining sequential patterns”. In: *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*. IEEE. 1995, pp. 3–14 (cit. on p. 15).

[4] Iyad Batal, Dmitriy Fradkin, James Harrison, Fabian Moerchen, and Milos Hauskrecht. “Mining recent temporal patterns for event detection in multivariate time series data”. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2012, pp. 280–288 (cit. on p. 43).

[5] Ricky W Butler and George B Finelli. “The infeasibility of experimental quantification of life-critical software reliability”. In: *ACM SIGSOFT Software Engineering Notes* 16.5 (1991), pp. 66–76 (cit. on pp. 1, 2).

[6] Gong Chen, Xindong Wu, and Xingquan Zhu. “Mining sequential patterns across data streams”. PhD thesis. University of Vermont, 2005 (cit. on p. 18).

[7] Erwin De Gelder, Jan-Pieter Paardekooper, Jeroen Ploeg, et al. “Ontology of scenarios for the assessment of automated vehicles”. In: (2018) (cit. on pp. 5–7).

[8] Hala Elrofai, Jan-Pieter Paardekooper, Erwin de Gelder, Sytze Kalisvaart, and Olaf Op den Camp. “SCENARIO-BASED SAFETY VALIDATION OF CONNECTED AND AUTOMATED DRIVING”. In: (2018) (cit. on pp. 7, 8).

[9] Hala Elrofai, Daniël Worm, and Olaf Op den Camp. “Scenario Identification for Validation of Automated Driving Functions”. In: *Advanced Microsystems for Automotive Applications 2016*. Springer, 2016, pp. 153–163 (cit. on pp. 1, 2).

[10] ENABLES3. *Validation & Testing Of Complex Automated Systems*. 2018. URL: `https://www.enable-s3.eu/about-project` (visited on July 23, 2018) (cit. on p. 1).

[11] Maria Frias Goyenechea. *ERP dataset metadata*. TNO. 2016 (cit. on pp. 23, 24).

[12] Jiawei Han, Jian Pei, and Yiwen Yin. “Mining frequent patterns without candidate generation”. In: *ACM sigmod record*. Vol. 29. 2. ACM. 2000, pp. 1–12 (cit. on p. 15).

[13] Sotiris Kotsiantis and Dimitris Kanellopoulos. “Discretization techniques: A recent survey”. In: *GESTS International Transactions on Computer Science and Engineering* 32.1 (2006), pp. 47–58 (cit. on p. 20).

[14] Haifeng Li and Hong Chen. “Mining non-derivable frequent itemsets over data stream”. In: *Data & Knowledge Engineering* 68.5 (2009), pp. 481–498 (cit. on p. 43).

[15] Tim Oates and Paul R Cohen. “Searching for structure in multiple streams of data”. In: *ICML*. Vol. 96. Citeseer. 1996, p. 346 (cit. on p. 18).

[16] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, et al. “Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth”. In: *icccn*. IEEE. 2001, p. 0215 (cit. on p. 18).

[17] Andreas Pütz, Adrian Zlocki, Julian Bock, and Lutz Eckstein. “System validation of highly automated vehicles with a database of relevant traffic scenarios”. In: *situations* 1 (2017), E5 (cit. on p. 1).

[18] Claes Tingvall and Narelle Haworth. “Vision Zero: an ethical approach to safety and mobility”. In: *6th ITE International Conference Road Safety & Traffic Enforcement: Beyond*. Vol. 1999. 2000, pp. 6–7 (cit. on p. 1).

[19] Hans-Georg Wahl, Kai-Lukas Bauer, Frank Gauterin, and Marc Holzapfel. “A real-time capable enhanced dynamic programming approach for predictive optimal cruise control in hybrid electric vehicles”. In: *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE. 2013, pp. 1662–1667 (cit. on p. 1).

[20] Mohammed J Zaki. “SPADE: An efficient algorithm for mining frequent sequences”. In: *Machine learning* 42.1-2 (2001), pp. 31–60 (cit. on p. 18).

[21] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li, et al. “New algorithms for fast discovery of association rules.” In: *KDD*. Vol. 97. 1997, pp. 283–286 (cit. on p. 15).