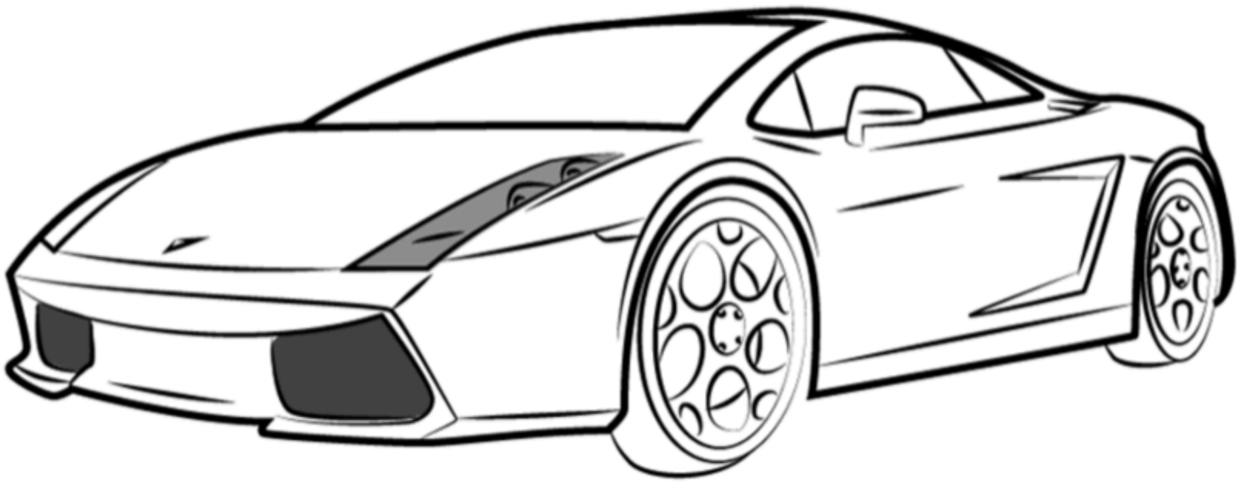
Lane Change Intention Recognition Models

Using Hidden Markov Models and Relevance Vector Machines

Rui Yu 4702069



DELFT UNIVERSITY OF TECHNOLOGY

MASTER THESIS

Lane Change Intention Recognition Models Using Hidden Markov Models and Relevance Vector Machines

---

Author:
Rui YU

Supervisor:
Dr. D. M. GAVRILA
Dr. A. TEJADA RUIZ

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Vehicle Engineering
Cognitive Robotics

September 14, 2019

Declaration of Authorship

I, Rui YU, declare that this thesis titled, "Lane Change Intention Recognition Models Using Hidden Markov Models and Relevance Vector Machines" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: [illegible]

Date: September 15, 2019

DELFT UNIVERSITY OF TECHNOLOGY

Abstract

Faculty of Mechanical, Maritime and Materials Engineering (3mE)

Cognitive Robotics

Master of Science

Lane Change Intention Recognition Models Using Hidden Markov Models and Relevance Vector Machines

by Rui YU

The development of intelligent vehicle and autonomous driving asked a higher requirement of ADAS on its functionality. Currently, ADAS systems are able to detect and segment urban and highway driving scenes. They cannot, in general, extract 'meaning' from this segmentation yet. Learning the intention of other road users will help ADAS understand surroundings and make a response. In a highway scenario, understanding what the preceding vehicle is about to do, is the minimum level of understanding the environment in order to take a decision about your own actions. Among the driving behaviors the preceding vehicle could do, lane change is a complex and dangerous one. Thus, we aimed to develop a real-time lane change intention recognition model. This report presents three models inspired by the Hidden Markov Models (HMMs) and Relevance Vector Machines (RVMs). Besides these two methods, we proposed a new model which combines them and overcome both of their main shortcomings. According to the testing result, the proposed model can correctly recognize more than 95% of the driving behaviors within 1 second the behavior starts, while the F1 score is also as high as 0.98. Besides the high accuracy, the model also has a good performance on the flexibility, testing complexity and the generalization ability.

Acknowledgements

Throughout doing this project I have received a great deal of support and assistance. I would first like to thank my daily supervisor, Dr. A. Tejada Ruiz, for his guidance through each stage of the process. You supported me greatly and were always willing to help me. I am extremely grateful for your kind words and scientific attitude.

Second, I want to thank the rest of my thesis committee. Dr. D. M. Gavrila and Dr. Julian Kooij are great and professional teachers. I learnt a lot from them during these two years. I also want to thank TNO who provided me this thesis topic. It is an interesting topic and I enjoy it a lot.

In addition, I would like to thank my parents, Jiansheng Yu and Wei Zhang, for their financial and spiritual support. Although home is far far away, I know you are always there for me. Finally, there are my friends, Z. Ai, X. Sui, N. Xia, and my boy friend Haoran Yuan, thanks for all the warm and fun you gave me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Advanced Driver Assistance Systems (ADASs) are a kind of vehicular system that collect environmental information and processes it by means of embedded algorithms to support drivers during the driving process. ADASs were first proposed around 1986 when a series of projects which were aimed at practical solutions to urban traffic jams was carried out by the European Union [1]. From the time of this initial idea to the point of current automated commercial cars, both the concept and functionality of ADAS have evolved and been significantly enriched. Currently, the ADAS is considered to be a collection of systems which can instruct, alarm or even override drivers in order to enhance road safety, driving comfort and traffic efficiency. At present, some types of ADAS such as Advanced Cruise Control (ACC) and Lane Keeping Assistance (LKA) are already mature enough to be used in commercial cars. However, there is still a long way to go before ADAS meets the requirements for high cognition to achieve further capabilities.

Under complex real-world traffic with various road users and infrastructures, an intelligent vehicle needs a sophisticated ADAS that can respond to the environment. We take a simple ADAS as an example. With the help of radar, ACC can detect obstacles in front. It automatically computes the distance between the vehicle and the obstacle, and then adjusts the velocity to maintain an appropriate distance headway. This is a complete procedure of the ADAS see (detect), interpret (compute) and respond (adjust the velocity) on the road. With the help of advanced sensors and algorithms, an advanced commercial car (such as Model S from Tesla) already can segment and classify different types of obstacles rather than just detecting them. In the future, the ADAS is going to reason about the near future and plan ahead for the driver. Thus, it is important for ADAS to recognize what the road users is going to do in the following seconds. If the ADAS knows the others' intention, it can further predict trajectories, understand surroundings and make a response. Therefore, based on the progress made in detecting objects using sensors or Vehicle Detection (VD) systems, we would like to move forward to recognize driving intentions.

1.2 Related Work

An intention recognition model can recognize other road users' driving behavior on the basis of their actions in an early stage. One typical application is to distinguish certain driving events from lane keeping, especially lane change behaviors. Some researchers have developed several models using multiple machine learning techniques, mainly Hidden Markov Models (HMMs) and Support Vector Machines (SVMs) to build intention recognition systems.

Pentland first proposed that human behaviors are best described as a sequence of control steps rather than as a sequence of sensor data. He used an HMM to model driving behaviors by splitting the behavior into control steps [2]. In [3], an HMM was used to predict whether a driver will stop before a stop bar when he is approaching an intersection. To solve the same issue, an Auto-Regressive HMM [4] whose observations are correlated was applied because it can better describe a real-world driving event than a standard HMM. Later, HMMs were also used to distinguish lane change events from lane keeping [5][6]. Nuria Oliver *et al.*extended Pentland's model [2] to Coupled HMMs [7] to achieve a higher recognition rate. He used driver gaze as an extra parallel set of recognition clues rather than merely sensor information. In [8], Kuge *et al.*further distinguished emergency lane changes from regular lane changes using HMMs.

Aoude and How introduced a method that combines SVMs and a Bayesian filtering (SVM-BF method) to recognize dangerous drivers in the DARPA Grand Challenge race [9]. Mandalia and Salvucci also chose the SVM-BF method in [10], but they used the variance of the data as the model input instead of the data themselves. Their result proved that it is a clever way to represent sequential data. Puneet Kumar *et al.*extended two-class SVM-BF classifiers to a multiclass one for the purpose of recognizing lane change [11]. A real-time classifier using Relevance Vector Machines (RVMs) which is an extension of standard SVM was developed by Brendan Morris *et al.*[12]. In [3], both an HMM and an SVM-BF classifier were built and tested for the purpose of characterizing compliant and violating driving behaviors. The result shows that the HMM has a higher classification rate than the SVM-BF classifier. However, SVM-BF can achieve a high classification rate while keeping false positive rates to certain minimal levels.

Recently, Husen and Lee introduced a new approach called syntactic pattern approach to recognize driver intentions. They defined pattern, sentence and grammar for driving behaviors. By representing and parsing the driving parameters, one can infer the driving behavior [13]. Julia Nilsson *et al.*developed a recognition system on the basis of the driving rules. It is a simple system without complex inputs and algorithm and easy to understand [14]. Besides, Gaussian Process (GP)[15], Recurrent Neural Networks (RNNs) [16], [17] and Dynamics Bayesian Networks (DBN) [18], [19] were also applied on this topic.

Based on the existing literature, one can observe that there are two shortcomings.

One is that none can recognize other vehicles' lane change intentions from the perspective of the host vehicle. In other words, they only let an intelligent vehicle know what its driver is going to do at the moment. However, recognize the intention of the host vehicle itself is not enough for understanding real-world traffic. We want the intelligent vehicle to know what surrounding drivers intend to do. The second is that their data lacked of variety. They collected data from simulator or database. None of them trained and tested their model on multiple vehicles at multiple places. This may limit the application range of their model. In other words, their model was specific to one certain situation rather than a general situation. We would like to enhance the model ability from these two aspects, this will help the recognition model better understand other road users and it will have a wider application in ADASs.

1.3 Problem Statement

The purpose of this thesis is to develop a maneuver intention recognition system for highway scenarios that allows a vehicle equipped with such system (called the host) to predict whether its preceding vehicle (called the target) will perform a lane keeping or lane changing maneuver up to 1 second after the maneuver starts.

We aim to recognize lane change intention because it is one of the riskiest maneuvers that a driver has to perform on highway. It involves changes in both longitudinal and lateral movements as well as interactions with the surroundings [20]. As an improvement to the current research, we would like to recognize the lane change intentions of the target vehicle (the leading one) from the perspective of the host vehicle (the following one) in a new scenario where these two vehicles are driving behind each other on a highway. Our goal is to successfully recognize lane change intention as soon as possible after the maneuver starts.

On the basis of the result of literature survey, we adopted two methods to build our recognition model. The first one is the Hidden Markov Models. Since it achieved the best accuracy among the literature, we also took it as a benchmark to measure the performance of the other methods. The reasons for this were:

- An HMM is a simple and direct method which can easily describe a driving behavior by segmenting it into several operations.
- Compared to other methods, the HMMs reached the highest recognition rate from study to study. According to the comparison between an HMM and an SVM-BF classifier, the HMM achieved a higher classification rate than the SVM-BF classifier [3].
- Based on the derivation in [21], an HMM also performed quickly because of its few parameters and low complexity.

The second method is the Relevance Vector Machines. The motivations of this were:

- The formulation of an RVM classifier is a probabilistic generalized regression model. This implies that it can be extended to the multiple-class case without the need to train and combine one-vs-all classifiers as an SVM classifier [22].
- Although an RVM has additional computational cost when compared to an SVM, it models a smoother hyperplane and fewer relevant (support) vectors, which result in fast computation during testing. Thus, an RVM is suitable for a real-time classification task.
- The RVMs learn from training data on the basis of Bayesian theory. The learning phase can be adjusted. Therefore, an RVM has high flexibility and high tolerance with uncertainties.

Besides these two methods, we took advantage of both and proposed a new method which combines them together. The proposed one had the same structure of the HMMs but adopted a multi-class RVM classifier to replace a part of the HMMs. Later in this thesis, we simply call it the HMM-RVM method. We compared the performance of three methods with the model performance from the literature. Their recognition results as well as their properties are compared and discussed in Chapter 4. The required data are extracted from the NGSIM database, which is a public database providing the driving parameters of the vehicles on Highway 101 and Interstate 80 in the US [23]. Through filtering and cleansing these data, we obtain feature vectors which are as the role of input during the model's training and testing phase. At the end of this report, we will answer these questions:

- What are the advantage and disadvantage of each method?
- Make a comparison between the models based on accuracy, complexity, flexibility and generalization ability. In general, which model performs best?
- How do our models perform compare with respect to others in the literature?

1.4 Overview

This report is organized as follows: Chapter 2 reviews Hidden Markov Models and Relevance Vector Machines by explaining their principles and decoding the algorithms. Chapter 3 introduces the structure of our model and also elaborates on how the model works step by step. Chapter 4 evaluates the recognition performance of our models and then makes an analysis of them. Chapter 5 presents the conclusion and recommendations for future work.

Chapter 2

Background

In this chapter, we provide the knowledge of the fundamentals of a driver intention recognition system. The first section presents the principle and development of driver intention recognition models. The second section summarizes the algorithms that we choose for this report. We also explain how the algorithms are applied in an intention recognition model.

2.1 Driver Intention Recognition

Michon described driving behaviors using three hierarchical levels: strategical (planning), tactical (maneuvering) and operational (control) [24]. As Figure2.1 shows, the control level is the lowest one, with the shortest time constant. It incorporates the most basic operations like acceleration and deceleration. The maneuvering level contains behaviors like overtaking and lane changing, which usually last a few seconds. The strategic level has a long time constant because it contains general plans, such as route choice. These three levels are top-down influenced. For example, the operations at control level are the results of maneuvers from the previous level.

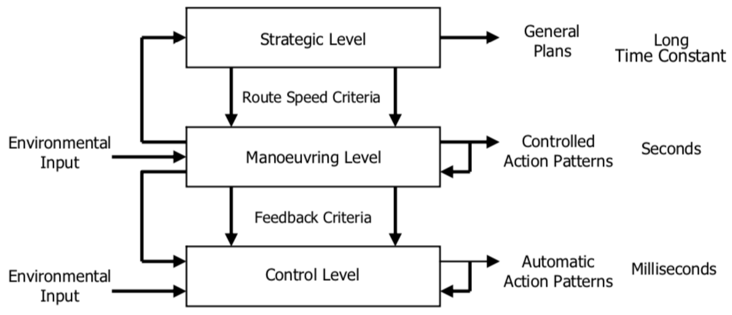


FIGURE 2.1: The hierarchical structure of a driving task [24].

During on-road driving the operations taken by the driver all belong to the control level. The driving parameter detected by sensors or VD systems can reflect the operations. It was proved that a series of control level operations can describe a driving maneuver process [2]. Thus, one can predict the driving maneuver and even the trajectory of the target vehicle by modeling the dynamics of the driving parameters. According to this theory, a driver intention recognition system can be built in order to help intelligent vehicles to infer the maneuver of others. The core of this model is basically an advanced algorithm that manages the sensor data and gives the recognition result.

At the early stage of recognition model research, the mainstream algorithm was the Kalman Filter, which can predict the short-term trajectory of the target vehicle based on current driving data. However, the Kalman Filter loses accuracy significantly as the prediction time increases [25]. Recently, with the help of artificial intelligence, driver intention recognition has been developed by various advanced machine learning techniques. The Hidden Markov Model is the most popular method providing high accuracy and a quick response [18]. The Support Vector Machine is another dominant method with good accuracy and, especially, the lowest false negative rate [9]. Gaussian Process [15], [26], Decision Tree [25], Recurrent Neural Networks [17], [27] are introduced in a driver intention recognition model as well. After recognizing intentions by artificial intelligent methods, a Kalman Filter can be added to predict the long-term trajectory of the target [25]. Usually the long-term trajectory generated from recognized intention is more useful than the only intention.

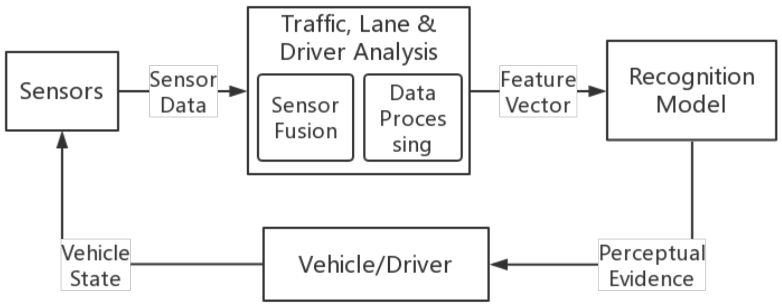


FIGURE 2.2: A Driver Intention Recognition Model in ADAS.

Figure 2.2 illustrates a flow chart of the ADAS contains an intention recognition model. The data are collected from the in-vehicle sensors. After being processed, it becomes a feature vector and is fed to the recognition model. The intention recognition model can recognize the target's intention (whether he is going to change the lane or not) and report it to the driver or the intelligent vehicle.

To evaluate the performance of the lane change intention recognition model, we measure the model from these aspects:

- Accuracy, as well as precision and recall. We expect the model to correctly recognize all the driving behaviors.

- Recognition time. The model should recognize the intention as soon as possible, or the recognition result is not valuable.
- The amount of free hyperparameters. A simple model with few hyperparameters is recommended.
- Generalization ability. The model should have flexibility and self-learning ability, or it is hardly applied in real life.
- Time complexity. The recognition model is a real-time model, so it requires low testing complexity.

2.2 Methodology

We developed three models using Hidden Markov Models (HMM) and Relevance Support Machines (RVM), and then compared their results. In the following we will present the basic principles of them and show how we implement them in a driver intention recognition system.

2.2.1 Hidden Markov Models

An HMM is a model of a finite-states stochastic process whose states cannot be observe directly, but have to be inferred from a set of observed (indirect) variables.

Figure 2.3 shows the structure of a Hidden Markov Model, where node S represents the state, A represents the transition probability matrix and B represents emission probability matrix. Between the states and the observations, there are some underlying relationships which we can learn about from observations, which is termed the training phase. With the knowledge of the underlying relationship, the states of a new process can be interpreted being fed with new observations. This is also called the evaluation phase.

Parameters

An HMM is characterized by the following:
(a) N, the number of states in the model. We denote the state at time t as $q_t \in S \triangleq \{S_1, S_2, \ldots, S_N\}$.
(b) The state transition probability distribution $A = \left[a_{ij}\right], 1 \leqslant i, j \leqslant N$ where

$$a_{ij} = P(q_{t+1} = S_j | q_t = S_i). \tag{2.1}$$

(c) The emission probability matrix in state j, $B = \left[b_j(\cdot)\right]$.
When the observation symbol is distinct, we use M to denote the number of distinct

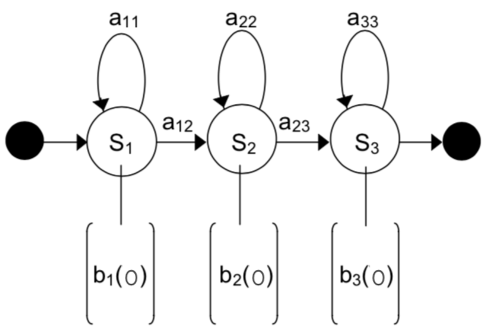


FIGURE 2.3: The structure of a Hidden Markov Model [28].
It shows a process transits among three different states. S represents the states of the process, $A = [a_{ij}]$ represents the transition probabilities between states, $B = [b_j(k)]$ represents the emission probabilities in states.

observation symbols per state. The observation at time t is $o_t \in V \triangleq \{V_1, V_2, \dots, V_M\}$, where V denotes the observation symbols. Hence, the emission probability in state j is represented by

$$b_j(o_t) = P(o_t = V_k | q_t = S_j), 1 \leqslant j \leqslant N, 1 \leqslant k \leqslant M. \tag{2.2}$$

When the observation symbol is continuous, the observation sequence is represented by $O \triangleq \{o_1, o_2, \dots, o_T\}$, where T is the length of the sequential observation. We define all the observation symbols correspond to state j as O_j, where $O_j = \{o_t \in O | q_t = S_j\}$. The distribution of O_j is considered to be a mixture of M multivariate Gaussian distribution where

$$O_j \sim \sum_{l=1}^{M} c_{jl} \mathcal{N}(\boldsymbol{\mu}_{jl}, \boldsymbol{\Sigma}_{jl}). \tag{2.3}$$

Here, c_{jl} denotes the coefficient of the l^{th} mixture Gaussian in state j, $\boldsymbol{\mu}_{jl}$ and $\boldsymbol{\Sigma}_{jl}$ are the mean and covariance of the l^{th} mixture Gaussian in state j. The emission probability $b_j(o_t)$ is represented by the probability density of the mixture Gaussian distribution, where

$$b_j(o_t) = f(o_t | O_j). \tag{2.4}$$

(d) The initial state distribution $\pi = \{\pi_i\}$ where

$$\pi_i = P(q_1 = S_i), 1 \leqslant i \leqslant N. \tag{2.5}$$

For convenience, we use the specification of the three probability measures

$$\lambda = (A, B, \pi) \tag{2.6}$$

to indicate the complete parameter set of the model [21], [29].

Algorithm

In a driver intention recognition system, an HMM meets three challenges, which are:

- Training Problem: How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$?
- Model Selection: Given the observation sequence $O = O_1 O_2 \dots O_T$, and a model $\lambda = (A, B, \pi)$, how do we compute $P(O|\lambda)$, the probability of the observation sequence, given the model?
- State Estimation: Given a model $\lambda = (A, B, \pi)$ and an observation sequence $O = O_1 O_2 \dots O_T$, how do we choose a corresponding state sequence $Q = q_1 q_2 \dots q_T$?

The first challenge is to find an HMM model to best describe how the observations are related to the states. In other words, we find a model to tell us what kind of influence a driving action has on vehicle parameters. It is also known as a training problem. It is the procedure of calculating the model parameters (namely, state transition and observation symbol probability distribution and initial state distribution) to maximize the probability of the observation sequence given the model $P(O|\lambda)$ [5]. The Baum-Welch method, an iterative procedure, is used for solving this problem.

The last two challenges are evaluation problems. The second challenge computes the probability that the model produced the observed sequence. It can also be considered to measure how well a given model explains the given observations. The third challenge is to solve a hidden part of the HMM model. When considering an HMM of a driver intention recognition system, this problem can be defined as finding the corresponding states (driving actions) based on an HMM model (relationships between actions and vehicle data) and the observations (vehicle data). Unfortunately, the solution cannot be exact, so we usually use the Viterbi algorithm, which aims to find the single best state sequence Q, to get a solution which is the best possible one [28]. All the formulations of evaluation solution, the Viterbi algorithm, and the Baum-Welch method can be found in [28].

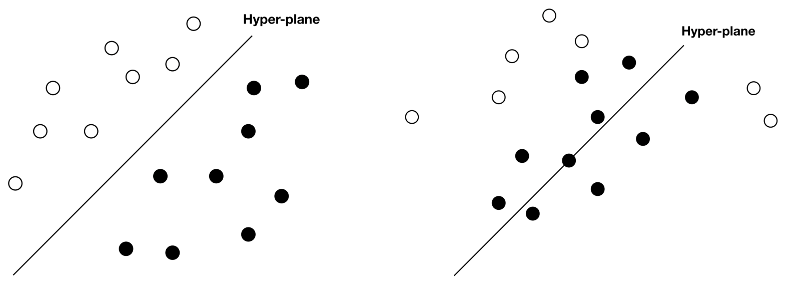
Application

For a driving car, each driving action is a state. For example, the state set S could be $S \triangleq \{\text{acceleration}, \text{deceleration}, \text{steer right}, \dots\}$. The observations O are the driving parameters. During the training phase, the HMM finds a set of optimized parameters to describe the relationship between driving parameters and the states. After that, given a new sequence of driving parameters, a well-trained HMM can output the corresponding actions, as well as how well the model explain the parameters and the states.

In the case of a lane change maneuver, about three states make up the whole procedure. We define the state set $S \triangleq \{\text{lane keeping, steer, steer back}\}$. In the recognition model, we trained an HMM for each driving behavior, so there are three parallel HMMs represent three driving behaviors. The input data are collected by vehicle sensors, pre-processed and labeled by states. After the training phase, given all three HMMs a new sequential data, each model estimates the probability of the data given the corresponding model. The highest probability means the driving parameter is most likely matches this behavior.

2.2.2 Relevance Vector Machines

The Relevance Vector Machine (RVM) can generate a linear model to describe a data set, which is similar to the Support Vector Machines (SVM). Both of them are state-of-art techniques for regression and classification, having wide application range with a kernel representation [22]. The RVM models have an identical functional form to the Support Vector Machines, but output probabilistic results on the basis of Bayesian learning. It can be viewed as an probabilistic upgrade of SVM. Thus, we firstly elaborate on the SVM and then the RVM.



(A) (B)

FIGURE 2.4: SVM techniques used in classification and regression tasks.
(a): The SVM classifier trains a hyperplane to separate two classes of samples in a classification task. (b): The SVM regression model trains a hyperplane to fit the samples in a regression task.

The Support Vector algorithm is a method to categorize samples from different classes. Figure 2.4a and 2.4b demonstrate how SVMs are applied on a classification or a regression task. The solid points and the hollow points represent samples with different labels. In a classification task, an SVM classifier finds a hyperplane which can best separate one class of samples from another [30]. In a regression task, an SVM regression model finds a hyperplane which can best fit all the 'True-labeled' samples. In Support Vector Machines, a sample is a vector whose each element represents a feature of it. The number of features, which is also known as the dimensionality of the data, can be very high in an SVM model. This is because the training algorithm of SVMs can automatically select the most distinct features and abandon the others.

However, SVM cannot make probabilistic predictions. In a regression task the SVM outputs a point estimate, and in a classification task, a 'hard' binary decision which only contains '-1' and '+1'. Ideally, we desire to make an estimation considering uncertainties in our prediction. To overcome this shortcoming, the RVM method was introduced. It is a probabilistic sparse kernel model identical in functional form to the SVM [22].

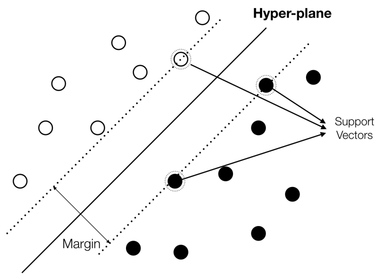
Parameters

Figure 2.5a illustrates the simplest SVM, a linear two-class classification. The solid points and the hollow points are from two classes, which are labeled by '+1' and '-1' respectively. The decision function f takes the form of

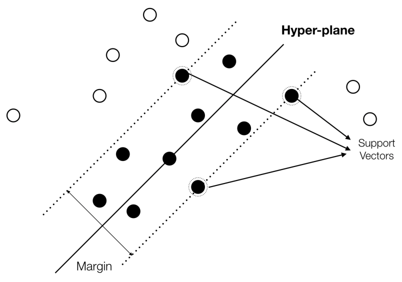
$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b, \mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}, \tag{2.7}$$

where $\mathbf{x}$ [1] can be any sample (driving data obtained from vehicles) with D features, $\mathbf{w} = (w_1, w_2, \ldots, w_D)$ is the weights, and b is the bias. The hyperplane that separates the two-class samples is represented by

$$\mathbf{w} \cdot \mathbf{x} + b = 0, \mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}. \tag{2.8}$$



(A)



(B)

FIGURE 2.5: How SVMs work on different tasks.
(a): An SVM classifier finds a hyperplane which has the largest margin to the closest samples (technically called 'support vectors') on both sides. (b): An SVM regression model finds a hyperplane which has the narrowest margin to the furthest samples (support vectors).

Similarly, it can be applied to a regression problem. As Figure 2.5b shows, a hyperplane is trained to fits all the solid points (generally the samples which are labeled by '1'). The linear function f and the hyperplane take identical forms to a classification problem, which are shown in Equation 2.7 and 2.8.

So far we introduced the simplest case. However, usually training data are not linearly separable in their input space. Figure 2.6 shows the samples cannot be linearly

[1]Throughout Section2.2.2, m denotes scalar, $\mathbf{m}$ vector and $\mathbf{M}$ a matrix. Given the Matrix $\mathbf{M}$, $\mathbf{m}_i$ denotes the row vector from the i^{th} row of $\mathbf{M}$ unless stated otherwise.

separated in the input space. In order to solve this, a nonlinear map ϕ is introduced, then the samples are mapped into a higher dimensional space $\mathscr{H}$ (called feature space):

$$\phi : \mathbb{R}^D \mapsto \mathscr{H}. \tag{2.9}$$

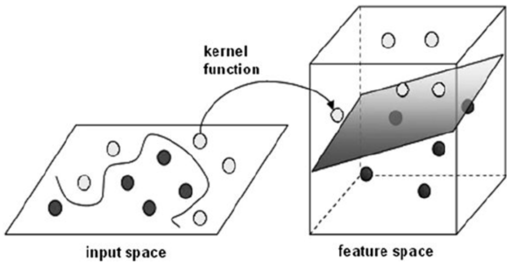


FIGURE 2.6: A demonstration of kernel function [31]

We define a "kernel function" K represents the dot products of the mapped inputs

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j). \tag{2.10}$$

By choosing a suitable kernel function, the data which are non-separable in the input space can become separable in the feature space, because it is easier to find a linear hyperplane to separate the data in a higher dimensional space. When the dimension exceeds the number of samples, it definitely can find a linear solution. Detailed information of the training algorithm and kernel functions are presented in Appendix A.

Lastly, an SVM model takes the form:

$$y_i = y(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i + b, \mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}. \tag{2.11}$$

In order to obtain a probability estimation of samples based on SVM, Michael E. Tippling proposed the RVM framework[22]. In an RVM regression model, given a data set $\mathbf{X} = (\mathbf{x}_1^T, \ldots, \mathbf{x}_N^T)^T$ we write the true labels as a vector $\mathbf{t} = (t_1, \ldots, t_N)$, and express it as the sum of an approximation vector $\mathbf{y} = (y(\mathbf{x}_1), \ldots, y(\mathbf{x}_N))$ and an error vector $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_N)$:

$$\mathbf{t} = \mathbf{y} + \boldsymbol{\epsilon}, \tag{2.12}$$

where

$$t_i = y(\mathbf{x}_i) + \epsilon_i = \mathbf{w} \cdot \mathbf{x}_i + b + \epsilon_i. \tag{2.13}$$

The errors $\boldsymbol{\epsilon}$ are conventional assumed as independent zero-mean Gaussian distributions, with variance σ^2: so $p(\boldsymbol{\epsilon}) = \prod_{i=1}^{N} \mathcal{N}(\epsilon_i|0, \sigma^2)$. The parameter σ^2 can be set in advance if known but more usually would be estimated from the data.

To extend this model into a non-linear case, a kernel function K is applied to the input samples $\mathbf{X}$. Appendix A.2 proves that only dot products of samples are used in the training phase. In an RVM model, kernel functions are used to replace the original samples, and also the weights $\mathbf{w}$ are changed to a another weight vector $\tilde{\mathbf{w}}$. One sample is represented by $\mathbf{k}_i$, whose elements equal to $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j), j = 1, \ldots, N$.

The big kernel matrix $\mathbf{K} = (\mathbf{k}_1^T, \ldots, \mathbf{k}_N^T)^T$ is composed by all new sample vectors. Thus, an RVM takes the form:

$$t_i = y(\mathbf{x}_i) + \epsilon_i = \tilde{\mathbf{w}} \cdot \mathbf{k}_i + \epsilon_i[32]. \tag{2.14}$$

Although an RVM model has identical form to an SVM model, they use completely different ways to learn. Bayesian interference helps an RVM model to formulate the likelihood and select hyper-parameters. In order to conveniently formulate all the probabilities during the training phase, an auxiliary parameter $\mathbf{A} \in \mathbb{R}^{N \times N}$ is introduced to the RVM algorithm. $\mathbf{A}$ is a diagonal matrix whose element α_i restrict the corresponding weights $\tilde{w}_i$ from $\tilde{\mathbf{w}}$. It is set that weight $\tilde{w}_i$ follows a Gaussian distribution with zero mean and variance α_i. Taking advantage of this definition, probability $P(\tilde{\mathbf{w}}|\boldsymbol{\alpha})$ as well as probability $P(\mathbf{t}|\boldsymbol{\alpha}, \sigma_2)$ can be expressed as multivariate Gaussian distributions. Figure 2.7 shows the hyper-parameters of the model and their relationships.

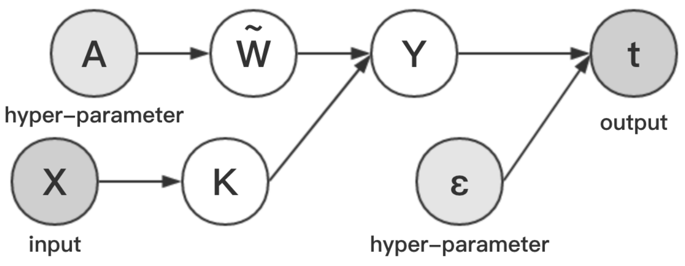


FIGURE 2.7: Plates diagram of an RVM model [33].

Algorithm

According to Equation 2.14, the size of the training data is changed to N through a kernel function. Therefore, the numbers of the entries of weight $\tilde{\mathbf{w}}$ and auxiliary parameter $\mathbf{A}$ should also equal to the number of samples N. However, most of the vectors of kernel $\mathbf{K}$ make limit contribution to classification results. In order to reduce computational cost and avoid the 'curse of dimensionality', feature selection is applied during the training phase. After the training phase, most of the features are pruned and only few of them are kept. The number of the features kept in the model is denoted as M, which is much smaller than the number of samples N. Training an RVM model is basically a process of evaluating each sample's contribution, selecting contributed samples and estimating the corresponding weight $\tilde{\mathbf{w}}$. The following roughly interpret the training algorithm of an RVM model.

In an RVM regression model, the prediction $\mathbf{t}$ follows a multivariate Gaussian distribution $\mathcal{N}(\mathbf{y}, \Sigma_\epsilon)$. Σ_ϵ is a diagonal covariance matrix whose elements all equal to σ^2 because it is assumed the error follows independent Gaussian distributions. The conditional probability of the prediction **t** is written as:

$$p(\mathbf{t}|\tilde{\mathbf{w}},\sigma^2) = (2\pi\sigma^2)^{-N/2} \exp\left\{-\frac{\|\mathbf{t}-\mathbf{y}\|^2}{2\sigma^2}\right\}. \tag{2.15}$$

The weights $\tilde{w}$ from $\tilde{\mathbf{w}}$ follow a standard normal distribution with zero mean and variance α_i^{-1}:

$$p(\tilde{\mathbf{w}}|\boldsymbol{\alpha}) = (2\pi)^{-M/2} \prod_{m=1}^{M} \alpha_m^{1/2} \exp\left(-\frac{\alpha_m \omega_m^2}{2}\right), \tag{2.16}$$

where M denotes how many rows of **K** are kept in the model. In a RVM model, most of the row vectors of **K** are removed to achieve sparsity, while only M vectors remained. These vectors are also known as 'the basis vectors'. The weight $\tilde{\mathbf{w}}$ is a $1 \times M$ vector. The value of M depends on the model, and is much less than the number of samples N.

The probability we are interested during the training phase is $p(\mathbf{t}|\boldsymbol{\alpha},\sigma^2)$, which is the marginal likelihood of the prediction given hyper-parameters. Based on Equation 2.14, prior knowledge of the hyper-parameters and the property of multivariate Gaussian distribution, we can infer that the prediction **t** also follows a multivariate Gaussian distribution with zero mean and $(\mathbf{K}\mathbf{A}^{-1}\mathbf{K}^T + \sigma^2\mathbf{I})$ covariance. We define **C** to represent $\mathbf{K}\mathbf{A}^{-1}\mathbf{K}^T + \sigma^2\mathbf{I}$ for convenient. The marginal likelihood of the prediction can be written as:

$$p(\mathbf{t}|\boldsymbol{\alpha},\sigma^2) = (2\pi)^{-N/2}(|\mathbf{C}|)^{-1/2} \exp\left(-\frac{1}{2}\mathbf{t}^T\mathbf{C}^{-1}\mathbf{t}\right). \tag{2.17}$$

The sparse Bayesian learning is formulated as the (local) maximization with respect to $\boldsymbol{\alpha}$ of the marginal likelihood, or equivalently, its logarithm $\mathcal{L}(\boldsymbol{\alpha})$:

$$\mathcal{L}(\boldsymbol{\alpha}) = \log p(\mathbf{t}|\boldsymbol{\alpha},\sigma^2) = -\frac{1}{2}\left[N\log 2\pi + \log|\mathbf{C}| + \mathbf{t}^T\mathbf{C}^{-1}\mathbf{t}\right]. \tag{2.18}$$

If we decompose the vector $\boldsymbol{\alpha}$ into a single prior α_i and the vector of priors without α_i, which is also called $\boldsymbol{\alpha}_{-i}$, we can write Equation 2.18 as:

$$\mathcal{L}(\boldsymbol{\alpha}) = \mathcal{L}(\boldsymbol{\alpha}_{-i}) + \frac{1}{2}\left[\log\alpha_i - \log(\alpha_i + s_i) + \frac{q_i^2}{\alpha_i + s_i}\right], \tag{2.19}$$

where we have the 'sparsity factor' s_i and the 'quality factor' q_i. They measure the quantities of sparsity and quality the i^{th} vector of **K** contributes. They are defined as:

$$s_i \triangleq \mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{k}_i^T,\ and\ q_i \triangleq \mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{t}, \tag{2.20}$$

where $\mathbf{C}_{-i}$ represents the quantity C without the contribution of the ith vector of **K**.

To maximize the marginal likelihood $\mathcal{L}(\boldsymbol{\alpha})$, we calculate the derivative of $\mathcal{L}(\boldsymbol{\alpha})$ with respect to α_i and find the α_i which can maximize the $\mathcal{L}(\boldsymbol{\alpha})$:

$$\begin{array}{ll} \alpha_i = \frac{s_i^2}{q_i^2 - s_i}, & if q_i^2 > s_i, \\ \alpha_i = \infty, & if q_i^2 \leqslant s_i. \end{array} \tag{2.21}$$

The posterior parameter distribution conditioned on the data is given by combining the likelihood Equation 2.15, 2.16 and prior within Bayes' rule:

$$p(\tilde{\mathbf{w}}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2) = p(\mathbf{t}|\tilde{\mathbf{w}}, \sigma^2) p(\tilde{\mathbf{w}}|\boldsymbol{\alpha}) / p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2), \tag{2.22}$$

which can be expressed as a Gaussian $\mathcal{N}(\mu, \Sigma)$ with

$$\boldsymbol{\Sigma} = (\mathbf{A} + \sigma^{-2}\mathbf{K}\mathbf{K}^T)^{-1}, \boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\mathbf{K}\mathbf{t}. \tag{2.23}$$

An optimized weight $\tilde{\mathbf{w}}$ is obtained by maximizing the posterior

$$\tilde{\mathbf{w}} = \arg\max_{\tilde{\mathbf{w}}} p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2). \tag{2.24}$$

Therefore, it can be updated by

$$\tilde{\mathbf{w}} = \sigma^{-2}(\mathbf{A} + \sigma^{-2}\mathbf{K}\mathbf{K}^T)^{-1}\mathbf{K}\mathbf{t} \tag{2.25}$$

Lastly, the noise level σ^2 is updated by

$$\sigma^2 = \|\mathbf{t} - \mathbf{y}\|^2 / (N - M + \sum_m \alpha_m \Sigma_{mm})^2. \tag{2.26}$$

At the beginning of a training phase, the variance σ^2 is estimated from data and all the priors α are set to infinity. Then a single basis vector $\mathbf{k}_i$ is selected, the related prior α_i, sparsity factor s_i and quality factor q_i are calculated. Based on the calculation, the weights $\tilde{\mathbf{w}}$ and noise σ^2 are updated. After that, the model will iteratively find other basis vectors and update the hyper-parameters until it meets its convergence criterion.

Application

In a driving recognition case, the input data is derived from in-vehicle sensors. All the sensor data need to be selected and pre-processed to become a time sequence of data. The processed training data should be labeled according to if it contains a specific driving behavior or not. By giving these training data, a three-class RVM (combine three one-vs-all RVMs) is trained to fit a regression model for each class, *i.e.* each driving maneuver. Hence, we let the driving behavior recognition problem

[2]The detailed steps of the maximization of $\mathcal{L}(\boldsymbol{\alpha})$ and the update of σ^2 will be elaborated on in Appendix B.

become a data classification problem. When we do the recognition, the RVM model will output a probability of the specific driving behavior happens at the moment.

Chapter 3

Model Structure

In this chapter, we elaborate on how our models work step by step. It starts from the data acquisition, which includes data cleansing, pre-processing, labeling and re-scaling, then goes through the three types of models. We introduce how to adapt the HMMs and the RVMs to a driver intention recognition model as well as a combined model which can overcome the limitations of the first two models.

3.1 Data Acquisition

Figure 3.1 illustrates all the data pre-processing steps in a flow chart. In the following we will explain the approach and the motivation of each step.

3.1.1 NGSIM Database

Next Generation Simulation (NGSIM) Vehicle Trajectories and Supporting Data [23] is the database we used for our model. It is provided by the US department of transportation. Data was collected through a network of synchronized digital video cameras and transcribed to vehicle trajectories. This vehicle trajectory data provide the precise location of each vehicle within the study area every one-tenth of a second, resulting in detailed lane positions and locations relative to other vehicles. Researchers for the Next Generation Simulation (NGSIM) program collected detailed vehicle trajectory data on various highways. By researching the database, we used the data which are collected on southbound US 101 and eastbound I-80 to extract the training and testing data.

Figure 3.2a and 3.2b illustrate the locations of the trajectory data we used for the recognition model. Figure 3.2a depicts the southbound direction of Highway 101 (Hollywood Freeway) in Los Angeles, California. The study area is approximately 640 meters in length, with five mainline lanes, one on-ramp and one off-ramp throughout the section. The trajectory data were collected from 7:50 AM to 8:05 AM in 2005 [34]. Figure 3.2b depicts the northbound direction of Interstate 80 in Emeryville,

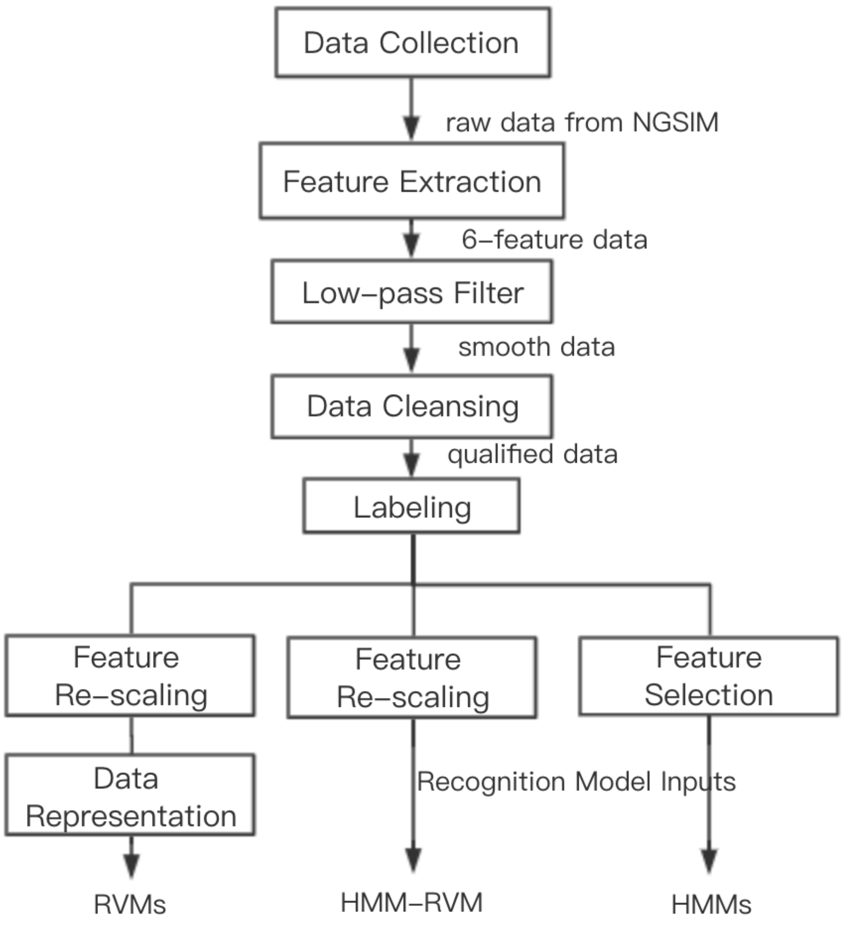
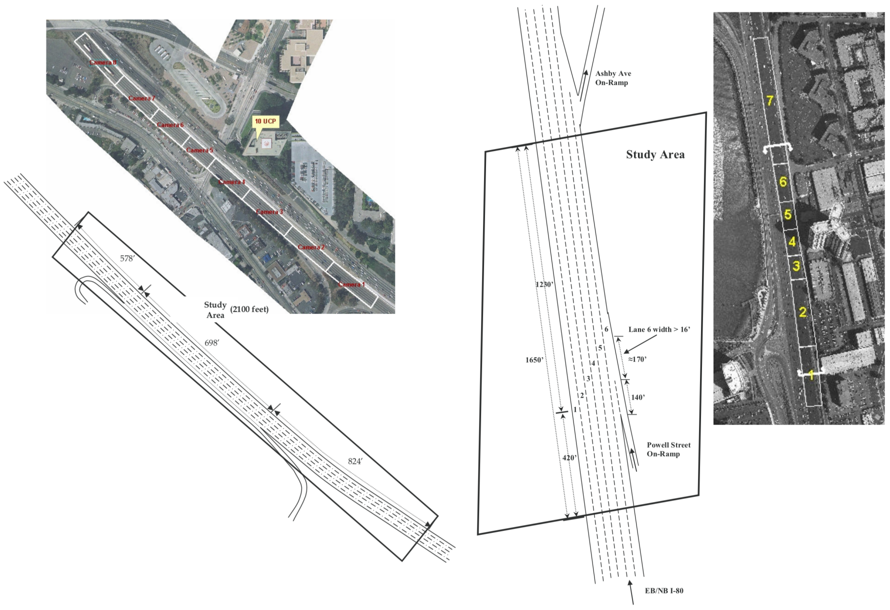


FIGURE 3.1: Flow Chart of the Data Acquisition.

California. The area is approximately 500 meters in length, with 6 lanes and one on-ramp. The off-ramp is located at the downstream of the study area. The data were collected during 4:00 PM and 5:30 PM in 2005 [35]. Given the NGSIM database, we went through a data pre-processing phase to extract specific parameters we need for our model. Because the NGSIM database was collected by camera and processed into trajectory data, it contains high level noise. Thus, a series of pre-processing and cleansing is essential.

3.1.2 Data Pre-processing and Cleansing

Since the NGSIM database gives the specific trajectory information about vehicle locations, we can easily find lane change behaviors and collect the driving information of the corresponding vehicles. All the truck data were removed and also the lane change behaviors happened on on-ramp or off-ramp were not considered, because we were only interested in the lane change behaviors happened between passenger cars on main-lane. Based on the trajectory data NGSIM provided, by applying



(A) (B)

FIGURE 3.2: (A): Study area schematic on Highway 101 [34]. (B): Study area schematic on Interstate 80 [35].

derivations and low-pass filters [1], we extracted the driving data including 6 features: absolute lateral and longitudinal velocities of preceding vehicles (also named target vehicles), absolute lateral and longitudinal accelerations of preceding vehicles, relative heading angles and relative yaw rates. These six features can fully describe the status of a vehicle during the driving. The importance of each feature will be tested and discussed in the following chapter. Figure 3.3 depicts how we extracted the lane change information from the NGSIM database.

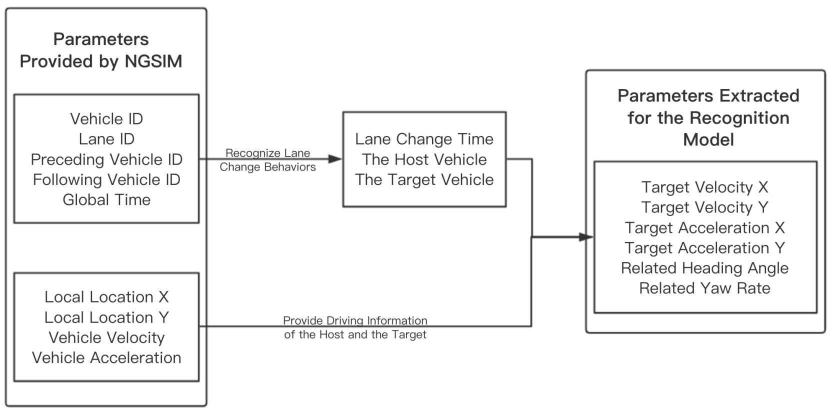


FIGURE 3.3: Diagram of Data Pre-processing.

So far, only part of the pre-processed data are valuable data. Figure 3.4 illustrates

[1]A first-order Butterworth filter is used to remove noise from the NGSIM database, and moving average filters are used to smooth the driving parameters. The code is shown in Appendix C.

a set of pre-processed parameters which represent a 15-second driving sequence including a lane change behavior. We searched all the changes in the parameters of 'Lane ID', which is considered to be the moment when the target vehicle crosses the lane marking. To completely cover the whole lane change process, we took 10 seconds before and 5 seconds after this moment as a preliminary lane change process. From the bottom-left figure, we can tall that the heading angle has a gradual increase from the time of 50 ms until the lane change moment. However, due to the low quality of the data, not all the parameter sets we extracted are as good as this one. For example, Figure 3.5a shows a set which has unexpected heading angles. We inferred this is caused by incorrect following vehicle information. Figure 3.5b shows a set which has two peaks of heading angles. This is because the driver failed to change the lane on his first try. Due to technical restrictions on labeling, this will cause extra troubles during the segmentation and labeling phase, so we also abandon this kind of data. These unexpected data will disturb the model's training phase, so we have to remove them from our dataset. This is called a cleansing phase.

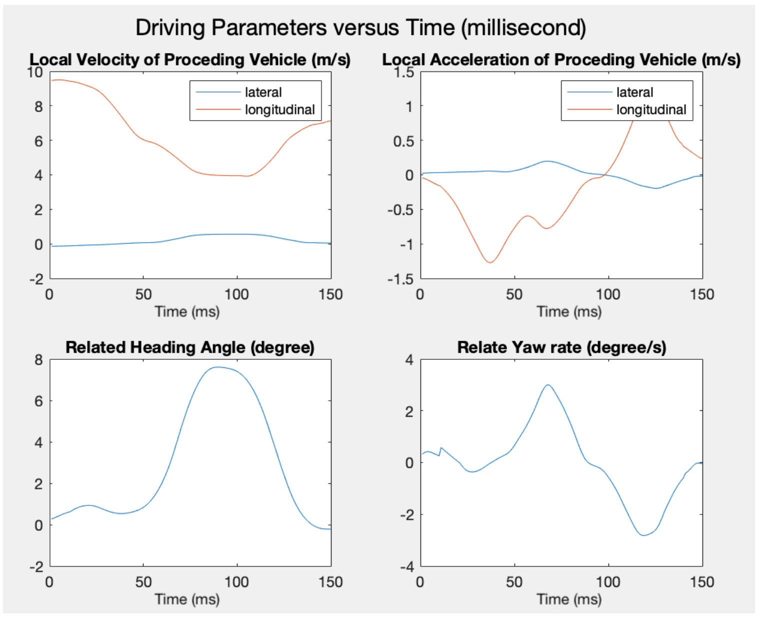
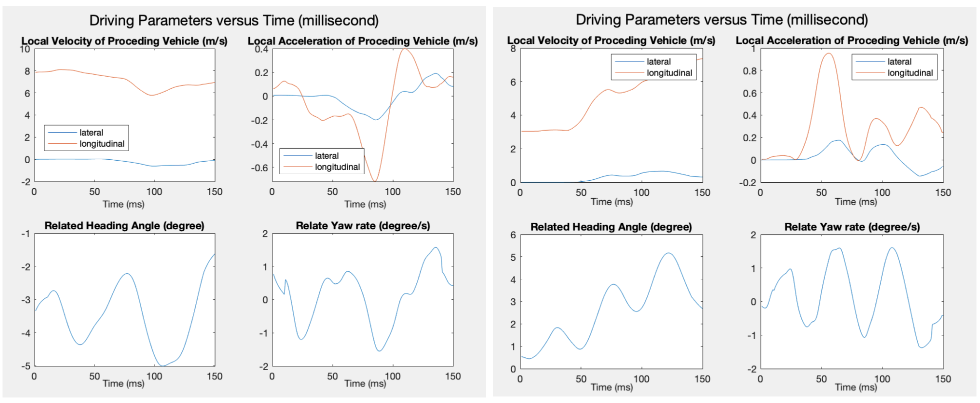


FIGURE 3.4: Valuable parameters extracted from the NGSIM.
In the first-row figures, the red lines represent the longitudinal components of the velocity and the acceleration, and the blue lines represent the lateral components. The lane change moment happened at the time of 100 ms.

We removed defective sequences based on the value of heading angle, because heading angle is the most obvious parameter indicate a lane change behavior. There are two conditions for an eligible set: (1) Before the lane change behavior, the target vehicle must keep driving in its lane for a while. So we required the heading angle should be around zero for at least 1 second before it arises. (2) The heading angle should decrease in a reasonable rate after it reaches the peak, which means it should take about 4 seconds to go back to zero value. It can guarantee us the parameters describe only one lane change behavior and there is no failed tries and other consecutive behaviors. Any sequences can not satisfy these two conditions were removed from our dataset. For the eligible ones, we trimmed them to an appropriate length



(A) (B)

FIGURE 3.5: Defective parameters extracted from the NGSIM.
(A): This set of parameters has unexpected heading angles which is caused by incorrect vehicle information. (B): This set of parameters has more than one peak of heading angles, which will lead unexpected troubles in the following phases.

in order to obtain a complete but non redundant lane change sequence. Figure 3.6 shows a sample of the selected and trimmed parameter sets. The length of the lane change sequence was slightly shorten from 150 ms to around 110 ms.

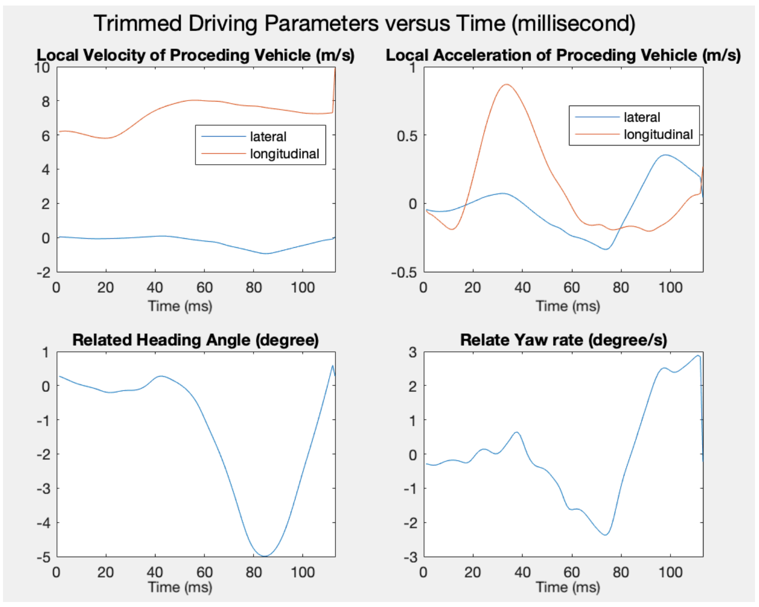


FIGURE 3.6: Illustration of a lane change sequence after cleansing and trimming.

For the lane keeping samples, we extracted parameters from the NGSIM database according to two conditions: (a) The heading angle should be within a small range during the whole chosen sequence. Here we defined the sequence lasts 150 ms, and the range of the heading angle is within $(0.25, +0.25)$ degree. (b) No lane change behaviors occurs 20 seconds before or after this chosen sequence, or the parameters might be influenced by the lane change behavior. Figure 3.7 illustrates the driving parameters of a lane keeping behavior.

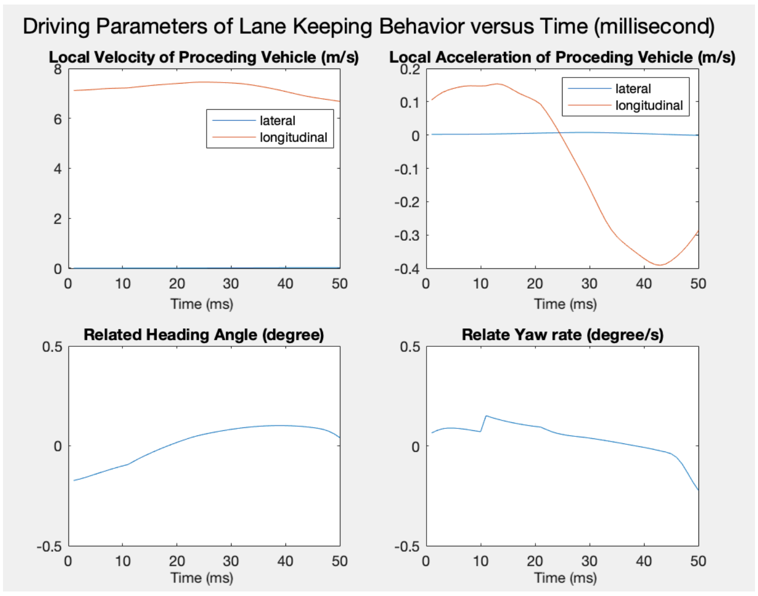


FIGURE 3.7: Illustration of a lane keeping sequence.

3.1.3 Labeling

As Figure 3.6 shows, a lane change behavior was roughly trimmed to a 110 ms long process. The sampling frequency of our data is 10 Hz. Thus, about 110 samples describe a lane change sequence. For each sample, we have numerical values of 6 features (shown in Figure 3.3). We labeled each sample according to these feature values. For the lane keeping behaviors, we only have one label for all samples. They are all labeled by 'lane keeping' label. For the lane change behaviors, besides the 'lane keeping' label we have another two labels 'steering' and 'steering back'. Each sample is labeled by one of these three labels mainly according to the values of heading angle and yaw rate. Thus, a lane change behavior is segmented into three parts: lane keeping (normal driving), steering and steering back. Figure 3.8 illustrates the relation between the heading angle and the label during a lane change sequence. A sample is labeled by 'lane keeping' (blue area) when the heading angle is around zero. All the samples belong to the period since the heading angle starts to increase until it reaches the peak are labeled by 'steering' (yellow area). After that, the samples with a decreasing heading angle are labeled by 'steering back' (red area).

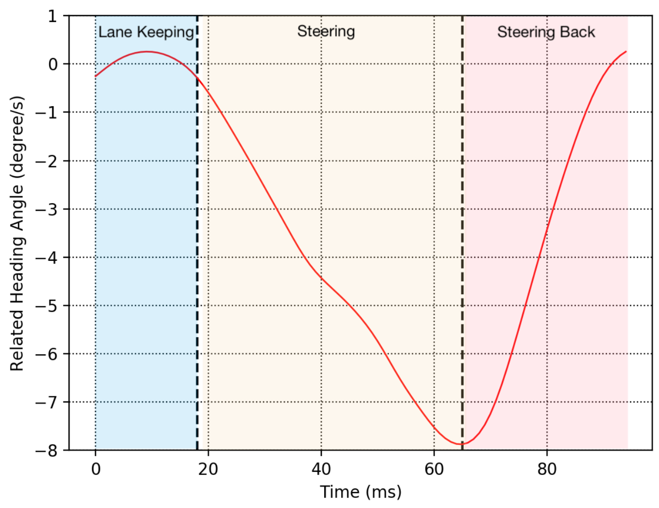


FIGURE 3.8: Illustration of A Lane Change Sequence After labeling. A sample is labeled by 'lane keeping' (blue area) when the heading angle is around zero. All the samples belong to the period since the heading angle starts to increase until it reaches the peak are labeled by 'steering' (yellow area). After that, the samples with a decreasing heading angle are labeled by 'steering back' (red area).

3.1.4 Feature Rescaling

Before the data can be fed to a model for training or testing, feature rescaling is the last but very crucial step. The contribution of a feature depends heavily on its variability relative to others, which results that the feature in greater numeric ranges dominates those in smaller numeric ranges. For example, if one feature has a range of 0 to 1, while another one has a range of 0 to 1,000,000, then the contribution of the first feature will be swamped by the second one. Therefore, it is essential to rescale the features so that their variability reflects their importance, or at least is not in inverse relation to their importance. When the relative importance of the features is unknown, it is common to rescale each feature to the same range or the same standard deviation. Otherwise, rescaling the features to let the more important ones have larger variances or ranges may help. Besides, feature rescaling also can avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors, large feature values might cause numerical problems [36].

We have known that 'heading angle' and 'yaw rate' are the most important features among the others, and the longitudinal (Y-axis) velocity and acceleration are the least important ones. Fortunately, the important feature values have already centered around zero, so we need not to do the standardizing. By trying different rescaling coefficients, we found a best combination for our features, which will be discussed in Chapter 4.

3.2 HMMs Method for Intention Recognition

Hidden Markov Models have been popular on the topic of speech recognition due to its simple structure and clear performance [37]. As was introduced in Chapter 2, the concepts of 'state' and observation' in a Hidden Markov Model perfectly fits to driving actions such as a lane change behavior. Hence HMMs have been introduced to recognize driving behaviors as well. We value the HMMs' properties of processing sequential data, statistic foundation and efficient algorithm. However, we do not want the states are defined by a black box algorithm rather than us, because this leads to a big amount of undetermined parameters. We built our recognition model by means of taking advantage of the good properties and also reducing the negative ones.

3.2.1 HMM Training

Generally HMMs is a unsupervised learning method, which means it does not need any labels of the training data. With the help of Baum-Welch methods, the model can iteratively re-estimate the its parameters until it finds a local maximum. However, the amount of our training samples is extremely limited (only 363 samples in total), a supervised learning method rather than unsupervised one may help the model reach better accuracy. Thus, we would like to estimate the model parameters by labels. From the Chapter 2.2 we have known that an HMM can be indicated by three measures: state transition probability matrix A, emission probability matrix B and the initial state distribution π. Since every sample of the training set has been labeled by state, we define $\gamma_t(i)$ as the number of transitions from state S_i at time t, and define $\xi_t(i,j)$ as the number of transitions from state s_i to state s_j from time t until time $t+1$, then $A = \{a_{ij}\}$ and $\pi = \{\pi_i\}$ can be directly computed by

$$\begin{aligned} \pi_i &= \frac{\gamma_0(i)}{\sum_i \gamma_0(i)}, \\ a_{ij} &= \frac{\sum_{t=1}^{t=T-1} \xi_t(i,j)}{\sum_{t=1}^{t=T-1} \gamma_t(i)}. \end{aligned} \tag{3.1}$$

Here in a lane change intention recognition, the 'observations' the model collected are driving parameters such as velocities and accelerations. Their values are continuous, so the emission probability matrix $B = \{b_j(\cdot)\}$ in each state is assumed to be a multivariate Gaussian distribution. For each state j, we have the mean $\boldsymbol{\mu}_j$ and the covariance matrix $\boldsymbol{\Sigma}_j$. The emission probability of the observation o_t in state j at time t is expressed by:

$$b_j(o_t) = f(o_t|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j). \tag{3.2}$$

Hence, learning the emission probability becomes a problem of learning the mean $\boldsymbol{\mu}_j$ and the covariance matrix $\boldsymbol{\Sigma}_j$ of a multivariate Gaussian distribution. Based on the observations, which are the features of the training data, we can easily obtain the mean and the variance of each feature as well as the covariance matrix of the whole training data.

This is a simple and direct way to estimate model parameters from the observations and the labels. Compared to the methods which estimate model parameters in a unsupervised way [2], our model have prior knowledge about the real label (state) of the each training sample, the only thing it should learn is the relationship between the observations (inputs) and labels. Supervised learning is a reliable and easy way to find the hyper-parameters in order to achieve a maximized $P(O|\lambda)$. For a small training set, it is crucial to reduce the complexity of the model. Because of this, we made some small changes to the HMM training phase in order to simplify it: (1) sacrificed the model's abilities of learning and generalization to make it become a supervised learning model (2) chose one multivariate Gaussian distribution instead of a mixture of multiple one as the observation symbol probability distribution, so the amount of hyper-parameters is significantly reduced.

We have introduced the principle of HMM training in Chapter 2.2. Next we will explain how we train the HMMs to become a lane change intention recognition model. Due to the unique structure of the states and their inner relations, an HMM is specific to one certain driving behavior. This means if we want to build a classification model we have to train multiple HMMs. In our recognition model, there are three HMMs describe three different behaviors. Before the training phase, we separated driving sequences which belong to different behaviors and fed them to the corresponding HMM. These three trained HMMs are parallel and equally contribute to the recognition.

3.2.2 Forward Procedure for HMM Testing

In the testing phase, given a sequence of driving parameters $O \triangleq \{o_1, o_2, \cdots, o_T\}$, a HMM is expected to output the probability of the observation sequence given this model λ, *i.e.* $P(O|\lambda)$. This probability shows how well this driving sequence fits the model. For example, if the trained HMM describes a left lane change behavior, the probability $P(O|\lambda)$ means how likely a left lane change behavior occurs during the sequence O.

The best algorithm for a HMM to make this prediction is called 'Forward Procedure'. A forward variable $\alpha_t(i)$ is introduced in this algorithm. It is defined as the probability of the partial observation sequence $\{o_1, o_2 \cdots o_t\}$ and state S_i at time t, given the model λ, *i.e.*

$$\alpha_t(i) = P(\{o_1, o_2 \cdots o_t\}, q_t = S_i|\lambda). \tag{3.3}$$

At the first time step of a sequence, the forward variable $\alpha_1(i)$ is initialized by

$$\alpha_1(i) = \pi_i b_i(o_1). \tag{3.4}$$

[2]Iterative procedures such as the EM(expectation-maximization) method or gradient techniques can solve this problem. They find a locally maximized probability $P(O|\lambda)$ and the corresponding model λ [21].

Then given a new observation o_{t+1} and the state transition probability A, the forward variable can be iteratively calculated in each time step as follow:

$$\alpha_{t+1}(j) = \left[\sum_i \alpha_t(i) a_{ij} \right] b_j(o_{t+1}). \tag{3.5}$$

Until it goes to the end of the sequence, the forward variable $\alpha_T(i)$ represents the probability of the sequence O ends at state i at time T, given the model λ. Hence, the probability $P(O|\lambda)$ we seek equals to:

$$P(O|\lambda) = \sum_i \alpha_T(i). \tag{3.6}$$

Figure 3.9 depicts the procedure of the a testing phase. The recognition model is composed of three trained HMMs which represent three driving behaviors and a compare module. The same observation sequence is given to all three HMMs and receives three probabilities. In the compare module, the behavior which has the highest probability among the threes will be given as the final result of the recognition model.

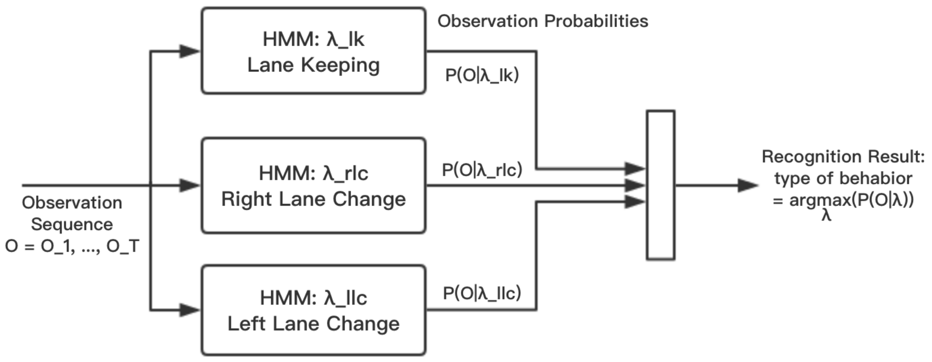


FIGURE 3.9: Illustration of the testing phase of the Hidden Markov Models.

3.3 RVMs Method for Intention Recognition

3.3.1 RVM Training

The RVM is fundamentally a two-class model. However, practically we have to tackle problems involving multiple classes. To build a multi-class model, for example a $k-$class classification model, methods which combine several binary classifiers were proposed in [38]. One is to train k one-vs-all models. For the k^{th} model, the k^{th}-class samples belong to '+1' class and all the other samples belong to '-1' class. A test sample needs to be tested by k models, a majority vote across the classifiers will decide the final result. Another one is to train $k(k-1)/2$ one-vs-one models for every two classes of samples. A test sample will be tested by $k(k-1)/2$ times and a vote scheme will give the result [39]. Compared to the binary results given by an SVM model, RVM has an advantage in probability estimation. The probability estimation given by the RVM provides an efficient way to make the majority vote, therefore we chose to build the multi-class classification model by combining 3 one-vs-all RVMs.

The training phase of an RVM model basically is a process weighting the contribution of the features. We define the contribution to the marginal likelihood as a new quantity $\boldsymbol{\theta}$. Based on the Equation 2.19 and 2.21 we can derive that the contribution is evaluated by:

$$\theta_i = {q_i}^2 - s_i. \tag{3.7}$$

A positive θ_i indicates the feature i contributes to the model, and a negative θ_i indicates the feature i does not contribute. For each one-vs-all RVM, the training data are same, only the labels differ. We assume that the features have identical contribution across all the classes, which means they share one $\boldsymbol{\theta}$ as well as one $\boldsymbol{\alpha}$ across the three RVMs. We can consider the three RVMs to be an integrated model which only need to be trained once rather than three times. The label vector $\mathbf{t}$ becomes an N-by-C label matrix $\mathbf{T}$, where N denotes the number of samples and C denotes the number of classes. The 'quality factor' q_i becomes a vector $\mathbf{q}_i$ which contains C elements. Then the contribution factor θ becomes:

$$\theta_i = \|\mathbf{q}_i\|^2 - Cs_i \tag{3.8}$$

The training algorithm takes the identical form to binary classification cases. The pseudo-code is shown in Algorithm 1. The iteration terminates when the update of $\boldsymbol{\alpha}$ is small enough, *e.g.* the changes in $\log \alpha$ is smaller than 10^{-4}, and there is no contributed but non-activated sample. The model find a local maximum of the marginal likelihood (*i.e.* Equation 2.18) when this convergence criterion is satisfied.

Although RVMs preform well on classification problems, one inevitable drawback of the RVM is the restriction of the input data. Firstly, the RVMs cannot deal with sequential data, which means we need to use some methods to represent sequential data. Secondly, this model is sensitive to the quantity magnitude of the feature. Before the training, the range and the variance of each feature should be regularized or re-scaled. Thirdly, the integrated RVMs are trained together using Bayesian interference. The trained model is sensitive to imbalanced data as well. Thus, the number of samples from different classes should be equalled in order to avoid bias. These restrictions require a series of data preparation steps. We must abandon part of the 'left lane change' data in order to balance the number of samples. Then we re-scaled the features according to the importance of the feature. Lastly, we used a 'sub-window approach' to represent the sequential data. As Figure 3.10 shows, a sub-window contains the observations during 20 time steps and a observation at one time step contains 6 features. The mean and the variance of each feature are calculated to replace all the original values because relationship measurements between values are more critical. The variance is an efficient measurement to reflect change

Algorithm 1 The Training Algorithm of the Multi-class RVMs

```
1: initialize the noise level $\sigma^2$
2: select a sample and initialize it $\alpha_i \leftarrow \frac{C\|\mathbf{k}_i\|^2}{\|\mathbf{k}_i\mathbf{t}\|^2/\|\mathbf{k}_i\|-C}$, all others are set to $\infty$
3: while convergence criterion is not satisfied do
4:     calculate sparsity factor $\mathbf{S} \leftarrow \mathbf{K}\mathbf{C}^{-1}\mathbf{K}^T$
5:     calculate quality factor $\mathbf{Q} \leftarrow \mathbf{K}\mathbf{C}^{-1}\mathbf{T}$
6:     for all activated sample $i$ ($\alpha_i \neq \infty$) do
7:         modify $s_i \leftarrow \frac{\alpha_i S_i}{\alpha_i - S_i}$
8:         modify $\mathbf{q}_i = \frac{\alpha_i \mathbf{q}_i}{\alpha_i - S_i}$
9:     calculate contribution factor $\boldsymbol{\theta}$ : $\theta_i \leftarrow \|\mathbf{q}_i\|^2 - Cs_i$
10:    if $\theta_j > 0$and$\alpha_j = \infty$ then
11:        find $j = \arg\max_j \theta_j$ for all $j$
12:    else if $\theta_j < 0$and$\alpha_j \neq \infty$ then
13:        find $j = \arg\min_j \theta_j$ for all $j$
14:    else
15:        $j$ = randomly select one from activated samples
16:    add/re-estimate or delete $\alpha_j \leftarrow \frac{C{s_i}^2}{\theta_i}$ or $\infty$
17:    update the noise level $\sigma^2$
```

patterns and also reduce noise and input size. As a result, only 12 features remain when the training phase starts.

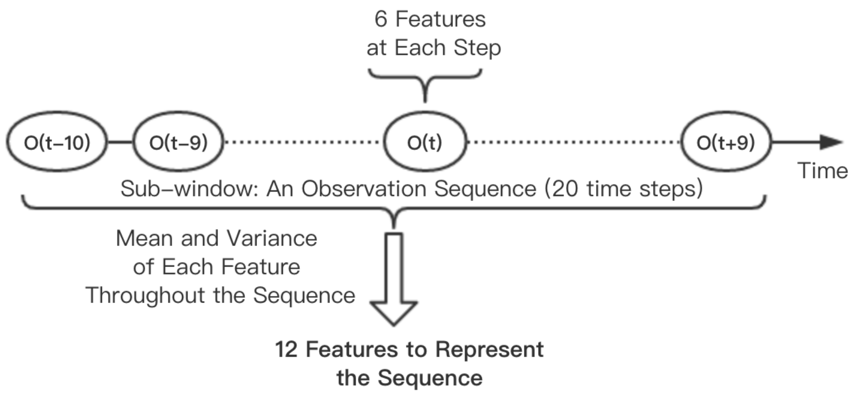


FIGURE 3.10: A representation of sequential data.

3.3.2 RVM Testing

When the training converges, we obtain M basis vectors $\mathbf{X}_{basis} = (\mathbf{x}_1^T, \mathbf{x}_2^T, \dots, \mathbf{x}_M^T)^T$ and also the weight $\tilde{\mathbf{W}}$ based on the parameter $\mathbf{A}$ and the Equation 2.25. Here, the weight $\tilde{\mathbf{W}}$ is a $C \times M$ matrix contains three weight vectors $\tilde{\mathbf{w}}$ which reflect relationships between the input data and the three classes. Given the testing set $\mathbf{X}_{test}$, the estimations are made as following:

$$\mathbf{Y} = \tilde{\mathbf{W}}\mathbf{K}(\mathbf{X}_{basis}, \mathbf{X}_{test}). \tag{3.9}$$

$\mathbf{Y}$ has three estimations for each test sample. They estimate the probability of the test sample given the 'right lane change', 'left lane change' or 'lane keeping' model. The final result is the class which has the highest estimation.

3.4 HMM-RVM Method for Intention Recognition

Although use of Hidden Markov Models has contributed greatly to driver intention recognition and also our recognition model, there are limitations of the HMMs. For example, the HMMs assume the distribution of observations can be well described by Gaussian Mixture models. Replacing the emission probability matrix B by other models can solve this limitation. As was discussed above, the RVMs are good at classification problems in a static situation rather than a dynamic one. We proposed a combined model which use a multi-class RVM model to estimate the emission probability. Hence, the major limitations of both models have been solved.

The combined model has an approximately same structure to the model in Figure 3.9, where three HMMs work in parallel. The difference lies inside the HMMs, a multi-class RVM classifier provides emission probability estimation for each HMM. Figure 3.11 illustrates the internal structure of one combined model. To conclude, the HMM-RVM combined method is an extension of the HMM method. The RVM model is considered to be a component of the HMM.

3.4.1 Combined Model Training

The whole model contains three HMMs and each HMM contains a 3-class RVM classifier. We trained these models separately. All the training samples have to be divided into three parts according to their true label. One class of training samples are only fed into the corresponding model.

The input data of the RVM model is a static observation with 6 features measured at one time step. One can distinguish the driving state of the observation according to the label we added to each observation. Based on these, the RVMs learns three weight vectors $\tilde{\mathbf{w}}$ which can map the observations O into the driving states $S = \{S_1, S_2, S_3\}$. The training algorithm decoded in Algorithm 1.

In the HMM-RVM combined model, the RVMs provide the emission probability B and replace the original multi-variate Gaussian models. This means the training

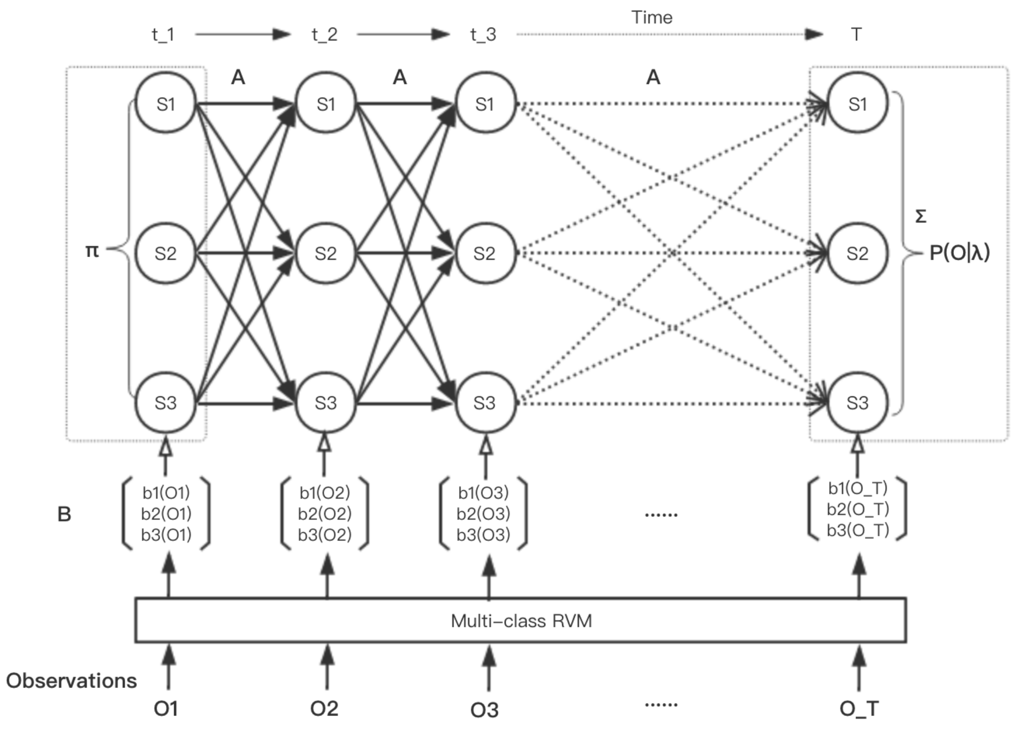


FIGURE 3.11: The internal structure of an HMM-RVM combined model.
The input is the observations $O = \{o_1, o_2, \dots, o_T\}$, *i.e.* driving parameters throughout the time $t = 1, 2, ..., T$. The output is the probability estimation of the observations given the model, *i.e.* $P(O|\lambda)$.

phase of the RVMs also replace the learning of hyperparameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. The training phase of the HMMs merely computes the initial state distribution π and state transition probability A on the basis of Equation 3.1.

3.4.2 Combined Model Testing

As Figure 3.11 shows, the observations $O = \{o_1, o_2, \dots, o_T\}$ are fed into the RVM classifier as test data. Then the RVM makes estimations of the input based on its model. These estimations serve as the emission probability B in the HMM part of the combined model. The final estimations are computed by the forward procedure, which decoded in 3.2.2. The RVM testing still follows Equation 3.9, where testing samples $\mathbf{X}_{test}$ are the observations at a single time step and the estimations $\mathbf{Y}$ are probability estimations of the observations given the states, *i.e.* $\mathbf{y}_t = \{P(o_t|q_t = S_1), P(o_t|q_t = S_2), P(o_t|q_t = S_3)\}$.

With the help of trained RVM classifiers, we tested the performance of our new recognition model. The testing procedure follows the forward procedure as was shown in Equation 3.4, 3.5 and 3.6. Similarly, the result of the recognition model is also given by choosing the behavior which has the highest probability.

Chapter 4

Results

The dataset and the pre-processing phase have been presented in Chapter 3, so we start from the evaluation method. Besides, we designed experiments to learn and prove the properties of different methods. The experiments, their result and discussion are all presented in the following. Lastly, we also compare our final result with those from the literature.

4.1 Evaluation Method

4.1.1 Cross Validation

Cross validation is a model evaluation method which can evaluate the generalization capability of a model. The generalization capability indicates how well the model will do when it makes new predictions for the unseen data. In other words, lack of generalization capability is considered to be a overfitting problem, which means the model corresponds too closely to a particular sample set rather than general samples. The overfitting problem occurs when the amount of training samples is small relative to the amount of hyper-parameters. Therefore, cross validation is important when the size of the dataset is limited. Besides, cross validation is not only a way to predict a model's future accuracy, but also a method to select the best model from a given set [40].

The main idea of cross validation is to split the dataset and to use different parts to train and test the model. Before the training begins, a portion of the data is removed and only the remain data is used for training. Then when the training is done, the removed data plays the role of the testing set to test the performance of the learned model. There are several approaches to split the dataset, such as holdout method, K-fold method and leave-one-out method. The holdout method is the simplest one. The dataset is divided into two parts: training set and testing set, and the model is trained and tested once. The K-fold method splits the dataset into K equal size subsets. The model is test by one of the k subsets and trained by the others, which means it will be trained and tested for K times in order to test all the samples. Compared to the holdout method, $K-$fold method is not affected by the way data get divided. This is because each sample in the dataset gets to be in a test set only once and to be in a training set $K-1$ times. Leave-one-out is a particular case of K-fold method with K equals to the sample amount of the whole dataset N. Because the variance of the model evaluation results reduces as K increases, the leave-one-out cross validation has a good estimation of model accuracy. However, it is extremely expensive during the training and testing phase because of the N times training. As a result, we chose K-fold cross validation for our model because it can balance the reliability of the validation and computational cost.

In k-fold cross validation, the whole dataset X is randomly split into k subsets (the folds) $X_1, X_2, \ldots, X_k$ of approximately equal size. The classifier is trained and tested k times; each time $t \in 1, 2, \ldots, k$, it is trained on $X - X_t$ and is tested on X_t. It is not conclusive how many fold can achieve an unbiased estimation of the real accuracy. However, the experiments done by Kohavi showed that the $k-$fold estimation is nearly the true accuracy when the k is equal to or larger than 10 [40]. Hence, we chose a 10-fold cross validation for our experiments.

4.1.2 Evaluation Metric

In a classification problem, confusion matrix is a commonly used method to visualize the model performance. It is a table whose rows represent the instances in a predicted class while columns represent the instances in an actual class. It can be converted to a table with two rows and two columns, which represent true predictions and false predictions. There are four components in the table: true positives, false positives, false negatives, and true negatives. Generally, Accuracy [1] is the easiest way to measure how well a classification model is. However, lane keeping behavior continuously happens on the road, while lane change behavior sometimes happens in highway driving. Plenty lane keeping samples but limited lane change samples lead to the imbalance of the test data in a real world recognition. If a model tends to classify all the samples to the 'lane keeping' class, the accuracy is still high but the model is totally useless. Hence we evaluated our model by 'precision' and 'recall'. They are calculated by the equations:

$$\begin{aligned} precision &= \frac{\#TruePositives}{\#TruePositives + \#FalsePositives} \\ recall &= \frac{\#TruePositives}{\#TruePositives + \#FalseNegatives} \end{aligned} \tag{4.1}$$

Precision measures how many of the lane changes we recognized are actually lane change behaviors. Recall measures how many of all the lane change behaviors we correctly recognized. To deal with imbalanced data, they are both essential to the model evaluation. We want to find the best combination of precision and recall, so

[1] $Accuracy = \frac{\#TruePositives + \#TrueNegatives}{\#AllSamples}$

F1 score is introduced. The F1 score is calculated as follow:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4.2}$$

It is a decent way to combine the precision and the recall in order to properly evaluate the performance of a classification model.

4.2 Results and Discussion

To research the properties of each model, we trained and tested them by different dataset. The dataset we used collected lane change and lane keeping driving data on two US highways. It contains 281 lane change sequences (68 right lane change and 213 left lane change) and 112 lane keeping sequences. Except for the single state recognition, we always used a 10-fold cross validation to ensure the reliability of the result.

4.2.1 Experiments on the HMMs Method

The HMMs adopt a multi-variate Gaussian distribution to describe the features and states. The Gaussian distribution can represent a class of data by its mean and covariance in spite of its scale, which means data pre-processing is not required for the HMMs. However, it is strict with the features. A large number of the features will affect the efficiency of the model. The uncertainty of the feature and unnecessary features will affect the performance of the model. We tested and compared the performance of the HMMs using different feature combinations. The original dataset is composed by six features, which are longitudinal and lateral velocities, longitudinal and lateral accelerations, heading angle and yaw rate. Obviously, lateral parameters are more critical than longitudinal ones in lane change recognition. Furthermore, the acceleration can better reflect the changes of the vehicle than the velocity. Thus, we obtained three different feature combinations according to the features' importance: (1) The most essential part: lateral acceleration, heading angle and yaw rate. (2) The most essential one and lateral velocity. (3) All the six features. Table 4.1 presents the difference between the results caused by three feature combinations. The 3-feature dataset achieved the best accuracy at 92.07%. The HMMs require feature selection before the training phase. This is also a drawback of the HMMs, especially the supervised learning HMMs. The HMMs lack the abilities of discovering and understanding data.

During the testing phase of the model, we found a defect of the multi-variate Gaussian model. The accuracy of the Gaussian model decreases significantly when the size of the training set gets larger and larger. Figure 4.1 depicts the relationship of the error and the training set size. There are more than 25,000 samples inside the 'left lane change' class, one fifth of them were selected as the testing set. The remain contained about 20,000 samples and we evenly divided them into thirty portions.

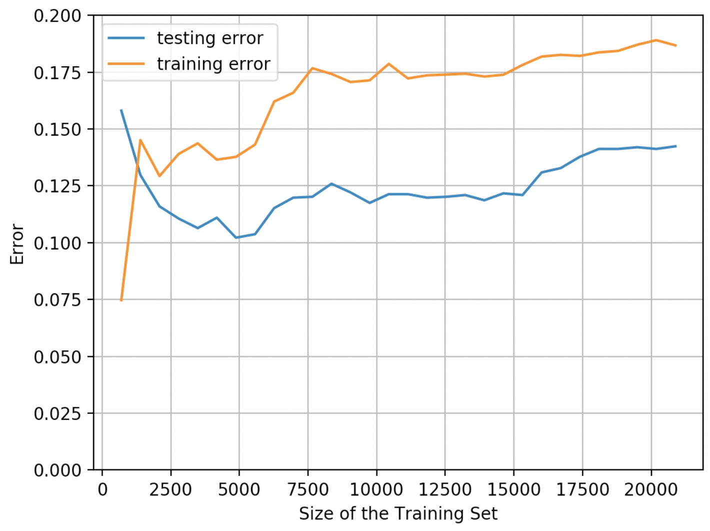


FIGURE 4.1: Error of the Gaussian Model using different size of the training set.
It was trained and tested on the 'left lane change' class.

We trained the Gaussian model by times, and each time we added a portion to the training set. The training set merely had about 700 sample at the beginning, and then it got larger and larger until it reached 20,000 samples at the last training. From Figure 4.1, we can see the the testing error decreases at beginning, after reach its lowest value when the training set contains 5000 samples, it rises again. This means the Gaussian model reach its optimum when the training set has 5000 samples (testing set has 2500 samples). With a smaller training set, the model shows overfitting, where training error is low but testing error is high. This is caused by insufficient training samples. With a larger training set, the model shows underfitting, where both training error and testing error are high. This indicates the model lack of complexity to explain the training data. The values show that the error significantly rises from 10.2% to 14.2%. It is proved that the Gaussian model has low ability to deal with large dataset, especially when the data contains high uncertainty.

TABLE 4.1: The performance of the HMMs method using different combination of features

	3-feature	4-feature	6-feature
Accuracy	92.07%	91.37%	90.58%

4.2.2 Experiments the RVMs Method

Due to the Bayesian learning method (maximum a posterior) adopted in the training algorithm, the RVMs are sensitive to the proportion of training samples. Taking an example of an imbalanced data set with 90% 'class 1' samples and 10% 'class 2' samples, the model will tend to classify all the sample into 'class 1' to achieve 90% accuracy. Unfortunately, the extremely small size of our dataset makes this problem even severe in our model. Table 4.2a presents the performance of the RVMs model using different proportion of inputs. First, we used 68 samples of each class as a baseline (sample proportion is 1:1:1). The accuracy is merely 87.25$, and the precision is 94.68%. Then, we increased the amount of the 'lane keeping' samples to let the sample proportion become 1:1:1.5. From Table 4.2a, we can clearly observe that the precision significantly enhance to 99.09% and the recall slightly decrease 0.6%. As a result, the accuracy have a great improvement. The RVMs tend to classify more samples into 'Lane Keeping' class with the ratio of 'Lane Keeping' samples increasing.

In an RVM model, the quantities and the variances of features determine their importance. In order to take full advantage of the data and also the model, the features were re-scaled by coefficients $\{1, 0.01, 5, 0.1, 10, 10\}$ [2]. We explain why we chose these coefficients in the following section. Table 4.2b shows difference results caused by re-scaling. We tested the same model in the same environment but using raw data and re-scaled data. The accuracy enhance from 91.53% to 92.37% after re-scaling the input features.

TABLE 4.2: The Performance of the RVMs using Imbalanced and Raw Data

(A) Comparison between Different Data Proportions

	Sample Proportion 1:1:1	Sample Proportion 1:1:1.5
Accuracy	87.25%	92.37%
Precision	94.68%	99.09%
Recall	98.34%	97.76%

(B) Comparison between Re-scaled and Raw Data

	Raw Data	Re-scaled Data
Accuracy	91.53%	92.37%
Precision	99.08%	99.09%
Recall	97.30%	97.76%

4.2.3 Experiments on the HMM-RVM Combined Model

The new model we proposed replaced the original Gaussian distribution by a multi-class RVM classifier to represent the observation symbol probability. We investigated the advantages and disadvantages of both models.

Table 4.3 presents the accuracy of the models fed by different features. From the previous section as well as this table, we found that the HMMs achieved the best accuracy when using three most essential features. However, with the help of re-scaling,

[2]The six features are: (1) lateral velocity (2) longitudinal velocity (3)lateral acceleration (4) longitudinal acceleration (5) relative heading angle (6) relative yaw rate.

3-feature data and 6-feature data barely make difference on the RVM's results. The comparison between the first row and the third row shows an appropriate re-scaling on the features enhances the accuracy from 83.96% to 85.79%, while the Gaussian model is not sensitive to the scaling.

TABLE 4.3: Comparison between the RVM classifier and the Gaussian model: The Influence of Input Features.

The models were trained and tested on the basis of the 'left lane change' samples. The RVM model was trained by imbalanced data (proportion 1:1.33:1).

	RVM Classifier	Gaussian Model
6 Features, Without Re-scaling	83.96%	82.09%
3 Features, Re-scaling	85.80%	82.67%
6 Features, Re-scaling	85.79%	82.09%

We also investigated the influence of the imbalanced sample. We used the precision and the recall instead of the accuracy to evaluate the performance. There are three labels in the training samples, which represent state 0, state 1 and state 2 respectively. We considered the 'state 1' is our target class because it means the driver is steering. Hence, the 'state 1' is the positive class and both the other two belongs to the negative class. Then we can compute the precision and the recall based on the confusion matrix. The values are illustrated in Table 4.4. While the precision and recall of the Gaussian Model nearly stay constant as the proportion of samples changes, the RVM classifier shows different results. As adding more 'state 1' samples into the training set, the RVM classifier tends to classify more samples into the 'state 1' class. This results a decrease in the 'precision' but an increase in the 'recall'. We can conclude that the RVMs method is more flexible than the HMMs Method.

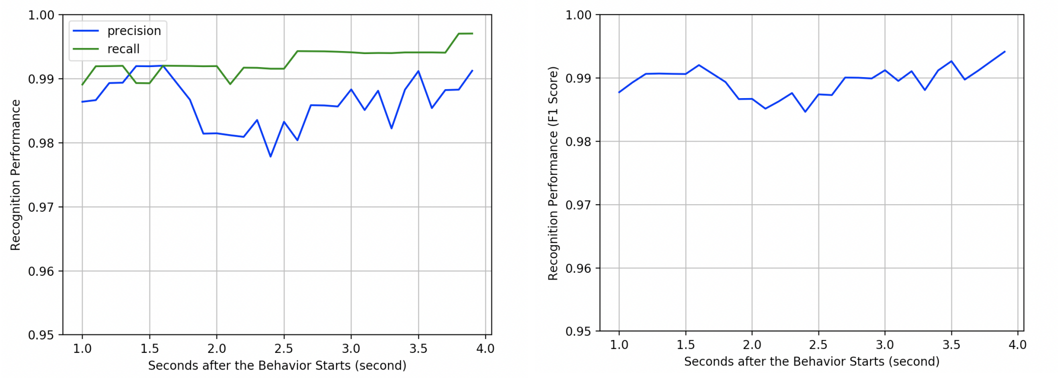
TABLE 4.4: Comparison between the RVM classifier and the Gaussian model: The Influence of Imbalanced Data.

The total amount of the training samples is 2500. For the Equally set, the proportion of the three classes (states) is 1:1:1; for the imbalanced set, the proportion is 1:1.333:1.

	RVM Classifier		Gaussian Model	
	Precision	Recall	Precision	Recall
Equally Training Samples	95.02%	94.79%	97.53%	93.05%
Imbalanced Training Samples	93.70%	96.55%	97.54%	93.01%

4.2.4 Comparison and Discussion

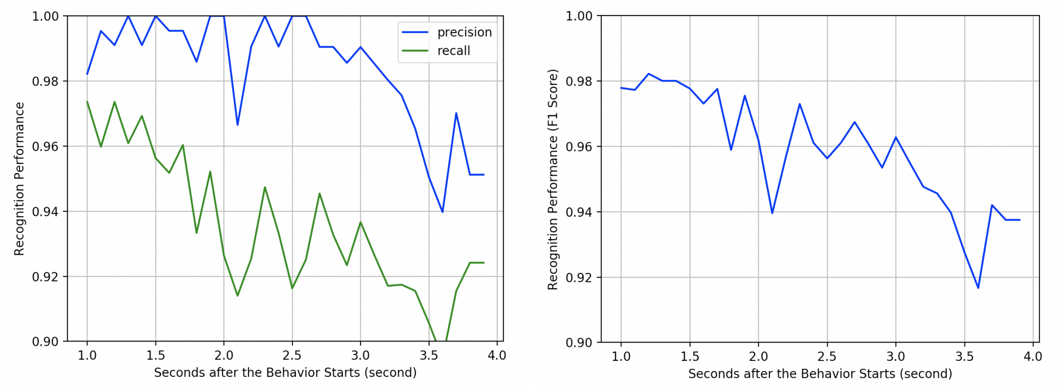
The models were trained and tested on the mixed datasets of '101' and 'i80', using 10-fold cross validation. For the RVM models, the re-scaling coefficient was $\{1, 0.01, 5, 0.1, 10, 10\}$. Each model was tested by different length of data for multiple times. The data always started being collected since 1 second before the behavior starts, *i.e.* 10 time steps before the start point for each sequence. The first test used a 20-step sub-window, which means the sequential data included 20 steps data and it covered the period from 1 second before the start point until 1 second after the start point. Then the sub-window was extended one step for each time until it reached 50-step long. Finally, it covered a 5-second period, which is approximately the whole process from the start point to the lane change point. For each test, we evaluated the performance by precision, recall and F1 score. The curves of them are drawn in Figure 4.2, 4.3 and 4.4.



(A) (B)

FIGURE 4.2: Performance evaluation of the HMMs method.
(A): The curve of the precision and the recall versus the length of the sequential data. (B): The curve of the F1 score versus the length of the sequential data.

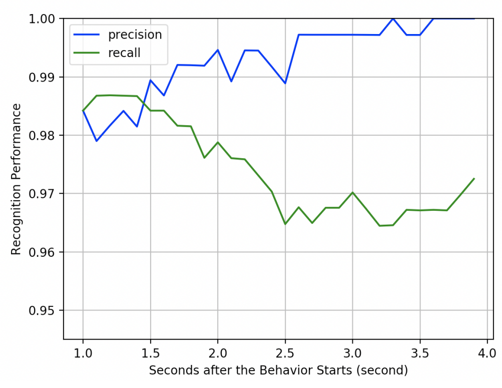
The HMMs achieved outstanding performance with stable precision and recall. From Figure 4.2 we can see that the F1 score keeps at a very high value of 0.99 and both the precision and the recall are around 0.99. Thus, we can concluded that the recognition model using the HMMs method is a stable and unbiased model, which also has remarkable performance.



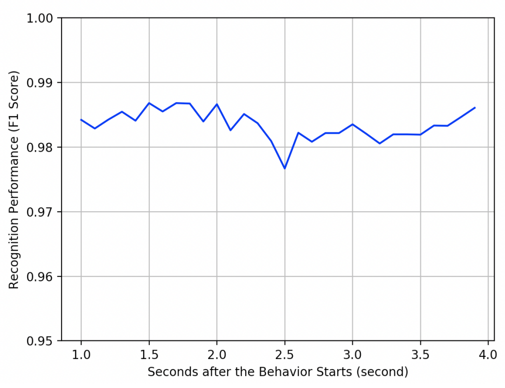
(A) (B)

FIGURE 4.3: Performance Evaluation of the RVMs Method.
(A): The curve of the precision and the recall versus the length of the sequential data. (B): The curve of the F1 score versus the length of the sequential data.

Figure 4.3 shows that the performance of the RVMs model as well as the precision and the recall slide when the sequential data becomes longer and longer. This is caused by the deficiency lies in the feature representation method. It is hard to represent the changes inside the data merely using a set of mean and covariance if the data contains too many time steps. That is why the model has a relative low F1 score of 0.92 at the end of the sequence.



(A)



(B)

FIGURE 4.4: Performance Evaluation of the HMM-RVM combined Method.
(A): The curve of the precision and the recall versus the length of the sequential data. (B): The curve of the F1 score versus the length of the sequential data.

The HMM-RVM combined method shows a comparable performance with the HMMs method. The F1 score of the combined method is around 0.985, which is sightly lower than that of the HMMs. By comparing Figure 4.4 with 4.2, we found that the precision and the recall show completely different trends. In the HMMs and the RVMs, they always change in the same direction. However, the precision increases but the recall decreases in the combined model when the input data become longer. It is caused by the sensitivity of the RVM model. The model is extremely sensitive to the amount of each class samples. At the beginning, the precision and recall are equal because both the training set and the testing set are balanced data (the amount of ′state 0′ sample and ′state 1′ sample are equal). Later, the testing set contains more and more ′state 1′ samples and it is not balanced any more. The RVM model tends to wrongly classified some ′state 1′ samples to the ′state 0′ class. Therefore, some ′lane change′ sequences are wrongly matched to the ′lane keeping′ behavior. This defect restricts the combined method can only be applied in a short time recognition model.

TABLE 4.5: Final result of the models.

The models were trained and tested on the mixed dataset and evaluated at one second after the maneuver starts.

	HMMs Method	RVMs Method	HMM-RVM Method
Accuracy	92.44%	93.84%	95.18%
Precision	98.64%	98.22%	98.94%
Recall	98.91%	97.36%	98.41%
F1 Score	98.77%	97.79%	98.67%

Table 4.5 illustrates the final result of our models. It shows the performance of each model at one second after the maneuver starts. The highest accuracy is achieved by the HMM-RVM combined method using mixed dataset. It can correctly classify more than 95% of the lane change or lane keeping behaviors within one second. By the first second after the maneuver starts, 98.94% of the lane change behaviors are correctly recognized (*i.e.* precision), while the recall is also as high as 98.41%.

Pentland and Liu used Hidden Markov Models to describe driving behaviors for the first time. According to their results, approximately 1.5 seconds after the beginning of action (or roughly 20% of the way through the action), mean recognition accuracy was 95.24% $\pm$ 3.1%. However, one thing need to be mentioned is their model can recognize multiple behaviors rather than only lane change. They had 72 stop, 262 turn, 47 lane change, 24 passing and 208 normal driving in their data set. The high accuracy was mainly attributed by the other behaviors instead of the lane change [2]. Kuge *et al.*introduced a driver model framework based on the principle of HMMs. At the time of 1.5 seconds after the behavior begins, his framework could recognize 98.5% of them. The false negative rate and false positive rate are merely 0% and 0.29% respectively [8]. Mandalia and Salvucci used Support Vector Machines for lane-change detection. Their model recognized the lane change behavior from lane keeping behaviors with 97.9% true positive rate [3]. About 87% of the true positives were correctly detected within the first 0.3 second from the start of the maneuver [10]. Nilsson *et al.*proposed a rule-based lane change intention recognition algorithm. They also used NGSIM interstate 80 dataset for the training and testing. The marked the moment the vehicle crosses the lane marking as the lane change moment. For a left lane change recognition, the accuracy was 89% 1 second prior to its lane change moment and 39% 2 seconds prior to the moment, while the false positive rate was around 3%. For a right lane change, the accuracy was 81% and 38% with only 0.7% false positive rate [14].

Among the literature, Nilsson used same dataset NGSIM 'i80' as we used. According to the data, it usually takes 3 or 4 seconds since the moment of behavior starts until the moment of lane change. Our model can achieve 95% accuracy with only 1.6% false positive rate at least 2 seconds before the lane change moment. We can conclude our model exceeds Nilsson's. Our models are also comparable with the models from Kuge, Mandalia and Salvucci.

TABLE 4.6: Testing the models on single or mixed dataset.

The models were trained and tested with a 20-step sub-window. The values are the accuracy at 1 second after the behavior starts.

	HMMs Method	RVMs Method	HMM-RVM Method
i80 Dataset	94.20%	92.50%	94.15%
i80 and 101 Dataset	92.44%	93.84%	95.18%

Besides, Table 4.6 compares the models' performance on different dataset. Although the HMMs method shows better result than the other two methods on the single 'i80' dataset, it loses about 2% accuracy and becomes the worst one on the mixed dataset.

[3] $\frac{\#TruePositives}{\#TruePositives+\#FalseNegatives}$, is also called as recall or sensitivity

On the contrast, the RVMs and the combined model perform even better when the dataset become more complex. The low generalization ability of the HMMs is also proved from this table.

Chapter 5

Conclusion

In this chapter, we conclude the properties of the models based on the experimental results.We also analyze the advantage and disadvantage of the proposed model. Besides, we recommend some ideas on how to improve each model. We hope the model will be further developed with the help of future works.

5.1 Conclusion

From the comparison and discussion section in Chapter 4, we have answered the proposed questions in Chapter 1.3. Here, we give a short conclusion to present the result and the property of our models.

5.1.1 The Property of the HMMs Method

We developed a recognition model which contains three paralleled supervised HMMs. Each HMM can represent a driving behavior by defining three driving states (operations, such as steering) and finding their distribution. We can infer the current driving state based on the observations (the driving parameters, such as velocity and heading angle), because we assume the observations follow a multi-variate Gaussian distribution. Given observations, we can deduce the corresponding driving state and then the driving behavior.

The main advantage of the HMMs method is simple. An HMM can depict a sequence of behavior in a statistics way by defining non-observable 'states' and finding relationships between the 'states' and some observable features. Before the training starts we replaced the model to do the 'state' definition, so the HMM only need to learn the relationships during the training. This significantly reduces the time consumption of the training phase, although it results in the low ability to learn new features. The time complexity of the supervised training is $O(T)$, and the time complexity of the testing phase is $O(N^2T)$ [1] [21].

[1] T denotes the length of the sequential data, and N denotes the number of states.

The main drawback of the HMMs lie in its assumptions. There are three assumptions: (a) The observations of the model are independent. The probability of the observation sequence is the product of the probability of the observation at each time step, *i.e.* $P(O) = \prod_{t=1}^{T} P(O_t)$. (b) The model is a first-order Markov Chain. The state at time t only depends on the state at time $t-1$. (c) A Gaussian distribution can well describe the observation symbol probability.

In a driving case the observations, alternatively the driving parameters, are not independent with each other. The velocity at time t is determined by the velocity and the acceleration at time $t-1$. The assumption of independent observations as well as the first-order Markov Chain are compromised between low complexity and high accuracy. For a on-road intention recognition model, collecting training data from multiple places is essential to enhance the application range of it. The Gaussian model showed a decline on its accuracy as the training set getting larger, which means the Gaussian model has low generalization ability.

According to the experiment results, its precision, recall and F1 score all achieve 99% high. The results prove that the HMMs method is a stable and reliable way to recognize driving behaviors. However, it is also a rigid and inflexible method due to its assumptions. Its performance on a real-world recognition task is still doubtful.

5.1.2 The Property of the RVMs Method

The multi-class RVMs classifier learns three hyper-planes to describe the three driving behaviors. We used a series of the pre-processing steps to represent a sequential driving data by a set of features. Given the training data, the RVMs learned the relationships between the features and the driving behaviors. Basically there are also three paralleled RVMs inside the integral model, but they can learn simultaneously. Given the testing data, the RVMs output probability estimations of the feature set based on the models. We deduce the driving behavior according to the highest estimation.

Compared the HMMs method, it utilizes the Bayesian theory in the training phase. Therefore, the model is flexible and adjustable. There are plenty of the parameters we can tune to better train the model. It has been proved that the RVMs are fast on testing but slow on training. The time complexity of the training phase and the testing phase are $O(N^3)$ and $O(M)$ [2] respectively [22]. Besides, the RVMs model also lost the ability of discovering new features due to the one-vs-all method we adopted for the multi-class extension.

A severe deficiency of the RVMs method is that the RVMs cannot process sequential data. So we have to represent the sequential data by their mean and variance. From Figure 4.3, we can know that the accuracy is higher at the beginning. This is because the 'sub-window' trick as well as the mean and the variance cannot well represent the changes inside the sequential data when the window size is large.

[2] N denotes the number of training samples, and M denotes the number of relevance vectors.

5.1.3 The Property of the HMM-RVM Combined Method

To solve the shortcomings of the above methods, we combined them by using a multi-class RVM classifier to replace the Gaussian model of the HMMs. Hence, the input of the multi-class RVM classifier becomes discrete. The simple but rigid Gaussian model was discarded and we adopted a more flexible model to give the observation symbol probability.

The main difficulty of the combined model is the training complexity. From the last section, we have known that the training complexity of a 3-class RVM is $O(N^3)$. Since we split the whole sequence of a driving behavior into multiple time steps, the enormous amount of training samples made the training phase become extremely expensive. We had to restrict the number of the training sample at an appropriate value to ensure a short training time and comparable accuracy. The testing complexity is $O(N^2T + M)$, which can be simplified to $O(N^2)$.

The combined method shows the highest accuracy of 95.18% among all methods. Even among the literature, it still is an outstanding result. Expect the high accuracy, it is a flexible model which can adjust the recall and precision. The most important advantage is it shows good generalization ability and high tolerance on complex input data. This is greatly helpful to apply it on real-life applications.

5.2 Recommendation

Although the model's performance have been already good, there is still a long way to go before applying it on wide-range applications. We would like to propose some topics for future works to help the intention recognition model become better.

- For the inaccurate assumptions of the HMMs, Kumagai and Akamatsu switched the probability of observation in a linear dynamic system (LSDS). Hence, the observation symbol probability was a Gaussian distribution considering the influence from the last observation, *i.e.* $b_i(O_t) = \mathcal{N}(\boldsymbol{\mu}_i + \mathbf{w}_i \cdot O_{t-1}), \boldsymbol{\Sigma}_i)$ [41]. In the future, the LSDS may be combined with any other model to estimate the conditional probability of the observations.

- We modified all the three models to let them become supervised learning models. This simplified the models but also affected their abilities of generalization and self-learning. To prevent from the loss and preserve the advantage, a semi-supervised learning method is worth trying.

- Damoulas and Girolami proposed a multi-class kernel machine that can solve multi-class classification problems while learning at the same time. This algorithm is based on a well-founded hierarchical Bayesian framework and able to instructively combine feature spaces [42]. Their results showed that their algorithm achieved high precision on classification problems, and the most important is, it did not yield bias when the input data is imbalanced. This algorithm can be integrated into the multi-class RVM classifier in order to improve the stability.

Appendix A

Appendix: Training Algorithm of the SVM Model

A.1 Lagrangian Formulation in the Training Algorithm

In an SVM model, we have a hyperplane takes the form of:

$$\mathbf{w} \cdot \mathbf{x} + b = 0, \mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}. \tag{A.1}$$

To find the best regression of the hyperplane and also avoid overfitting, the loss function is expressed as the sum of an error term and a penalty term [43]. Thus, finding the hyperplane of an SVM model becomes a problem of minimizing the loss function [36], [44], which is

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2}, \tag{A.2}$$

subject to

$$\begin{array}{ll} y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geqslant 1, & \text{for classification} \\ |y_i - \mathbf{w} \cdot \mathbf{x}_i - b| \leqslant \epsilon, & \text{for regression.} \end{array} \tag{A.3}$$

This becomes an optimization problem, subject to constrains A.3. Then we will switch to a Lagrangian formulation of the problem, because we can replace the constrains by introducing positive Lagrange multipliers $\alpha_i, i = 1, \ldots, N$. [1] It gives Lagrangian:

$$L_P \equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i y_i (\mathbf{w} \cdot \mathbf{x}_i + b) + \sum_{i=1}^{N} \alpha_i \tag{A.4}$$

We need to solve this 'dual' problem: minimize L_P, and simultaneously require that the derivatives of L_P with respect to $\mathbf{w}, b$ vanish, all subject to the constraints $\alpha_i \geqslant 0$. The derivatives vanishing gives the conditions:

$$\begin{array}{l} \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \\ \sum_i \alpha_i y_i = 0. \end{array} \tag{A.5}$$

[1] In the following, we take an SVM classification model as an example. The algorithm for an SVM regression model is similar and is introduced in [44].

Since we got these conditions on $\mathbf{w}$ and b, we substitute them into Equation A.4 and it turns to

$$\begin{array}{c} L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ \text{s.t.} \sum_i \alpha_i y_i = 0, \forall \alpha_i \geqslant 0 \end{array} \tag{A.6}$$

Hence the optimization problem turns to the maximization problem of L_D. Lastly, due to the property of derivatives of α_i vanishing, we can calculate the numerical solutions of α_i and also w_i [30].

A.2 Kernel Function

Equation A.6 shows that we use the dot product of samples $\mathbf{x}_i$ and $\mathbf{x}_j$ rather than one single sample $\mathbf{x}_i$ to find the corresponding weight $\mathbf{w}_i$. This is convenient for us to apply SVM techniques into non-linear cases.

From Equation 2.9, we know that the samples need to be mapped into a high dimensional space to achieve linear separable. We take the simplest kernel as an example, if we choose the kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^2$, it is easy to find a mapping ϕ from $\mathbb{R}^2$ space to $\mathscr{H}$ space:

$$\phi(\mathbf{x}) = \begin{pmatrix} {x_1}^2 \\ \sqrt{2}x_1x_2 \\ {x_2}^2 \end{pmatrix}. \tag{A.7}$$

However, if we have a high dimensional input space, the mapping ϕ will be much more complex than this case. This is why we use dot products to replace single samples in the training algorithm. We only need to select a kernel function K instead of knowing what exactly the mapping ϕ is.

There are four commonly used kernel functions:

- linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$.
- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i \cdot \mathbf{x}_j) + r)^d, \gamma > 0$.
- radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \gamma > 0$.
- sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i \cdot \mathbf{x}_j + r)$.

γ, r, d are the kernel parameters need to be decided before applying the kernel function. One thing need to be mentioned is that a linear kernel cannot project the data into a higher dimensional space. It is only applied on a linear separable case. Using a linear kernel instead of the original data is convenient for training. When choosing a kernel function, generally the RBF kernel is a reasonable first choice. The reasons are two-fold. First, compare to a linear kernel, it can handle non-linear cases. Second, compare to a polynomial and sigmoid kernel, it has less kernel parameters. This will decrease the complexity of the model. Nevertheless, if the number of features is large, linear kernel is good enough because the data has no need to be mapped into a higher dimensional space [36].

Appendix B

Appendix: Hyper-parameter Update of the RVM Model

B.1 Update of A

We re-write the definition of **C** from $\mathbf{K}\mathbf{A}^{-1}\mathbf{K}^T + \sigma^2\mathbf{I}$ to

$$\begin{aligned} \mathbf{C} &= \sigma^2\mathbf{I} + \textstyle\sum_m \alpha_m \mathbf{k}_m^T\mathbf{k}_m \\ &= \sigma^2\mathbf{I} + \textstyle\sum_m \alpha_m^{-1}\mathbf{k}_m^T\mathbf{k}_m + \alpha_i^{-1}\mathbf{k}_i^T\mathbf{k}_i \\ &= \mathbf{C}_{-i} + \alpha_i^{-1}\mathbf{k}_i^T\mathbf{k}_i \end{aligned} \tag{B.1}$$

Using established matrix determinant and inverse identities, we have:

$$\begin{aligned} |\mathbf{C}| &= |\mathbf{C}_{-i}||1 + \alpha_i^{-1}\mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{k}_i^T|, \\ \mathbf{C}^{-1} &= \mathbf{C}_{-i}^{-1} - \tfrac{\mathbf{C}_{-i}^{-1}\mathbf{k}_i^T\mathbf{k}_i\mathbf{C}_{-i}^{-1}}{\alpha_i + \mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{k}_i^T}, \end{aligned} \tag{B.2}$$

which gives

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}) &= -\tfrac{1}{2}[N\log 2\pi + \log|\mathbf{C}_{-i}| + \mathbf{t}^T\mathbf{C}_{-i}^{-1}\mathbf{t} \\ &\quad - \log\alpha_i + \log(\alpha_i + \mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{k}_i^T) - \tfrac{(\mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{t})^2}{\alpha_i + \mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{k}_i^T}] \\ &= \mathcal{L}(\boldsymbol{\alpha}_{-i}) + \tfrac{1}{2}\left[\log\alpha_i - \log(\alpha_i + \mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{k}_i^T) + \tfrac{(\mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{t})^2}{\alpha_i + \mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{k}_i^T}\right] \\ &= \mathcal{L}(\boldsymbol{\alpha}_{-i}) + \ell(\alpha_i). \end{aligned} \tag{B.3}$$

$\mathcal{L}(\boldsymbol{\alpha}_{-i})$ is the log marginal likelihood with α_i, and $\mathbf{k}_i$ removed from the model and we have now isolated the terms in α_i in the function $\ell(\alpha_i)$. The gradient of the marginal likelihood was computed as:

$$\frac{\partial\mathcal{L}(\boldsymbol{\alpha})}{\partial\alpha_i} = \frac{\partial\ell(\alpha_i)}{\partial\alpha_i} = \frac{1}{2}\left[\frac{1}{\alpha_i} - \frac{1}{\alpha_i + \mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{k}_i^T} - \frac{(\mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{t})^2}{(\alpha_i + \mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{k}_i^T)^2}\right]. \tag{B.4}$$

To simplify this, we defined 'sparsity' quantity s_i and 'quality' quantity q_i as:

$$s_i \triangleq \mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{k}_i^T, \textit{ and } q_i \triangleq \mathbf{k}_i\mathbf{C}_{-i}^{-1}\mathbf{t}. \tag{B.5}$$

Hence, the gradient of the marginal likelihood can be re-written as:

$$\frac{\partial \mathcal{L}(\boldsymbol{\alpha})}{\partial \alpha_i} = \frac{\alpha_{i-1} s_i^2 - (q_i^2 - s_i)}{2(\alpha_i + s_i)^2}. \tag{B.6}$$

By equating Equation B.6 to zero, we get two solutions for α_i:

$$\begin{aligned} \alpha_i = \tfrac{s_i^2}{q_i^2 - s_i}, &\quad q_i^2 > s_i, \\ \alpha_i = \infty, &\quad q_i^2 \leqslant s_i. \end{aligned} \tag{B.7}$$

The criterion from Equation B.7 determines the hyper-parameter $\boldsymbol{\alpha}$, and thus $\tilde{\mathbf{W}}$. On the basis of this, the iterative maximization of the marginal likelihood function $\mathcal{L}(\boldsymbol{\alpha})$ with respect to the hyper-parameters $\boldsymbol{\alpha}$ helps the RVMs to selected 'relevance vectors' and optimize the weights $\tilde{\mathbf{W}}$ [45].

B.2 Update of σ^2

The marginal likelihood has been given in Equation 2.18:

$$\mathcal{L}(\boldsymbol{\alpha}) = \log p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2) = -\frac{1}{2}\left[N \log 2\pi + \log|\mathbf{C}| + \mathbf{t}^T \mathbf{C}^{-1} \mathbf{t}\right], \tag{B.8}$$

with the quantity $\mathbf{C} = \mathbf{K}\mathbf{A}^{-1}\mathbf{K}^T + \sigma^2 \mathbf{I}$. The first term of the marginal likelihood is independent of the noise level σ^2. The determinant identity gives

$$|\mathbf{A}||\mathbf{K}\mathbf{A}^{-1}\mathbf{K}^T + \sigma^2\mathbf{I}| = |\sigma^2\mathbf{I}||\mathbf{A} + \mathbf{K}\mathbf{K}^T/\sigma^2|. \tag{B.9}$$

Thus, the second term of the marginal likelihood can be factorized as following:

$$\log|\mathbf{K}\mathbf{A}^{-1}\mathbf{K}^T + \sigma^2\mathbf{I}| = -\log|\boldsymbol{\Sigma}| - N\log\frac{1}{\sigma^2} - \log|\mathbf{A}|, \tag{B.10}$$

where $\boldsymbol{\Sigma}$ is the posterior weight covariance $\boldsymbol{\Sigma} = (\mathbf{A} + \sigma^{-2}\mathbf{K}\mathbf{K}^T)^{-1}$. The Woodbury inversion identity gives

$$(\mathbf{K}\mathbf{A}^{-1}\mathbf{K}^T + \sigma^2\mathbf{I})^{-1} = \mathbf{I}/\sigma^2 - \sigma^{-2}\mathbf{K}(\mathbf{A} + \sigma^{-2}\mathbf{K}\mathbf{K}^T)^{-1}\mathbf{K}^T\sigma^{-2}. \tag{B.11}$$

Then the third term of the marginal likelihood can be expressed as

$$\begin{aligned} \mathbf{t}^T(\mathbf{K}\mathbf{A}^{-1}\mathbf{K}^T + \sigma^2\mathbf{I})^{-1}\mathbf{t} &= \sigma^{-2}\mathbf{t}^T\mathbf{t} - \sigma^{-2}\mathbf{t}^T\mathbf{K}\boldsymbol{\Sigma}\mathbf{K}^T\sigma^{-2}\mathbf{t} \\ &= \sigma^{-2}\mathbf{t}^T(\mathbf{t} - \mathbf{K}\boldsymbol{\mu}) \\ &= \sigma^{-2}\|\mathbf{t} - \mathbf{K}\boldsymbol{\mu}\|^2 + \sigma^{-2}\mathbf{t}^T\mathbf{K}\boldsymbol{\mu} - \sigma^{-2\text{-}T}\mathbf{K}^T\mathbf{K}\text{-} \\ &= \sigma^{-2}\|\mathbf{t} - \mathbf{K}\boldsymbol{\mu}\|^2 + \text{-}^T\mathbf{A}\text{-}, \end{aligned} \tag{B.12}$$

where $\boldsymbol{\mu}$ is the posterior weight mean $\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\mathbf{K}\mathbf{t}$. Hence, the derivative of the marginal likelihood with respect to $\log\sigma^{-2}$ is

$$\frac{\partial \mathcal{L}}{\partial \log \sigma^{-2}} = \frac{1}{2}[N\sigma^2 - \|\mathbf{t} - \mathbf{K}\boldsymbol{\mu}\|^2 - \mathrm{Tr}(\boldsymbol{\Sigma}\mathbf{K}^T\mathbf{K})]. \tag{B.13}$$

$\mathrm{Tr}(\mathbf{\Sigma K}^T\mathbf{K})$ can be re-written as $-M - \sum_m \alpha_m \Sigma_{mm}$ [46]. By equaling the derivative to zero, we obtain the new noise level

$$\begin{aligned} \sigma^2 &= \frac{\|\mathbf{t}-\mathbf{K}\boldsymbol{\mu}\|^2}{(N-M+\sum_m \alpha_m \Sigma_{mm})} \\ &= \frac{\|\mathbf{t}-\mathbf{y}\|^2}{(N-M+\sum_m \alpha_m \Sigma_{mm})}. \end{aligned} \tag{B.14}$$

Following Equation B.7 and B.14, the hyper-parameters of the RVMs are updated.

Appendix C

Data Acquisition Code

Data pre-processing is completed by Matlab. We take the trajectory on Interstate 80 at 5:15pm as an example.

```
%% read .csv file
filename = 'trajectories0515.csv'; % 0750/0400/0500/0515
g_time = 1113437565600; % /ms % 0515
Fs = 10; % frequency /Hz
T = 1/Fs; % period /s
% rows for the first vehicle
i = 2;
j = 1155;
% store vehicle data in SpeedData, LaneData, Length and Width
for n = 1:1200
    % read vehicle data from csv file
    A = csvread(filename, i-1, 0, [i-1,0,j-1,16]);
    % convert global time to local time /s
    A(:,3) = (A(:,3) - g_time)/1000;
    % convert feet to meter
    A(:,[4:9,13]) = 0.3048* A(:,[4:9,13]);
    % record the vehicle id
    Veh_ID = A(1,1);
    % signal length
    L = A(1,15);
    t = (0:L-1)*T;
    % local position filter
    Fc = 0.3;
    [b,a] = butter(1,Fc/(Fs/2));
    A(1:end-1,4) = filter(b,a,A(1:end-1,4));
    A(1:end-1,5) = filter(b,a,A(1:end-1,5));
    % store data
    % SpeedData: local time, local xy, global xy, speed, acc
    % LaneData: lane, proceding vehicle, following vehicle, headways
    SpeedData{Veh_ID} = A(1:end-1,3:9);
    LaneData{Veh_ID} = A(1:end-1,10:14);
    Length{Veh_ID} = A(1,16); % length of the vehicle
    Width{Veh_ID} = A(1,17); % width of the vehicle
    % rows and columns for next vehicle
    i = j;
    j = i + A(end,15);
end

%% recognize lane changes
n_lc = 0; % count the number of lane change
for n = 1:1740
    % check if veh_id exists in the database
    if isempty(SpeedData{n})
        continue
    else
        Veh_ID = n;
    end
    % m: locate the row of the vehicle data
    for m = 100:size(LaneData{Veh_ID},1)-50
        % recognise lane change behavior (no onramp and outramp)
        if range(LaneData{Veh_ID}(m:m+1,1))==1 & LaneData{Veh_ID}(m,1)~isme
            Veh_F = LaneData{Veh_ID}(m,3); % find following vehicle
            % lane change period
            % Time_1: 10s before the lane change point
            % Time_2: 5s after the lane change point
            Time_1 = SpeedData{Veh_ID}(m-99,1);
            Time_2 = SpeedData{Veh_ID}(m+50,1);
            % check if the following vehicle's data is in the database
            % p,q: the position of rows in following vehicle data
            if Veh_F & ~isempty(SpeedData{Veh_F})
                p = find(SpeedData{Veh_F}(:,1) == Time_1);
                q = find(SpeedData{Veh_F}(:,1) == Time_2);
            end
            if p & q & q-p==149
                n_lc = n_lc + 1; % count the lane change behaviors
                % 1: time step 15s (10s before, 5s after)
                % 2~5: leading position (local/gobal)
                % 6:9: following position (local/gobal)
                % 10,11: following headway (space/time)
                LaneChange{n_lc}(:,1:5) = SpeedData{Veh_ID}(m-99:m+50,1:5),
                LaneChange{n_lc}(:,6:9) = SpeedData{Veh_F}(p:q,2:5);
                LaneChange{n_lc}(:,10:11) = LaneData{Veh_F}(p:q,4:5)
            end
        end
    end
end

%% save the driving parameters of lane changes
save filtereddata0515.mat LaneChange
```

```
%% extract velocity, acceleration and heading angles
Local_vel = cell(1, size(LaneChange,2));
Local_acc = cell(1, size(LaneChange,2));
Ref_heading = cell(1, size(LaneChange,2));
Yaw_rate = cell(1, size(LaneChange,2));
for i = 0:size(LaneChange,2)
    % local_velocity: leading xy, following xy
     Local_vel{i}(1:149,1) = diff(LaneChange{i}(:,2))./diff(LaneChange{i}(:,1))
     Local_vel{i}(1:149,2) = diff(LaneChange{i}(:,3))./diff(LaneChange{i}(:,1))
     Local_vel{i}(1:149,3) = diff(LaneChange{i}(:,6))./diff(LaneChange{i}(:,1))
     Local_vel{i}(1:149,4) = diff(LaneChange{i}(:,7))./diff(LaneChange{i}(:,1))
    % smooth local_velocity
     Local_vel{i}(:,1:4) = movmean(Local_vel{i}(:,1:4),20);
     Local_vel{i}(end+1,:) = Local_vel{i}(end,:);
    % local_acceleration: leading xy
     Local_acc{i}(1:149,1) = diff(Local_vel{i}(:,1))./diff(LaneChange{i}(:,1));
     Local_acc{i}(1:149,2) = diff(Local_vel{i}(:,2))./diff(LaneChange{i}(:,1));
    % smooth local_acceleration
     Local_acc{i}(:,1:2) = movmean(Local_acc{i}(:,1:2),20);
     Local_acc{i}(end+1,:) = Local_acc{i}(end,:);
    % ref_heading angle
     lead_heading = atand(Local_vel{i}(:,1)./Local_vel{i}(:,2));
     follow_heading = atand(Local_vel{i}(:,3)./Local_vel{i}(:,4));
     Ref_heading{i} = lead_heading - follow_heading;
    % smooth ref_heading angle
     Ref_heading{i} = movmean(Ref_heading{i},20);
    % yaw rate (derivitive of ref_heading)
     Yaw_rate{i}(1:149) = diff(Ref_heading{i})./diff(LaneChange{index}(:,1));
     Yaw_rate{i}(end+1) = Yaw_rate{i}(end);
end

%% data cleansing/converter to csv
% count left changes and right changes
l = 0;
r = 0;
% Left/Right: store input features
Left = [];
Right = [];
% convert matrix to string matrix
Left = string(Left);
Right = string(Right);
% store headings of the features
Left(1,:) = ["veh_id","vel_x","vel_y","acc_x","acc_y","heading","yaw_rate"];
Right(1,:) = ["veh_id","vel_x","vel_y","acc_x","acc_y","heading","yaw_rate"];
% converter
for i = 1:size(Local_vel,2)
    % distinguish left/right lane changes
     if Ref_heading{i}(100) < 0
```

```
        % Condition 1: heading angle is small before lc point
        % at least 1 second
        for t = 60:-1:10
            if max(abs(Ref_heading{i}(t-10:t)))<0.3
                start = t-10;
                break
            else
                start = [];
            end
        end
        % Condition 2: heading angle is back to 0 at the end
        if ~isempty(start)
            for t = 120:150
                if abs(mean(Ref_heading{i}(t-5:t)))<0.3
                    stop = t;
                    l = l+1;
                    i1 = size(Left,1)+1;
                    i2 = index1+stop-start;
                    Left(i1:i2,1) = string(l);
                    Left(i1:i2,2:3) = string(Local_vel{i}(start:stop,1:2));
                    Left(i1:i2,4:5) = string(Local_acc{i}(start:stop,1:2));
                    Left(i1:i2,6) = string(Ref_heading{i}(start:stop));
                    Left(i1:i2,7) = string(Yaw_rate{i}(start:stop));
                    break
                end
            end
        end
    % right lane change, same as the left ones
    elseif Ref_heading{i}(100) > 0
        for t = 60:-1:10
            if max(abs(Ref_heading{i}(t:t+10)))<0.3
                start = t;
                break
            else
                start = [];
            end
        end
        if ~isempty(start)
            for t = 120:150
                if abs(mean(Ref_heading{i}(t-5:t)))<0.3
                    stop = t;
                    r = r+1;
                    i1 = size(Right,1)+1;
                    i2 = index1+stop-start;
                    Right(i1:i2,1) = string(r);
                    Right(i1:i2,2:3) = string(Local_vel{i}(start:stop,1:2)
                    Right(i1:i2,4:5) = string(Local_acc{i}(start:stop,1:2)
                    Right(i1:i2,6) = string(Ref_heading{i}(start:stop));
                    Right(i1:i2,7) = string(Yaw_rate{i}(start:stop));
                    break
                end
            end
        end
    end
end

%% save data/write files
save inputdata.mat Left Right
csvwrite('leftlc', Left);
csvwrite('rightlc', Right);
```

Bibliography

[1] K. A. Brookhuis, D. De Waard, and W. H. Janssen, "Behavioural impacts of advanced driver assistance systems–an overview", *European Journal of Transport and Infrastructure Research*, vol. 1, no. 3, pp. 245–253, 2001.

[2] A. Pentland and A. Liu, "Modeling and prediction of human behavior", *Neural Computation*, vol. 11, no. 1, pp. 229–242, 1999. DOI: `10.1162/089976699300016890`. eprint: `https://doi.org/10.1162/089976699300016890`. [Online]. Available: `https://doi.org/10.1162/089976699300016890`.

[3] G. S. Aoude, V. R. Desaraju, L. H. Stephens, and J. P. How, "Behavior classification algorithms at intersections and validation using naturalistic data", in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, IEEE, 2011, pp. 601–606.

[4] Y. Kishimoto and K. Oguri, "A modeling method for predicting driving behavior concerning with driver's past movements", in *2008 IEEE International Conference on Vehicular Electronics and Safety*, 2008, pp. 132–136. DOI: `10.1109/ICVES.2008.4640888`.

[5] D. Mitrovic, "Reliable method for driving events recognition", *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 198–205, 2005, ISSN: 1524-9050. DOI: `10.1109/TITS.2005.848367`.

[6] Q. N.Y. S. Haijing Hou Lisheng Jin and M. Lu, "Driver intention recognition method using continuous hidden markov model", *International Journal of Computational Intelligence Systems*, vol. 4, no. 3, pp. 386–393, 2011. DOI: `10.1080/18756891.2011.9727797`. [Online]. Available: `https://doi.org/10.1080/18756891.2011.9727797`.

[7] N. Oliver and A. P. Pentland, "Graphical models for driver behavior recognition in a smartcar", in *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, IEEE, 2000, pp. 7–12.

[8] N. Kuge, T. Yamamura, O. Shimoyama, and A. Liu, "A driver behavior recognition method based on a driver model framework", SAE Technical Paper, Tech. Rep., 2000.

[9] G. S. Aoude and J. P. How, "Using support vector machines and bayesian filtering for classifying agent intentions at road intersections", Tech. Rep., 2009.

[10] H. M. Mandalia and M. D. D. Salvucci, "Using support vector machines for lane-change detection", *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 49, no. 22, pp. 1965–1969, 2005. DOI: `10.1177/154193120504902217`. eprint: `https://doi.org/10.1177/154193120504902217`. [Online]. Available: `https://doi.org/10.1177/154193120504902217`.

[11] P. Kumar, M. Perrollaz, S. Lefevre, and C. Laugier, "Learning-based approach for online lane change intention prediction", in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, IEEE, 2013, pp. 797–802.

[12] B. Morris, A. Doshi, and M. Trivedi, "Lane change intent prediction for driver assistance: On-road design and evaluation", in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 895–901. DOI: `10.1109/IVS.2011.5940538`.

[13] M. N. Husen and S. Lee, "Continuous car driving intention recognition with syntactic pattern approach", in *2016 International Conference on Information and Communication Technology (ICICTM)*, 2016, pp. 71–76.

[14] J. Nilsson, J. Fredriksson, and E. Coelingh, "Rule-based highway maneuver intention recognition", in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 950–955.

[15] J. Ko, D. J. Klein, D. Fox, and D. Hähnel, "Gaussian processes and reinforcement learning for identification and control of an autonomous blimp", *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 742–747, 2007.

[16] U. Dogan, H. Edelbrunner, and I. Iossifidis, "Towards a driver model: Preliminary study of lane change behavior", in *2008 11th International IEEE Conference on Intelligent Transportation Systems*, 2008, pp. 931–937. DOI: `10.1109/ITSC.2008.4732700`.

[17] A. Zyner, S. Worrall, J. Ward, and E. Nebot, "Long short term memory for driver intent prediction", in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1484–1489. DOI: `10.1109/IVS.2017.7995919`.

[18] B. H.L. Q. G. Xie H. Gao and J. Wang, "A driving behavior awareness model based on a dynamic bayesian network and distributed genetic algorithm",

[19] A. Bender, G. Agamennoni, J. R. Ward, S. Worrall, and E. M. Nebot, "An unsupervised approach for inferring driver behavior from naturalistic driving data", *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3325–3336, 2015, ISSN: 1524-9050. DOI: `10.1109/TITS.2015.2449837`.

[20] J. Nilsson, M. Brännström, E. Coelingh, and J. Fredriksson, "Lane change maneuvers for automated vehicles", *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1087–1096, 2017, ISSN: 1524-9050. DOI: `10.1109/TITS.2016.2597966`.

[21] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition", *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989, ISSN: 0018-9219. DOI: `10.1109/5.18626`.

[22] M. Tipping, *Relevance vector machine*, US Patent 6,633,857, 2003.

[23] *Next generation simulation (ngsim) vehicle trajectories and supporting data*, `https://data.transportation.gov/Automobiles/Next-Generation-Simulation-NGSIM-Vehicle-Trajector/8ect-6jqj`, Accessed: 2018-09.

[24] J. A. Michon, "A critical view of driver behavior models: What do we know", *Human Behaviour and Traffic Safety*, Jan. 1985.

[25] A. Houenou, P. Bonnifait, V. Cherfaoui, and W. Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition", in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 4363–4369. DOI: `10.1109/IROS.2013.6696982`.

[26] T. Georgiou and Y. Demiris, "Predicting car states through learned models of vehicle dynamics and user behaviours", in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 1240–1245. DOI: `10.1109/IVS.2015.7225852`.

[27] Dogan, J. Edelbrunner, and I. Iossifidis, "Autonomous driving: A comparison of machine learning techniques by means of the prediction of lane change behavior", in *2011 IEEE International Conference on Robotics and Biomimetics*, 2011, pp. 1837–1843. DOI: `10.1109/ROBIO.2011.6181557`.

[28] L. R. Rabiner and B.-H. Juang, "An introduction to hidden markov models", *Ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[29] J. Bilmes, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models", Tech. Rep., 1998.

[30] C. J. Burges, "A tutorial on support vector machines for pattern recognition", *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[31] G. Zararsiz and E. Cosgun, "Introduction to statistical methods for microrna analysis", *Methods in molecular biology (Clifton, N.J.)*, vol. 1107, pp. 129–155, Jan. 2014. DOI: `10.1007/978-1-62703-748-8_8`.

[32] M. E. Tipping, A. C. Faul, *et al.*, "Fast marginal likelihood maximisation for sparse bayesian models.", in *AISTATS*, 2003.

[33] I. Psorakis, T. Damoulas, and M. A. Girolami, "Multiclass relevance vector machines: Sparsity and accuracy", *IEEE Transactions on neural networks*, vol. 21, no. 10, pp. 1588–1598, 2010.

[34] I. Cambridge Systematic, "Ngsim program us route 101 data analysis", 2005.

[35] ——, "Ngsim program interstate 80 data analysis", 2005.

[36] C.-W. Hsu, C.-C. Chang, C.-J. Lin, *et al.*, "A practical guide to support vector classification", 2003.

[37] B. H. Juang and L. R. Rabiner, "Hidden markov models for speech recognition", *Technometrics*, vol. 33, no. 3, pp. 251–272, 1991. DOI: `10.1080/00401706.1991.10484833`. eprint: `https://www.tandfonline.com/doi/pdf/10.1080/00401706.1991.10484833`. [Online]. Available: `https://www.tandfonline.com/doi/abs/10.1080/00401706.1991.10484833`.

[38] Chih-Wei Hsu and Chih-Jen Lin, "A comparison of methods for multiclass support vector machines", *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002, ISSN: 1045-9227. DOI: `10.1109/72.991427`.

[39] J. Weston and C. Watkins, "Multi-class support vector machines", Citeseer, Tech. Rep., 1998.

[40] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection", Morgan Kaufmann, 1995, pp. 1137–1143.

[41] T. Kumagai and M. Akamatsu, "Modeling and prediction of driving behavior", in *Proc. IMEKO/IEEE/SICE 2nd Intl. Symp. Measurement, Analysis, and Modeling of Human Functions*, 2004, pp. 357–361.

[42] T. Damoulas and M. A. Girolami, "Combining feature spaces for classification", *Pattern Recognition*, vol. 42, no. 11, pp. 2671–2683, 2009.

[43] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems", *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970. DOI: 10.1080/00401706.1970.10488634. eprint: https://www.tandfonline.com/doi/pdf/10.1080/00401706.1970.10488634. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00401706.1970.10488634.

[44] M. Awad and R. Khanna, "Support vector regression", in *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Berkeley, CA: Apress, 2015, pp. 67–80, ISBN: 978-1-4302-5990-9. DOI: 10.1007/978-1-4302-5990-9_4. [Online]. Available: https://doi.org/10.1007/978-1-4302-5990-9_4.

[45] A. C. Faul and M. E. Tipping, "Analysis of sparse bayesian learning", in *Advances in neural information processing systems*, 2002, pp. 383–389.

[46] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine", *Journal of machine learning research*, vol. 1, no. Jun, pp. 211–244, 2001.