# Prediction of vehicles' trajectories based on driver behaviour model

Zhengyu Li

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

# Prediction of vehicles' trajectories based on driver behaviour model

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Zhengyu Li

August 14, 2014

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology


# Abstract

As a component of Dutch Automatic Vehicle Initiative (DAVI) project, this study aims at improving highway driving safety of autonomous vehicle. It is observed that some misbehaved drivers do not use turn indicators forehead a lane change on highway. For a self-driving car in such situations, a lane change is potentially dangerous without an accurate estimation of other vehicles' movements .If lane changes can be detected or predicted at its initial phase, DAVI vehicle can be advised in time to perform corresponding maneuvers to avoid collision. In this study, a Support Vector Machine (SVM) based method was proposed to fulfil this task. Predictions are generated through analysing vehicles' motion parameters.

At the first place, a number of driving simulator tests were carried out to set up database of typical vehicle maneuvers features, including heading angle, yaw rate, lateral velocity and lateral acceleration. 14 driver databases were used in SVM training after removal of unrealistic info.

An off-line trained SVM was obtained to perform lane change predictions. In addition to off-line training, an incremental training SVM was introduced. Compared with off-line learning, its incremental feature enables the classifier to update itself with freshly recorded lane change data in a relatively short time. This enables the vehicle to be "learning while predicting" .

Based on the databases, SVM based approaches were verified to be feasible of predicting lane changes. With the most optimal parameter combination, this method is able to perform predictions with 100% sensitivity ( predicted all lane changes successfully). Average advance time is approximate to 1.5 seconds. Average computational time is less than 0.005 seconds, which is acceptable for automatic driving. Besides, the performance of sliding window method was evaluated for variation of its size, and a general applicability of overall prediction method was also examined on data from different drivers.

# Table of Contents

**Acknowledgements** **ix**

**1 Introduction** **1**
1-1 Goal and motivation . . . 1
1-2 State of art . . . 2
1-3 Thesis overview . . . 4

**2 Driving simulator test** **5**
2-1 Simulation environment construction . . . 5
2-2 Data analysis . . . 7
2-2-1 General description . . . 7
2-2-2 Data importation . . . 8
2-2-3 Lane change data . . . 9

**3 SVM based prediction method** **15**
3-1 General description . . . 16
3-2 Multi-class SVM . . . 18
3-3 Incremental SVM training . . . 20
3-3-1 Karush-Kuhn-Tucker Conditions . . . 21
3-3-2 Adiabatic Increments . . . 21
3-3-3 Incremental learning algorithm . . . 22

**4 Implementation** **25**
4-1 Before training . . . 25
4-2 Choice of kernel function and parameter selection . . . 26
4-3 Prediction and results . . . 27
4-3-1 Generating ground truth . . . 28
4-3-2 Point by point prediction . . . 29
4-3-3 Sliding window . . . 30
4-4 Data features selection . . . 31

**5 Validation results** **33**
5-1 Set 1 . . . 34
5-1-1 Off-line training . . . 34
5-1-2 Incremental learning . . . 36
5-2 Set 2 . . . 40
5-2-1 Off-line training . . . 40
5-2-2 Incremental learning . . . 41
5-3 Summary . . . 45

**6 Conclusions and Future Work** **49**
6-1 Conclusions . . . 49
6-2 Future works . . . 50

**A SVM related problems** **51**
A-1 Probability estimation . . . 51
A-2 Dual problem . . . 52

**B Validation related** **53**
B-1 Prediction results and ground truth . . . 53

**Bibliography** **57**

**Glossary** **61**
List of Acronyms . . . 61

# List of Figures


2-1 Driving simulator . . . . . 6
2-2 Road design software . . . . . 7
2-3 Descriptions of "Fleet" . . . . . 8
2-4 Example Vehicle trajectory . . . . . 8
2-5 Window for extracting lane change data . . . . . 9
2-6 Example on extracting lane change data . . . . . 10
2-7 Lane change left (LCL) groups . . . . . 11
2-8 Lane keeping (LK) groups . . . . . 11
2-9 Lane change right (LCR) groups . . . . . 12
2-10 Histograms of LCL at varying time steps . . . . . 12
2-11 Changes of LCL parameters with respect to time . . . . . 13
2-12 Histograms of LCR at varying time steps . . . . . 13
2-13 Changes of LCR parameters with respect to time . . . . . 14

3-1 Hyperplane of SVM . . . . . 16
3-2 One vs. One . . . . . 18
3-3 One vs. All . . . . . 19
3-4 DAG Classification . . . . . 19
3-5 Different groups of vectors . . . . . 21

4-1 Contour map used in finding the best parameter combination . . . . . 27
4-2 Self test results . . . . . 27
4-3 Lateral position and ground truth . . . . . 28
4-4 Point by point applied on single LCL . . . . . 29
4-5 Point by point prediction output and ground truth . . . . . 30
4-6 Sketch of sliding window approach . . . . . 30

4-7 Results of sliding window size 5 . . . . . 31

5-1 Prediction of a single LCL . . . . . 35
5-2 Prediction of a single LCL with online trained classifier . . . . . 37
5-3 Prediction of a single LCL . . . . . 41
5-4 Prediction of a single LCL with online trained classifier . . . . . 43
5-5 Performance evaluation varying window size – Scenario 1 . . . . . 45
5-6 Performance evaluation varying window size – Scenario 2 . . . . . 46
5-7 Performance evaluation varying window size – Scenario 3 . . . . . 46
5-8 Performance evaluation of each scenarios with sliding window size 3 . . . . . 47

# List of Tables


1-1 Performance comparison of different methods . . . . . 3

2-1 Recorded parameters . . . . . 7

4-1 Grid search results . . . . . 26
4-2 Performance comparison in Off-line training . . . . . 32
4-3 Performance comparison in on-line training . . . . . 32

5-1 Computer configuration . . . . . 34
5-2 Off-line training performance . . . . . 34
5-3 Prediction performance on single LCL . . . . . 35
5-4 Prediction performance on continuous trajectory . . . . . 36
5-5 Incremental training performance . . . . . 36
5-6 Predict results for single LCL . . . . . 36
5-7 Prediction performance on continuous trajectory with Inc. classifier . . . . . 37
5-8 Incremental training performance . . . . . 38
5-9 Predict results for single LCL . . . . . 38
5-10 Prediction performance on continuous trajectory with Inc. classifier . . . . . 38
5-11 Comparison between different learning methods - Set 1 . . . . . 39
5-12 Off-line training performance . . . . . 40
5-13 Prediction performance on single LCL . . . . . 40
5-14 Prediction performance on continuous trajectory . . . . . 40
5-15 Incremental training performance . . . . . 41
5-16 Predict results for single LCL . . . . . 42
5-17 Prediction performance on continuous trajectory with Inc. classifier . . . . . 42
5-18 Incremental training performance . . . . . 42

5-19 Predict results for single LCL . . . . . 42
5-20 Prediction performance on continuous trajectory with Inc. classifier . . . . . 43
5-21 Comparison between different learning methods in Set 2 . . . . . 44

B-1 Prediction performance comparison - Driver #4 . . . . . 54
B-2 Prediction performance comparison - Driver #8 . . . . . 55
B-3 Prediction performance comparison - Driver #11 . . . . . 56

# Acknowledgements

This thesis is the result of my master study in Delft Center for Systems and Control (DCSC) at Delft University of Technology (TU Delft). I would like to thank all people who supported me both technically and morally during my two years study. Without your help it would not have been possible for me to accomplish this far.

My thesis committee guided me through this project. I would like to express my sincere gratitude to prof.dr.ir. J. Hellendoorn, dr.ir. M. Mazo Jr. and dr. R. G. Hoogendoorn for their advices and practical support. Their patience and support helped me overcome challenges and finish this dissertation.

I am grateful to all staff members and colleagues in DCSC. Their friendship and assistance make this international experience unforgettable. Further I want to thank all participants took part in the driving simulator test for their time and patience. I want also thank my all friends for their company and great support.

Finally I would like to thank my family for all their love and encouragement. It is hardly be expressed in words how grateful I am for their constant guidance and support, not only financially but also emotionally. My parents have been a constant source of inspiration and this work is especially dedicated to them.

Delft, University of Technology — Zhengyu Li
August 14, 2014

# Chapter 1

# Introduction

This project is a component of Dutch Automatic Vehicle Initiative (DAVI) project. DAVI project focuses on developing, improving and demonstrating self-driving vehicles in Netherlands [1]. The DAVI project has been initiated by Delft University of Technology (TU Delft) , RDW, Connket, TNO and Toyota Motor Europe.

## 1-1 Goal and motivation

Driving brings people freedom and independence. However, apart from the rejoice it brings, traffic safety is an aspect that one can not neglect. Driving safety has been a major concern from the day vehicle was invented. Besides,with developments achieved in automatic driving in recent years, traffic safety of self-driving vehicle has drawn increasing attention from different fields.

Drivers need to interact with each other continuously while driving. Their individual behaviour will have a significant influence on traffic safety. Driver's behavior can be regarded as a combination of decisions and actions. A qualified driver will take proper actions based on accurate forecasts of traffic situation. The same goes for automatic driving. A fully automated vehicles should be capable of ensuring driver's security and comfort while maneuvering through traffic flow. In order to improve self-driving safety, a prediction method is vital. This method should not only be able to provide precise prediction in a sufficient time in advance, but also be satisfactory in efficiency and reliability.

Lane change is one of the most commonly executed actions in driving. A side-by-side collisions might occur if a driver performs improper lane change. According to a statistical result in 2008 [2], side-by-side collision take 5% of total injury accidents in EU. So it is essential to find a method that can provide predictions of surrounding vehicles' potential lane changes. This study will focus on the initial phase of a lane change. The self-driving vehicle need to be able to predict future movements of other vehicles at early stage: Are they going to perform a lane change? If yes, on which direction? Only in this way, the DAVI vehicle can be prepared for lane changes and would be able to perform corresponding actions to avoid possible collision.

Thus, the problem statement of this project can be reached:

*Predict short term future lane change maneuvers of other drivers, based on the externally attitude and movements observed from surrounding vehicles.*

Apart from this project, there are many sub-projects, focus on different driving related purposes and vehicle components ongoing in DAVI project. In order to make them interact with each other reliably, a contract-based design methodology is needed. This contract-based composition requires each subsystem of a bigger design provide solid interfaces: assumptions and guarantees. Each subsystem should be able to connect with others through matching its assumptions with guarantees provided by others. The assumptions and guarantees of this specific project are listed as follows:

**Assumptions:**

- Sensor data will be continuously sampled. There will be no distortion in transferring data to processors so that current traffic situation can be represented accurately.
- The variety and amount of parameters contained in the sensor data are sufficient to conduct a proper prediction. e.g. vehicle position, velocity, heading angle,etc.
- The pre-recorded data that will be used in training procedure is sufficient to capture driving behavior in real traffic scenario.

**Guarantees:**

- A predictive result will be generated via an prediction algorithm in advance of the lane change actually occurs with respect to the externally observable cues acquired from surrounding vehicles.
- Computational time will be small enough to guarantee driving safety.

## 1-2 State of art

Several studies have been done to predict possible lane changes. Different methods such as Bayesian Network, Support Vector Machine, Hidden Markov Model, Mind-tracing and Fuzzy Logic were proposed to solve this problem.

Bayesian Network can be regarded as a graphical representation of probability. In the work provided by Dagli et al. [3] and Tezuka et al. [4], Bayesian Network is used in recognizing drivers' intended actions. Lateral movement of a vehicle is represented in Bayesian Network through multiple probability nodes. The probability distribution of each node is determined by off-line analysis of driver behaviour data. Finally, the prediction is done by computing a probability of a certain maneuver with respect to the possibility of every related nodes.

Pentland et al. [5, 6] considered driving process as a collection of different driving states. As actions will occur in a particular sequence, the likeliness of a state change can be estimated. The probability of driving states transitions is modelled with Hidden Markov Model (HMM). Oliver et al. [7, 8] further extended their work with a more realistic test scenario in which a vehicle that equipped with sensors was employed.

Mind-tracing method is proposed by Salvucci et al. [9]. It is a computational framework that is capable of predicting possible intentions. Different versions of cognitive model that contain streams of possible intentions and actions are compared with human behaviour simultaneously. The closest one will be used for further inferring divers' intentions.

According to Hou et al. [10], fuzzy logic can be an alternative method to model driver behaviours. Their work focused on the decision making process of drivers at lane drop sections on a highway. A triangular membership function was applied in their study. Fuzzy rules were generated via learning from examples training procedure. The output was evaluated

Mandalia et al. [11] evaluated the soundness of applying Support Vector Machine (SVM) in lane change prediction. After proper cues that related to lane changes were chosen, recorded trajectory data can be divided into different groups. With these groups a classifier can be trained. Predictions were made through classification of cues at present.

However, comparing the performance of these methods is a complicated task. The results are measured in different situations. The resulting accuracies given in related papers are also measured at different time steps. For instance, a prediction accuracy at two seconds before vehicle's lateral position overlaps with the lane border is given in one paper. However, in other publication only the accuracy at one seconds before "lane crossing moment" is provided. Moreover, the time of "lane crossing"" is commonly defined as the moment vehicle crosses lane edge, but in some publications the definition is rather vague.

In order to draw a comparison, the best accuracies of each methods are presented in the table. Please note that it is not possible to compare them at the same time step. All methods are tested in different simulation environments with different data sets. The results listed can only be a reference in determine which method is the most suitable for our project.

**Table 1-1:** Performance comparison of different methods

| Method | Author | Best accuracy |
|---|---|---|
| Bayesian Network | Dagli et al.[3] | 80% @ 1.5s before lane crossing |
| | Tezuka et al.[4] | 89% @ 0.5s after lane change starts |
| SVM | Mandalia et al.[11] | 87% @ 0.3s after lane change starts |
| Hidden Markov | Pentland et al.[5] | 89.4% @ 2s after lane change starts |
| | Pentland et al.[6] | 95.2% @ 2s after lane change starts |
| | Oliver et al.[7, 8] | Unknown @ 0.4s before any sign occurs |
| Mind-tracing | Salvucci et al.[9] | 82% @ 1.1s before lane crossing |
| Fuzzy Logic | Hou et al.[10] | 86.8% @ unknown |

Table 1-1 shows a comparison of all the methods. More detailed introductions on these methods are listed in literature survey report [12].

The results suggest Bayesian network and SVM seems to be suitable for lane change prediction. They are able to predict the lane change in a relative shorter time thus will lead to a longer advance time.

A SVM based method could be a suitable candidate for classifying driver behaviours for a number of reasons.

- Firstly, a SVM method performs well while handling large amount of data.

- Secondly, SVM yields high accuracy results for problem with many features. It is necessary to incorporate different physical parameters as features while describing drivers' actions. Conventional statistical classifiers might be inadequate while processing high dimensional data.
- Moreover, SVMs are robust to over fitting problem since they rely on margin maximization rather than finding a decision boundary directly from the training samples.
- Finally, in recent studies, incremental training is proved to be possible for SVM method [13]. Thus this prediction method is able to update itself continuously. So far no attempts have been done introducing incremental SVM in predicting lane changes. The applicability will be examined in this paper.

## 1-3 Thesis overview

This study focuses on developing and evaluating a SVM based prediction algorithm. This algorithm uses external cues to predict surrounding vehicles possible lane changes.

This thesis is organised as follows:

In Chapter 2, a driving simulator test is introduced. This simulator test is held to gather drivers' driving behaviours. Four parameters related to lane changes were recorded in 25 tests. All unrealistic data were discarded in a following filtering procedure. Remaining data set contains 14 drivers' driving data.

Chapter 3 contains mathematical introductions on binary SVM classifier and how it is modified into a multi-class SVM for implementation. To breakthrough the limitation brought by applying an off-line training method, an incremental training approach is proposed.

Implementation details concerning are presented in Chapter 4. In order to further improve the accuracy of SVM method, procedures concerning parameter and kernel selection for SVM are provided. Moreover, sliding window approach is introduced to reduce appearance of false positives.

In Chapter 5, validation results as well as performance comparisons between off-line trained and incrementally trained classifiers are listed . In addition, this prediction method is applied on another set with different configuration to prove the general applicability.

Chapter 6 winds up this thesis with conclusions and future works based on the findings of previous chapters.

# Chapter 2

# Driving simulator test

In order to perform reliable predictions, it is essential to observe the behaviours of different drivers and analyse them to find clues that may indicate an upcoming lane change maneuver. Moreover, it is also very important to inspect how different drivers will behave in similar traffic situations. To fulfil these objectives, a driving simulator test is designed. In this test, a virtual highway scenario is carried out. While volunteer participants driving in this scenario, several physical parameters related to lane change maneuver are recorded for further assessment. After this test, a database containing several drivers' driving styles is constructed.

The benefit of employing driving simulator test is its high controllability. It is possible to present the same traffic scenario to different participants with the lowest costs compare to other test methods. Also, tests can be held without limitation on weather, availability of test fields and safety concerns. Thus the tests can be completed with high efficiency. Further, the test can be repeated for as many times as needed.

The manufacturer of this driving simulator is ST software B.V., located in Groningen, the Netherlands. This simulator is a property of Faculty of Civil Engineering and Geosciences, Delft university of technology. There are several components equipped in this simulator: a road designing software, a traffic scenario designer and a driving cabin. In road designing software, users can construct desired road structures as well as surrounding environment. The traffic scenario designer is a script-based programmer. Users can define a certain traffic scenario and specify movements of each vehicle presented in it. Finally, the driving cabin is designed to approximate the interior structure of a Volkswagen vehicle. There are three LCD displays set in front of the driver to present the virtual world. Figure 2-1 shows the structure of this simulator.

## 2-1 Simulation environment construction

It is unavoidable that the driving feedback of a simulator is different from a real vehicle. In order to improve the reliability of recorded data, a "test drive" is introduced at the beginning of each experiment. In this "test drive", participant will be asked to drive on a virtual training

**Figure 2-1:** Driving simulator

track with the same setting as real tests. This training track is designed for participants to get familiar with simulator operations. Participants need to drive on this track for at least one lap before start the real testing track. The geometry parameters of this training track is designed based on a real racing circuit, in which the diver will have opportunities to practice all fundamental maneuvers like acceleration, braking, handling road curvatures and intersections. It will take approximate 10 minutes to finish this training.

A screen shot of road design software's user interface is presented in Figure 2-2. The road structure designed for this test is displayed at the bottom right window in Figure 2-2. Those solid black lines are the highways built for test. The green parts, or polygons, are "tiles" that are graph layers planted beneath road structures. These "tiles" generate road side grassland that enhance driving experience as they give participants more realistic feeling.

At the initial part, test "vehicle" will be set at the bottom left section of the map, where is constructed as an urban area. In this part there are some intersections and turns for the participant to "warm-up". After several turns the driver will be guided to a highway section, where pre-defined traffic scenarios appear and recording initiates. The objective of participant is to drive through traffic to reach the end of the highway. It will take approximate 15 minutes to finish this test.

The traffic on highway is defined through different "fleets". An example of a "fleet" is presented in Figure 2-3. Vehicles with color black are pre-allocated. Their appearance positions, velocity and behaviours are specified through scripts in ST Control software. It should be noted that for different "fleets", vehicles' speed and leading distance varies. It is designed in this way to observe how one driver react to different traffic scenarios.

The vehicle in blue color represents the testing vehicle. Since the velocity of pre-defined vehicles are set to be slower than the maximum speed, it is possible for test subjects to overtake other vehicles in each "fleet" . These overtake maneuvers can be regarded as a combination of Lane Change Left (LCL), Lane Keeping (LK) and Lane Change Right (LCR).

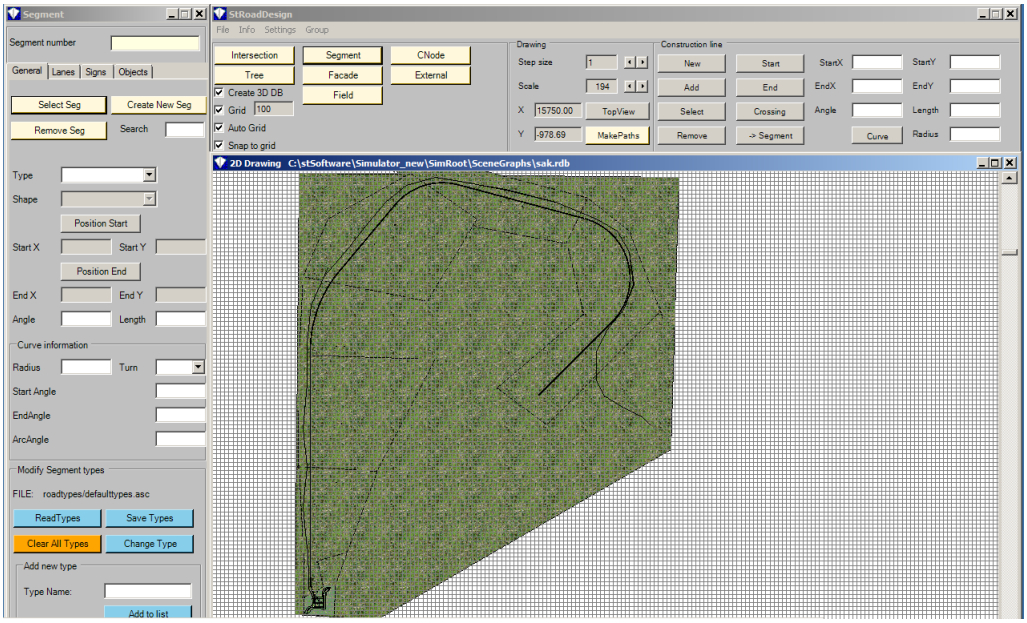


**Figure 2-2:** Road design software

Thus, the next step is to extract each maneuver from recorded overtaking related data.

## 2-2 Data analysis

### 2-2-1 General description

25 participants took part in this driving simulator test voluntarily. As described in previous section, each participants' driving data is recorded from starting point to the end point of highway. All recorded parameters are listed in Table 2-1. An example of recorded trajectory is presented in Figure 2-4. Vehicle moves in direction from left to right. The upper lane is

**Table 2-1:** Recorded parameters

| Parameters | Description |
|---|---|
| Time | Current time step (s) |
| Lateral position | Lateral distance between center of front bumper of testing vehicle and the center line of the rightmost lane. The lane width is set as 3.6 meters. (m) |
| Heading | Heading of vehicle with respect to the road (degree) |
| Yaw rate | Rotation speed of vehicle longitudinal axis (rad/s) |
| Lateral velocity | Lateral velocity of test vehicle. If vehicle moves to LEFT: positive values. If vehicle moves to RIGHT: negative values. (m/s) |
| Lateral acceleration | Lateral acceleration computed as yaw rate * longitudinal velocity of test vehicle ($m/s^2$) |
| Velocity | Current velocity of test vehicle (m/s) |

**Figure 2-3:** Descriptions of "Fleet"

left lane and lower one is right lane.

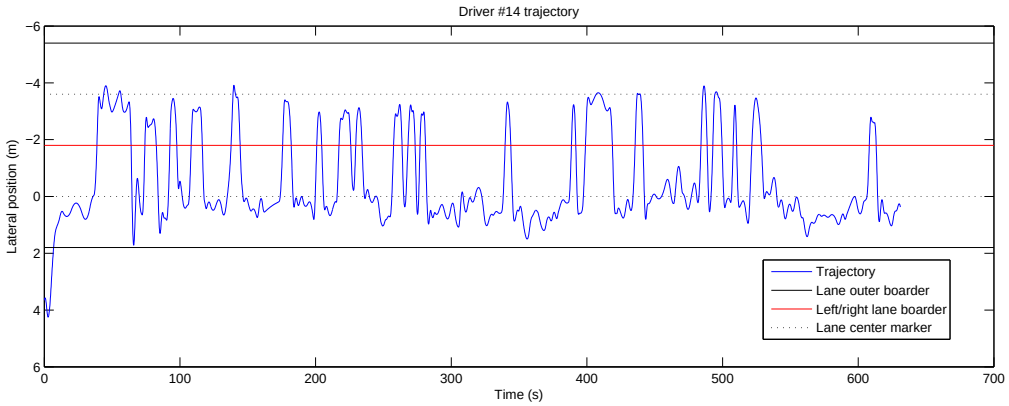


**Figure 2-4:** Example Vehicle trajectory

In order to extract each lane change maneuvers from the database. A lateral position based criterion is used. This criterion defines an actual lane change occurs when the lateral position of test vehicle overlaps with the boarder that separates left lane and right lane. Further illustration is shown in Figure 2-5. In this figure, $T1$ is the time before a lane change and $T2$ means the time after the lane change happens. All the data points in this window are defined as a lane change left (LCL) maneuver.According to Toledo's study on average lane change duration[14], in this project, $T1$ is set as 3.5 seconds and $T2$ is 1 second. The total size of this window is 4.5 seconds.

### 2-2-2 Data importation

All data is stored in a local disk drive that connected to the simulator. The data is recorded in ACS-II format and saved as DAT files. The first step is to import these data into Matlab

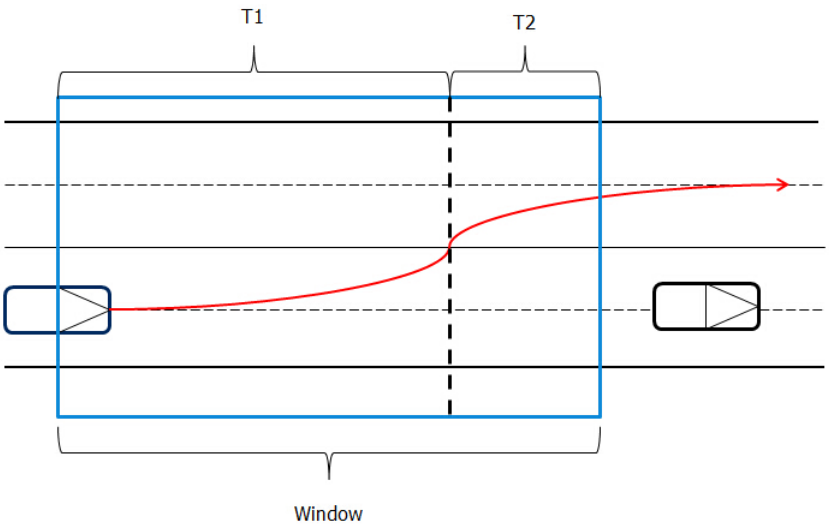


**Figure 2-5:** Window for extracting lane change data

for further analysis. During this importation, the information concerning measurement on lateral position, heading angle, yaw rate, lateral velocity and lateral acceleration are imported to workspace.

An important remark need to be made here is that it is essential to filter the raw data before.

Even all test drivers will have a "test drive" before the test, it is still observed that participants performed unrealistic maneuvers during the experiment. This phenomenon is possibly caused by the differences in traffic situation observation and interpretation between real world driving and simulation. Some drivers complete lane change in less than 1 second, which too rapid to be real. It is also occurred that a driver drives off the road or crashes into surrounding vehicle.

In the importation part, all the data with off-roads driving, crashes are discarded. Lane change time less than 2 seconds are also filtered out in the database. After this procedure, there are 14 drivers meets the selection criterion in the database and will be used in further assessment.

### 2-2-3 Lane change data

With the criterion explained in previous section, one get filter out every lane change sections, as shown in Figure 2-6. There are in total 273 LCL , 222 LK and 253 LCR samples can be extracted from the remaining 14 driver database.

Figure 2-7, 2-8 and 2-9 present the variations of different parameters in each lane change groups.

As shown in these figures, the actual lane change take place at 3.5 second. For a rough inspection, it can be observed that there exists an increasing value of heading which starts at around 2 second and reaches its peak at 3.5 second. Similar increasing or decreasing in parameter values can be seen in yaw rate, lateral velocity and lateral acceleration as well.

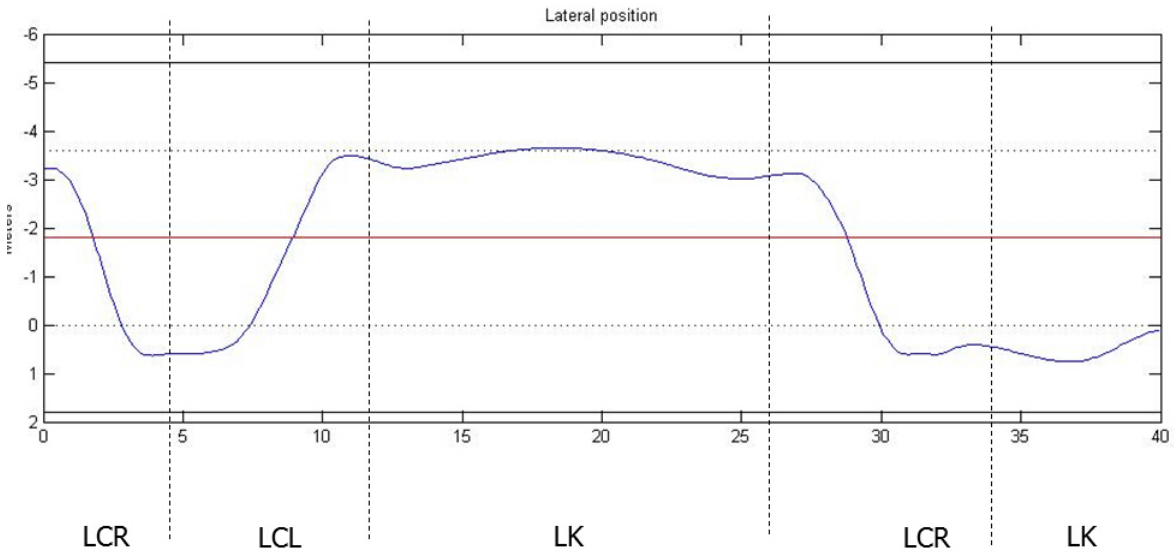


**Figure 2-6:** Example on extracting lane change data

For an in-depth investigation, histograms of different parameters can be drawn. Figure 2-10 and Figure 2-12 give the histograms of LCL and LCR respectively. Please keep in mind that the lane change occurs at T=3.5s. A decreasing in values of heading angle and lateral velocity (listed in column 1 and column 3 respectively) at increasing time steps (from 1.5 second to 3.5 second with step of 0.5 second) can be observed. To be more illustrative, the mean value of each parameters in LCL is computed every 0.2 second from 1.5 second to 3.5 second. The way these parameters change over time is presented in Figure 2-11. Similarly, Figure 2-13 presents the parameters of LCR. Axes on the left side of Figure 2-11 and Figure 2-13 are for heading angle, lateral velocity and lateral acceleration. Those on right side are for yaw rate. Figures are presented in this way because the absolute value of yaw rate is relatively small compared with other three features.

The distributional features of these parameters suggest it might be possible to classify these lane changes with these externally observed vehicle physical parameters used as indicators. The values of heading angle, yaw rate, lateral velocity and lateral acceleration can be used to train a classifier. Detailed description of classifier algorithm and implementations are presented in following chapter.

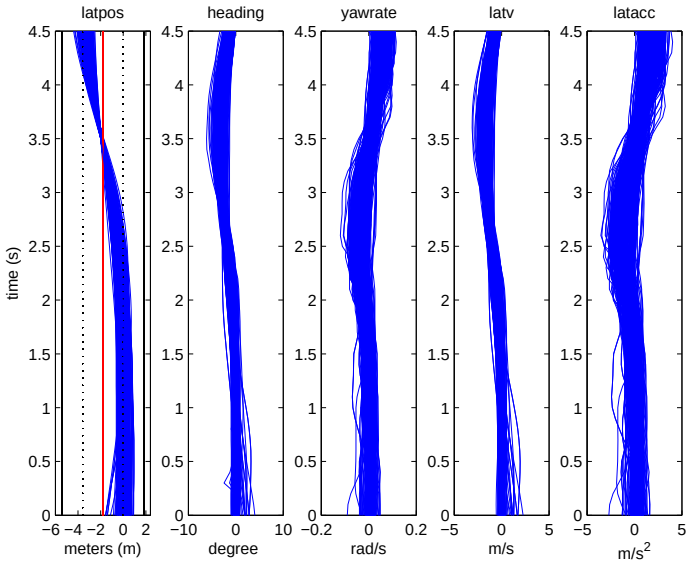


**Figure 2-7:** Lane change left (LCL) groups

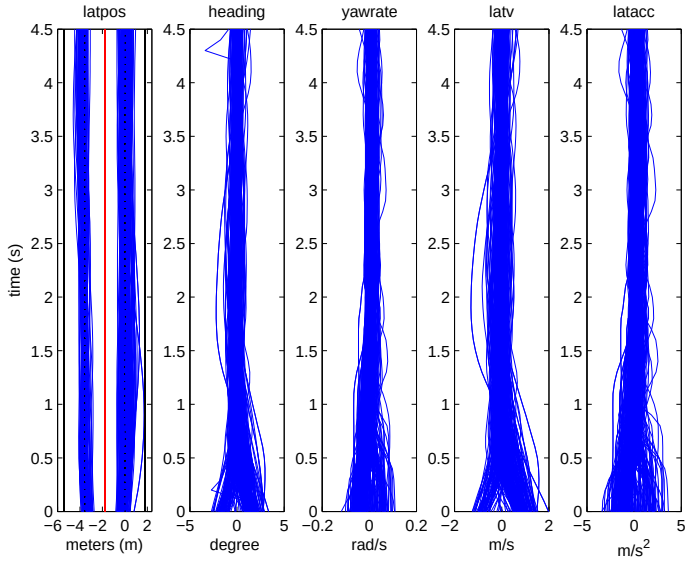


**Figure 2-8:** Lane keeping (LK) groups

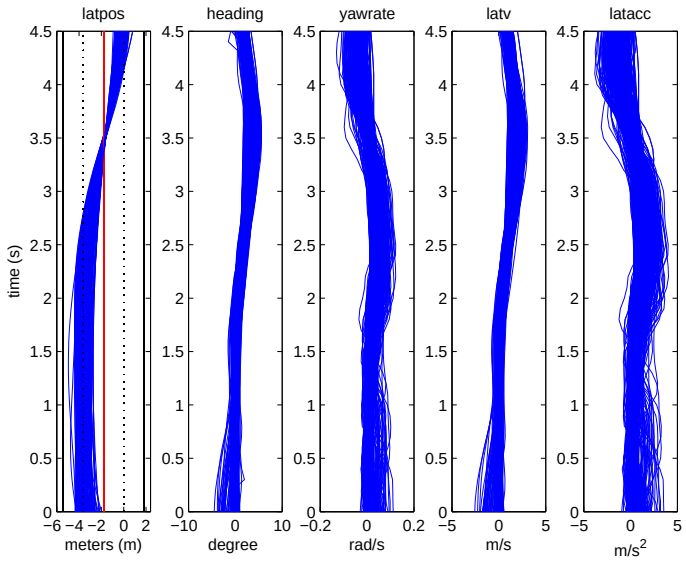


**Figure 2-9:** Lane change right (LCR) groups

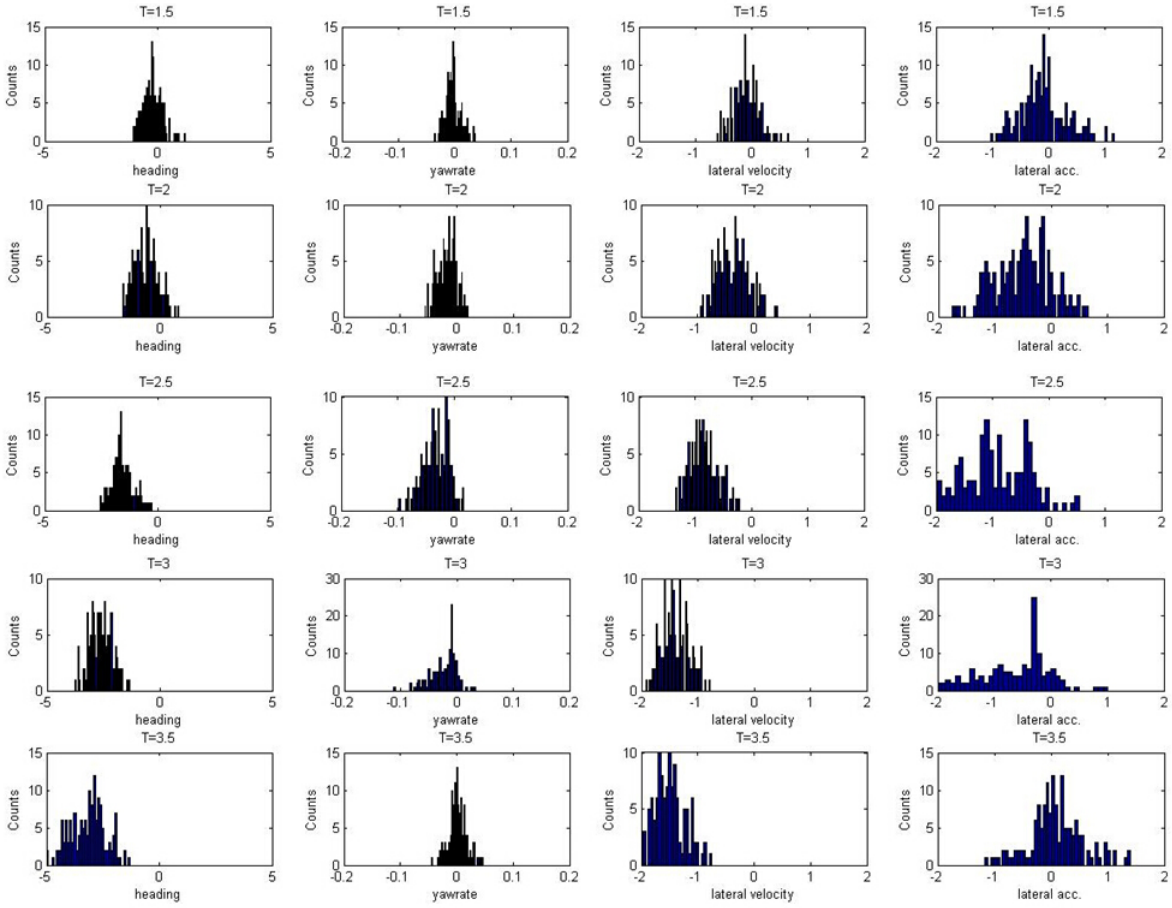


**Figure 2-10:** Histograms of LCL at varying time steps

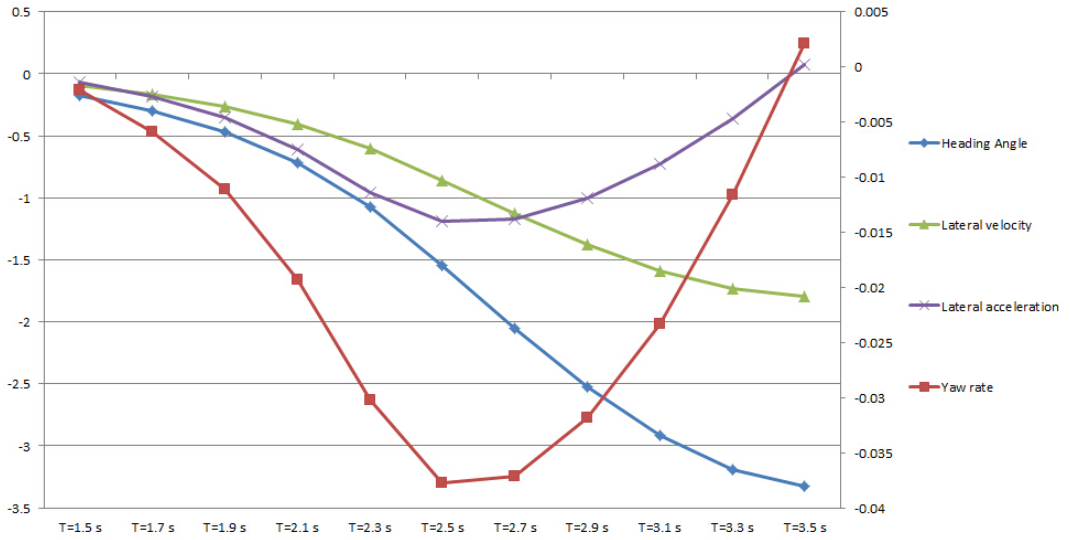


**Figure 2-11:** Changes of LCL parameters with respect to time

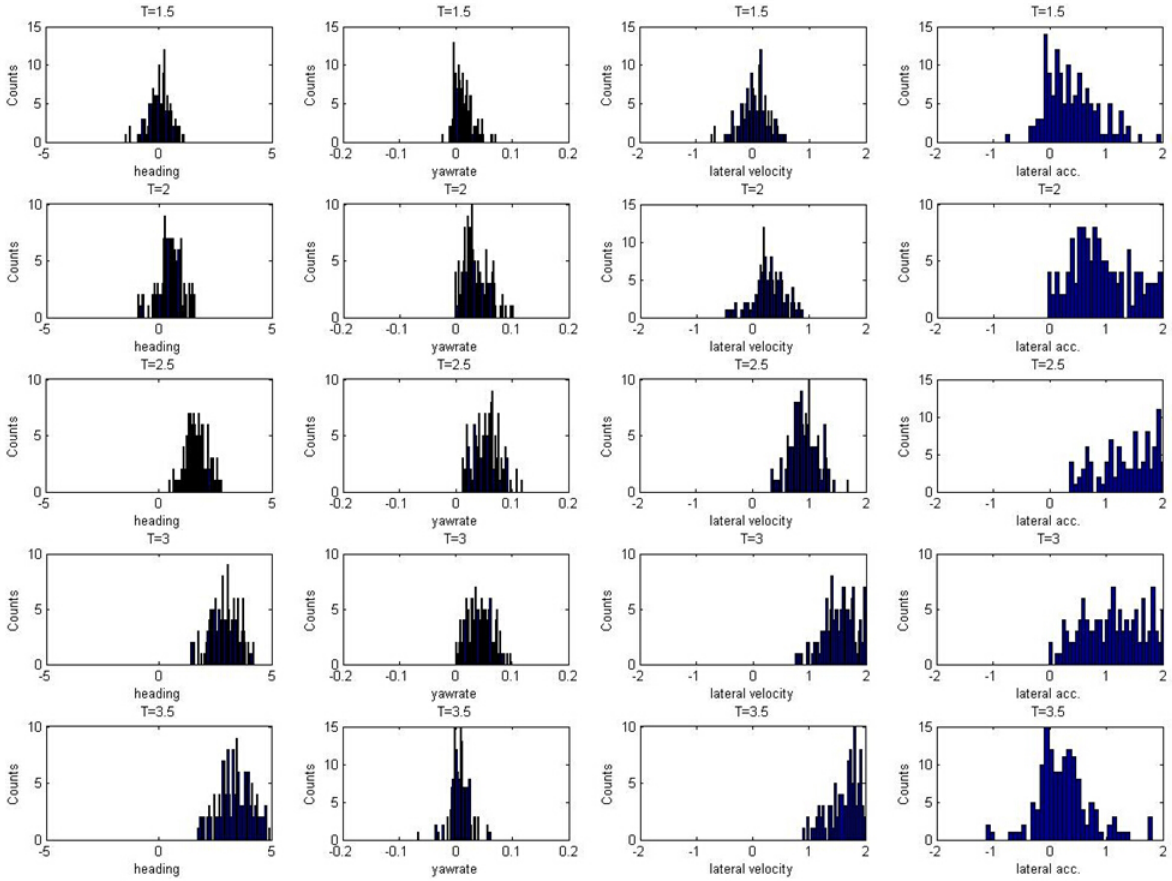


**Figure 2-12:** Histograms of LCR at varying time steps

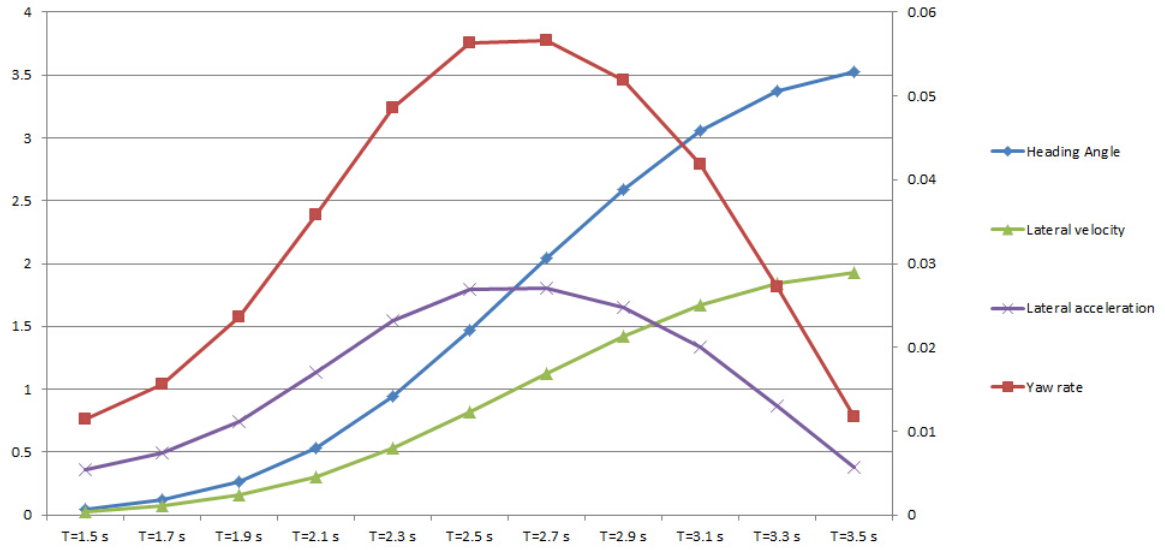


**Figure 2-13:** Changes of LCR parameters with respect to time

# Chapter 3

# SVM based prediction method

As illustrated in previous chapter, all recorded trajectories are divided into three groups: lane change left (LCL), lane keeping (LK) and lane change right (LCR). For the purpose of prediction, a method that is capable to infer one driver's upcoming maneuvor based on his/her previous actions is critical. Inferences need to be made with those recorded parameters regarded as cues of a lane change. In practice, all the newly updated data need to be categorized into those three groups. Thus, this problem boils down to a classification problem in which each class represents a corresponding action group.

A classifier need to be chosen so that this project can move on to the next step. However, there are numbers of different classifiers to choose from. Commonly used ones are Support Vector Machine (SVM), Naive Bayes (NB), Hidden Markov Model (HMM) etc.. In this project, the SVM classifier is chosen for further implementation with following motivations:

- From Chapter 2 it can be seen that several physical parameters are used to describe the movements of a vehicle. In order to make a successful prediction, a classifier that can allocate data points into correct groups using these parameters is needed.
- SVM method is able to deal with large amount of input data, which is suitable for this project since there will be numbers of recorded drivers' behavior.
- The solution given by SVM method is global optimal, since the objective function of SVM is a convex function.
- According to different literatures, for instance in the work by G.S. Aoude [15], the SVM method outperforms the HMM method. It gives better performance and higher accuracy when predicting drivers' behaviour in an ADAS problem. In [16] the HMM method achieved approximate 70% detecting lane changes while in [17], using the same sensors data, a SVM 87% of accuracy.

## 3-1 General description

Support Vector Machines (SVM) is a kernel based classification technique. This method is first introduced by Cortes and Vapink in 1995 [18]. This machine learning approach becomes popular quickly and has been applied in various fields such as speech , video and hand writing recognition. SVM is a powerful tool in classification and regression area.

The core concept of this method is to find a hyperplane that separates data points into different groups in a hyperspace. If a data set is linearly inseparable, a kernel function can be applied to map these data into a higher dimensional space and solve it as a linear problem.

The general idea of SVM algorithm is a linear learning machine based on the principle of optimal separation of classes. It is designed to solve classification problem initially. With training data obtained, the objective of this method is to divide these data into different groups by finding a separating hyperplane. This hyperplane, defined as a plane in a multidimensional space, is often referred as a decision surface or an optimal margin hyperplane. Also, SVM is capable of classifying data adequately on a high dimensional feature space with a limited number of training data [18].

The sketch of this concept is presented in Figure 3-1.

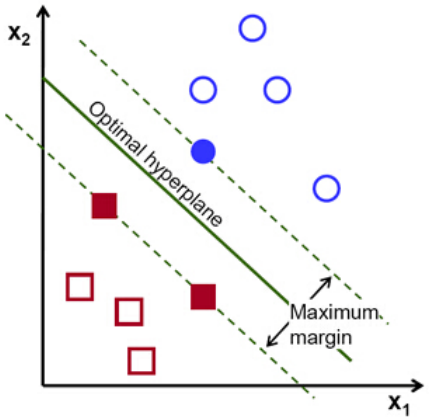


**Figure 3-1:** Hyperplane of SVM

However, a linear separating hyperplane is not sufficient to make separation under some practical circumstances, especially those in which each data sample has more than 2 features. All data need to be transformed into a higher dimensional space in which a new hyperplane can be found. This transformation is done via a kernel function.

A kernel function is defined as $K(x, x_i)$. It is used in finding a general mathematical description of the linear hyperplane, as stated in following equation.

$$f(x) = \sum_{i=1}^{k} c_i K(x, x_i) \tag{3-1}$$

Where $x_i$ is the input vectors with features. $K$ stands for the kernel function and $c$ is a set of parameters to be determined base on inputs.

A linear SVM can be described using a set of training data points $(x_i, y_i)$, with $i = 1, 2, ..., n$ denotes the number of data points. $y_i \in \{-1, 1\}$ indicate the groups $i$th vector belongs to. The goal is to find a hyperplane that has the maximum margin between two classes. An arbitrary hyperplane that separates data points into two classes is provided in Equation 3-2.

$$f(x) \equiv W \cdot x + b = 0 \tag{3-2}$$

Where $W$ is the normal vector of the hyperplane; $x$ is a set of input vectors; $b$ is the perpendicular distance between original and the plane. With the description of initial hyperplane obtained, constraints can be attained by varying $W$ and $b$.

$$x_i \cdot W + b \geq +1 \qquad \text{when} \quad y_i = +1 \tag{3-3}$$
$$x_i \cdot W + b \leq -1 \qquad \text{when} \quad y_i = -1 \tag{3-4}$$

These equations can be summarized into one inequality constraints:

$$y_i(x_i \cdot W + b) - 1 \geq 0 \quad \forall\, i \tag{3-5}$$

Thus, a general mathematical description of SVM method can be concluded. For general, define $S = \{(x_1, y_1), ...(x_m, y_m)\} \subseteq X \times Y$ with $m$ data samples. $x$ is training examples as $y$ represents corresponding label. An optimization problem, described in Equation 3-6, that aims at finding a hyperplane need to be solved.

$$\min_{W,\zeta \geq 0, b} \frac{1}{2} W^T W + C \sum_{i=1}^{m} \zeta_i \tag{3-6}$$

$$\begin{aligned} s.t. \qquad & y_1(W^T x_1 + b) \geq 1 - \zeta_1 \\ \vdots \quad & \\ & y_m(W^T x_m + b) \geq 1 - \zeta_m \end{aligned}$$

$W$ are parameters of the hyperplane;$b$ being intercept term; $C$ as the regularization parameter and $\zeta$ stands for slack variable. $\zeta$ reforms the SVM from a hard margin classifier into a soft margin one that tolerates training errors and operates robustly.

To solve the optimization problem, it can be simplified into a dual form. Since $W$ is a linear combination of the vectors of the training set, one get $W = \sum_{i=1}^{N} y_i \alpha_i x_i$ [19]. So the dual form can be written as Equation 3-7 with Lagrange multiplier $\alpha$ and $Q_{ij} = y_i y_j \Phi(x_i) \cdot \Phi(x_j)$.

$$\min_{0 \leq \alpha_i \leq C} \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i Q_{ij} \alpha_i - \sum_{i=1}^{N} \alpha_j + b \sum_{i=1}^{N} y_i \alpha_i \tag{3-7}$$

So the resulting dual form of the SVM is $f(x) = \sum_{i=1}^{N} y_i \alpha_i \Phi(x_i) \cdot \Phi(x) + b$.

## 3-2 Multi-class SVM

One may notice that from the description of SVM algorithm, a SVM classifier is originally designed to solve binary classification problem. However, in this project, there are three groups, namely LCL, LK and LCR. A multi-class SVM approach need to be applied to settle this problem.

A number of approaches that modify SVM into multi-class have been proposed. It is still a topic that draws continuing attention. For now, commonly used techniques are one against one, one against all and direct acyclic graph (DAG) based classifier [20]. The similarity among these approaches is they all create numbers of sub binary classifiers, the number is related to the amount of classes.

**One against One** One against one method is also known as One versus One (OvO) method. It was introduced in the work by Knerr et al. [21]. This method constructs $k(k-1)/2$ classifiers. Each of these classifiers make use of data only from two classes. For instance, a classifier is trained only using data with label $i$ as positive and label $j$ as negative examples. In this classifier, the following binary classification problem showed in Equation (3-8) needs to be solved.

$$\begin{aligned} \min_{W^{ij},b^{ij},\zeta^{ij}} \quad & \frac{1}{2}(W^{ij})^T W^{ij} + C\sum_t \zeta_t^{ij}(W^{ij})^T \\ & (W^{ij})^T\Phi(x_t) + b^{ij} \geq 1 - \zeta_t^{ij}, \text{ if } y_t = i \\ & (W^{ij})^T\Phi(x_t) + b^{ij} \leq -1 + \zeta_t^{ij}, \text{ if } y_t = j \\ & \zeta_t^{ij} \geq 0 \end{aligned} \tag{3-8}$$

To be more illustrative, for example there are 3 groups, the classifiers are trained using group 1 & 2,1 & 3 and 2 & 3. As presented in Figure 3-2.

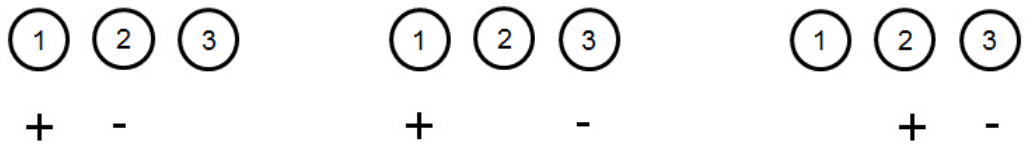


**Figure 3-2:** One vs. One

**One against all** Another commonly chosen method is one against all (OaA), also called One versus All (OvA) method. This technique is first implemented in the work done by Bottou et al. [22]. Different from the OvO method, this one only need $k$ classifiers for $k$ classes. In the training of $i$th classifier, all samples with corresponding label will be regarded as positive examples while all remaining data set as negative samples. Equation (3-9) shows how the $i$th

classifier is computed.

$$\begin{aligned} \min_{W^i,b^i,\zeta^i} \quad & \frac{1}{2}(W^i)^T W^i + C\sum_t \zeta_t^i (W^i)^T \\ & (W^i)^T\Phi(x_j) + b^i \geq 1 - \zeta_j^i, \text{ if } y_t = i \\ & (W^i)^T\Phi(x_j) + b^i \leq -1 + \zeta_j^i, \text{ if } y_t = j \\ & \zeta_t^i \geq 0, \ j = 1, ..., l \end{aligned} \tag{3-9}$$

Similarly, for a 3 group case the classifiers can be trained in the form showed in Figure 3-3.

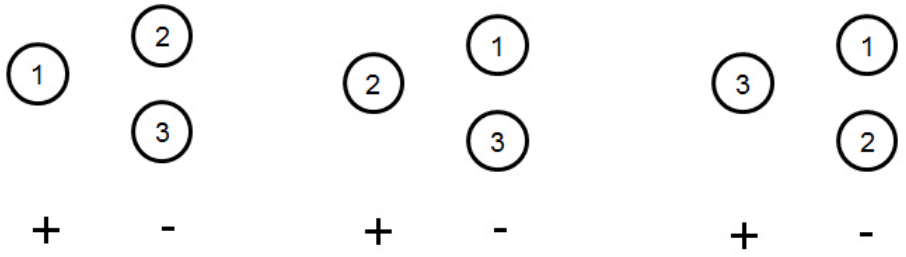


**Figure 3-3:** One vs. All

**DAG** DAG method is provided in the work done by Platt et al [23]. The training phase of this approach is identical to one-vs-one method by constructing $N(N-1)/2$ binary classifiers. While in testing phase, a rooted binary graph with $N(N-1)/2$ internal nodes and $N$ leaves is applied. The classification initials at the top leave. Moving direction is determined by classification result. This iteration continues until the leave that represents predicted class is reached. Thus there will be $N-1$ classifications before a result can be reached.

A sketch of DAG is presented in Figure 3-4.

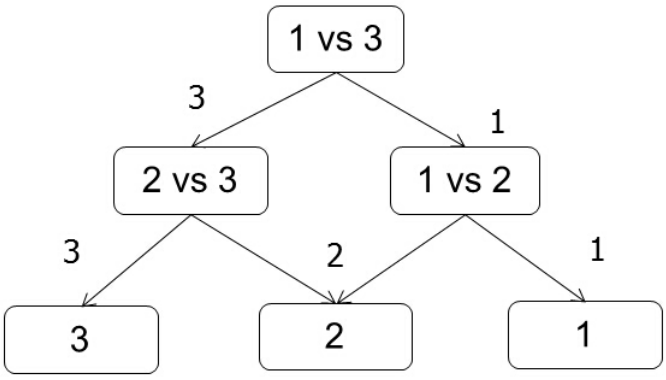


**Figure 3-4:** DAG Classification

A performance comparison of these approaches is provided in the work by Lin et al. [20]. Experiments on large data sets show one against one and DAG approach are more suitable

for practical usage. Moreover, in the tests provided by Demirkesen, one against one approach outperforms other methods in classifying image information [24]. In further implementation, one against one method is applied.

## 3-3 Incremental SVM training

SVM method is originally developed as an off-line method that is trained with a pre-determined database before been used to perform a classification. So the limitation is that the resulting classifier is not able to update itself when facing various situations.

Therefore, there is a need for a classifier that is able to augment itself with new data constantly. There are mainly two practical advantage of using incremental training method.

The first one is a great saving in computation time and consumptions. As illustrated in previous sections, the learning procedure of SVM boils down to an optimization problem. The storage increase dramatically with the expansion of training sample size. This will lead to a limitation especially Automatic driving vehicles . This can be solved by incremental learning effectively since it does not need to store all historical data. Therefore precious on-board storage space can be saved for other tasks.

Secondly, a complete training set is required before training in order to obtain a classifier with high accuracy. However, the sampling of data could be a continual process. The collecting of drivers' behaviour can be a proper example. Driving style varies not only from people to people, but also places to places. The DAVI vehicle will face unpredictable traffic situation from time to time. The capability of an off-line trained classifier is always limited by fixed number of training samples. Thus, there will be a possibility that the pre-recorded database is not sufficient to perform an accurate prediction in some certain traffic situations. Hence a desirable classifier should have the ability to learn incrementally.

To improve the predictive ability in different traffic situations, an incremental learning SVM method is introduced. As its name suggests, this method is able to keep updating itself with freshly recorded data. Since the sensor will be consistently analysed and recorded, it is possible to let the classifier learn actively after each prediction.

Several studies have been done on incremental SVM algorithm. The first approach is provided by Naddeem Syed et al. [13]. They developed the fundamental computation structure of this method. Further, in 2003, Christopher Diehl and Gert Cauwenberghs [25] [26] extended their work to a more general expression and a Matlab project of incremental SVM was initiated by Christopher. In this project, the Matlab scripts of incremental learning are created based on Christopher's work.

### 3-3-1 Karush-Kuhn-Tucker Conditions

According to convex optimization theory [27], the solution of optimization problem in Equation (3-7) can be defined as following equations.

$$g_i = \frac{\partial W}{\partial \alpha_i} = \sum_{j=1}^{N} Q_{ij}\alpha_j + y_i b - 1 \begin{cases} > 0 & \alpha_i = 0 & S_R \\ = 0 & 0 \leq \alpha_i \leq C & S_S \\ < 0 & \alpha_i = C & S_E \end{cases} \tag{3-10}$$

$$h = \frac{\partial W}{\partial b} = \sum_{j=1}^{N} y_j \alpha_j \equiv 0 \tag{3-11}$$

Equation (3-10) is called the Karush-Kunh-Tucker (KKT) condition of SVM. As Figure 3-5 (taken from [26]) suggests, the training set of a SVM can be divided into three groups: $S_R$ is reserve support vectors; $S_S$ is margin support vectors and $S_E$ is error support vectors.

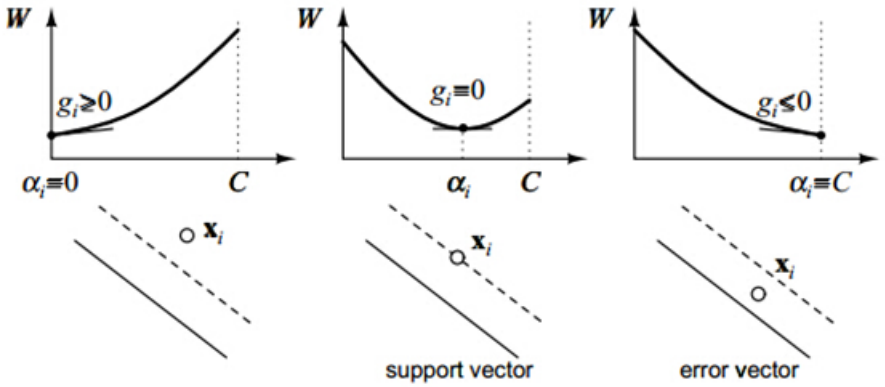


**Figure 3-5:** Different groups of vectors

### 3-3-2 Adiabatic Increments

When a new sample $(x_c, y_c)$ is added into the original training set $S = \{(x_1, y_1), ..., (x_l, y_l)\}$, but $\alpha_c$ and $g_c$ do not satisfy the KKT condition. Adjustments should be made on both $\alpha_c$ and $g_c$ to find the optimal solution of the incremental training example. $\alpha_S = [\alpha_1, ...\alpha_l]^T$ is the $\alpha$ of all samples in original training set. The adiabatic increments method is applied to ensure all samples in set $S_S$ meets KKT condition during the process of tuning $\alpha_c$.

While making adjustments on $\alpha_c$, in order to guarantee all samples in set $S$ satisfy KKT condition, only $\alpha$ and Lagrangian multiplier $b$ of samples contained in margin support vectors $S_S$ can be altered. KKT condition can be written in a differential form, with $Q_{ij} = y_i y_j K(x_i, x_j)$

$$\Delta g_i = Q_{ic}\Delta\alpha_c + \sum_{j \in S} Q_{ij}\Delta\alpha_j + y_i \Delta b, \quad \forall i \in D \cup \{c\} \tag{3-12}$$

$$0 = y_c \Delta\alpha_c + \sum_{j \in S} y_i \Delta\alpha_j \tag{3-13}$$

Further, since $g_i = 0$ holds for all the samples in $S_s$, Equation 3-12 can be rewritten as

$$\varrho \cdot \begin{bmatrix} \Delta b \\ \Delta\alpha_{S_S} \end{bmatrix} = - \begin{bmatrix} y_c \\ Q_{S_s c} \end{bmatrix} \tag{3-14}$$
$$\varrho = \begin{bmatrix} 0 & y_{S_S}^T \\ y_{S_S} & Q_{S_S S_S} \end{bmatrix}$$

Thus, the linear relation between $\alpha_c$, $\alpha_{S_S}$ and $b$ can be obtained.

$$\Delta\alpha_{S_S} = \beta_{S_S}^c \Delta\alpha_c \tag{3-15}$$
$$\Delta b = \beta_b^c \Delta\alpha_c \tag{3-16}$$
$$\begin{bmatrix} \beta_b^c \\ \beta_{S_S}^c \end{bmatrix} = -R \cdot \begin{bmatrix} y_c \\ Q_{S_S c} \end{bmatrix}$$
$$R = \varrho^{-1}$$

where $\alpha_{S_S}$ denotes $\alpha$ for samples in margin support set $S_S$. The relation between $\alpha_c$ and $g_i$ can be derived as well:

$$\Delta g_i = \gamma_i^c \Delta\alpha_c \tag{3-17}$$

with $\gamma_i^c = Q_{ic} + \sum_{j \in S_S} Q_{ij}\beta_j^c + y_i\beta_b^c, \forall i \in S$

### 3-3-3 Incremental learning algorithm

Optimal solutions can not be directly derived from adiabatic increment method since the components of $S_S$,$S_R$ and $S_E$ may change as the increasing of $\alpha_c$. A possible solution for this phenomenon is to capture the minimum changes occurred in those three sets. As to find an upper limit of $\alpha_c$ so that only one sample will be transmitted among $S_S$.$S_R$ and $S_E$. There are three possible scenarios:

(1)The first is when $\alpha_i$ of sample $i$ in set $S_S$ reaches upper or lower limit as follows:

$$\Delta\alpha_i^{\max} = \begin{cases} C - \alpha_i \\ -\alpha_i \end{cases} \tag{3-18}$$

Thus, according to Equation 3-15, the maximum increase before a sample moves from $S_S$ to $S_R$ or $S_E$ is :

$$\Delta\alpha_c^{S_S} = \min \frac{\Delta\alpha_i^{\max}}{\beta_i^c} \tag{3-19}$$

When $\Delta\alpha_i^{\max} = C - \alpha_i$, $i$ moves from $S_S$ to $S_E$; $\Delta\alpha_i^{\max} = -\alpha_i$, $i$ moves from $S_S$ to $S_R$

(2) Sample $i$ in $S_R$ or $S_E$ may move to $S_S$ if the corresponding $g_i$ equals to 0. So the maximum augment possible is :

$$\Delta\alpha_c^{S_{R,E}} = \min \frac{-g_i}{\gamma_i^c} \tag{3-20}$$

If initially $g_i < 0$, $i$ moves from $E$ to $S$; If initially $g_i > 0$, $i$ moves from $R$ to $S$

(3) New data $c$ meets KKT condition. The resulting maximum update is :

$$\Delta\alpha_c^{a,g} = \min \left\{ \frac{-g_c}{\gamma_c^c}, C - \alpha_c \right\} \tag{3-21}$$

Finally, the maximum increment of $\Delta\alpha_c$ is the minimum of these three values.

$$\Delta\alpha_c^{\max} = \min\{\Delta\alpha_c^{S_S}, \Delta\alpha_c^{S_{R,E}}, \Delta\alpha_c^{a,g}\} \tag{3-22}$$

The variation of $\alpha_c$ will result in a component change in $S_S$. Matrix $R$ should be altered correspondingly. There are two possible alterations of set $S_S$:

- A sample is added to set $S_S$

$$R^+ \leftarrow \begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\gamma_t} \begin{bmatrix} \beta_b^t \\ \beta_{S_S}^t \\ 1 \end{bmatrix} \cdot \begin{bmatrix} \beta_b^t \\ \beta_{S_S}^t \\ 1 \end{bmatrix}^T \tag{3-23}$$

- A sample is moved out from $S_S$.

$$R_{ij}^- \leftarrow R_{ij} - R_{tt}^{-1} R_{it} R_{tj} \tag{3-24}$$

where $t$ is the removed sample.

**Algorithm 1** Incremental learning algorithm

1: **Step 1:** Initialize $\alpha_c$ of new sample $(x_c, y_c)$ to 0.
2: **Step 2:**
3: *If* $(x_c, y_c)$ satisfies KKT condition $(g_c > 0)$
4: New sample is not a margin or error vector, **END**
5: **Step 3:**Increase $\Delta\alpha_c$ until
6: *If* $g_c = 0$
7: Add $c$ to $S_S$, update $R$, **END**
8: *Elseif* $\alpha_c \leq C$
9: Add $c$ to $S_E$, **END**
10: *Elseif* Scenario (1)
11: Move $i$ to $S_R$ or $S_E$,update $R$, **END**
12: *Elseif* Scenario (2)
13: Move $i$ from $S_R$ or $S_E$ to $S_S$,update $R$, **END**
14: **END**

Procedures of incremental learning algorithm are listed in Algorithm 1. When a new sample $(x_c, y_c)$ is added to $S$, the optimal solution can be derived through Algorithm 1.

# Chapter 4

# Implementation

## 4-1 Before training

As described in the Chapter 2-1, the next step is to modify acquired data groups into a form that suitable for SVM training.

Firstly, a SVM training sample contains features and label. Features are values that describe sample position in the hyperspace. One sample can have more than 2 features which make it becomes high dimensional. In this project, a sample has 4 features, they are heading angle, yaw rate, lateral velocity and lateral acceleration. Labels are numeric values that indicate which group one sample belongs to. The label of LCL is defined as 3, LK as 2 and LCR is 1 respectively in this thesis. This definition makes it easier to interpret validation results from graphs. This aspect will be further elaborated in following sections.

Further, SVM training requires each data group to have relatively same length. However, filtered data in Chapter 2 does not meet this requirement. Each LCL and LCR sample contains data duration of 4.5 seconds. Since the time step is 0.1 second, there are 46 data points in every LCL and LCR sample. The difference in length between LCL and LCR samples is trivial. But it is not the case for LK group. LK is filled with data taken from the sections between each lane change maneuver. So the recording duration of each LK sample is indefinite. One LK sample may contain around 2000 data points. To ameliorate this problem, a decimate resample technique is applied on LK group. The resample ratio is related to the length of LCL and LCR.

After training finished, a self-test will be performed in order to verify the performance of this classifier. So the final step is to construct training set and testing set. K-fold validation technique is a primary method to evaluate performance of a classifier. In a K-fold validation, the total data set will be divided into $K$ groups. One of these groups will be used to test the classifier trained by remaining $K-1$ groups. Commonly $K = 5$ is selected. In this study, 80% of the total samples will be used to train the classifier while the remaining 20% for testing purpose. Different ratios will lead to changes in validation results.

## 4-2 Choice of kernel function and parameter selection

Kernel functions play an important role in SVM algorithm by mapping linear inseparable data into higher dimensional space in which it becomes possible to separate them. There are several kinds of kernel function available. Commonly used ones are:

- Linear kernel: $K(x_i, x_j) = x_i \cdot x_j$
- Polynomial kernel: $K(x_i, x_j) = (ax_i \cdot x_j + b)^b$
- Radial Basis Function (RBF) kernel: $K(x_i, x_j) = e^{-\frac{|x_i - x_j|^2}{2\sigma^2}}$
- Sigmoid kernel: $K(x_i, x_j) = tanh(ax_i \cdot x_j + b)$

It is a difficult task to choose between these kernels. In the work done by Pal [28], five different kernels are tested: linear kernel, polynomial kernel, RBF kernel, sigmoid kernel and linear spline kernel are examine in the report. Results suggest that the RBF kernel performs well. It achieved the highest accuracy when handling multi-spectral data from remote sensing.

Another problem before training initiation is parameter selection. The choices of parameter $C$ and $g$ in training will have a critical influence on prediction accuracy. $C$ controls the trade-off between margin maximization and error minimization. $g$ is a representation of $\sigma$ in RBF kernel.

For the purpose of finding the most optimal parameter combination, a grid search method is introduced. There will be two searching stages to accomplish the search effectively. The first one is a "rough" search: setting $c$ as the $x$ axis; $g$ on $y$ axis. Each parameter varies from $2^{-10}$ to $2^{10}$ by step size of 0.8. After this search, a contour accuracy map can be obtained. Next step is to draw a refined search on the section with highest accuracy. The step size on this reduced scan area shrinks down to 0.5.

This grid search method is applied on each kernel function. The combination which leads to the highest accuracy in self-test will be regarded as the most suitable kernel function for the training. The results are listed in Table 4-1.

**Table 4-1:** Grid search results

| Kernel Function | Accuracy |
|---|---|
| Linear | 69.7572% |
| Polynomial | 68.8742% |
| RBF | 74.3929% |
| Sigmoid | 71.9647% |

The contour map of RBF kernel is shown in Figure 4-1. Self test result is present in Figure 4-2.

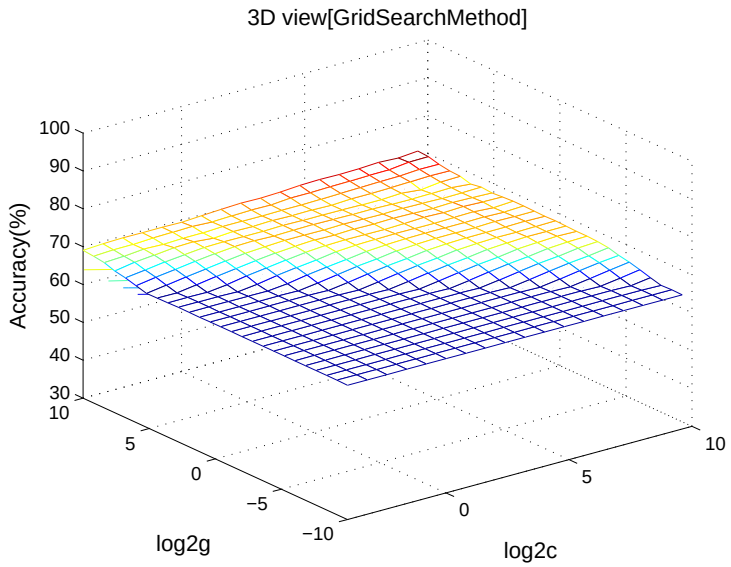


**Figure 4-1:** Contour map used in finding the best parameter combination

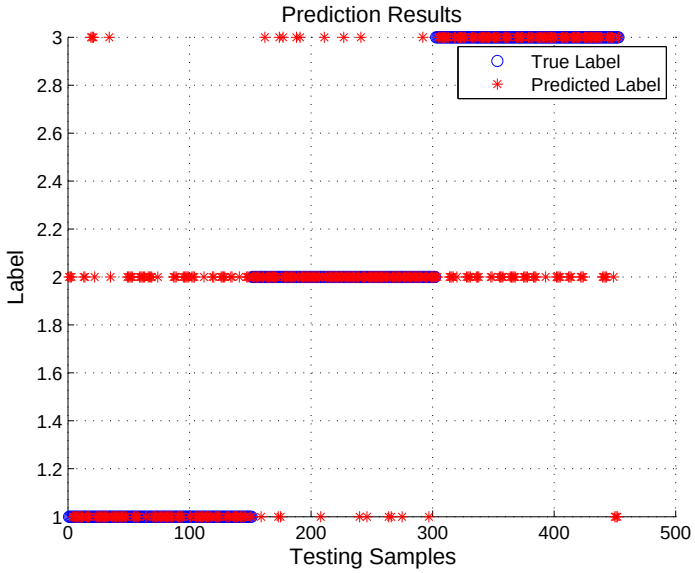


**Figure 4-2:** Self test results

The result of grid search suggests that RBF kernel yields the best accuracy. The best parameter combination is obtained as well.

## 4-3 Prediction and results

Self test is only a numerical verification of classifiers' ability. In order to examine its performance in practical application, the classifier will be applied on a complete driver data set

which is not included in the training set. The features and label definition are identical to training data. The objective for this classifier is to find all possible lane changes in this data set by analysing corresponding variations in features value.

Two methods are proposed to accomplish the prediction process. The first is point by point prediction. Results are made by classifying each data point individually in this method. On the other hand, sliding window method is proposed to focus on the relations between adjacent data points.

To be more illustrative, a detailed procedure will be given. The data section from driver #14 will be used as testing data set. More thorough validation results are presented in the next chapter.

### 4-3-1 Generating ground truth

Lane changes must be presented in a graphical way so that comparisons can be drawn between predict results and actual ones. Figure 4-3 shows how the ground truth of lane changes are generated.

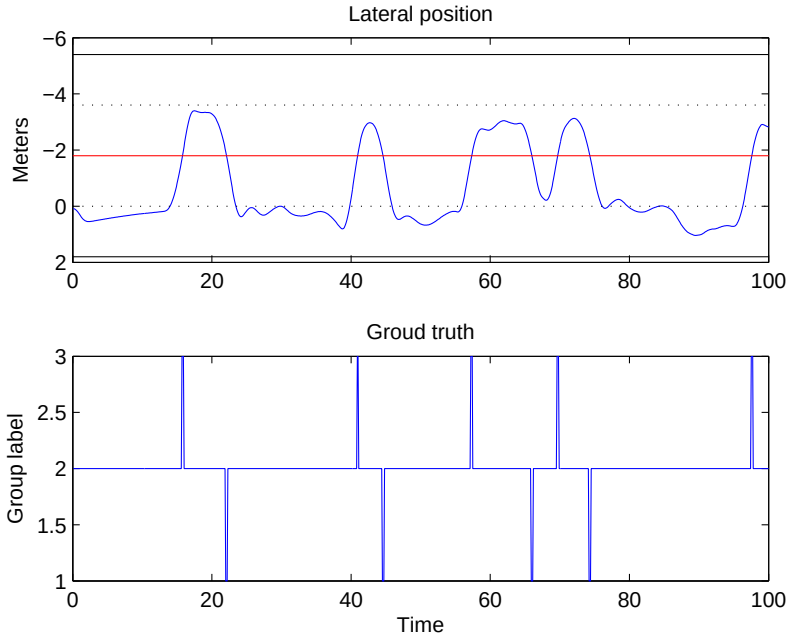


**Figure 4-3:** Lateral position and ground truth

In Figure 4-3, an actual lane change left takes place between 10 to 20 seconds. The value of ground truth is assigned at the moment when vehicle's lateral position overlaps with the border between two lanes (the red line). In this way, exact moment of lane changes can be presented. The value of ground truth is defined as the heading direction of virtual vehicle. If the vehicle turns left, the value is set to 3. 1 will be assigned if it goes opposite direction. Apart from these "lane crossing" moments, the value will be 2.

### 4-3-2 Point by point prediction

As the name of this method suggests, data points will be fed into the classifier in a chronological order individually. So the prediction results are acquired solely in the order of increasing time step.

Figure 4-4 gives a point by point prediction result of a single LCL maneuver. As lateral position suggests, the vehicle drivers from right lane to left lane. So the prediction goes as the same direction. At each data point, three possibilities concerning in which group current point belongs to. The final result will be decided on the label with the highest possibility. For instance, the point at 1 second belongs to LK since the dominance of LK possibility. As time moves to 1.2 s, the output changes to LCL. The actual lane change take place at 3.5 s. Thus the alarm is able to ring at 2.3 seconds before. Prediction results of a section of continuous trajectory are presented in Figure 4-5.

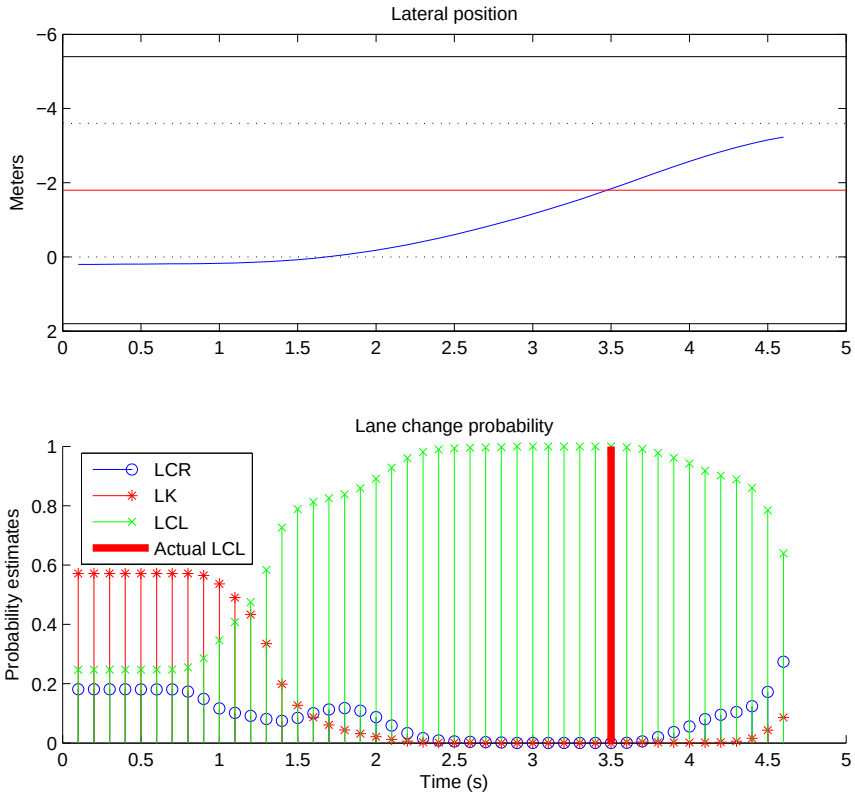


**Figure 4-4:** Point by point applied on single LCL

**Remark :** Predictions are only made before a lane change actually happens. For instance, the predicted label changes from 2 to 3 at 13.4 s, which means a lane change is predicted at this time step. The predict value will return to default value 2 after the lane change is finished, i.e. at 17 s. This change is not a prediction action.

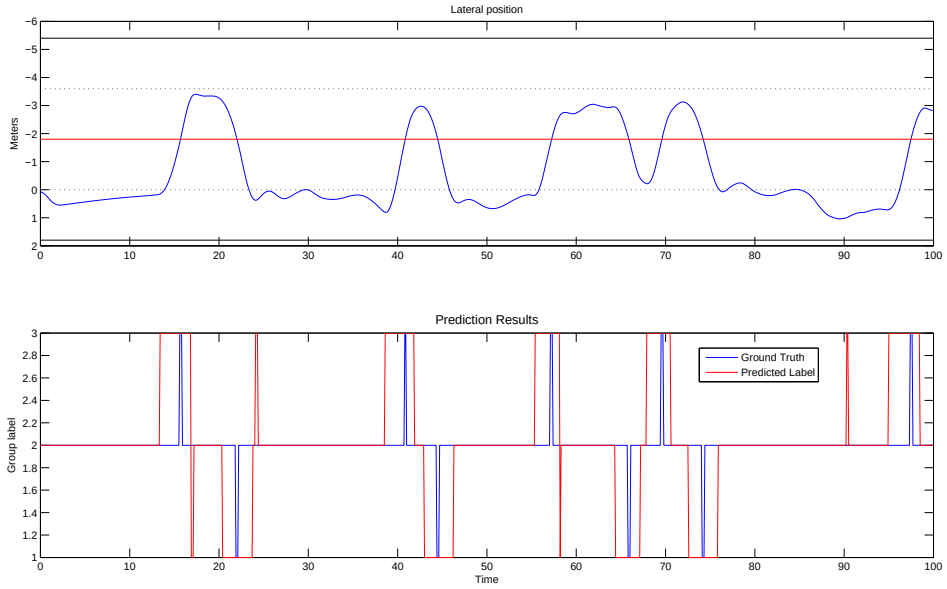


**Figure 4-5:** Point by point prediction output and ground truth

### 4-3-3 Sliding window

It can be observed in Figure 4-5 there are some false positives in prediction results. This is mainly caused by the inaccuracy of the classifier and the nature of data. It is also commonly occurs that a driver behaves as he/she is going to perform a lane change, however, apart from wiggling, no actions were taken. To ameliorate this phenomenon, a sliding window approach is introduced. A sketch of this method is presented in Figure 4-6.

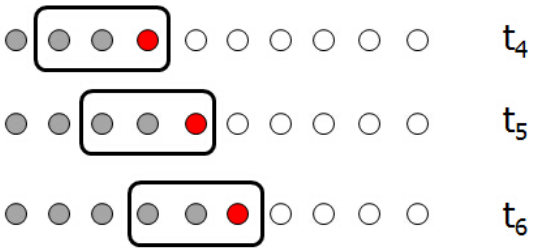


**Figure 4-6:** Sketch of sliding window approach

In this method, the result is reached by taking relations between adjacent data points into consideration. A vote mechanism is applied in this approach. As shown in Figure 4-6, the window size is 3. So the output label of time step $t_4$ (the red dot) is determined by the results of $t_4$, $t_3$ and $t_2$. The result will be assigned as the majority output of these three points.

Figure 4-7 shows the results after the sliding window with size 5 is applied. It can be seen that the number of false positives is reduced.

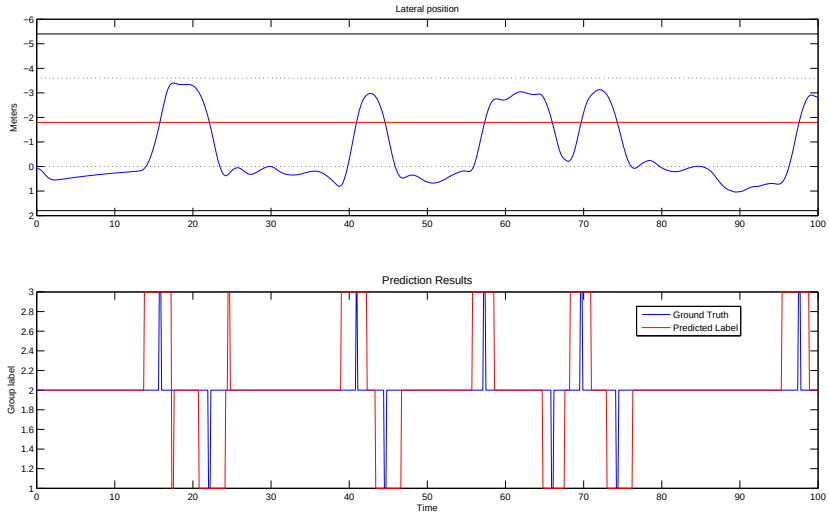


**Figure 4-7:** Results of sliding window size 5

The choice of sliding window size also has an influence on prediction accuracy and computational time. A more in-depth analysis is presented in Chapter 5.

## 4-4 Data features selection

For now there are 4 features in total in dataset. They are heading angle, yaw rate, lateral velocity and lateral acceleration. All these features can be recorded conveniently in the driving simulator. However, yaw rate might not be an easy accessible parameter with current vehicle sensors in real prediction practise. Moreover, it can be seen in Figure 2-11 and Figure 2-13 in Chapter 2, the value of yaw rate is relatively small compared to other three parameters. In SVM training, the smaller the value of a feature compared with others, the less influence it will have on prediction results. Thus a question arises: is it possible to train without yaw rate? To answer this, a test is needed to verify how the performance will change if yaw rate is no longer regarded as a feature.

There are 14 drivers data available in data set. They named as #1,#2,...#14. As validation test set 1 in Chapter 5, training set is chosen as driver #1 to driver #13; test set is driver #14. Comparison of off-line training method and on-line training method are listed in Table 4-2 and Table 4-3 respectively.

It can be observed from this feature that there is a trivial prediction performance change with the removal of yaw rate. On the other hand, a reduction in number of features result in less storage demand. The classifier size shrinks approximate 15%. So in further validation, only heading angle, lateral velocity and lateral acceleration will be employed.

**Table 4-2:** Performance comparison in Off-line training

| | Sensitivity (%) | Advance time (s) | Computation time (s) | False positives (times) |
|---|---|---|---|---|
| With yaw rate | | | | |
| Point by point | 100 % (42/42) | 1.481 | 0.0023 | 34 |
| Sliding window (size 3) | 100 % (42/42) | 1.4643 | 0.0035 | 28 |
| Sliding window (size 5) | 100 % (42/42) | 1.3762 | 0.0042 | 25 |
| Sliding window (size 7) | 100 % (42/42) | 1.2643 | 0.0052 | 21 |
| Sliding window (size 9) | 97.619 %(41/42) | 0.82857 | 0.0057 | 16 |
| Without yaw rate | | | | |
| Point by point | 100 % (42/42) | 1.5952 | 0.0021 | 32 |
| Sliding window (size 3) | 100 % (42/42) | 1.3786 | 0.0031 | 28 |
| Sliding window (size 5) | 100 % (42/42) | 1.2786 | 0.0041 | 25 |
| Sliding window (size 7) | 100 % (42/42) | 1.0571 | 0.0052 | 16 |
| Sliding window (size 9) | 97.619 %(41/42) | 0.9619 | 0.0057 | 13 |

**Table 4-3:** Performance comparison in on-line training

| | Sensitivity (%) | Advance time (s) | Computation time (s) | False positives (times) |
|---|---|---|---|---|
| With yaw rate | | | | |
| Point by point | 100 % (42/42) | 1.7238 | 0.0039 | 47 |
| Sliding window (size 3) | 100 % (42/42) | 1.5238 | 0.005 | 46 |
| Sliding window (size 5) | 100 % (42/42) | 1.3833 | 0.0057 | 44 |
| Sliding window (size 7) | 100 % (42/42) | 1.1857 | 0.0061 | 39 |
| Sliding window (size 9) | 97.619 %(41/42) | 0.8833 | 0.0067 | 32 |
| Without yaw rate | | | | |
| Point by point | 100 % (42/42) | 1.7262 | 0.0047 | 50 |
| Sliding window (size 3) | 100 % (42/42) | 1.5238 | 0.005 | 46 |
| Sliding window (size 5) | 100 % (42/42) | 1.3833 | 0.0057 | 44 |
| Sliding window (size 7) | 100 % (42/42) | 1.1857 | 0.0061 | 39 |
| Sliding window (size 9) | 97.619 %(41/42) | 0.88571 | 0.0075 | 31 |

# Chapter 5

# Validation results

14 drivers data remained after the filtering process. They are numbered as #1,#2,...,#14. As described in the chapter concerning SVM algorithm, there are two ways to obtain a classifier: off-line training and incremental training (on-line training).

In this chapter, the performance of different prediction approaches will be tested. This validation process will be done in 2 sets.

There are two classifier evaluation indexes applicable for this study: sensitivity and precision. Sensitivity is an evaluation of a classifier's positive finding capability. On the other hand, precision is introduced to find the ratio of true positives in all positive labelled samples. The definition of each index is presented in Eq. 5-1 and Eq. 5-2 respectively.

$$Sensitivity = \frac{True\ positives}{Total\ positives} \tag{5-1}$$

$$Precision = \frac{True\ positives}{True\ positives + False\ positives} \tag{5-2}$$

In set 1, 2 test items will be examined. The first is to verify how the classifier performs when predicting a single lane change maneuver. In this part, the first lane change of driver # 14 will be tested. The second part is to check if this classifier is able to predict all the lane changes in one continuous trajectory. In this process, all lane changes recorded in data set of driver #14 is applied.

General applicability of this prediction algorithm is another major concern. In order to validate this aspect, all procedure in set 1 will be done again on driver set #1, instead of driver #14. In addition to these 2 sets, 3 more sets are done and results are listed in Appendix B.

All calculations are done on a computer with configuration listed in Table 5-1. Resulting training and calculation time may vary when algorithms run on computers with other configurations.

**Table 5-1:** Computer configuration

| System | |
|---|---|
| CPU | Intel Core i5-4258U CPU @ 2.40GHz |
| RAM | 1600MHz DDR3 8.00 GB |
| Hard Drive (HD) type | SSD |
| HD read speed | 731 MB/s |
| HD write speed | 671 MB/s |
| Matlab version | 2013a 64 Bit |
| Operating system | Windows 8.1 Pro 64 Bit |

## 5-1 Set 1

### 5-1-1 Off-line training

In this scenario, RBF kernel function is applied in the SVM algorithm. The function parameters can be tuned in order to reach the best cross validation accuracy. The higher the cross validation accuracy, the higher the self test accuracy the classifier can achieve.

Off-line training process contains two steps : training and a self test. Before the training initiates, all the training samples from 13 drivers' data set will be divided into training samples, takes 80% in total sample number, and testing samples randomly.

Further, in this test the performance of point by point approach and sliding window approach, with varying window size, will be tested as well.

#### Scenario 1

**Training set:** Driver #1 to #13
**Testing set:** Driver #14

#### Training part

**Table 5-2:** Off-line training performance

| | Training time (s) | Self-test accuracy |
|---|---|---|
| Parameter selection | 62476.708 | 74.07% |
| Training | 563.24104 | |

Parameters selection: Best $C$= 776.047, Best $g$= 1024

#### Predicting part

**Table 5-3:** Prediction performance on single LCL

| | Computation time (s) | Advance time (s) |
|---|---|---|
| Point by point | 0.0030 | 2.1 |
| Sliding window size 3 | 0.0043 | 2 |
| Sliding window size 5 | 0.0051 | 1.9 |

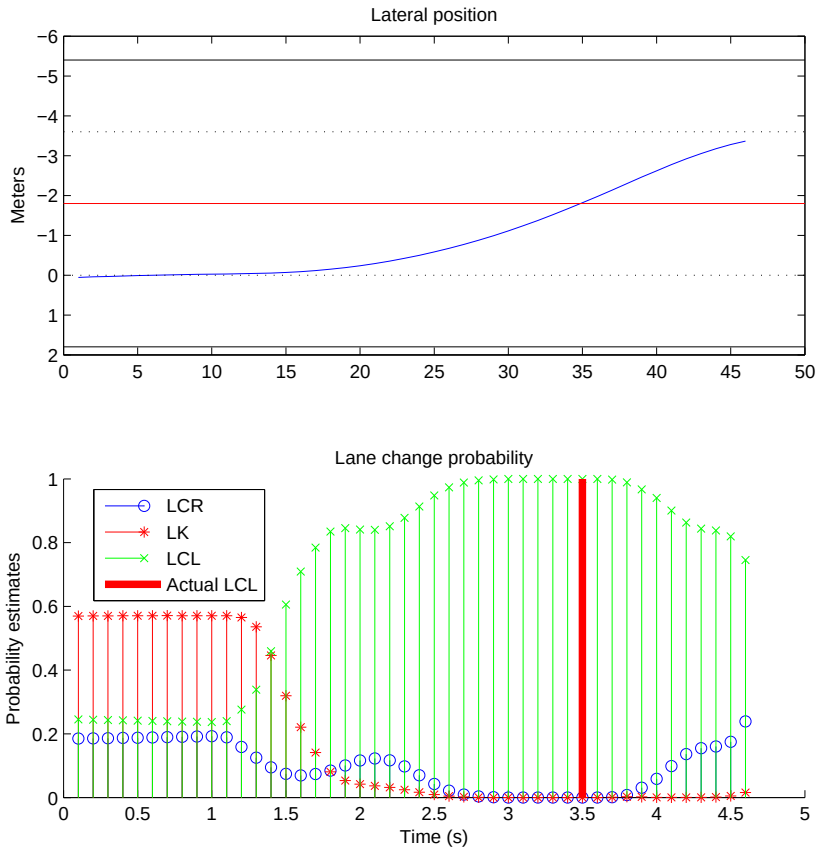


**Figure 5-1:** Prediction of a single LCL

The probability output is presented in Figure 5-1. The moment that actual lane change occurs is shown in red solid line. The probability that each data point belongs to LCL group is labelled as green stem. The same goes for other two groups. It can be observed in this figure that the possibility of LCL becomes the highest at 1.4 second. This means that this classification approach is able to reveal a potential lane change with 2.1 seconds in advance of its appearance.

**Table 5-4:** Prediction performance on continuous trajectory

| | Sensitivity (predicted/actual) | Avg. advance time (s) | Avg. computation time(s) | False positives (times) |
|---|---|---|---|---|
| Point by point | 100 % (42/42) | 1.5952 | 0.0021 | 32 |
| Sliding window (size 3) | 100 % (42/42) | 1.3786 | 0.0031 | 28 |
| Sliding window (size 5) | 100 % (42/42) | 1.2786 | 0.0041 | 25 |
| Sliding window (size 7) | 100 % (42/42) | 1.0571 | 0.0052 | 16 |
| Sliding window (size 9) | 97.619 %(41/42) | 0.9619 | 0.0057 | 13 |

### 5-1-2 Incremental learning

#### Scenario 2

In this scenario, the classifier is trained in an incremental way. The training process will be completed by training 13 drivers' data set one by one incrementally.

**Training set:** Driver #1 to #13 (1 by 1 training)
**Testing set:** Driver #14

#### Training part

**Table 5-5:** Incremental training performance

| | Training time (s) | Self-test accuracy |
|---|---|---|
| Individual training | (Avg.)44.6655 | 70.13% |
| Total training | 584.8379 | |

#### Prediction part

**Table 5-6:** Predict results for single LCL

| | Computation time (s) | Advance time (s) |
|---|---|---|
| Point by point | 0.0041 | 2 |
| Sliding window (size 3) | 0.0055 | 1.9 |
| Sliding window (size 5) | 0.0063 | 1.8 |

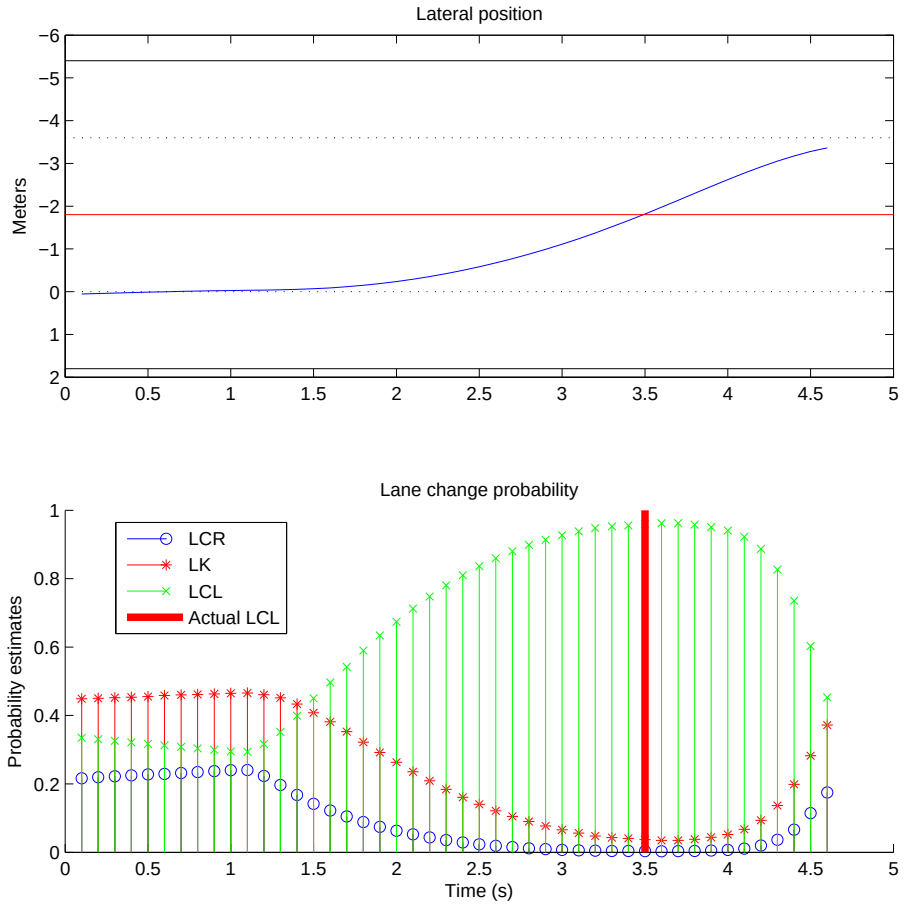


**Figure 5-2:** Prediction of a single LCL with online trained classifier

**Table 5-7:** Prediction performance on continuous trajectory with Inc. classifier

| | Sensitivity (predicted/actual) | Avg. advance time (s) | Avg. computation time(s) | False positives (times) |
|---|---|---|---|---|
| Point by point | 100 % (42/42) | 1.7262 | 0.0047 | 50 |
| Sliding window (size 3) | 100 % (42/42) | 1.5238 | 0.005 | 46 |
| Sliding window (size 5) | 100 % (42/42) | 1.3833 | 0.0057 | 44 |
| Sliding window (size 7) | 100 % (42/42) | 1.1857 | 0.0061 | 39 |
| Sliding window (size 9) | 97.619 %(41/42) | 0.88571 | 0.0075 | 31 |

### Scenario 3

In this scenario, the classifier is trained in same procedure as Scenario 2. In addition, while predicting driver #14, after each actual lane change those data will be used to update the classifier.

**Training set:** Driver #1 to #13 (1 by 1 training) and lane changes in #14
**Testing set:** Driver #14

### Training part

**Table 5-8:** Incremental training performance

| | Training time (s) | Self-test accuracy |
|---|---|---|
| Individual training | (Avg.)41.665 | 70.21% |
| Total training | 541.651 | |

### Prediction part

**Table 5-9:** Predict results for single LCL

| | Computation time (s) | Advance time (s) |
|---|---|---|
| Point by point | 0.0041 | 2 |
| Sliding window (size 3) | 0.0055 | 1.9 |
| Sliding window (size 5) | 0.0063 | 1.8 |

In this scenario, the results are the same as previous scenario when predicting a single LCL. Because the update only take place at the end of each lane change maneuver.

**Table 5-10:** Prediction performance on continuous trajectory with Inc. classifier

| | Sensitivity (predicted/actual) | Avg. advance time (s) | Avg. computation time(s) | Avg. inc. training time (s) | False positives (times) |
|---|---|---|---|---|---|
| Point by point | 100 % (42/42) | 1.7762 | 0.0048 | 2.769 | 56 |
| Sliding window (size 3) | 100 % (42/42) | 1.5024 | 0.0058 | 2.7229 | 51 |
| Sliding window (size 5) | 100 % (42/42) | 1.3571 | 0.0067 | 2.8088 | 44 |
| Sliding window (size 7) | 100 % (42/42) | 1.2167 | 0.0071 | 2.8365 | 42 |
| Sliding window (size 9) | 97.619 %(41/42) | 0.92857 | 0.0081 | 2.8372 | 35 |

### Summary of results in Set 1

Validation results are compared in Table 5-11.

**Table 5-11:** Comparison between different learning methods - Set 1

| | Offline learning | Online learning | |
|---|---|---|---|
| | Scenario 1 | Scenario 2 | Scenario 3 |
| Training Process | | | |
| Total training time (s) | 563.241 | 541.651 | 541.651 |
| Self-test accuracy | 74.39% | 70.21% | 70.21% |
| Prediction average Computation time(s) | | | |
| Point by point | 0.0021 | 0.0039 | 0.0048 |
| Sliding window (size 3) | 0.0031 | 0.005 | 0.0058 |
| Sliding window (size 5) | 0.0041 | 0.0057 | 0.0067 |
| Sliding window (size 7) | 0.0052 | 0.0061 | 0.0071 |
| Sliding window (size 9) | 0.0057 | 0.0067 | 0.0081 |
| Prediction average Advance time (s) | | | |
| Point by point | 1.5952 | 1.7238 | 1.7762 |
| Sliding window (size 3) | 1.3786 | 1.5238 | 1.5024 |
| Sliding window (size 5) | 1.2786 | 1.3833 | 1.3571 |
| Sliding window (size 7) | 1.0571 | 1.1857 | 1.2167 |
| Sliding window (size 9) | 0.9619 | 0.8833 | 0.92857 |
| Sensitivity | | | |
| Point by point | 100 % (42/42) | 100 % (42/42) | 100 % (42/42) |
| Sliding window (size 3) | 100 % (42/42) | 100 % (42/42) | 100 % (42/42) |
| Sliding window (size 5) | 100 % (42/42) | 100 % (42/42) | 100 % (42/42) |
| Sliding window (size 7) | 100 % (42/42) | 100 % (42/42) | 100 % (42/42) |
| Sliding window (size 9) | 97.619 %(41/42) | 97.619 %(41/42) | 97.619 %(41/42) |
| Precision (%) | | | |
| Point by point | 55.26 | 47.19 | 42.27 |
| Sliding window (size 3) | 60 | 47.72 | 44.56 |
| Sliding window (size 5) | 62.69 | 48.84 | 48.24 |
| Sliding window (size 7) | 66.67 | 51.85 | 49.4 |
| Sliding window (size 9) | 72.41 | 56.76 | 53.95 |

## 5-2 Set 2

This section is performed to examine the general applicability of this prediction algorithm. In this set, instead of driver data set #14, data set #1 will be employed as the testing samples. Thus, the training samples will still be 13 sets. The testing procedures in this set are identical to Set 1.

### 5-2-1 Off-line training

#### Scenario 4

**Training set:** Driver #2 to #14
**Testing set:** Driver #1

#### Training part

**Table 5-12:** Off-line training performance

| | Training time | Test accuracy |
|---|---|---|
| Parameter selection | 76053.54 | 71.82% |
| Training | 524.85093 | |

Parameters selection: Best $C$= 776.047, Best $g$= 1024

#### Predicting part

**Table 5-13:** Prediction performance on single LCL

| | Computation time (s) | Advance time (s) |
|---|---|---|
| Point by point | 0.0030 | 2.7 |
| Sliding window size 3 | 0.0043 | 2.6 |
| Sliding window size 5 | 0.0055 | 2.5 |

**Table 5-14:** Prediction performance on continuous trajectory

| | Sensitivity (predicted/actual) | Avg. advance time (s) | Avg. computation time(s) | False positives (times) |
|---|---|---|---|---|
| Point by point | 100% (38/38) | 1.9737 | 0.0032 | 99 |
| Window size 3 | 100% (38/38) | 1.8579 | 0.0043 | 89 |
| Window size 5 | 100% (38/38) | 1.6895 | 0.0049 | 79 |
| Window size 7 | 100% (38/38) | 1.5921 | 0.0057 | 66 |
| Window size 9 | 100% (38/38) | 1.5053 | 0.0065 | 54 |

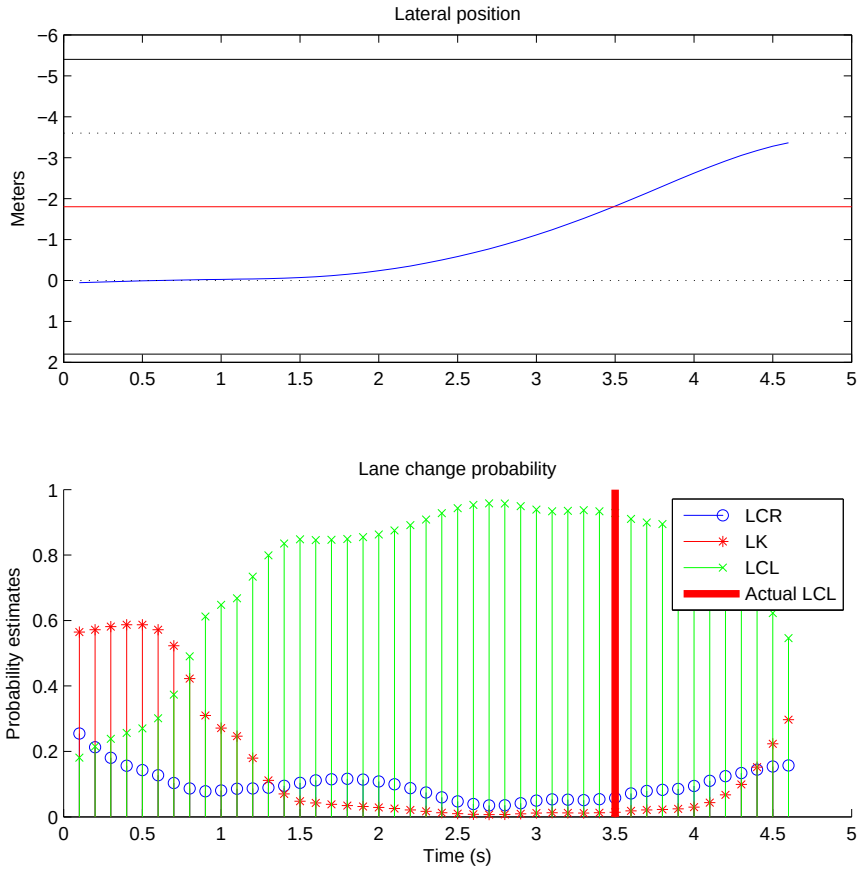


**Figure 5-3:** Prediction of a single LCL

## 5-2-2 Incremental learning

### Scenario 5

**Training set:** Driver #2 to #14 (1 by 1 training)
**Testing set:** Driver #1

### Training part

**Table 5-15:** Incremental training performance

| | Training time (s) | Self-test accuracy |
|---|---|---|
| Individual training | (Avg.)42.343 | 69.942% |
| Total training | 550.466 | |

### Prediction part

**Table 5-16:** Predict results for single LCL

| | Computation time (s) | Advance time (s) |
|---|---|---|
| Point by point | 0.0043 | 2.6 |
| Sliding window (size 3) | 0.0055 | 1.9 |
| Sliding window (size 5) | 0.0063 | 1.8 |

**Table 5-17:** Prediction performance on continuous trajectory with Inc. classifier

| | Sensitivity (predicted/actual) | Avg. advance time (s) | Avg. computation time(s) | False positives (times) |
|---|---|---|---|---|
| Point by point | 100% (38/38) | 1.8947 | 0.0043 | 103 |
| Window size 3 | 100% (38/38) | 1.6737 | 0.0048 | 100 |
| Window size 5 | 100% (38/38) | 1.5737 | 0.0055 | 91 |
| Window size 7 | 100% (38/38) | 1.4763 | 0.0063 | 84 |
| Window size 9 | 97.3684%(37/38) | 1.0316 | 0.0066 | 74 |

### Scenario 6

**Training set:** Driver #2 to #14 (1 by 1 training) and lane changes in #1
**Testing set:** Driver #1

#### Training part

**Table 5-18:** Incremental training performance

| | Training time (s) | Self-test accuracy |
|---|---|---|
| Individual training | (Avg.)41.665 | 70.21% |
| Total training | 541.651 | |

#### Prediction part

**Table 5-19:** Predict results for single LCL

| | Computation time (s) | Advance time (s) |
|---|---|---|
| Point by point | 0.0041 | 2 |
| Sliding window (size 3) | 0.0055 | 1.9 |
| Sliding window (size 5) | 0.0063 | 1.8 |

As described in Scenario 3, the results are the same as previous scenario when predicting a single LCL.

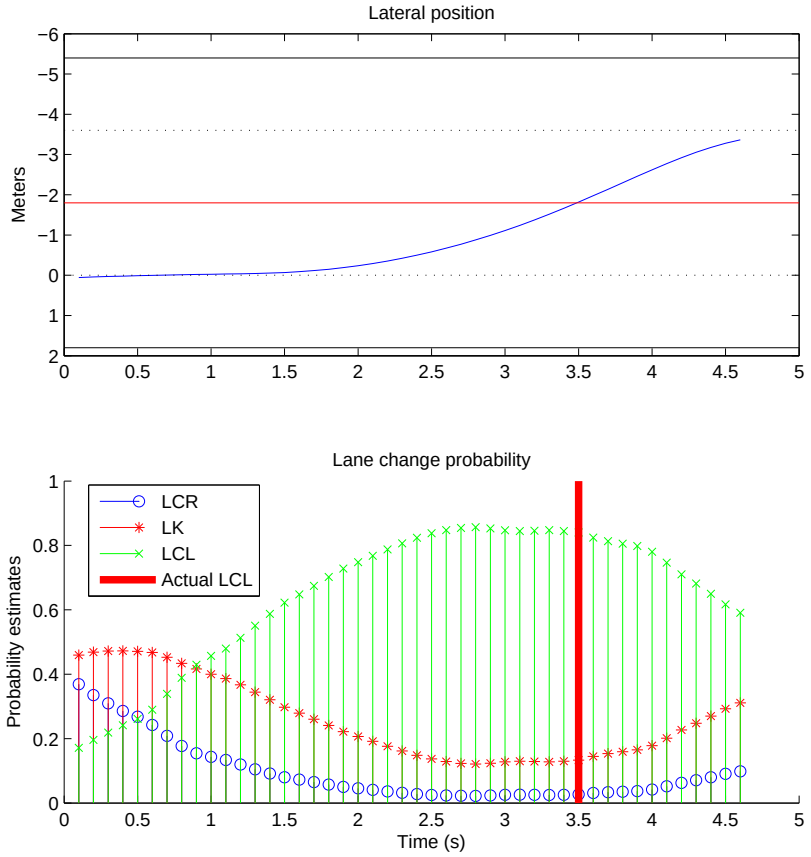


**Figure 5-4:** Prediction of a single LCL with online trained classifier

**Table 5-20:** Prediction performance on continuous trajectory with Inc. classifier

| | Sensitivity (predicted/actual) | Avg. advance time (s) | Avg. computation time(s) | Avg. inc. training time (s) | False positives (times) |
|---|---|---|---|---|---|
| Point by point | 100% (38/38) | 2.0381 | 0.0045 | 2.5842 | 123 |
| Window size 3 | 100% (38/38) | 1.9132 | 0.0049 | 2.5592 | 116 |
| Window size 5 | 100% (38/38) | 1.6553 | 0.0055 | 2.5748 | 105 |
| Window size 7 | 100% (38/38) | 1.5395 | 0.0061 | 2.5729 | 95 |
| Window size 9 | 97.36%(37/38) | 1.2211 | 0.0066 | 2.5554 | 83 |

### Summary of results in Set 2

Validation results of Set 2 are listed in Table 5-21.

**Table 5-21:** Comparison between different learning methods in Set 2

| | Offline learning | Online learning | |
|---|---|---|---|
| | Scenario 4 | Scenario 5 | Scenario 6 |
| Training Process | | | |
| Total training time (s) | 524.85 | 541.651 | 541.651 |
| Self-test accuracy | 71.82% | 70.21% | 70.21% |
| Prediction Process average Computational time(s) | | | |
| Point by point | 0.0032 | 0.0043 | 0.0045 |
| Sliding window (size 3) | 0.0043 | 0.0048 | 0.0049 |
| Sliding window (size 5) | 0.0049 | 0.0055 | 0.0055 |
| Sliding window (size 7) | 0.0057 | 0.0063 | 0.0061 |
| Sliding window (size 9) | 0.0065 | 0.0066 | 0.0066 |
| Prediction Process average Advance time (s) | | | |
| Point by point | 1.9737 | 1.8947 | 2.0421 |
| Sliding window (size 3) | 1.8579 | 1.6737 | 1.9132 |
| Sliding window (size 5) | 1.6895 | 1.5737 | 1.6553 |
| Sliding window (size 7) | 1.5921 | 1.4763 | 1.5395 |
| Sliding window (size 9) | 1.5053 | 1.0316 | 1.2211 |
| Sensitivity (%) Sensitivity | | | |
| Point by point | 100 % (38/38) | 100 % (38/38) | 100 % (38/38) |
| Sliding window (size 3) | 100 % (38/38) | 100 % (38/38) | 100 % (38/38) |
| Sliding window (size 5) | 100 % (38/38) | 100 % (38/38) | 100 % (38/38) |
| Sliding window (size 7) | 100 % (38/38) | 100 % (38/38) | 100 % (38/38) |
| Sliding window (size 9) | 97.36 %(37/38) | 97.36 %(37/38) | 97.36 %(37/38) |
| Precision (%) | | | |
| Point by point | 27.74 | 26.95 | 23.60 |
| Sliding window (size 3) | 29.92 | 27.54 | 24.68 |
| Sliding window (size 5) | 32.48 | 29.46 | 26.57 |
| Sliding window (size 7) | 36.54 | 31.15 | 28.57 |
| Sliding window (size 9) | 41.30 | 33.93 | 31.40 |

## 5-3 Summary

As listed in Table 5-11 and Table 5-21, classifier performance in above scenarios can be evaluated in five indexes: training time, computational time, advance time, sensitivity and precision.

In addition to listing figures in one table, radar charts are introduced in this analysis. Radar chart is a graphical method of displaying multivariate data in the form of a two-dimensional chart of many quantitative variables represented on axes starting from the same point. Pros and cons can be observed directly from this chart type.

However, data in each evaluation index has its own scale. For instance, training time is around 550 seconds but computation time is only 0.005 seconds. It is not possible to present them in the same chart without a rescaling process. In this process, a scoring system is applied, in which lowest score is 1 and highest being 4. Every values in the index will be assigned a number from 1 to 4 according to their original values.

Another problem that makes it difficult to compare these indexes in the same chart is the difference in evaluation criterion. For instance, the smaller training time is the better efficiency there will be. But for advance time one expect it to be as large as possible. To solve problem, computation time and training time are changed to computation efficiency and training efficiency respectively. Shorter training time leads to higher training efficiency and the same goes for computation efficiency.

### Sliding window size

The first usage of radar chart in this study is to find a proper sliding window size. Results of varying window size applied in scenario 1 are presented in Figure 5-5. Outcomes for scenario 2 and scenario 3 are given in Figure 5-6 and 5-7.

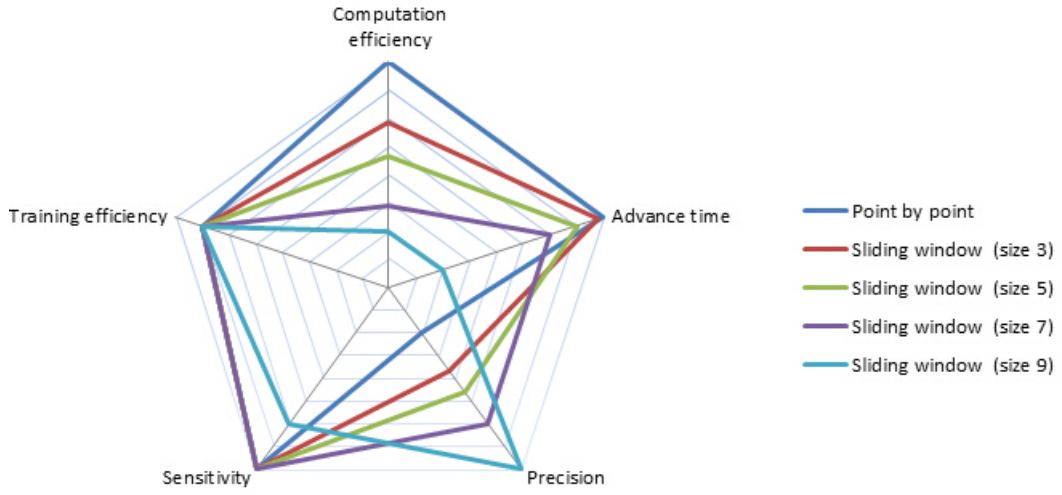


**Figure 5-5:** Performance evaluation varying window size – Scenario 1

As Figure 5-5 suggests, point by point method is best at swiftness but suffers from its low precision. A larger window size can reach higher precision at the cost of sacrificing computa-

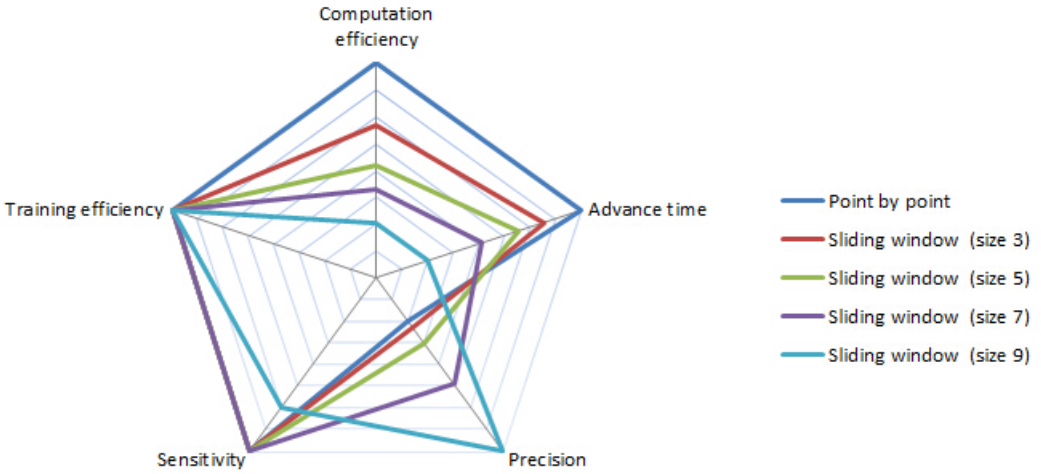


**Figure 5-6:** Performance evaluation varying window size – Scenario 2

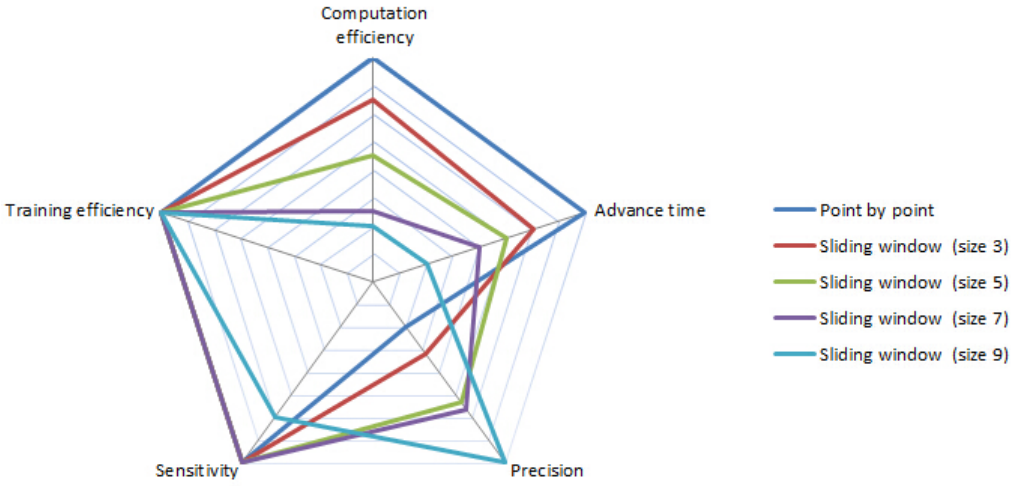


**Figure 5-7:** Performance evaluation varying window size – Scenario 3

tion efficiency and time in advance. The sensitivity even drops when size is chosen as 9. For this project, it is intolerable to have a true negative since a missed prediction of lane change will lead to dangerous situation for an automatic driving vehicle. Thus sliding window with size equal or larger to 9 is not an appropriate choice.

There will always be a trade-off between efficiency and precision when selecting window size. The same phenomenon can be observed in Figure 5-6 and Figure 5-7 as well. Window size 3 and 5 are relatively proper choice because of their balanced performance in each index. The same properties can be observed in validation results of Set 2 and additional experiments listed in Appendix B.

### Off-line and On-line

Comparisons between 3 scenarios, with sliding window size of 3 are showed in Figure 5-8.

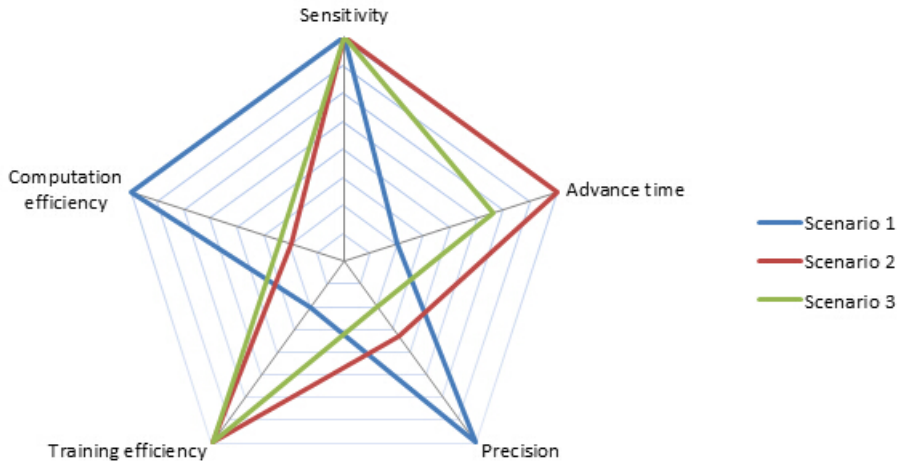


**Figure 5-8:** Performance evaluation of each scenarios with sliding window size 3

The advantages of on-line trained classifier are its training efficiency and prediction advance time. With the contribution from its active learning character, incremental trained classifier is more keen to potential lane changes. However, higher sensitivity will lead to greater false positive rate. A further reduction in false positive numbers is essential before this method is implemented on automatic vehicles. Another result from this figure is that Scenario 2 outperforms Scenario 3 in advance time and precision. This suggests incremental training with whole driver set yields better result than updating after each lane changes.

To further demonstrate the general applicability of this prediction method, 3 more tests are done on driver #4, driver #8 and driver #11. These driver numbers are chosen randomly. Test procedures are identical to Set 1 and Set 2. But only final comparison tables are presented for concision. The results of these experiments are listed in Appendix B.

# Chapter 6

# Conclusions and Future Work

## 6-1 Conclusions

In this study, an effective SVM based lane change prediction algorithm is developed. This prediction method is able to predict the short term future behaviour of other drivers on the externally observed vehicle trajectory of the surrounding traffic. The core concept of this method is to use certain physical parameters of surrounding vehicles as cues to reveal their imminent actions.

Using SVM based prediction approach in this study exhibited promising results. Conclusions can be drawn from those outcomes:

- Driving simulator test is a reasonable method to gather information concerning drivers' driving behaviours. The validity of acquired data set in capturing drivers' behaviour has been proved in further assessments. Features ( heading angle, yaw rate, lateral velocity and lateral acceleration ) recorded were able to describe vehicle maneuvers effectively.
- SVM based approach is verified to be feasible of predicting lane changes. According to validation tests Set 1, with the most optimal parameter combination, this method is able to perform predictions with 100% sensitivity ( predicted all lane changes successfully). Average advance time is approximate to 1.5 seconds. Average computational time is around 0.005 seconds, which is acceptable for automatic driving.
- In addition to the off-line training method, an incremental training approach was introduced. This approach enables the classifier to update itself after each prediction. As validation results suggest, this on-line trained classifier is able to provide 100% sensitivity output as well. This training algorithm is more practical for autonomous driving. Because the capability of an off-line trained classifier is always limited. DAVI vehicle will face unexpected traffic scenarios unavoidably, an on-line trainable classifier will be more applicable in predicting complicated traffic situations.

- Validation results in Set 2 demonstrated the general applicability of overall SVM prediction method. Further demonstrations are presented in Appendix B.
- Sliding window approach outperforms point by point method in effectively reducing false positives.

## 6-2 Future works

Due to the limitation of time, costs and safety concerns, some tasks were not implemented in this study. Moreover, in the purpose of developing a prediction method with better performance. There are still some possible future research tasks:

- Testing with more drivers and different traffic scenarios can be held for better evaluation. More lane change related parameters can be studied. It is also possible to investigate lane changes from psychological aspect.
- Instead of a driving simulator test, a real field test on studying lane change behaviours can be arranged. The shortcoming of simulator test is its validility Data recorded from field test will be more realistic than from a simulator. The whole procedures might be more complex but data gained in this test will be more applicable DAVI project.
- It is also of interest to study lane changes in a more macro aspect. So a lane change prediction can be made in combination of ego-vehicle observation and macro traffic situation.
- Develop a method to further decrease the number of false positives. As suggested in validation results, incremental SVM is more sensitive to lane changes (results in more false positives) than off-line training. A reduction in false positives will boost overall performance of incremental SVM.
- Implement this prediction algorithm in C/C++ in order to integrate it with the current DAVI platform.

# Appendix A

# SVM related problems

## A-1 Probability estimation

Classical SVMs can produce numerical results. For this project, a probabilistic output in addition to output label will be more favourable for users.

To reach this objective, an approach based on LIBSVM toolbox is implemented . This description is based on texts by Chih-Chung Chang and Chih-Jen Lin [29]. For more information about SVM algorithm, their texts are highly recommended.

Suppose there are $k$ classes of data, with input defined as $x$, possibility can be estimated through Equation A-1.

$$p_i = P(y = i|x), \quad i = 1, ..., k \tag{A-1}$$

With one against one strategy applied, the pairwise class probabilities can be obtained via Equation A-2.

$$r_{ij} \approx P(y = i|y = i \text{ or } j, x) \tag{A-2}$$

With $r_{ij}$ collected, $p_i$ can be obtained by solving an optimization problem in Equation A-3.

$$\begin{aligned} \min_{p} \quad & \tfrac{1}{2} \textstyle\sum_{i=1}^{k} \sum_{j:j\neq i} (r_{ji} p_i - r_{ij} p_j)^2 \\ \text{subject to} \quad & p_i \geq 0, \forall i, \textstyle\sum_{i=1}^{k} p_i = 1 \end{aligned} \tag{A-3}$$

which can be rewritten as

$$\min_{p} \quad \frac{1}{2} p^T Q p \tag{A-4}$$

where

$$Q_{ij} = \begin{cases} \sum_{s:s\neq i} r_{si}^2 & \text{if } i = j, \\ -r_{ji}r_{ij} & \text{if } i \neq j \end{cases} \tag{A-5}$$

It is proved in the work by Wu et al.[30] that $p_i \geqslant 0, \forall i$ are redundant. There will be a Lagrange multiplier $b$ of the equality constraint $\sum_{i=1}^{k} p_i = 1$ such that

$$\begin{bmatrix} Q & e \\ e^T & 0 \end{bmatrix} \begin{bmatrix} p \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{A-6}$$

In which $e$ is a $k \times 1$ vector of ones.

An alternative method was proposed by Wu et al. [30]. Using the relation of

$$-p^T Qp = -p^T Q(-be) = bp^T e = b \tag{A-7}$$

to obtain the optimal solution $p$

$$(Qp)_t - p^T Qp = Q_{tt}p_t + \sum_{j:j\neq t} Q_{tj}p_j - p^T Qp = 0, \quad \forall t \tag{A-8}$$

This equation can be applied to help us to avoid solving A-6 via a direct method.

This method starts with $p$ initiated under $\sum_{i=1}^{k} p_i = 1$. It iterates with procedure described in Equation A-9. Normalized $p$ is computed for each time step. Until $p_t$ satisfy Equation A-6.

$$p_t \leftarrow \frac{1}{Q_{tt}}[-\sum_{j:j\neq t} Q_{tj}p_j + p^T Qp] \tag{A-9}$$

## A-2 Dual problem

An optimization problem with linear constraints can be transformed to a dual problem [31]. Linear constraints will be applied on the variables in objective function during the transformation. For instance, an optimization problem with linear constraints is defined as follows:

$$\begin{aligned} \min \quad & f(x) \\ \text{S.b.t} : \quad & g_i(x) \leq 0 \\ & h_j(x) = 0 \end{aligned} \tag{A-10}$$

where $f(x)$ is a quadratic function in $x$; $g(x)$ and $h(x)$ are linear constraints; $i$ and $j$ are indexes for inequality and equality constraints respectively. The corresponding dual form is defined as :

$$D = \frac{1}{2}f(x) + \sum_i \alpha_i g_i(x) + \sum_j \beta_j h_j(x) \tag{A-11}$$

# Appendix B

# Validation related

## B-1 Prediction results and ground truth

In addition to Set 2 in Chapter 5, to further examine the general applicability, the same validation procedure have been done on driver #4, driver #8 and driver #11. Comparison results are listed in Table B-1, Table B-2 and Table B-3 respectively.

**Table B-1:** Prediction performance comparison - Driver #4

| #4 | Offline learning | Online learning | |
|---|---|---|---|
| | Scenario 7 | Scenario 8 | Scenario 9 |
| Training Process | | | |
| Total training time (s) | 770.20 | 602.78 | 602.78 |
| Self-test accuracy | 71.81% | 70.66% | 70.66% |
| Prediction average Computation time(s) | | | |
| Point by point | 0.0019 | 0.0039 | 0.0039 |
| Sliding window (size 3) | 0.0029 | 0.0045 | 0.0058 |
| Sliding window (size 5) | 0.0036 | 0.0051 | 0.0067 |
| Sliding window (size 7) | 0.0044 | 0.0056 | 0.0055 |
| Sliding window (size 9) | 0.0052 | 0.0059 | 0.0059 |
| Prediction average Advance time (s) | | | |
| Point by point | 1.6406 | 1.85 | 1.7406 |
| Sliding window (size 3) | 1.2813 | 1.5219 | 1.5024 |
| Sliding window (size 5) | 1.0781 | 1.4219 | 1.3571 |
| Sliding window (size 7) | 0.99687 | 1.325 | 1.2375 |
| Sliding window (size 9) | 0.89687 | 0.77812 | 0.65 |
| Sensitivity | | | |
| Point by point | 100 % (32/32) | 100 % (32/32) | 100 % (32/32) |
| Sliding window (size 3) | 100 % (32/32) | 100 % (32/32) | 100 % (32/32) |
| Sliding window (size 5) | 100 % (32/32) | 100 % (32/32) | 100 % (32/32) |
| Sliding window (size 7) | 100 % (32/32) | 100 % (32/32) | 96.875 % (31/32) |
| Sliding window (size 9) | 100 % (32/32) | 100 % (32/32) | 96.875 % (31/32) |
| Precision (%) | | | |
| Point by point | 42.67 | 39.5 | 36.78 |
| Sliding window (size 3) | 50.79 | 41.56 | 44.56 |
| Sliding window (size 5) | 57.14 | 41.56 | 48.24 |
| Sliding window (size 7) | 66.67 | 47.06 | 44.44 |
| Sliding window (size 9) | 71.11 | 58.18 | 55.17 |

**Table B-2:** Prediction performance comparison - Driver #8

| #8 | Offline learning | Online learning | |
|---|---|---|---|
| | Scenario 10 | Scenario 11 | Scenario 12 |
| Training Process | | | |
| Total training time (s) | 753.15 | 647.27 | 647.27 |
| Self-test accuracy | 71.28% | 69.39% | 69.39% |
| Prediction average Computation time(s) | | | |
| Point by point | 0.0022 | 0.0065 | 0.0061 |
| Sliding window (size 3) | 0.0034 | 0.0086 | 0.0064 |
| Sliding window (size 5) | 0.0041 | 0.0090 | 0.0075 |
| Sliding window (size 7) | 0.0049 | 0.0094 | 0.0079 |
| Sliding window (size 9) | 0.0059 | 0.0097 | 0.0087 |
| Prediction average Advance time (s) | | | |
| Point by point | 1.6625 | 1.6875 | 1.8675 |
| Sliding window (size 3) | 1.4625 | 1.5238 | 1.7125 |
| Sliding window (size 5) | 1.3300 | 1.5875 | 1.5875 |
| Sliding window (size 7) | 1.0225 | 1.4125 | 1.4725 |
| Sliding window (size 9) | 0.9225 | 1.2625 | 1.265 |
| Sensitivity | | | |
| Point by point | 100 % (40/40) | 100 % (40/40) | 100 % (40/40) |
| Sliding window (size 3) | 100 % (40/40) | 100 % (40/40) | 100 % (40/40) |
| Sliding window (size 5) | 100 % (40/40) | 100 % (40/40) | 100 % (40/40) |
| Sliding window (size 7) | 97.5 % (39/40) | 100 % (40/40) | 100 % (40/40) |
| Sliding window (size 9) | 97.5 % (39/40) | 100 % (40/40) | 100 % (40/40) |
| Precision (%) | | | |
| Point by point | 61.54 | 47.62 | 40.40 |
| Sliding window (size 3) | 60.00 | 48.19 | 42.55 |
| Sliding window (size 5) | 66.67 | 48.19 | 42.11 |
| Sliding window (size 7) | 75.00 | 54.05 | 49.38 |
| Sliding window (size 9) | 78.00 | 58.82 | 55.56 |

**Table B-3:** Prediction performance comparison - Driver #11

| #11 | Offline learning | Online learning | |
|---|---|---|---|
| | Scenario 13 | Scenario 14 | Scenario 15 |
| Training Process | | | |
| Total training time (s) | 861.30 | 726.16 | 726.16 |
| Self-test accuracy | 71.62% | 70.11% | 70.11% |
| Prediction average Computation time(s) | | | |
| Point by point | 0.0023 | 0.0047 | 0.0048 |
| Sliding window (size 3) | 0.0046 | 0.0073 | 0.0058 |
| Sliding window (size 5) | 0.0061 | 0.0073 | 0.0067 |
| Sliding window (size 7) | 0.0075 | 0.0075 | 0.0076 |
| Sliding window (size 9) | 0.0074 | 0.0083 | 0.0081 |
| Prediction average Advance time (s) | | | |
| Point by point | 1.4342 | 1.5132 | 1.7762 |
| Sliding window (size 3) | 1.2526 | 1.3763 | 1.5024 |
| Sliding window (size 5) | 1.1368 | 1.1684 | 1.3571 |
| Sliding window (size 7) | 1.0368 | 1.1342 | 1.1711 |
| Sliding window (size 9) | 0.94211 | 0.97105 | 1.0447 |
| Sensitivity | | | |
| Point by point | 100 % (38/38) | 100 % (38/38) | 100 % (38/38) |
| Sliding window (size 3) | 100 % (38/38) | 100 % (38/38) | 100 % (38/38) |
| Sliding window (size 5) | 100 % (38/38) | 100 % (38/38) | 100 % (38/38) |
| Sliding window (size 7) | 100 % (38/38) | 100 % (38/38) | 100 % (38/38) |
| Sliding window (size 9) | 100 % (38/38) | 100 % (38/38) | 100 % (38/38) |
| Precision (%) | | | |
| Point by point | 54.29 | 44.19 | 42.27 |
| Sliding window (size 3) | 67.86 | 45.78 | 44.56 |
| Sliding window (size 5) | 73.08 | 52.05 | 48.24 |
| Sliding window (size 7) | 77.55 | 58.46 | 55.07 |
| Sliding window (size 9) | 82.61 | 71.69 | 63.33 |

# Bibliography


[1] R. Hoogendoorn and et al., "Towards safe and efficient driving through vehicle automation: The dutch automated vehicle initiative."

[2] eIMPACT, "Accident data compilation - challenges and solutions," Paris, 2008.

[3] I. Dagli, M. Brost, and G. Breuel, "Action recognition and prediction for driver assistance systems using dynamic belief networks," in *In Proceedings of the Conference on Agent Technologies, Infrastructures, Tools, and Applications for E-Services*, pp. 179–194, Springer-Verlag, 2002.

[4] S. Tezuka, H. Soma, and K. Tanifuji, "A study of driver behavior inference model at time of lane change using bayesian networks," in *Industrial Technology, 2006. ICIT 2006. IEEE International Conference on*, pp. 2308–2313, 2006.

[5] A. Liu and A. Pentland, "Towards real-time recognition of driver intentions," in *Proceedings of the IEEE Conference on Intelligent Transportation Systems, Boston, USA*, p. pages 236âĂŞ241, Nov 1997.

[6] A. Pentland and A. Liu, "Modeling and prediction of human behavior," *Neural Comput*, vol. 11, pp. 229–242, 1999.

[7] N. Oliver and A. Pentland, "Driver behavior recognition and prediction in a smartcar," *Enhanced and Synthetic Vision*, vol. 4023, pp. 280–290, 2000.

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

[9] D. Salvucci, "Inferring driver intent: A case study in lane-change detection," in *in Proceedings of the Human Factors Ergonomics Society 48th Annual Meeting*, pp. 2228–2231, 2004.

[10] Y. Hou, P. Edara, and C. Sun, "A genetic fuzzy system for modeling mandatory lane changing," *International IEEE Conference on Intelligent Transportation Systems*, 2012.

[11] H. M. Mandalia and D. Salvucci, "Using support vector machines for lane change detection," in *In Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting*, 2005.

[12] Z. Li, "Prediction of vehicles' trajectories based on driver behavior models : literature survey," 2014. Delft university of technology.

[13] N. Syed, S. Huan, K. Liu, and S. Kay, "Incremental learning with support vector machines," 1999.

[14] T. Toledo and D. Zohar, "Modeling duration of lane changes," *Transportation Research Record: Journal of the Transportation Research Board*, pp. pp 71–78, 1999.

[15] G. Aoude, *Threat Assessment for Safe Navigation in Environments with Uncertainty in Predictability*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, July 2011.

[16] H. Berndt, J. Emmert, and K. Dietmayer, "Continuous driver intention recognition with-hidden markov models," in *Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems*, 2008.

[17] H. Mandalia, "Pattern recognition techniques to infer driver intentions," tech. rep., Drexel University, 2004.

[18] C. Cortes and V. Vapnik, "Support-vector networks," in *Machine Learning*, pp. 273–297, 1995.

[19] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer New York, 1998.

[20] C. Hsu and C. Lin, "A comparison of methods for multiclass support vector machines," *Neural Networks, IEEE Transactions on*, vol. 13, pp. 415–425, Mar 2002.

[21] S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer learning revisited: A stepwise procedure for building and training a neural network," in *Neurocomputing: Algorithms, Architectures and Applications*, vol. F68 of *NATO ASI Series*, pp. 41–50, Springer-Verlag, 1990.

[22] L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Comparison of classifier methods: a case study in handwritten digit recognition," in *Pattern Recognition, 1994. Vol. 2 - Conference B: Computer Vision amp; Image Processing., Proceedings of the 12th IAPR International. Conference on*, vol. 2, pp. 77–82 vol.2, Oct 1994.

[23] J. Platt, N. Cristianini, and J. Shawe-taylor, "Large margin dags for multiclass classification," in *Advances in Neural Information Processing Systems*, pp. 547–553, MIT Press, 2000.

[24] C. Demirkesen and H. Cherifi, "A comparison of multiclass svm methods for real world natural scenes," in *Advanced Concepts for Intelligent Vision Systems*, vol. 5259, pp. 752–763, 2008.

[25] C. Diehl and G. Cauwenberghs, "Svm incremental learning, adaptation and optimization," in *Neural Networks, 2003. Proceedings of the International Joint Conference on*, vol. 4, pp. 2685–2690 vol.4, July 2003.

[26] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems (NIPS*2000)*, vol. 13, 2001.

[27] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[28] M. Pal, "Kernel methods in remote sensing: A review," *ISH Journal of Hydraulic Engineering*, vol. 15, pp. 194–215, 2009.

[29] C. Chang and C. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[30] T. Wu, C. Lin, and R. Weng, "Probability estimates for multi-class classification by pairwise coupling," *Journal of Machine Learning Research*, 2004.

[31] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical recipes*. Cambridge University Press, 3rd ed., 2007.

# Glossary

## List of Acronyms

| | |
|---|---|
| **SVM** | Support Vector Machine |
| **DAVI** | Dutch Automatic Vehicle Initiative |
| **HMM** | Hidden Markov Model |
| **OvO** | One versus One |
| **OvA** | One versus All |
| **KKT** | Karush-Kunh-Tucker |
| **RBF** | Radial Basis Function |
| **LCL** | Lane Change Left |
| **LK** | Lane Keeping |
| **LCR** | Lane Change Right |
| **TU Delft** | Delft University of Technology |