Otto-von-Guericke University of Magdeburg

Faculty of Computer Science
Department of Knowledge and Language Processing

Diploma Thesis

Application of Support Vector Machines to Discriminate Vehicle Crash Events

Author:

Christian Moewes

October 15, 2007

Supervisors:

Bernard Ramirez Araya
SV C RS CC Safety Electronics
Siemens VDO Automotive AG
Siemensstraße 12
D-93055 Regensburg

Dr. Clemens Otte
CT IC 4 Learning Systems
Siemens AG
Otto-Hahn-Ring 6
D-81739 München

Advisers:

Prof. Dr. Eyke Hüllermeier
Fachbereich Mathematik und Informatik
Philipps-Universität Marburg
Hans-Meerwein-Str. (Lahnberge)
D-35032 Marburg

Prof. Dr. Rudolf Kruse
Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2
D-39106 Magdeburg

Moewes, Christian:
Application of Support Vector Machines to Discriminate Vehicle Crash Events
Diploma Thesis, Otto-von-Guericke University
Magdeburg, 2007.

Abstract

This diploma thesis was written within the scope of my work on a project in the Department of Restraint Systems at Siemens VDO Automotive AG in Regensburg, Germany from March 26 until September 28, 2007.

The security requirements of motor vehicles in the road traffic rises up gradually nearly every year due to the massively increasing number of cars, the continuously becoming faster automobiles and the steadily growing amount of safety regulations. With the objective of fulfill these requirements, automotive suppliers like Siemens VDO work with the utmost care on the implementation of restraint systems together with motor vehicle manufacturers to protect the driver, passengers, pedestrians and other road users.

Nowadays, restraint systems like multistage drivers' and passengers' airbags, belt tensioner and side airbags can be found in almost every modern car. Many steps in the prototypical vehicle development are necessary to come up with an absolutely reliable restraint system. The embedded systems have to be adopted and calibrated to every type of vehicle and many different crash scenarios.

The crash data usually contain conflicts which are equivalent to some requirements that are hard to fulfill. The discussions with the customers about changes of demands to solve these conflicts commonly take several weeks. A conflict found at the end of a project very often entails the project's deathblow.

In my thesis, I will concentrate on conflict analysis in order to improve and accelerate the calibration process. I will present machine learning algorithms based on support vector machines that try to find conflicts in crash data.

Zusammenfassung

Diese Diplomarbeit entstand im Rahmen meiner Arbeit an einem Projektes im Zeitraum vom 26. März bis zum 28. September 2007 in der Abteilung Restraint Systems bei Siemens VDO Automotive AG in Regensburg.

Die Sicherheitsanforderungen aller Fahrzeuge im Straßenverkehr nehmen von Jahr zu Jahr aufgrund des massiv steigenden Verkehrsaufkommens, der immer schneller werdenden Autos und der stets wachsende Anzahl an Sicherheitsbestimmungen stetig zu. Um diesen Anforderungen gerecht zu werden, arbeiten Automobilzulieferer wie Siemens VDO mit den Fahrzeugherstellern akribisch an der Implementierung von Rückhaltesystemen zum Schutz von Fahrzeuginsassen, Fußgängern und weiteren Verkehrsteilnehmern.

Rückhaltesysteme wie mehrstufige Fahrer- und Beifahrerairbags, Gurtstraffer und Seitenairbags lassen sich heutzutage beinahe in jedem modernen Auto finden. Um jedoch ein absolut zuverlässiges Rückhaltesystem zu entwickeln, sind viele Schritte in der prototypischen Entwicklung des Fahrzeuges von Nöten. So müssen die eingebetteten Systeme auf jeden einzelnen Fahrzeugtyp und auf sehr viele verschiedene Crash-Situationen angepasst und kalibriert werden.

Für gewöhnlich enthalten die Crash-Daten sogenannte Konflikte. Dabei handelt es sich um Anforderungen, die nur schwer oder gar nicht zu erfüllen sind. Die Diskussionen über deren Änderungen mit den Kunden, um diese Konflikte zu lsen, dauern normalerweise mehrere Wochen. Ein Konflikt, der erst am Ende eines Projektes gefunden wurde, ist sehr häufig gleichbedeutend mit dem Todesstoßes des Projektes.

In meiner Arbeite werde ich mich auf die Konfliktanalyse konzentrieren um den Kalibrierungsprozess zu verbessern und zu beschleunigen. Ich werde maschinelle Lernverfahren basierend auf Support Vector Machines vorstellen, die versuchen, Konflikte in Crash-Daten zu finden.

Acknowledgment

I would like to thank my parents, Carola and Günther Moewes, who made it possible that I could finish my studies. Even though we did not have the same opinion from time to time, I want to let them know that I love them. Without their support, I would not have come that far in my life.

I want to give thanks to my advisers Prof. Dr. Eyke Hüllermeier and Prof. Dr. Rudolf Kruse who read this thesis prior to its publication. I appreciate the suggestions they gave to me, and their propositions to finalize this thesis.

My thanks especially go to my supervisors Bernard and Clemens. Our stimulating discussions helped me to understand many problems concerning crash calibration, theory and implementation as well. I am very grateful to their critical comments on this thesis in order to finalize it. I want to thank Bernard in particular since he assisted me nearly every day and prevented me from entering one-way streets.

I acknowledge the time and discussions I had with my boss, Michael Feser, who was concerned about my work and myself. He was always willing to help me when I had organizational problems with my work.

Finally, I would like to show my appreciation to Pavla Adamcová, the most important person in my life. Pavla, I am very grateful for the patience, the encouragement and the support you gave to me. Every moment I could spend with you is precious to me. Miluju tě.

"If people do not believe that mathematics is simple, it is only because they do not realize how complicated life is." [6]

John von Neumann

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ACU	Airbag (Electronic) Control Unit
AZT	Allianz Zentrum für Technik
DEUTA	Deutsche Tachometer-Werke GmbH
ECS	Early Crash Sensors
ERM	Empirical Risk Minimization
EuroNCAP	European New Car Assessment Program
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
NHTSA	National Highway Traffic Safety Administration
OSA	Otto Schulze Autometer
ODB	Offset (40 %) Deformable Barrier
OTA	Offenbacher Tachometer Werke
RS	Department of Restraint Systems
RTTF	Requested Time to Fire
SRM	Structural Risk Minimization
SV	Support Vector/Siemens VDO
SVM	Support Vector Machine
TTF	Time to Fire
VDO	Vereinigte Deuta OTA

1 Introduction

This thesis proposes methods for the discrimination of vehicle crash events by means of support vector machines. Before explaining these methods thoroughly, we first have a look at the given problem definition that we try to solve.

In this chapter a basic introduction to the topic of the thesis is given. First of all, the importance of vehicle safety is concisely presented in the first section. After that, in Section 1.2 the development of a restraint system is briefly described. The main motivation of this thesis is given in Section 1.3. In addition, a short history of the company Siemens VDO Automotive AG can be found in Section 1.4 since the suggested methods of this thesis were developed together with the automobile supplier in Regensburg, Germany. Finally, an overview of the thesis in Section 1.5 concludes this chapter.

1.1 The Importance of Vehicle Safety

Motor vehicles are used for transport to a huge degree. Especially this means of transportation has a major drawback in terms of the number of motor vehicle crashes and their effects like crash costs, accidental injuries and automobile fatalities. To give an example, in 2001, the costs of road crashes sums up to more than 170 billion Euros in the European Union. Moreover, more than 38,000 EU citizen lost their lives causes by these crashes. Due to that, transport crashes were the leading cause of death and hospital admission for citizen under 50 years in the same year [2].

To give a more recent example, Figure 1.1 shows the road fatalities in the European Union in the year 2005. The map expresses road fatalities per million population of every country of the EU. Not surprisingly, the interpretation of this figure can lead to manifold claims about country-specific conditions of drivers, roads, vehicles and so forth. Without analyzing the shown numbers in detail, it becomes clear that a complete prevention of road fatalities is a hopeless mission.

Nevertheless, a decrease of these numbers and their consequences is obviously a necessity for the economical, ethical, medical and last not least engineering sake. The importance of safe vehicles should be a major concern and represents an approach with a striking effect. In order to make vehicles safer, a number of new devices have been installed. Active safety facilities,

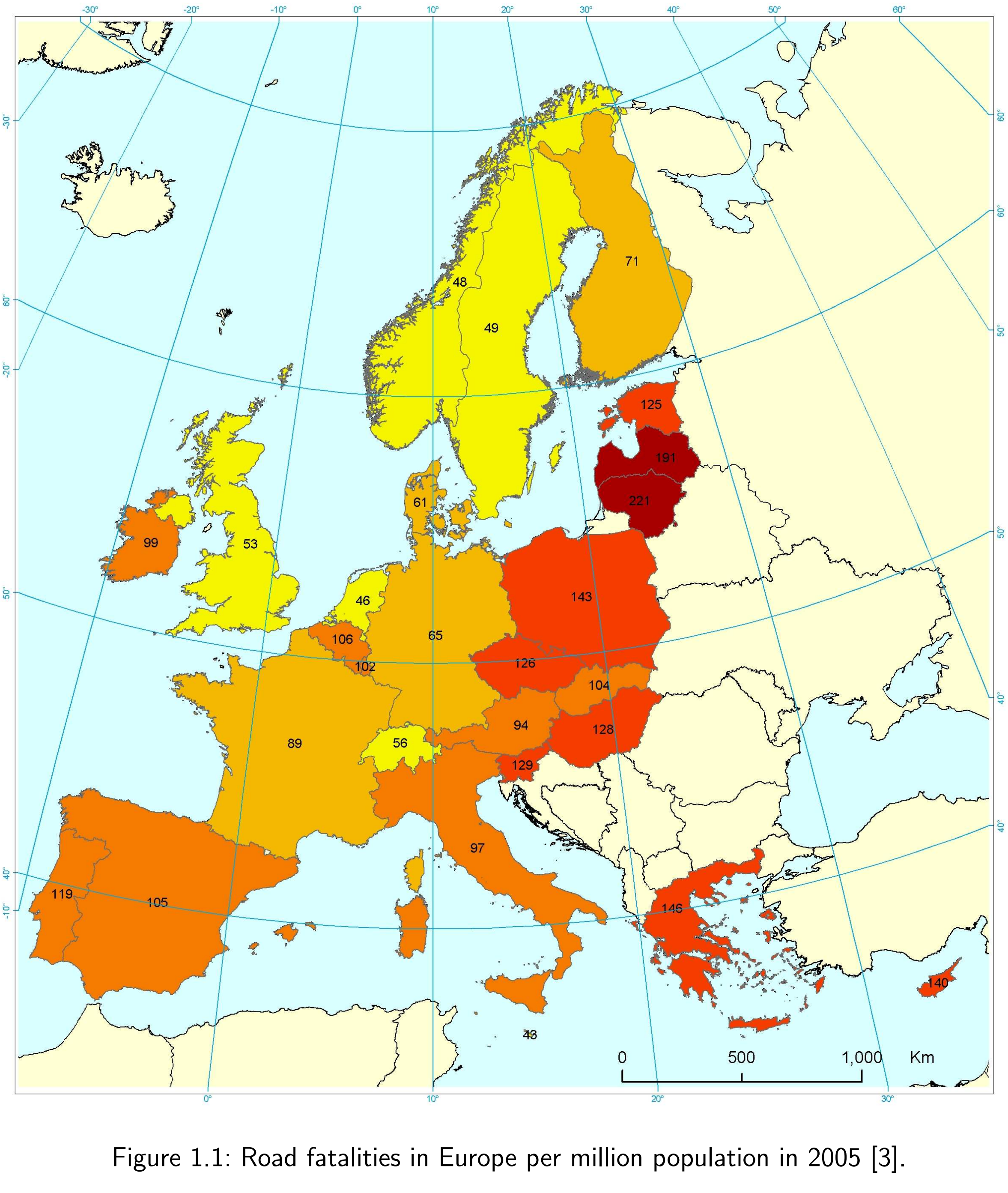


Figure 1.1: Road fatalities in Europe per million population in 2005 [3].

for instance, purpose to lower the possibilities of a crash event. Popular representatives like the electronic brake distribution (EBD) and the anti-lock breaking system (ABS) perform very well.

Another group of devices which are referred to passive safety facilities aims to reduce the consequences in a crash event in terms of injury severity. Safety belt and crushing zone are only two examples. Figure 1.2 shows the phases of a crash which are directly connected to the type of applied safety devices. Active safety devices might be applied to the driving dynamic by the automobile system in order to prevent crashes, whereas passive safety facilities come into play when a crash occurs to lower the crash consequences.

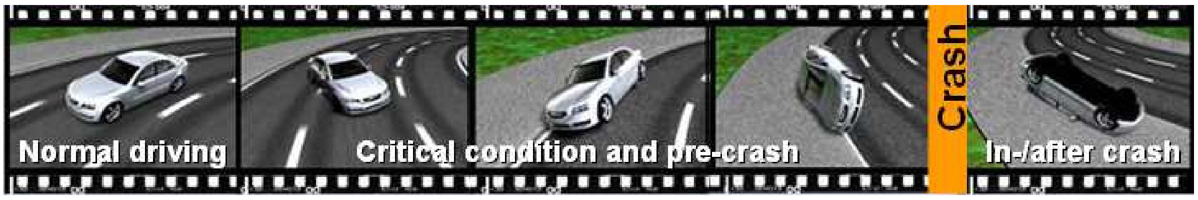


Figure 1.2: Phases of a crash [4]. In normal, critical or pre-crash situations, active safety systems may be used by the driver. On the other hand, passive safety systems are triggered automatically whilst in- and post-crash situations, respectively.

The combination of passive safety devices that restrain the occupant during a crash is named restraint system. The task of a restraint system is to reduce the risk of severe or fatal injuries by consume the occupant's kinetic energy. Since their introduction, safety belt and airbag are the best known facilities.

Since 1973, when the first airbags could be found as an accessory in the Chevrolet Impala, the restraint system of safety belt and airbag has become increasingly reliable, also due to innovations of additional facilities like the belt tensioner. This device prevents a relaxed belt shortly after a crash is observed.

In the nineties of the last century, restraint systems with only one operation stage could be found in automobiles. Occupant's seating position, mass and age do have an influence on the crash consequences concerning injuries [22], however, a single operation mode might not be sensitive enough in most of the crashes. Moreover, the velocity has a major impact on the risk of injuries. It may even happen that different crashes set conflicting demands on the restraint system. Furthermore, severe or even fatal injuries can be caused by a restraint system itself.

Thus multistage restraint systems are in focus of research and development today. By heuristics, the most suitable stage of operation is chosen when the crash is discovered. These heuristic rules are based on measurements of occupant and crash features. For instance, it is necessary to correctly set the point of time when the airbag has to be deployed.

1.2 The Development of Restraint Systems

In fact, it is a very ambitious task to develop a restraint system. There are a dozens of features that come into play. The characteristics of automobiles differs a lot from each other. The courses of crashes are thence different as well. Last but not least the characteristics of the occupant have a big influence on each restraint system.

At the end of the development process, possible changes to the automobile or the restraint system are exceedingly costly. Hence the development has to be processed extremely carefully. Another crucial point is that factors like expenses, durability, comfort, size, mass and ergonomics are in conflict with each other and with the operation of the restraint system as well.

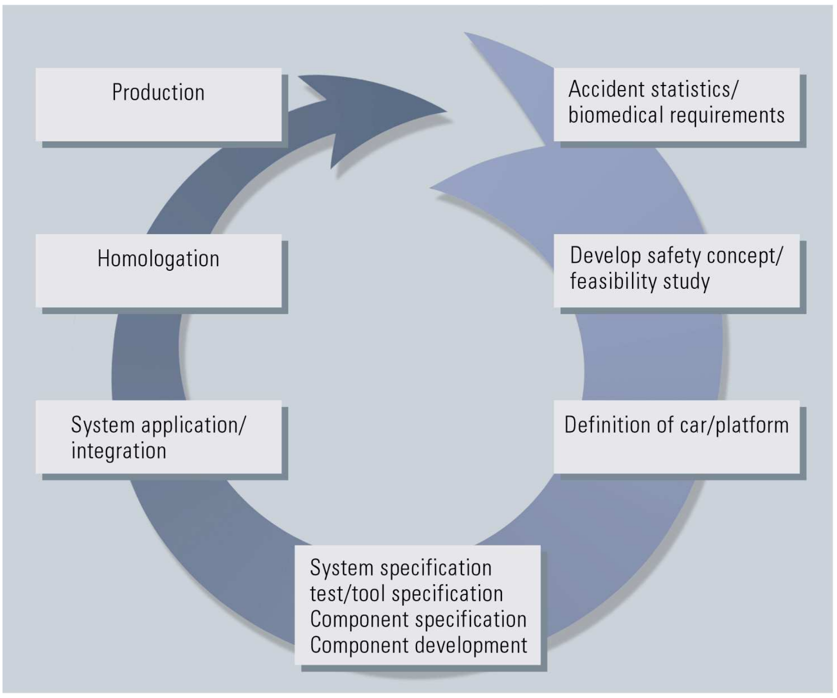

Production
Accident statistics/ biomedical requirements
Homologation
Develop safety concept/ feasibility study
System application/ integration
Definition of car/platform
System specification
test/tool specification
Component specification
Component development

Figure 1.3: The development cycle of a restraint system [4].

Figure 1.3 shows the process of a restraint system in a cycle. Presuming biomedical requirements and accident statistics, a safety concept is developed. In this step without any prototypical automobile [33], simulations are run to define the restraint system. Here, the geometry of the interior and the seating position are defined as part of the car platform. The requirements for the new restraint system as input for the simulation model are given by the customer. For instance, the customer determines a requested time to fire (RTTF) which is an upper limit for a certain restraint system when it has to be activated after a crash is detected.

Then the first crash test with a prototype and its pre-optimized restraint system is performed.

Its results generate the input for a subsequent validation crash test. The general behavior of the developed restraint system depends on the amount of sled tests which are performed. Fine tuning is performed together with both sled tests and simulation. Finally the restraint system is confirmed by a crash test.

The goal of that development is a restraint system which can face nearly every crash and occupant for the specified automobile platform. On the other hand, the system has to meet all other criteria like expenses, durability etc. In order to aim the requirements and to develop such a system, lots of different tests like numerical simulations, real world component tests and full scale crash tests have to be performed.

1.2.1 Crash Tests

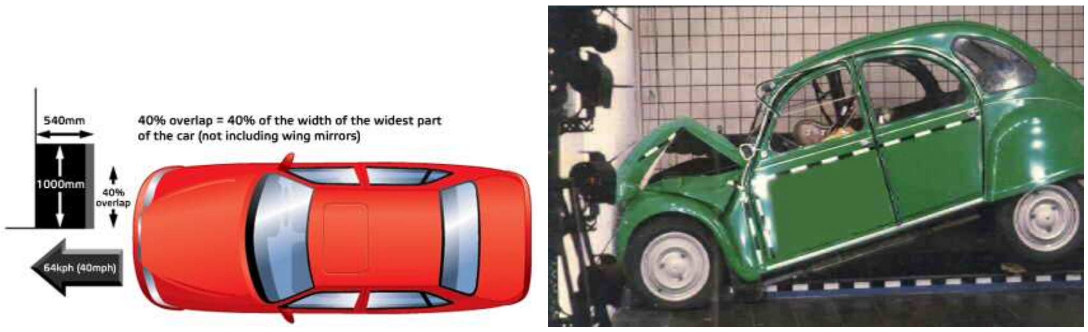
Almost all crashes take place in a fraction of one second. In this short time, a huge number of physical processes occur. This section first clarifies important aspects of a crash, the restraint system and the occupant, respectively. After that, relevant characteristics of a crash are cleared up.

Frontal crashes cause the most fatal injuries with more than 50% [5]. Crash tests were introduced to have a legislative basis for the term of a minimal safe car. All crash tests are standardized and take place under specified conditions with crash test dummies as passenger substitutes. These crash tests are performed by car manufacturers and insurance institutes as well in order to reveal weaknesses to the public.

Real world crashes brought out that mainly 4 parameters determine the severity of occupant injuries: the relative velocity, the relative angle, the relative overlap and the effective stiffness of the crash partners. Thus various crash tests have been developed. Figure 1.4(a) shows a basic framework of a frontal crash test as it is performed by EuroNCAP [1]. In this scenario, the relative angle α is 0, the relative velocity v_0 amounts 64 km/h and the relative overlap or offset $\Delta L/L$ is equivalent to 40%.

By changing these 4 parameters, different types of crash tests can be obtained. For instance, a deformable or a rigid barrier can be chosen either on the left or right side of the automobile front. Crash tests with rigid barriers are referred to *wall crashes* as it is shown in Figure 1.4(b). Whenever the barrier is deformable, *ODB (offset deformable barrier) crashes* are performed. The offset $\Delta L/L$ is usually either 40% or 100%. An ODB crash test tries to emulate a crash where two cars crashes frontal against each other.

If the relative angle is different from zero (usually 30 degrees), so-called *angular crashes* are executed. The impact velocity can have different values normally starting from 15 km/h up to 80 km/h. One also distinguishes between so-called *low- and high-speed crashes*, respectively depending on v_0.

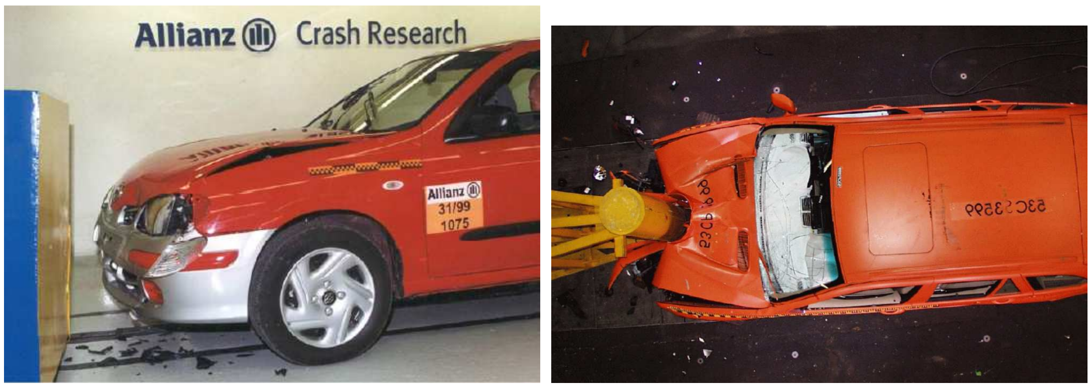

540mm
40% overlap = 40% of the width of the widest part of the car (not including wing mirrors)
1000mm
40% overlap
64kph (40mph)

(a) (b)

Figure 1.4: EuroNCAP [1] frontal crash tests [16]. The left figure shows the general configuration of a frontal crash with 40% overlap. On the right-hand side a crash test with a rigid barrier is shown. It is the analog to wall crashes.

Of course, apart from wall, ODB and angular crash tests, there are many more crash scenarios. The *Allianz Zentrum für Technik (AZT) GmbH* (http://azt.allianz.de) developed a very important type of crash test more than 20 years ago. The so-called *AZT (Danner) crash test* is performed by $v_0 = 15km/h$ and $\Delta L/L = 40\%$. In process of time, this test became standard in many countries.

Whenever an AZT test is performed, no restraint system shall be triggered since it might cause heavy injuries compared to those resulting from a high-speed crash. Figure 1.5(a) shows an example of such an AZT crash test. Such a low-speed crash can occur in many situations and it has to be ensured that neither airbag nor belt tensioner will trigger. Usually, a car can be repaired much easier after an AZT crash than after a high-speed crash since the outer damages to the car are mainly smaller dents.



(a) (b)

Figure 1.5: AZT (a) and frontal pole crash tests (b). The former type of crash is performed at 14 up to 16 km/h. Neither airbag, nor belt tensioner shall fire at these velocities. A frontal pole crash test shall reveal insights into frontal crashes with trees. (Both pictures are taken from [16])

Accidents where a car hit a tree are emulated by so-called *pole crash tests*. Basically front and lateral pole crash tests are performed. Figure 1.5(b) gives a real-world example of a frontal pole test. A lateral pole test, for instance, is shown in Figure 1.6.

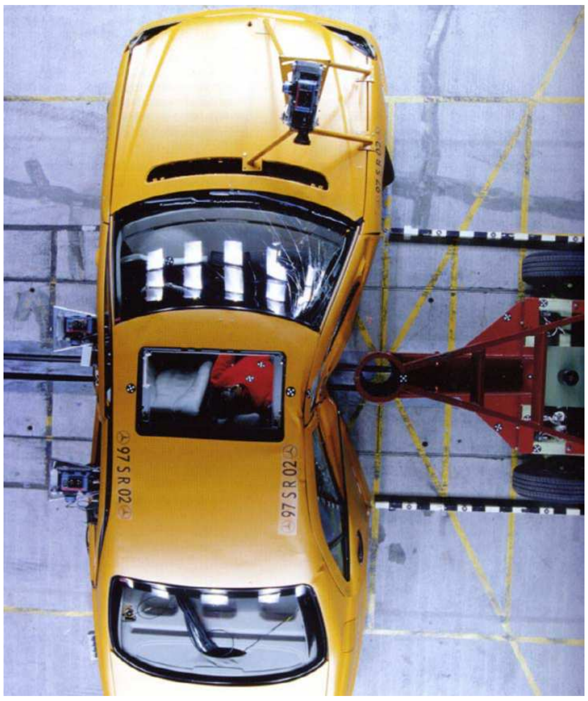


Figure 1.6: Lateral pol impact test [16] to gain information about real-world lateral tree crashes.

Aside from ODB, angular, wall and AZT crashes, there exists another big group of tests that are referred as *misuse tests*. This collection of tests tries to includes real-world scenarios where no restraint system shall be triggered. This is due to the fact that there might be another crash event directly after a misuse. This type of accidents is called *concatenated crash*. The latter event of such a concatenated crash usually requires the activation of passive safety devices.

Before we will explain why a misuse normally does not trigger any restraint system, some types of misuse crashes are briefly noted by some examples: driving on road of cobblestones, driving over a curb, road drop off, driving into a pothole, deer impact, wild boar impact, crash with cats or dogs, hammer blow hits car, steel or wooden beam falls onto automobile, etc.

Suppose that a restraint system like the airbag and belt tensioner were triggered after a misuse event, say a wild boar impact. According to that, there won't be any restraint for the driver or passenger if a severer crash (e.g. a collision with a tree) is subsequent to the misuse. Therefore every misuse is commonly treated like an AZT crash in terms of the fire decision.

The challenge is to distinguish between fire and no-fire events by a machine. Consequently, there have to be devices situated in the automobile that allow to draw conclusions about the type of crash event. These devices that are called sensors will be discussed in the next section in short.

1.2.2 Sensor Equipment of a Modern Car

Every modern car is equipped with a bunch of sensors and embedded systems. As might be expected, the question if and how a restraint system shall activate is answered based on sensor signals. The decision itself is made in the so-called tunnel-area right between driver and passenger seat. The electronic system for decision making is called airbag control unit (ACU).

The ACU, as shown in Figure 1.7(a), is the heart of the smart airbag system. Here, the system processes every incoming signal of each sensor. Based on that, potential commands are emerged. Moreover, diagnostics on the entire safety system are carried out continuously.

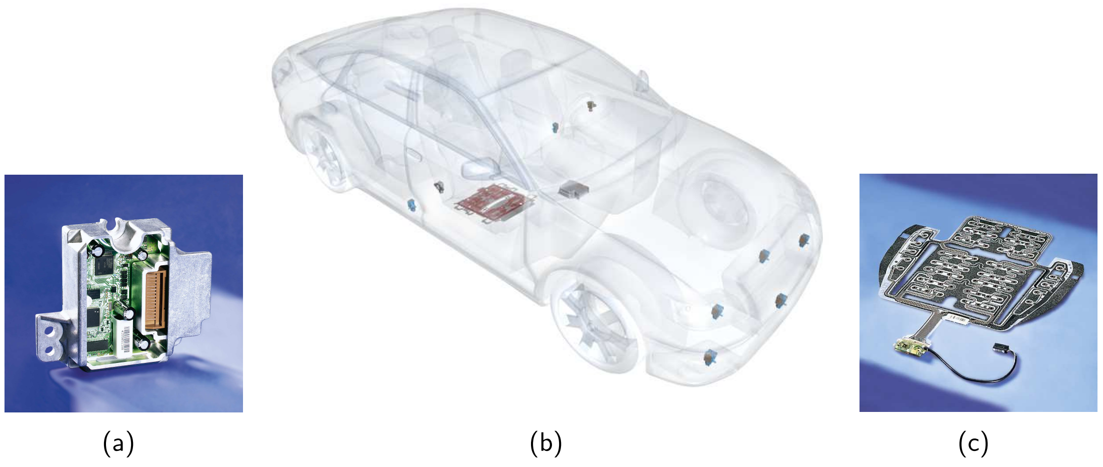


Figure 1.7: ACU on the left-hand side, sensor positions in the car shown in the middle, and occupant classification mat on the right hand side. (All pictures are taken from [4])

Figure 1.7(b) shows where the sensors and the ACU can be found in a standard automobile. The ACU itself contains a collection of internal sensors that, among others, record big changes of acceleration of the car. Further signals may be needed, however, to support or suppress a fire decision.

Such a signal can be obtained by a weight classification mat as shown in Figure 1.7(c). It is used to adapt the restraint system to the passenger. The airbag might be deactivated when the seat is occupied by a child below a predefined weight. It is used for multistage airbags as well to have an optimal passenger protection.

A possible side crash detection is carried out by a combination of two types of sensors. Acceleration satellites (G-Sats) measure the acceleration in the vehicle structure during a crash on the left and right side of the car. A modern G-Sat that has been developed by Siemens VDO is shown in Figure 1.8(b).

Whilst G-Sats are usually mounted in the right and left center pillar, the other type of sensor for side crashes can be found in the doors of the car. So-called pressure satellites (P-Sats) (see Figure 1.8(c)) measure the inner pressure of the doors continuously. Since a side impact will commonly reduce the inner volume of a door, the increasing pressure is a reliable check for it.

As front crashes occur much more often that side or rear crashes, it might be favorable to have another kind of sensor which is very sensitive to that specific type of crash. The so-called early crash sensor (ECS) is located in the vehicle's front end section. It provides information during the course of the accident. It can be used to detect a crash very early[1].

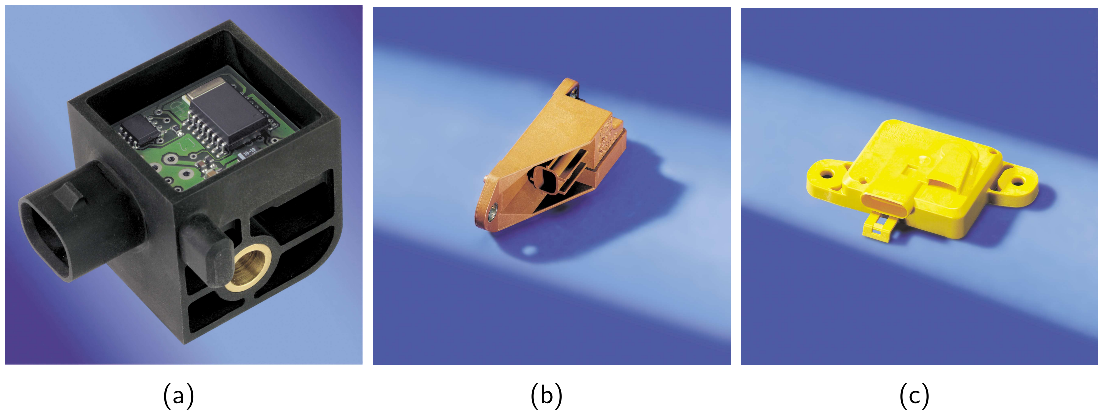


Figure 1.8: Common sensors of a modern car [4]. The left-hand figure (a) shows an early crash satellite (ECS). In the middle figure (b), a G-Sat is shown. The right-hand figure (c) shows a P-Sat.

When full scale crash tests are performed to gain physical data, the used sensors in the car differ very much in prices, value ranges and data resolution from the ones in manufactured vehicles. The reasons and consequences of this procedure are manifold. As every prototypical automobile platform is changed during the development process[2], *series sensors* with low value ranges and a inferior resolution would be a reason for severe problems in data[3]. On the other side, the costs of sensors in mass-produced vehicles are a tiny fraction of the expenses for so-called *reference sensors* in a prototype[4].

[1]Note that standard middle and lower-class cars, however, usually do not feature any ECS since they result in a higher price of the car.

[2]Prototypes might get a heavier engine, a stiffer front end section. Motor vehicle manufacturers even vary the sensor positions. These and further platform changes have a heavy impact on sensor signals.

[3]A sensor with low value range might suffer from saturation. Crucial signals may just be noticed by a high resolution. Thermic noise or offset addition can be other difficulties.

[4]One expensive sensor can cost up to 1000 Euros, whereas all sensors for restraint systems in the final

On the other side, restraint systems in a manufactured car have to deal with data coming from series sensors. As the calibration is carried out in the prototypical phase of the development, data generated by reference sensors will not have most of the problems described above. Thus crash test data have to be transformed in order to imitate sensors that can be found in ready-made automobiles. Common transformations to simulate data gained by series sensing are the following two examples. Usually the data resolution is lowered from 10 kHz down to 4 kHz. Moreover, the value range of acceleration is changed from ± 500 down to ± 75[5].

The transformed data is a necessity for the whole calibration process. It is demanded for all calibration steps that follow after crash tests with reference sensors. Hence it is the basis for the calibration of the ACU which will be clarified in the next section.

1.2.3 Calibration of the ACU

The calibration of the electronic airbag control unit is crucial to the performance of the restraint system. The calibration belongs to the engineering field of passive safety. Even though it is already well established, passive safety is a very dynamical field. It must cope with an increasing complexity of customer requirements due to new functionality and requests for shorter development times.

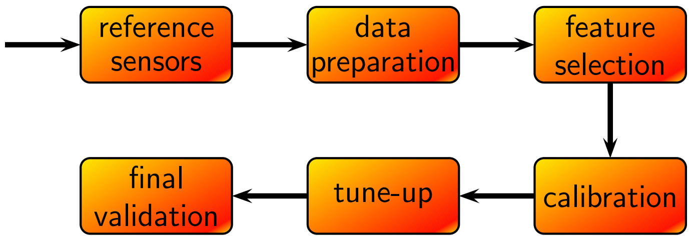


Figure 1.9: Flowchart of the calibration process.

One calibration process can take several weeks depending on the complexity of the given task. Figure 1.9 shows a flowchart of the complete calibration process. Raw data coming from the reference sensors has to be prepared to simulate cars with series sensors.

Since not all different environmental scenarios can be performed by crash tests due to their extreme expenses, every crash signal is scaled by different factors, e.g. $\{\pm 5, \pm 10, \pm 15, \pm 25\}$. This process has been applied in practice with big success. Moreover, the scaled versions also ensure a certain robustness of the ACU against unseen signals. Thus one single crash test usually leads to a set of different scaled crash tests.

Features are computed subsequently out of all scaled raw data signals. Then a good set of features has to be found by feature selection. The chosen features can differ from the chosen

automobile might sum up to a price of 100 Euros.

[5]Note that other transformations are possible as well.

automobile platform, its setup, the used sensors, and their positions in the vehicle.

The next step is the calibration of the deployment algorithm in the ACU. The algorithm for the fire decision has to be calibrated to satisfy every RTTF. No-fire crashes shall never fire since this might result in fatal injuries caused by the restraint system. So-called must-fire crashes have to be deployed by the ACU. A benefit is gained by correct discrimination of so-called may-fire crashes that need to be correctly classified in any case. Before the car with the calibrated restraint system is finally validated by a crash test, the deployment algorithm is tuned up in order to improve speed and to increase the number of correctly recognized crashes. For most platforms, however, not all requirements can be met completely.

The current technology used in the ACU is a fuzzy-rule based algorithm with a certain number of input variables. The corresponding set of fuzzy rules has to be constituted by human experts. Data analysis tools must be applied to find conflicting crashes that cannot be discriminated correctly. Finding critical crashes at the end of a project is equivalent to its deathblow.

As a consequence, a fast analysis of conflicts is necessary before the actual process of calibration starts. The next section will introduce the fundamental concept of one approach to conflict analysis. All ideas presented and implemented in the following chapters are based on that concept.

1.3 Toward Conflict Analysis

The fulfillment of customer goals can only be achieved by both streamlining the conventional development methods as well as by introducing innovative cutting-edge techniques. Among the new approaches being explored at Siemens VDO Restraint Systems is the application of support vector machines (SVM). They are applied to the analysis of crash sensor data with a minimum user supervision.

Their task is to produce a fast and reliable assessment for the potential of discrimination between so-called fire and no-fire events. In a fire event the vehicle restraint system must be triggered to protect the passengers. On the other hand, in a no-fire event the restraint system shall not be activated.

Customer requirements cannot always be fulfilled due to their complexity. An incorrectly classified crash constitutes a conflict which has to be analyzed. Early detected conflicts can be clarified with the customer much easier than conflicts that are discovered at the end of a project. It becomes clear that conflict analysis is an important step even before the actual calibration can begin.

Siemens VDO Restraint Systems together with Siemens Corporate Technology (CT) have developed a joint project for conflict analysis. The goal is to create a learning machine that is capable of discriminating between a fire and a no-fire event. As a result of this cooperation a data driven optimization tool (DDO) has been developed. It implements the methodology of support vector machines and is discussed more in detail in the next chapter.

1.3.1 Focus of this Thesis

The motivation of this thesis is to investigate new ideas that will lead to a further development of the SVM approach. The main challenge is to apply SVM to the discrimination of vehicle crash events. Within this frame, on the one hand, the thesis comprises investigations based on already existing improvement ideas. On the other hand, however, new promising techniques based on SVM are modeled for the given scenario as well.

The machine learning tools that are contributed in this thesis aim to lower the development time of calibration procedures by observing conflicts in an early phase of the project development when the calibration has not started yet. Moreover, they shall help and support the development of a restraint system. Although calibration tries to find a rule-based system for all kinds of crashes (see Section 1.2.1), this thesis only embraced a subset of them. Basically, we restrict the problem to discriminate crash events on the following types of frontal crashes:

- AZT (Danner) tests
- wall crashes
- angular crashes
- ODB crash tests

Before an overview of the thesis is presented, the next section will give a short history of Siemens VDO and its department Restraint Systems.

1.4 Siemens VDO Automotive AG

Siemens VDO is a company with a history that reaches far into the past. In this section we will briefly name the most important milestones that leaded to one of the best and well known automotive suppliers.

In 1847, *Siemens & Halske* founded the *Telegraphen-Bauanstalt* in Berlin, Germany. As a revolution at that time in 1905, the *Siemens Schuckert-Werke* started series production of "Victoria", the world's first electric automobile.

In 1902, Otto Schulze patents the eddy-current speedometer. The speedometer is marketed in Germany from 1908 by the *O.S. Autometerwerke E. Seignol* in Frankfurt/Main. Eighteen years later in 1920, Adolf Schindling founds the *OSA-Apparate GmbH* together with Georg Häußler and Heinrich Lang. In 1923, the first OSA speedometer is built. In 1925, the company is renamed *OTA Apparate*. In 1928, it merged together with the *Deuta-Werke* to form *VDO Tachometer AG – Vereinigte Deuta OTA*, abbreviated to VDO. The brand name VDO is created in 1929.

In the year 1951, the company is renamed *VDO Tachometer Werke Adolf Schindling GmbH*. In 1957, the 1st plant abroad was built in Australia (VDO Instruments Australia Ltd., Melbourne). Three years later, the 2nd plant abroad was installed in Brazil (VDO do Brasil, Sao Paulo).

In 1966, Siemens & Halske, Siemens-Schuckert-Werke and Siemens-Reiniger Werke merge to form *Siemens AG*. In 1972, the VDO Tachometer GmbH is renamed VDO Adolf Schindling GmbH. In 1973, the company becomes a public limited company: *VDO Adolf Schindling AG*. Its shares are traded on the stock exchange in 1986.

In 1989, Siemens establishes the independent Automotive Engineering business unit. In 1991, *Mannesmann* takes a majority shareholding in VDO Adolf Schindling AG. Finally, VDO is taken over completely by Mannesmann AG in 1994. Another three years later, the company is renamed *Mannesmann VDO AG*.

In 2000, Siemens AG spins off the Automotive Engineering business unit: *Siemens Automotive AG*. Just one year later, the merger between Siemens Automotive and Mannesmann VDO took place to form *Siemens VDO Automotive AG*.

In 2004, Siemens VDO Automotive AG acquires the former *Chrysler* electronic plant in Huntsville, USA. In 2006 Formal integration of Siemens VDO Automotive AG into Siemens AG as Siemens VDO Automotive Group: *Siemens AG - Siemens VDO Automotive*.

This year on July 25th, the German wheel producer *Continental* has taken over the Siemens VDO for approximately 11.4 billions. Although SV was supposed to be listed at the stock exchange market, Siemens AG decided to sell one of its 10 divisions. Now, SV and Conti form the 5th biggest automotive supplier in the world with nearly 140,000 employees and annual sales of around 25 billion Euros[6].

Nowadays, Siemens VDO is organized into the following 4 sections:

[6]The numbers are based on statistics of the year 2006.

- Powertrain
- Interior Electronics & Infotainment
- Chassis & Safety
- Commercial Vehicles

The section *Chassis & Safety* moreover is separated into the divisions *Safety & Chassis Electronics* and *Motor Drives*. The former division is divided into the undermentioned 5 departments.

- Restraint Systems
- Chassis Systems
- Washer Systems
- Advanced Driver Systems
- Electronic Wedge Brake

The next section will give a short overview of *Restraint Systems* as this thesis is based on a work in Regensburg.

1.4.1 Department of Restraint Systems

The department of *Restraint Systems* basically deals with the research and development of crash sensing electronics, sensor satellites, occupant sensing and pedestrian protection. Its headquarters are situated in Regensburg, Germany. Figure 1.10 shows the manufacturing and development centers of *Restraint Systems*.

The newest key innovative idea of *Restraint Systems* is the crash impact sound sensor (CISS). The signals produced by this sensor enable an earlier discrimination of a fire event. The earlier a reliable fire decision can be made, the lower the probability for fatalities.

With its technologies for passive safety, *Restraint Systems* holds the leading market position worldwide.

Figure 1.10: Restraint Systems worldwide: electronic manufacturing and development centers [4]

1.5 Overview of this Thesis

Chapter 2 gives a summary of problem definitions and scientific publications that deal with them. Moreover, it shall motivate and classify the basic approach which was chosen for this thesis. Especially the concept of how to learn from examples is introduced in Section 2.1. Following, basic approaches for time series analysis are given in Section 2.4.

In Chapter 3, techniques and procedures that are utilized in the context of this thesis are specified. In particular, the concept of support vector machines will be elaborated in Section 3.3.

Chapter 4 deals with the implementation of the thesis and is based on the fundamentals that were introduced in the chapter before. Section 4.1 delineates the overall ideas of the implementation in detail. Section 4.2 defines fundamental concepts for conflict analysis.

In Chapter 5, the proposed methods are evaluated experimentally on a real-word application. In the used crash database, the proposed methods lead to good accuracies and low model complexities compared to the existing discrimination method at Siemens VDO.

In the final Chapter 6 we will conclude the thesis and motivate possible improvements and extensions.

2 Related Work

The goal of this chapter is to give an overview of related problem settings and a selection of ideas that study them. Thus a potential categorization of this work shall be possible. Presentation and comparison of the different thoughts stand in the foreground of the general summary. For deeper insights into the ideas, we will refer to literature and Chapter 3, respectively. There, ideas on which this thesis is based on are discussed more in detail.

The problem to discriminate vehicle crash events automatically is essentially classified into the field of machine learning. In first instance, crash events are observed whilst real-world crash tests. Based on these data, more or less complex rules for triggering restraint systems are derived empirically by human experts. Hence it is straightforward to construct a machine (an algorithm) that solves this task by learning from given examples.

The whole concept of learning from examples is formulated in statistical learning theory which has been developed by Vapnik [36]. In this framework, one basically considers three statistical problems[1]. Besides the problems of regression and density estimations, pattern recognition is handled by this theory as well. This task is also know as classification problem.

Before we will introduce the problem of pattern recognition in detail in the next chapter, we will shortly discuss about the general model of learning from examples in Section 2.1. We will basically follow the formulations and notation as they can be found in [36].

However, the discrimination of vehicle crash events is a special pattern recognition problem. This is due to the fact that certain assumptions of the general setting of pattern recognition does not hold in this case. In Section 2.1.2, we will roughly explain which assumptions make this problem particular.

A tool, called DDO, that was developed by SV and CT already establishes a solution to the problem of crash discrimination. Implementation details how DDO discriminates fire crashes from no-fires is given in Section 2.3.

In contrast to the latter approach to pattern recognition where no temporal coherence is used, it is also possible to look at crashes as sequences in time, so-called time series. This approach and some selected solutions to it are outlined in Section 2.4. To finalize this chapter, we will summarize all presented ideas in Section 2.5.

[1]Note, however, that there are several more than just three statistical scenarios.

2.1 Learning From Examples

Vapnik [36] describes very elegant a general model how to learn from examples. This model consists of three elements. First of all, there exists a generator G which influences the environment where our learning machine shall operate. It generates data vectors $x \in \mathcal{X}$ independently and identically distributed (i.i.d.). This process depends on an unknown probability distribution function $F(x)$.

The second element of the model is the so-called supervisor's operator S (short supervisor) which receives every vector x as input and outputs some values y. Like the probability distribution function $F(x)$, the operator S is unknown as well. However, we can state that the operator S exists and does not change. Hence every output y on vector x is returned according to the conditional probability distribution $F(y|x)$.

The last element in Vapnik's model is the learning machine LM. It observes l pairs

$$(x_1, y_1), \ldots, (x_l, y_l) \equiv z_1, \ldots, z_l \tag{2.1}$$

which consists of input vectors x and the supervisors's output y. We will refer to these pairs as training set. This set itself is drawn randomly and independently by the unknown joint probability distribution $F(x, y) = F(x)F(y|x)$. In the learning step, the machine tries to produce a suitable operator that is a good approximation to the supervisor's operator S.

This operator (or function) that imitates the supervisor is chosen by the machine from an implemented set of possible functions whilst the learning process. Thus the learning machine will choose a function from a set of admissible functions $\{g(z) \mid z \in \mathcal{Z}\}$ where z corresponds to our pair (x, y) and $\mathcal{Z}$ is a subset of the vector space $\mathcal{R}^n$. Moreover, a functional $R = R(g(z))$ as quality criterion is defined by the expected loss over the whole probability distribution function,

$$R(g(z)) = \int L(z, g(z))dF(z), \tag{2.2}$$

where the function $L(z, g(z))$ is called loss function. By minimizing the functional $R(g(z))$ without knowing $F(z)$ but having $z_1, \ldots, z_l$, the learning problem would be solved.

Vapnik [36] describes that one cannot directly minimize Equation (2.2). Instead of the set of functions $\{g(z)\}$, one uses a parametric form $\{g(z, \alpha) \mid \alpha \in \Lambda\}$. With this, the functional (2.2) can be written as

$$R(\alpha) = \int Q(z,\alpha)dF(z), \quad \alpha \in \Lambda, \tag{2.3}$$

whereas

$$Q(z,\alpha) = L(z, g(z,\alpha)) \tag{2.4}$$

is again our loss function. By choosing an appropriate function $Q(z,\alpha_0)$ out of the set $\{Q(z,\alpha) \mid \alpha \in \Lambda\}$, Equation (2.3) shall be minimized. This cannot be done analytically since the probability distribution function is unknown and only the observations $z_1,\ldots,z_l$ are given. A way how to solve this problem is the principle of empirical risk minimization (ERM) which will be discussed in Chapter 3, Section 3.1.1.

As it seems to be clear, the problem of minimizing (2.3) given a random independent sample is rather general. It covers three fundamental statistical problems: pattern recognition, regression estimation and density estimation. In our environment of crash tests, the discrimination of crash tests into fire and no-fire events does belong to the problem of pattern recognition. Further explanations about this problem will be given in Chapter 3, Section 3.1.

With the objective of categorizing an unseen pattern into a certain class, the new pattern should be similar to a certain degree to the training set (2.1). Hence a similarity measure is needed to compare patterns with each other[2]. The type of similarity measure that is used for SVM is discussed shortly in Section 2.1.1.

The problem of pattern recognition based on given examples can be solved very efficiently by various algorithms. One very promising method that started to be accepted in several communities in the mid-nineties is named support vector machine (SVM). As nearly all ideas that can be found in this thesis are based on SVM, a detailed explanation about this specific learning machine can be found in Chapter 3, Sections 3.2 and 3.3, respectively.

A standard version of a support vector machine for classification is already implemented into a tool abbreviated DDO. It was developed by a joint project between Siemens VDO Restraint Systems and Corporate Technology. Since the task was to improve and adopt DDO in order to support the calibration process, a concise explanation about this tool will be given in Section 2.3.

2.1.1 Measuring Similarity of Patterns

One approach of measure the similarity of patterns is to consider a function

[2]Similarity for the class labels is straightforward since two labels can be either identical or different.

$$K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \qquad (x, x') \mapsto K(x, x'). \tag{2.5}$$

The similarity of the given patterns x and x' is represented by a real number. The function K should be symmetric ($K(x, x') = K(x', x) \quad \forall x, x' \in \mathcal{X}$). Such a function is called kernel [24, 10, 17].

A simple similarity measure is the dot product of two given vectors $x, x' \in \mathbb{R}^n$ that is defined as

$$\langle x, x' \rangle \equiv \sum_{i=1}^{n} x_i \cdot x'_i. \tag{2.6}$$

This similarity measure is not appropriate for most problems and hence other, more complex measures are needed. Even the length of a vector x, however, can be computed with the dot product by Pythagoras,

$$\|x\| \equiv \sqrt{\langle x, x \rangle}. \tag{2.7}$$

Since patterns can be anything e.g., elements of a multivariate vector space or words, it is necessary to map the patterns to vectors in some dot product space $\mathcal{H}$ by

$$\Phi : \mathcal{X} \to \mathcal{H}, \qquad x \mapsto \Phi(x). \tag{2.8}$$

Remark that $x \in \mathcal{X}$ can be already element of a dot product space. However, we can still map its space to another more general dot product space. The space $\mathcal{H}$ is called feature space. Now we can define the similarity measure for the dot product in the feature space $\mathcal{H}$ by

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle . \tag{2.9}$$

Since Φ can be defined arbitrarily, this mapping gives us a huge spectrum of different similarity measures. Furthermore the patterns can be handled geometrically which might be more intuitive than the original domain space.

Aside the so-called linear kernel (2.6), the following three kernels were successfully used in many real-world applications:

1. kernels that return polynomials of degree d:

$$K(x, x') = (\langle x, x' \rangle + 1)^d. \tag{2.10}$$

2. kernels generating radial basis functions:

$$K(x, x') = K(\|x - x'\|), \tag{2.11}$$

for example

$$K(\|x - x'\|) = \exp\left(-\gamma \|x - x'\|^2\right). \tag{2.12}$$

3. kernels that generate two-layer neural networks:

$$K(x, x') = S(v \langle x, x' \rangle + c) \tag{2.13}$$

where $S(\cdot)$ is an sigmoid function, e.g. $\tanh(\cdot)$ or $\frac{1}{1+\exp(\cdot)}$.

Note, however, that a symmetric function can only be used as a kernel if Mercer's condition [24] hold true. In short, the matrix $K_{ij} = K(x_i, x_j),\ 1 \geq i, j \geq l$ has to be positive definite. If this condition is not valid for the chosen kernel, the SVM cannot guarantee a good generalization to the test data.

Kernel functions play an elementary role for support vector machines. The way how similarity is defined has a major influence of the classification result of the machine. Basically, kernels are used to map nonlinear separable data into a higher dimensional space where the data points become linear separable and thus easy to classify.

Consider hereto the following example of two classes separated by an ellipse as decision boundary [32]. The scenario is illustrated in Figure 2.1 as well. The learning machine shall estimate this boundary based on empirical data. By the nonlinear mapping

$$\phi(x) = \phi(x_1, x_2) = (1,\ \sqrt{2}x_1,\ \sqrt{2}x_2,\ x_1^2,\ x_2^2,\ \sqrt{2}x_1x_2), \tag{2.14}$$

the ellipse in $\mathbb{R}^2$ becomes a hyperplane in $\mathbb{R}^6$ since an ellipse can be written as linear equation. Here the kernel can be defined by substituting (2.14) into Equation (2.9). This leads to the polynomial kernel of degree two

$$K(x, x') = \langle \phi(x), \phi(x') \rangle = (\langle x, x' \rangle + 1)^2. \tag{2.15}$$

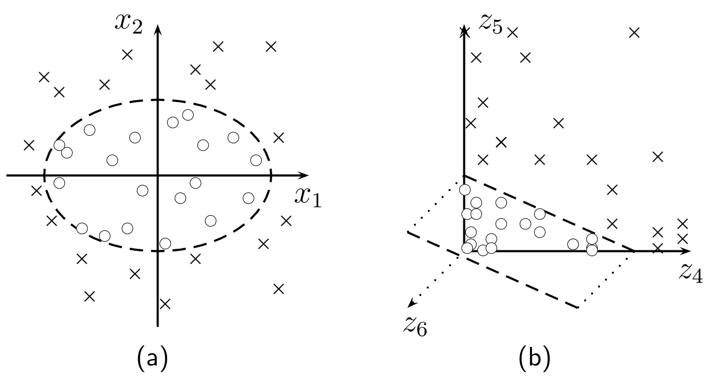


Figure 2.1: Linear separability induced by a kernel [32]. Whilst data in the left plot (a) cannot be separated by a linear classifier, a nonlinear mapping of the points into a 6-dimensional space $(z_1, z_2, z_3, z_4, z_5, z_6) = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)$ enables a linear separability. Note that in the right plot (b) only three dimensions z_4, z_5, z_6 are plotted.

By applying the kernel as similarity measure between two data points, the direct computation of the high-dimensional space can be omitted. Thus the support vector machine has to find a separation between both classes of mapped data. Further insights into SVM are given in Chapter 3, Section 3.3.

2.1.2 Specifics of Discrimination of Crash Events

The discrimination of crash events into fires and no-fires based on crash signals is a special classification problem. By definition, every pattern of the training set (2.1) is considered to be independent from each other. This is a fundamental assumption that guarantees that we can construct a learning machine based on the unknown joint probability distribution $F(x, y)$.

This assumption does not hold true in the case of crash events. The consecutive patterns of a crash do have a temporal coherence and are dependent from each other. A misclassification of one single no-fire pattern of a crash event will active the restraint system. This might cause severe injuries to driver and passenger, respectively. Therefore the minimal requirement of a restraint system is that all no-fires have to be classified correctly. In terms of machine learning, the costs of misclassification are very high.

On the other hand, there are the fire patterns. In contrast to the no-fires, it is not necessary to correctly classify all fires of a crash. A fire decision, however, has to be made since a misclassification of all fire patterns would lead to heavy injuries or fatalities. As a consequence, it is sufficient enough to detect at least one fire pattern correctly. Here, the costs of misclassification are therefore very low.

To sum up, the costs of misclassification are highly unbalanced. Moreover, there exists a temporal dependency for fire and no-fire patterns, respectively. No-fire patterns have to be recognized correctly by all means. Whereas some patterns of a fire crash do not need to be classified right provided that at least one pattern will be distinguished.

Another problem may be caused by fire crashes that are recognized at a time after the requested time to fire. These fire crashes are called *late fires* since the restraint system would be deployed later than the RTTF. Late fires can be the cause of severe injuries as well.

2.2 Definition of a Crash

Before an implementation of a SVM is elucidated in the next section, let us define some general concepts that are necessary for the thesis. First of all, it is essential to know how the data structure of a crash could look like.

Suppose there are given K crashes, both fire and no-fire. Every crash is labeled with a class variable $\omega \in \{0, 1\}$. Say 0 represents a no-fire and 1 a fire crash, respectively. Its crash data is equivalent with a time series $\mathcal{X} = (x_1, \ldots, x_l)$ which consists of l multidimensional vectors $x_i \in \mathbb{R}^n$, whereas $1 \leq i \leq l$. For every crash, there exists a sequence of timestamps $T = (t_1, \ldots, t_l)$. The point in time $t_i \in \mathbb{R}_+$ denotes the time after the beginning of the crash event in milliseconds when x_i is observed. Thus every t_i corresponds to one x_i.

In addition, customer requirements for every crash are given. These requirements pertain the requested time to fire (RTTF) for every stage $s \in \{1, \ldots, S\}$. For instance, a common five-stage restraint system ($S \equiv 5$) with all stages and their passive safety devices is shown in Table 2.1.

Table 2.1: Five-stage restraint system. Depending on the severity of the crash, different stages will be deployed. The first stage only deploys the belt tensioner, whereas the other stages deploy a two-stage airbag and differentiate between a belted and a non-belted passenger, respectively.

stage s	passive safety devices in use
1	belt tensioner
2	airbag stage 1 (passenger non-belted)
3	airbag stage 1 (passenger belted)
4	airbag stage 2 (passenger non-belted)
5	airbag stage 2 (passenger belted)

Let us denote the RTTF for a stage s by t_s. Hence a sequence of fire times $\mathcal{T} = (t_1, \ldots, t_S)$ belongs to every crash where $t_s \in \mathbb{R}_+ \cup \{0\}$, $1 \leq s \leq S$. A fire time $t_s = 0$ will ensure that the stage s shall not deploy. If $\mathcal{T} = (0,\ 0, \dots, 0)$, then this crash is denoted as no-fire crash[3]. Summarizing the definitions of above, we can denote a crash with all requirements as follows.

Definition 2.1 (Definition of a crash) *Let the index of a crash be a positive integer $k \in \{1, 2, \dots, K\}$ and let $l \in \mathbb{N}$ be the length of the crash, respectively. Let $T = (t_1, \dots, t_l)$ be the sequence of timestamps $t_i \in \mathbb{R}_+$ and $\mathcal{X} = (x_1, \dots, x_l)$ be the sequence of multivariate values $x_i \in \mathbb{R}^n$. Finally, let $\mathcal{T} = (t_1, \dots, t_S)$ be the sequence of requested times to fire $t_s \in \mathbb{R}_+ \cup \{0\}$ for every stages $1 \leq s \leq S$. It follows that a crash $\mathcal{C}$ is denoted by a quintuple $(k, T, \mathcal{X}, \mathcal{T}, \omega)$ where the class label ω is either 1 for a fire or 0 for a no-fire crash.*

2.3 Data Driven Optimization Tool

In 2004, Siemens VDO (SV) and Corporate Technology (CT) started cooperating to optimize sensor configurations in restraint systems. Different sensor setups heavily change the discrimination of fire crashes and no-fire crashes as we already explained in Chapter 1, Section 1.2. The tool *Data Driven Optimization of Sensor Configurations (DDO)* that was developed by this joint venture shall answer the following question. Do crash signals generated by a particular sensor setup provide sufficient information to discriminate fire crashes from no-fire crashes?

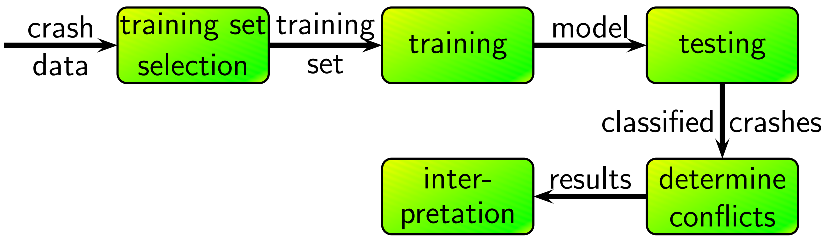


Figure 2.2: Flowchart of the DDO tool.

This discrimination usually cannot be carried out on raw signals coming from the sensors (cf. Chapter 1, Section 1.2.2) due to the complexity of the problem and the physical ability of interpretation that has to be ensured. Signals have to be represented as so-called features. For instances, a very simple feature might be the integration of a signal over the last n milliseconds. The final set of features for a certain platform might contain much more complex features that have to be chosen by calibration.

The DDO tool can be applied to several use cases. For example, given several crash measurements from different sensor configurations, DDO can be used to find the best configuration

[3]Recall that AZT crashes are no-fire crashes.

with respect to the best separability from fire and no-fire crashes. Figure 2.2 shows the general work flow of the DDO tool. In this chapter, only the first three steps of this flowchart will be discusses. The part of conflict determination is discussed in Chapter 4, Section 4.2.

The following sections will elucidate selected implementation details of the DDO tool as they are explained more precisely in [26, 18] as well. First of all, the selection of a training set is discussed in the next section. After that, the learning step with model selection is examined in Section 2.3.2. Finally, Section 2.3.3 will give insights into a heuristic how to classify an unseen pattern into a crash class.

2.3.1 Selecting a Training Set

In general, the way how to select a training set (cf. Equation (2.1)) is up to the DDO user. The selection can be influenced by either changing given parameters or directly implementing own procedures. The set of global parameters is determined in an INI file, whereas there exist parameters in the source code as well.

DDO reads the general configurations that are not crash-specific from an INI file. This INI file contains mandatory and optional parameters, respectively. A selection of the INI parameters is given in the following. Note that there are many more parameters. However, the undermentioned selection only includes parameters that influence the training set.

In order to construct a selection of fire patterns for each crash, a so-called fire range is constructed. This range includes all patterns that are bounded around the RTTF t_s for stage s, say $t \in \{t_s^-, \dots, t_s, \dots, t_s^+\}$. The computation of the fire range depends on the two mandatory parameters $t^<$ and $t^>$, respectively[4]. Both values specify the start and end of the fire range in percent of the RTTF t_s,

$$\forall s \in \{1, \dots, S\} : \; t_s^- = \max\left\{ t_s \left(1 - \frac{t^<}{100}\right), 0 \right\}, \quad t_s^+ = t_s \left(1 + \frac{t^>}{100}\right). \tag{2.16}$$

The values t_s^- and t_s^+ can be furthermore changed by two optional parameters t^- and t^+, resp[5]. By default, these constants are set to zero. On the one hand, the beginning of the fire range t_s^- is extended by the additive offset $t^- \leq 0$. The offset $t^+ \geq 0$, on the other hand, is added to the end of the fire range t_s^+. Thus,

$$\forall s \in \{1, \dots, S\} : \; t_s^- \longleftarrow t_s^- + t^-, \quad t_s^+ \longleftarrow t_s^+ + t^+. \tag{2.17}$$

[4]It is important to note that $t^<$ and $t^>$ are independent from the chosen stages. In the INI file, $t^<$ is denoted as `tPRE` and $t^>$ as `tCONST`, resp.

[5]The constants t^- and t^+ are independent from the chosen stages as well. The INI expressions are named `tConstPRE` and `tConstPOST`.

Until now, we have only considered the selection of fire patterns. Before a parameter for no-fire selection is introduced, let us recall that we deal with a very unbalanced classification problem as an explanation be found in Section 2.1.2. Due to this mismatch, a selection of no-fires can only be done very carefully.

A fire event usually lasts around 100 to 200 milliseconds since v_0 might be 60 km/h or even more. After 200 ms, every restraint system is far too late from passenger protection anyway due to the massive vehicle deformation. Therefore a rapid reaction after few dozens of milliseconds for a fire crash is a necessity in order to protect the driver and passengers. On the other hand, signals of a no-fire event have to be observed continuously since they might rapidly change to the characteristics of a fire event. Due to that reason, the length of a no-fire crash is much longer than a fire crash.

Unfortunately, signals of a no-fire are even recorded long time after the main events already happened. Hence all sensor features will record a constant signal. After a certain point in time, they do not carry any information. It is quite easy to find this point in time by looking at the signals of all no-fire crashes.

In order to incorporate this point in time, there exists one parameter in DDO that allows to cutoff all no-fires after a certain time. This parameter [6] can notable reduce the computation time. Nevertheless, it should be clear that a too small cutoff may remove important data. It is clear that one has to be careful with the application of this parameter.

In the procedure which creates the training set, there is another constants that has an influence on the selected fire patterns. It cannot be specified in the configuration file. It is necessary to understand how the fires are selected in order to see the constant's influence. This will be explained by the following elucidations.

A priori, only a maximum number of fire patterns $l_{\max}$ of every crash can join the training set. This constant[7] is heuristically set to $l_{\max} = 20$. In fact, the $l_{\max}$ "best ranked" patterns of every crash are added to the training set. The ranking is based on a linear SV classification of every single fire pattern against a subset of all no-fires.

Such a procedure where only a limited amount of patterns are added to the training set is referred to *active learning* [11]. The ranking criteria, so-called *stress*, is basically proportional to the no-fires that are misclassified due to the single fire pattern. The lower the stress of a fire, the higher it will be ranked. Thus the first $l_{\max}$ ranked fires are added to the training set.

Finally, the training set consists of all no-fires and the best $l_{\max}$ patterns of every fire crash. The next section will elucidate the way how model selection in DDO is carried out.

[6]The cutoff parameter in DDO is named `cutoffnofire`. By default, this parameter is deactivated.
[7]In the MATLAB implementation of DDO, this constant is called `MAX_NUM_SELECT`.

2.3.2 Performing Model Selection

Model selection is an important procedure in machine learning. Out of a collection of possible models that shall describe the data, the most suitable one should be selected. This demand depends on the problem that has to be solved. In our case of discrimination of crash events, we will prefer a model that will minimize the following two criteria:

1. the number of misclassified (critical) crashes n_{CC}
2. the number of critical errors n_{CE}.

The number of critical errors n_{CE} will sum up every misclassified no-fire and all l fire pattern of each fire crash if it is complete misclassified. This again, expresses the unbalance of our problem (cf. Section 2.1.2).

Since the DDO tool will apply a C-SVM (see Chapter 3, Section 3.2.1) based on the Gaussian kernel (2.12), there exist two SVM parameters that have to be specified. First of all, the $C > 0$ that enables complexer decision functions. Second, the γ value of the Gaussian kernel function which is indirect proportional to the width of the Gaussian. As a consequence, smaller C and γ values are preferred since they usually refer to an easy classify.

The exploration of these two SVM parameters is done by so-called *grid search*. The grid is defined on tuples $(C, \gamma) \in \{1, \ldots, 5\} \times \{0.1,\ 0.2, \ldots, 0.9\}$. For every tuple, a SVM is learned on the training set. The resulting model is evaluated on a subsample of the final test set in terms of n_{CC} and n_{CE}, respectively.

Whereas the real test set is used to compute the test error after the model is obtained, a subsample of this test set is used in the model selection step. By doing so, the estimate of the unknown error is too optimistic. Hence a poor model in model selection may perform very well in the test step. However, such a model is rejected by the model selection. This procedure is basically done due to runtime performance. For alternative model selection methods, it is recommended to read [20] as a starting point.

If the current SV model of the grid has recognized all crashes correctly, then the grid search will stop. Otherwise the model is compared to the so far best model with respect to our evaluation criteria.

2.3.3 Prediction of the Crash Class

After model selection has returned a model, it has to be evaluated on the test set. Here, the question is if a test pattern is classified as a fire or no-fire. Aside of that, it might be an advantage to know to which crash the pattern belongs. This comes especially into play when a pattern is misclassified. This is the key idea of conflict analysis.

Classifying a pattern into different crashes generally demands a multi-class approach instead of a binary method. A common implementation of a multi-class SVM is the *one-against-one* approach. There, $k(k-1)/2$ binary classifier are trained to separate each potential pair of k type of crashes. We usually deal with $k \approx 50$ crashes which make this approach unattractive since 1225 binary classifiers would be created.

The DDO tool therefore establishes another approach. Every standard SV model consists of a subset of the training set. These patterns, so-called *support vectors*, are labeled with a weight. These weighted points create the decision boundary[8]. Beside the weight, every pattern additionally stores the originally type of crash where it belongs to.

Whenever a new pattern is presented to the SVM, this model decides if it is a no-fire or fire pattern. To determine the pattern's type of crash, its distances to all support vectors that belong to its class are computed. The smallest distance will answer the crash type which is identical to the one of the closest support vector. This approach is motivated since the SVM is based on the Gaussian kernel (2.12) that defines the Euclidean distance between patterns.

It is important to note that the prediction of the crash class does only work with a RBF-kernel (2.11) due to the explanation above. This restricts the variety of possible approaches extremely. Moreover, the performance of this heuristic is questionable since the Gaussian kernel is not suitable for every problem. It is more or less a first approach to an unknown problem. Euclidean distance based similarity measures are naturally motivated if there does not exist any a priori information. A specialized kernel for our problem can include information that facilitates a discrimination between fires and no-fires. The restriction concerning the Gaussian kernel is the reason why a multi-class approach as it was just sketched becomes more attractive.

As it was already shown, assumptions for standard pattern recognition do not hold in our case. Therefore another approach is presented in the next section that is not based on individual patterns. The following ideas try to exploit the information of time.

2.4 Time Series Analysis

Generally, a classification problem starts with the assumption that all patterns are independently and identically distributed (i.i.d.) and thus there exists no coherence between any of them. However, as we already expressed, the patterns coming from crash tests are related

[8]Further explanations about support vectors will be given in the next chapter.

to each other by the information of time.

Therefore the assumption that every pattern is drawn i.i.d. does not hold. As a consequence, an approach that considers the temporal coherence of the patterns might lead to better results. Instead of regarding (2.1) as sample for training, we will take sequences in time (so-called time series) as patterns. A complete crash is such a sequence of points in time.

Since we have to have a reliable prediction as early as possible, it is necessary to split every crash into smaller time series that are labeled in the same way. This technique is shortly elucidated in the next section.

Another problem with time series analysis concerning support vector machines is the kernel function that measures the similarity of two patterns. As it was shown in Section 2.1.1, a kernel function takes only two multidimensional vectors as input rather than two matrices.

Either the function Φ that maps every time series into a multidimensional vector is chosen in an appropriate way or more complex kernels have to be used. One idea of the former approach will be discussed in the next chapter. Solutions with complexer kernels are presented in Section 2.4.2.

2.4.1 Sliding Window Technique

On the one hand, the incorporation of temporal knowledge will give further information to discriminate fire from no-fire events. Thus instead of classify only one pattern, we will look at w patterns successive in time. On the other hand, we have to ensure that the time when our machine makes a prediction does not exceed any RTTF. Hence it is clear that we have to find a good compromise between $w \equiv 1$ (possibly earliest prediction) and $w \approx$ crash length (large temporal coherence).

Suppose that the time Δt between two patterns of a crash event is constant (e.g. $1ms$) and that we have set an appropriate w which we will call *window length*. Due to this assumptions, the first w patterns represent the first small time series. The choice of the next w patterns as sequence depends on another parameter that has to be fixed.

This second parameter, say Δw which is usually called *window lag* defines the time between the beginning of one sequence and its successor. With fixed window length and lag, one can easily break down a whole crash into small time sequences. It is clear that a window lag of 1 will shift the starting point of the window to the next time stamp. In contrast, $\Delta w \geq w$ will lead to non-overlapping sequences in time. Usually one sets $1 \leq \Delta w \leq w$ to ensure that the sliding windows overlap each other.

This transition of a window to the next w patterns is also referred to the *sliding window* *technique*. For every crash $\mathcal{C} = (k, T, \mathcal{X}, \mathcal{T}, \omega)$ with $k \in \{1, 2, \ldots, K\}$, we construct matrices

$$\begin{aligned}
X_{k1} &= (x_{k,1}\ x_{k,2}\ \ldots\ x_{k,w}) \\
X_{k2} &= (x_{k,1+\Delta w}\ x_{k,2+\Delta w}\ \ldots\ x_{k,w+\Delta w}) \\
X_{k3} &= (x_{k,1+2\Delta w}\ x_{k,2+2\Delta w}\ \ldots\ x_{k,w+2\Delta w}) \\
&\vdots \\
X_{km} &= (x_{k,1+(m-1)\Delta w}\ x_{k,2+(m-1)\Delta w}\ \ldots\ x_{k,w+(m-1)\Delta w})
\end{aligned} \tag{2.19}$$

Every X_{ki} has the class label ω of the k-th crash $\mathcal{C}$. The information of time, e.g. when a fire decision can be made, is encoded into the last row w of each matrix X_{ki}. Since a fire decision is possible directly after recognizing a window, the TTF is thus equivalent to the last window's point of time. As a consequence, our new training set consists of triples $(t_{k,w+(i-1)\Delta w},\ X_{ki},\ \omega_k)$.

Since we want to apply support vector machines to discriminate fire crashes from no-fires, we cannot use standard kernels to compare matrices. The next section will present some techniques that are suitable for similarity measures between time series.

2.4.2 Kernels for Time Series Analysis

As already introduced in Section 2.1.1, every support vector machine is based on a kernel function in order to measure similarity between patterns. All kernels so far suffer from the same disadvantage. They are only defined for multivariate patterns $x \in \mathbb{R}^n$. When we deal with time series, however, our patterns are no longer multivariate vectors anymore. We rather have the problem of measuring similarity of sequences with different lengths[9].

Every pattern will be a sequence matrix, the rows of which are the original multivariate values. With the objective of computing the similarity of arbitrarily long sequences rather than vectors of fixed dimensionality, a kernel function for the SVM has to handle this problem also. The following subsections will present different kernels that are able to measure the similarity of sequences of different lengths.

[9]Note, however, that we do not consider different lengths of sequences since we fixed the window size w. Nevertheless all presented ideas can handle sequences of different lengths as well.

Outer Product of Trajectory Matrix

The possibly simplest approach to handle sequences of multivariate values is pursued by [7]. Here, every sequence (or trajectory matrix) $X = (x_1, \dots, x_w)$ with $x_i \in \mathbb{R}^n$ is multiplied by its transpose,

$$Z = XX^T. \tag{2.20}$$

Therefore this outer product matrix Z no longer depends on the length of the sequence. All diverse matrix elements $[Z]_{ij} = z_{ij},\ i = 1, 2, \dots, d,\ j = i, i+1, \dots, d$ will be transformed into a $(w(w+1)/2)$-dimensional vector $(z_{1,1},\ z_{1,2}, \dots, z_{w-1,w},\ z_{w,w})$. In other terms, the elements of the upper right triangular matrix of Z will build the input for the pattern vectors. These newly created vectors can easily be handled by a standard SVM with the kernels presented in Section 2.1.1.

It was tested with a SVM to classify segments of acoustic speech that durations are varying. The performance of the SVM was compared to an HMM that altogether could not be beaten by the outer product approach. Therefore this method shall only be listed here in order to show a possible way of modeling sequences.

Gaussian Dynamic Time Warping

A standard approach to measure the distance between time series is the technique of dynamic time warping (DTW) [25]. Whereas the computation of the Euclidean distance between two signals cannot handle a distortion in time, DTW allows nonlinear alignments between two time series to fit similar sequences. However, the sequences can even be locally out of phase when DTW is applied. It was introduced to the speech processing community and is now widely used in many fields of time series analysis. Before we introduce a kernel based on DTW, the formal problem of DTW is described.

Suppose we are given two time series, say $X = (x_1, \dots, x_l)$ of length l and $X' = (x'_1, \dots, x'_{l'})$ of length l', respectively. In order to align X and X' by DTW, an l-by-l' matrix D is constructed such that the element $[D]_{ij}$ corresponds to the Euclidean distance $d(x_i, x'_j) = \sum_{d=1}^{n} \left([x_i]_d - [x'_j]_d\right)^2$.

By applying DTW, one tries to find a so-called warping path

$$W = (w_1, \dots, w_L) \tag{2.21}$$

that leads across the matrix D whereas

$$w_k \in \{1,\dots,l\} \times \{1,\dots,l'\}, \qquad \max(l,l') \le L \le l + l' + 1. \tag{2.22}$$

Thus DTW defines a mapping between X and X' to align the elements of the time series. The optimal path is the one which minimizes the costs w_k. Usually, dynamic programming is used to find an approximation to the optimal path that is defined by

$$\mathrm{DTW}(X_1, X_2) = \min \sum_{k=1}^{L} [D]_{w_k}. \tag{2.23}$$

The application to SVM is straightforward [9] as only the Euclidean distance $\|\cdot\|$ of kernels generating radial basis functions (2.11) has to be substituted by the DTW distance $\mathrm{DTW}(X, X')$. The Gaussian kernel (2.12) is adopted to the DTW distance as follows

$$K_{\mathsf{DTW}}(X, X') = \exp(-\gamma\, \mathrm{DTW}^2(X, X')). \tag{2.24}$$

This kernel function is called Gaussian dynamic time warping (GDTW) kernel. The major drawbacks of this special similarity measure is the cost of the DTW computation and a theoretical reason that is severe. Mercer's condition [24] is not guaranteed for this type of kernel. The DTW distance leads to an inadmissible kernel. In practice however, it still showed good results, especially in the domain of handwriting recognition.

Dynamic Time Alignment Kernel

Another kernel that is based on DTW like the last one was introduced by [34]. However, this similarity measure is slightly different. Whereas the GDTW kernel is based on kernels generating radial basis functions (2.11), here the sum of an arbitrary kernel that is evaluated on every component of the warping path $W = (w_1, \dots, w_L)$ will determine the similarity of two sequences X and X', respectively.

Since the two sequences are dynamically aligned in time, the kernel is called dynamic time alignment kernel (DTAK) and can be written as

$$K_{\mathsf{DTAK}}(X, X') = \frac{1}{M} \sum_{k=1}^{L} K(x_{\varphi(k)}, x'_{\vartheta(k)}) \tag{2.25}$$

whereas $w_k = (\varphi(k), \vartheta(k))$ is the k-th element of the warping path and M is a normalizing factor.

The main advantage over the GDTW kernel that only permits RBF kernels is the fact that K can be any kernel. Nevertheless the DTAK does not represent kernel that satisfies Mercer's condition [24] either. The high expenses for the computation of the warping path are identical to the GDTW kernel. It was basically applied to phoneme recognition with a small amount of data.

2.4.3 Fisher Scores

The following idea is founded on an hybrid approach [19]. First, a parametrized probability model $P(X|\theta)$ that tries to generate given data is learned. Second, its parameters θ transform the patterns into a multidimensional feature vector, so-called *Fisher scores*, which is the input for a SVM that tries to discriminate theses feature vectors.

Hidden Markov models (HMM) [27] or Gaussian mixture models (GMM) are the probably best known examples of such generative models. The latter type of model is shortly introduced in the next subsection before the computation of the Fisher scores is clarified.

Gaussian Mixture Model

Mixture distributions provide a powerful, parametric framework for statistical modeling and analysis [23]. A mixture of distributions is a combination $\sum_{i=1}^{m} p_i f_i(x)$ of other distributions f_i. In our case, we consider parametric f_i with unknown parameter θ_i. Thus a parametric model of mixtures is denoted by

$$\sum_{i=1}^{m} p_i f(x|\theta_i). \tag{2.26}$$

We further assume that every f_i is a normal distribution (Gaussian) with θ_i that corresponds to its unknown mean and covariance. Now suppose that we are given l observations $X = (x_1, \ldots, x_l)$ generated by sampling a the normal distributions f_i. The probability density is a mixture of Gaussian distributions which consist of a set of prior probabilities $p = (p_1, \ldots, p_m)$ and a set of parameters $\theta = (\theta_1, \ldots, \theta_m)$.

Since the parameters are unknown, we have to chose a set of parameter values that makes the sample most likely. This is also known as maximum likelihood estimation. Formally, we have to chose θ that maximizes the likelihood function

$$L(p,\theta|X) = \prod_{i=1}^{n} \sum_{j=1}^{m} p_j f(x_j|\theta_j). \tag{2.27}$$

or the log-likelihood

$$\ln L(p,\theta|X) = \sum_{i=1}^{n} \ln \left(\sum_{j=1}^{m} p_j f(x_j|\theta_j) \right). \tag{2.28}$$

Maximizing the expected value of L or $\ln L$ can be performed by the expectation maximization (EM) algorithm [15]. Unfortunately, the EM algorithm only converges to a local maximum of the likelihood and it reacts very sensitive to noise.

Having locally optimal parameters θ to a certain degree of approximation, the Gaussian mixture model is trained and able to serve our purpose of computing the Fisher scores.

Computation of the Scores

In order to capture the generative process in a metric between examples, [19] suggest to use the gradient space of the generate model $P(X|\theta)$. The gradient of the log-likelihood with respect to the parameter θ establishes sufficient statistics for the sample X.

Formally, the class of probability models $P(X|\theta)$ defines a metric given by the Fisher information matrix

$$I = E_P(U_\theta(X)U_\theta(X)^T) \tag{2.29}$$

where the Fisher score is given by

$$U_\theta(X) = \nabla_\theta \log P(X|\theta). \tag{2.30}$$

Fisher Kernel

By mapping every pattern of X onto its Fisher score by the probability model $P(X|\theta)$ we can define the so-called natural kernel (Fisher kernel)

$$K(X, X') = U_\theta(X)^T I^{-1} U_\theta(X). \tag{2.31}$$

In contrast to the GDTW kernel and the DTAK, it is shown in [19] that the Fisher kernel is a valid kernel function. It was successfully applied to audio classification and the detection of protein homology in the field of bioinformatics.

Since probability models can easily handle sequences, the Fisher kernel based on the Fisher scores constitutes a natural way how to discriminate fire and no-fire time series by a support vector machine.

2.5 Summary

In this chapter, we introduced ideas how the problem to discriminate fire crashes from no-fires can be modeled and which specifics this problem brings up. First, we formally explained the theoretical concept of learning from examples in Section 2.1. There, we also brought out a way of measuring the similarity of these examples in terms of evaluating a kernel function. This will be the basis for the support vector machines as they will be introduced in the next chapter. Moreover, the particulars of the discrimination between fires and no-fires were shown.

A formal definition of a crash was given in Section 2.2. This definition determines the type of data structure that is necessary in order to solve both the problem of discrimination and conflict analysis.

In Section 2.3 the DDO tool was briefly explained. It is the basis of all improvements and implementations that will form the remainder of this thesis. Originally, DDO is based on a support vector classification that does not consider time series.

Section 2.4 addresses the idea of time series analysis. First of all, the sliding window technique for constructing short sequences in time is explained. After that a group of kernel functions that would be suitable for time series was concisely presented. Unfortunately, either a method was too simple or it is not guaranteed that the learning machine will generalize well on unseen data. Moreover, by applying DTW or the two kernels that are based on that, the differences between fires and no-fires get lost. Very often a fire crash only reaches a certain signal strength somewhat earlier than a no-fire crash. Thus DTW kernels most probably will not perform well here.

Last not least the Fisher kernel was introduced since it is a natural way to combine the generative power of a parametrized probability model with the discriminative power of a support vector machine. It can easily handle sequences of multivariate values and represents an admissible kernel as well.

3 Fundamentals

The present chapter is dedicated to techniques and procedures that are applied in the context of this thesis. The majority of the equations and the trains of thought are adopted from Vapnik [36].

Whereas the general concept of learning from examples was introduced in the last chapter, the problem of pattern recognition is specified from the general problem in Section 3.1. Two universal principles for solving this problem are presented.

Section 3.2 defines an important type of learning machines, the linear separating hyperplanes, that solve the problem of pattern recognition. Moreover, a special class of hyperplanes are specified in Section 3.2.1 that are more robust against outliers and noise, respectively. These soft margin hyperplanes that do not have to classify every pattern perfectly are very suitable for our problem since the sample of fire patterns does not have to be completely recognized.

The concept of support vector machines will be elaborated in Section 3.3 which concludes this chapter. A support vector machine (SVM) enables the mapping of all patterns from the input space into a higher dimensional feature space where the optimal soft margin hyperplane is obtained by solving a convex optimization problem.

3.1 Pattern Recognition

Let us introduce the general problem of pattern recognition as it might be found in [36] before we examine our scenario of crash tests. Formally, one can state that our environment where the crash tests are performed is characterized by a probability distribution function $F(x)$. Outcome x appears randomly and independently. Each outcome is classified into one of ω classes using the conditional probability distribution function $F(\omega|x)$, whereas $\omega \in \{0,\ 1,\ \ldots, k-1\}$.

Both information about the environment $F(x)$ and the decision rule $F(\omega|x)$ are unknown but fixed. Since both functions exist, we can conclude that the joint probability function $F(x, \omega) = F(\omega|x)F(x)$ does also exist. By definition, every outcome (x, ω) of the function $F(x, \omega)$ is independently and identically distributed (i.i.d.) as well as every pattern x itself. Therefore their appearance is not related to each other.

Consider a set of decision rules $\{\phi(x,\alpha) \mid \alpha \in \Lambda\}$ which only take k values $\{0, 1, \dots, k-1\}$. Moreover, we take the simplest loss function

$$L(\omega, \phi) = \begin{cases} 0 & \text{if } \omega = \phi \\ 1 & \text{if } \omega \neq \phi \end{cases} \tag{3.1}$$

which is also referred to indicator function. By minimizing the risk functional

$$R(\alpha) = \int L(\omega, \phi(x,\alpha))\, dF(\omega, x) \tag{3.2}$$

on the set of decision rules $\{\phi(x,\alpha) \mid \alpha \in \Lambda\}$, the problem of pattern recognition can be solved. However, the distribution function $F(\omega, x)$ is unknown and we are only given the i.i.d. sample

$$(\omega_1, x_1), \dots, (\omega_l, x_l). \tag{3.3}$$

The probability of a classification error for a given decision rule $\phi(x,\alpha)$ is computed by the risk functional (3.2) defined on the loss (3.1).

Let us recall the scenario of learning from examples as it was introduced in Chapter 2, Section 2.1. There, the vector z coincides with the $(n+1)$-dimensional vector (ω, x). Furthermore the set of functions $\{Q(z,\alpha) \mid \alpha \in \Lambda\}$ is changed accordingly to Equation (3.1) to $\{Q(z,\alpha) = L(\omega, \phi(x,\alpha)) \mid \alpha \in \Lambda\}$. With these assumptions, we can reduce our problem to the problem of minimizing the risk functional.

The risk functional (3.2) cannot be minimized directly as the probability function $F(x)$ is unknown. In order to get an estimate of the risk, one considers two induction principles which will be explained in the next two sections.

3.1.1 Principle of Empirical Risk Minimization

Instead of minimizing the risk functional (2.3), one minimizes the functional

$$R_{\text{emp}} = \frac{1}{l} \sum_{i=1}^{l} Q(z_i, \alpha), \quad \alpha \in \Lambda \tag{3.4}$$

which is named the *empirical risk functional* and evaluated on training data (2.1) that are drawn from the distribution function $F(z)$. The principle of minimizing the risk functional is referred to the *empirical risk minimization (ERM) principle*.

When we face the problem of pattern recognition as shown in the last section, then we want to minimize the empirical risk functional

$$R_{\text{emp}}(\alpha) = \frac{1}{l}\sum_{i=1}^{l} L(\omega_1, \phi(x_i, \alpha)), \quad \alpha \in \Lambda. \tag{3.5}$$

By minimizing (3.5) with $L(x, \phi)$ as defined in Equation (3.1), one would have the least errors on the training data (3.3). Thus the function $Q(z, \alpha) = L(\omega, \phi(x_i, \alpha))$ has to be chosen by the ERM principle. It can be chosen by another principle as well, e.g. the principle that will be introduced in the next section.

3.1.2 Principle of Structural Risk Minimization

The principle of structural risk minimization (SRM) [35] aims to have a trade-off between the complexity of the chosen function $Q(z, \alpha)$ and the result of empirical risk (3.4). Usually, the chosen functions by SRM differs from the one that would be selected by the ERM principle.

We would like to control the minimization of the risk functional (3.2). If the sample size $l \to \infty$, then we can use the ERM principle that minimizes the empirical risk (3.4). However, if the sample size is small, a small empirical risk on the training data (2.1) does not guarantee a small risk on unseen data.

In this case, we have to introduce another concept that enables us to control the minimization of the empirical risk. This concept is called Vapnik-Chervonenkis (VC) dimension that characterizes the capacity of a set of functions. It can be defined as follows.

Definition 3.1 (VC dimension(e.g., [36])) *The VC dimension of a set of indicator functions $\{Q(z, \alpha) \mid \alpha \in \Lambda\}$ is equal to the largest number h of vectors $z_1, \ldots, z_l$ that can be separated into two different classes in all the 2^h possible ways using this set of functions*[1].

With the concept of VC dimension, the following additive inequality holds true for all functions $0 \leq Q(z, \alpha) \leq 1,\ \alpha \in \Lambda$ with probability at least $1 - \eta$ and a finite VC dimension h

[1]The VC dimension is the maximum number of vectors that can be shattered by the set of functions.

$$R(\alpha) \leq R_{\text{emp}}(\alpha) + \frac{B\mathcal{E}(l)}{2}\left(1 + \sqrt{1 + \frac{4R_{\text{emp}}(\alpha)}{B\mathcal{E}(l)}}\right) \tag{3.6}$$

where

$$\mathcal{E}(l) = 4\frac{h\left(\ln\frac{2l}{h} + 1\right) - \ln\eta/4}{l} \tag{3.7}$$

and the constant $\eta \in (0, 1]$ converges to zero with increasing sample size l. When we deal with pattern recognition, the upper bound B of the indicator functions $Q(z, \alpha)$ equals to 1 which simplifies inequality (3.6). Furthermore, note that a sample of size $l \to \infty$ would result in $R(\alpha) \approx R_{\text{emp}}$.

Whereas the first term in (3.6) only depends on one chosen function of the set of functions, the second term basically depends on the VC dimension of the complete set of functions. Now, by minimizing both the left and the right term on the right-hand side of Equation (3.6), the VC dimension h becomes a controlling variable.

The principle of SRM can be introduced as follows. Let $\mathcal{S}$ be a structure on the set S of functions $\{Q(z, \alpha) \mid \alpha \in \Lambda\}$ with a structure $\mathcal{S}$. Moreover let

$$S_1 \subset S_2 \subset \ldots \subset S_n \subset \ldots \tag{3.8}$$

be the set of nested subsets of functions where $S_k = \{Q(z, \alpha) \mid \alpha \in \Lambda_k\}$ and $S* = \bigcup_k S_k$. Then the sequences of values of VC dimensions h_k and of the bounds B_k, respectively for $S_k \in \mathcal{S}$ is nondecreasing with increasing k:

$$\begin{array}{c} h_1 \leq h_2 \leq \ldots \leq h_n \leq \ldots \\ B_1 \leq B_2 \leq \ldots \leq B_n \leq \ldots \end{array} \tag{3.10}$$

Thus the principle of SRM chooses the $S_k \in \mathcal{S}$ for which the smallest bound on the risk functional is attained. Figure 3.1 shows an example of the dependencies between the VC dimension h and the risk functional that has to be minimized. To conclude the argumentations we emphasize that this principle guarantees the best solution of the learning problem for a given set of training data.

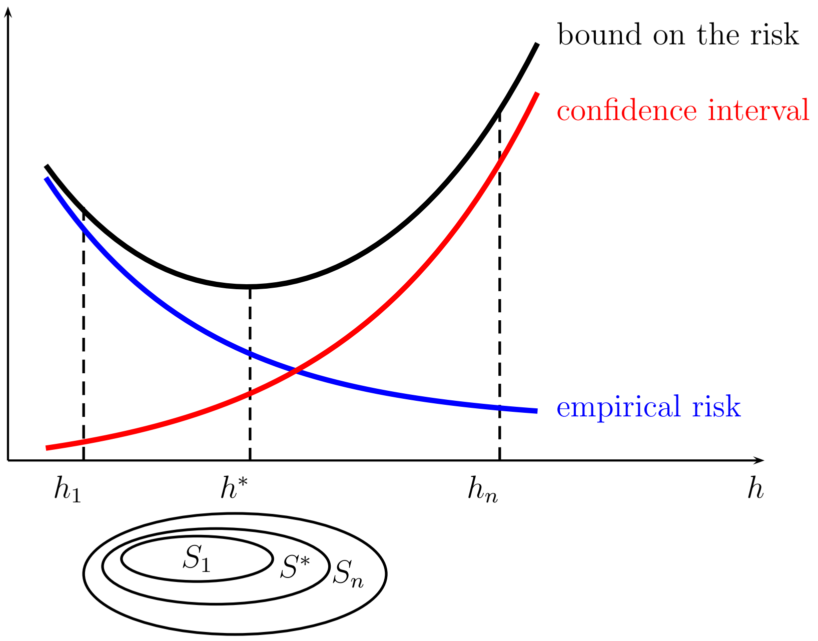


Figure 3.1: Principle of structural risk minimization [36]. The bound on the risk functional equals the sum of the empirical risk and of the confidence interval. There exists an indirect dependency between VC dimension and empirical risk, whereas the confidence interval directly depends on h. Note that the minimum of the risk is obtained by minimizing both empirical risk and the confidence interval.

3.2 Separating Hyperplanes

Separating hyperplanes as learning machines came up with Rosenblatt's perceptron algorithm [28]. They offer solutions to the statistical problems that we considered in the beginning of Chapter 2. Generalizations of this simple machine (artificial neural networks, e.g.) are standard machine learning methods today.

Support vector machines are based on separating hyperplanes as well. In this section, we will first introduce the concept of a linear separating hyperplane. After that soft margin hyperplanes for a nonlinear separable set of patterns is presented. So let us start with the linear case and the definition of a hyperplane.

Two finite subsets from the training data[2]

$$(y_1, x_1), \dots, (y_l, x_l), \qquad x \in \mathbb{R}^n, \quad y \in \{\pm 1\}, \tag{3.11}$$

first subset X_1 for which $y = 1$ and second one X_2 for which $y = -1$, are called separable [36] by the hyperplane $\langle x, \phi \rangle = c$ if there exist both a unit vector ϕ and a constant c such that

[2]We will use indicator functions of the form $\operatorname{sgn}(x) \in \{\pm 1\}$ rather than $\theta(x) \in \{0, 1\}$.

the inequalities

$$\begin{aligned} \langle x_i, \phi \rangle &> c, \qquad \text{if } x_i \in X_1, \\ \langle x_j, \phi \rangle &< c, \qquad \text{if } x_j \in X_2 \end{aligned}$$

hold true. Since the number of possible hyperplanes is infinite and we are looking for the "best" one, the concept of the optimal hyperplane [36] has to be introduced.

Denote for any unit vector ϕ the values

$$\begin{aligned} c_1(\phi) &= \min_{x_i \in X_1} \langle x_i, \phi \rangle \,, \\ c_2(\phi) &= \max_{x_j \in X_2} \langle x_j, \phi \rangle \,. \end{aligned}$$

Assume that the unit vector ϕ_0 maximizes the so-called margin

$$\rho(\phi) = \frac{c_1(\phi) - c_2(\phi)}{2}, \qquad \|\phi\| = 1 \tag{3.12}$$

when the inequalities (3.2) are fulfilled. Thus the vector ϕ_0 and the arithmetic mean of c_1 and c_2 ascertain the so-called *optimal hyperplane*. It separates both classes of vectors x correctly and maximizes the margin (3.12). In addition, it can be shown very easily that the optimal hyperplane is unique. Figure 3.2 shows an example of an optimal hyperplane given two linearly separable classes.

Intuitively, a separating hyperplane with a large margin will perform well on the test data. Since test and training data are randomly drawn from the same underlying distribution, the biggest fraction of the test set will be situated close to the training patterns. If we consider a form of input noise $\Delta x \in \mathbb{R}^n$ "around" a new pattern x that is bounded in its norm by some $r > 0$, then a classification of all training points with a margin $\rho > r$ implies a totally correct classification of every test pattern.

So we are looking for a pair that consists of ψ_0 and a constant b_0 such that the inequalities

$$\begin{aligned} \langle x_i, \psi_0 \rangle + b_0 &\geq +1 \qquad \text{if } y_i = +1, \\ \langle x_j, \psi_0 \rangle + b_0 &\leq -1 \qquad \text{if } y_j = -1 \end{aligned} \tag{3.14}$$

do hold true and the vector ψ_0 has to have the smallest vector norm

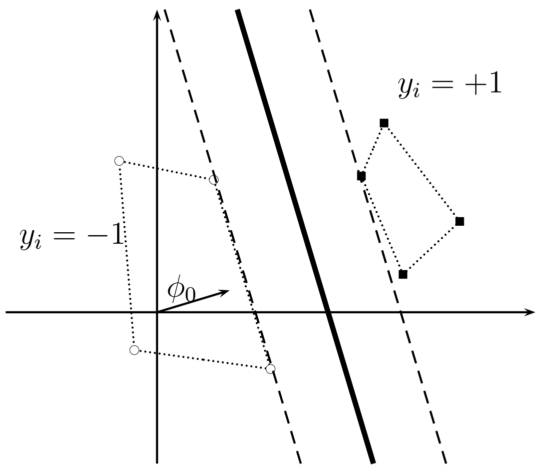


Figure 3.2: The optimal separating hyperplane is unique and separates the data points with the maximal margin [32].

$$\|\psi_0\|^2 = \langle \psi_0, \psi_0 \rangle . \tag{3.15}$$

It can be shown [36] that the optimal hyperplane characterized by ψ_0 coincides with

$$\phi_0 = \frac{\psi_0}{\|\psi_0\|}. \tag{3.16}$$

The margin between the optimal hyperplane and the separated training data equals

$$\rho(\phi_0) = \frac{1}{\|\psi_0\|}. \tag{3.17}$$

Therefore we finally arrived at the following quadratic optimization problem: minimize the quadratic function (3.15) subject to the linear constraints

$$y_i(\langle x_i, \psi_0 \rangle + b_0) \geq 1, \qquad i = 1, \ldots, l. \tag{3.18}$$

Instead of solving this problem in the *primal space* of ϕ and b, one usually considers the *dual space* where the problem is more convenient to solve as Lagrangian

$$L(\psi, b, \alpha) = \frac{1}{2} \langle \psi, \psi \rangle - \sum_{i=1}^{l} \alpha_i \left(y_i(\langle x_i, \psi \rangle + b) - 1 \right), \tag{3.19}$$

whereas $\alpha = (\alpha_1, \dots, \alpha_l)$ is the vector of Lagrange multipliers $\alpha_i \geq 0$. The Lagrangian L has to be maximized with respect to α and minimized with respect to ψ and b. Therefore, the derivatives of L with respect to the primal variables ψ and b must disappear,

$$\frac{\partial L(\psi, b, \alpha)}{\partial \psi} = \psi - \sum_{i=1}^{l} y_i \alpha_i x_i \overset{!}{=} 0,$$

$$\frac{\partial L(\psi, b, \alpha)}{\partial b} = \sum_{i=1}^{l} y_i \alpha_i \overset{!}{=} 0,$$

which results in the equalities

$$\psi = \sum_{i=1}^{l} y_i \alpha_i x_i, \tag{3.20}$$

$$\sum_{i=1}^{l} y_i \alpha_i = 0. \tag{3.21}$$

These two equations clarify that the solution does have an expansion in terms of the training set. Fortunately, only the non-zero Lagrangian multipliers correspond to constraint

$$\alpha_i \left(y_i \left(\langle x_i, \psi \rangle + b \right) - 1 \right) = 0. \tag{3.22}$$

Their patterns x_i are called Support Vectors. Equation (3.22) implies that they lie directly on the margin. It is for this reason that we call a resulting algorithm support vector machine (SVM). Note that the remaining fraction of training examples is not relevant for the solution since their multipliers meet $\alpha_i = 0$. Equation (3.22) is also referred to Karush-Kuhn-Tucker (KKT) conditions.

Finally, we can reformulate the primal into the so-called dual optimization problem by substituting (3.21) and (3.20) into the Lagrangian (3.19), we get the dual

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle . \tag{3.23}$$

subject to $a_i \geq 0,\ i = 1, \ldots, l$ and Equation (3.21). Denote the optimal solution with ψ_0, b_0 and α_i^0, respectively. Then we can expand the optimal hyperplane with the nonzero weights on support vectors by

$$\psi_0 = \sum_{i=1}^{l} y_i \alpha_i^0 x_i. \tag{3.24}$$

and the optimal hyperplane has the form

$$f(x, \alpha_0) = \sum_{i=1}^{l} y_i \alpha_i^0 \langle x_i, x \rangle + b_0. \tag{3.25}$$

Vapnik [36] notes that since neither the separating hyperplane (3.25) nor the objective function (3.23) depend on the dimensionality of the vector but on the scalar product of two vectors, enables to separate the sample in higher-dimensional (even infinite-dimensional) Hilbert spaces.

3.2.1 Soft Margin Hyperplanes

Given a real-world data set, it might not be possible to find a separating hyperplane. Even if a solution can be found, it might not be the optimal one for these specific data. One single outlier that is labeled incorrectly can have heavy impacts on the hyperplane. Hence a much more tolerant concept is required to handle noisy data up to a certain fraction.

Cortes and Vapnik [14] developed a new type of hyperplane that allows certain samples to violate (3.18). A hyperplane $\langle \psi, x \rangle - b = 0,\ \|\psi\| = 1$ is called Δ-*margin separating hyperplane* if it classifies vectors x by

$$y = \begin{cases} +1 & \text{if } \langle \psi, x \rangle - b \geq \Delta \\ -1 & \text{if } \langle \psi, x \rangle - b \leq \Delta \end{cases}$$

The following theorem gives an upper bound of the VC dimension of a Δ-margin separating hyperplane in $\mathbb{R}^n$.

Theorem 1 (e.g.,[36]) *Let vectors $x \in \mathcal{X}$ belong to a sphere of radius R. Then the set of Δ-margin separating hyperplanes has the VC dimension h bounded by the inequality*

$$h \leq \min\left(\left[\frac{R^2}{\Delta^2}\right], n\right) + 1.$$

This theorem is the starting point to develop the SRM principle in order to obtain a feasible result for a Δ-margin hyperplane that will perform well in terms of generalization. Therefore the so-called slack variables $\xi_i \geq 0$ are introduced by

$$F(\xi) = \sum_{i=1}^{l} \xi_i \tag{3.26}$$

which is defined in order not to obtain a trivial solution where all slack variables reached huge values. Thus we have to minimize the functional $F(\xi)$ subject to the constraints

$$y_i \left(\langle x_i, \psi \rangle - b \right) \geq 1 - \xi_i, \quad i = 1, \ldots, l. \tag{3.27}$$

and the constraint

$$\langle \psi, \psi \rangle \geq \frac{1}{\Delta^2}. \tag{3.28}$$

We now could make every ξ_i big enough to meet the constraint on (y_i, x_i). It follows that the solution is given by the reformulated Lagrangian

$$\begin{aligned} L(\psi, b, \alpha, \beta, \gamma) = &\sum_{i=1}^{l} \xi_i - \frac{1}{2}\gamma(A^2 - \langle \psi, \psi \rangle) \\ &- \sum_{i=1}^{l} \alpha_i \left(y_i(\langle \psi, x_i \rangle + b) - 1 + \xi_i \right) - \sum_{i=1}^{l} \beta_i \xi_i \end{aligned} \tag{3.30}$$

where $\psi,\ b,\ \xi_i$ have to be minimized and $\alpha_i, \beta_i, \gamma$ have to be maximized. Considering the usual extremal conditions with respect to $\psi,\ b$ and ξ_i, we obtain the equalities

$$\psi = \frac{1}{\gamma} \sum_{i=1}^{l} \alpha_i y_i x_i, \qquad \sum_{i=1}^{l} \alpha_i y_i = 0, \qquad \alpha_i + \beta_i = 1, \tag{3.31}$$

which can be substituted into (3.30). This leads to the functional

$$W(\alpha, \gamma) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2\gamma} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \frac{\gamma A^2}{2}, \tag{3.32}$$

that has to be maximized with respect to the constraints

$$\sum_{i=1}^{l} y_i \alpha_i = 0, \qquad 0 \leq \alpha_i \leq 1, \gamma \geq 0. \tag{3.33}$$

The γ in (3.32) can be substituted by

$$\gamma = \frac{\sqrt{\sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle}}{A} \tag{3.34}$$

which leads to optimizing

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - A \sqrt{\sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle} \tag{3.35}$$

subject to

$$\sum_{i=1}^{l} y_i \alpha_i = 0, \qquad 0 \leq \alpha_i \leq 1, \qquad i = 1, \ldots, l. \tag{3.36}$$

After optimization of $W(\alpha)$, the vector of parameters $\alpha = (\alpha_1^0, \ldots, \alpha_l^0)$ forms the generalized optimal hyperplane

$$f(x) = \frac{A}{\sqrt{\sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i^0 \alpha_j^0 y_i y_j \langle x_i, x_j \rangle}} \sum_{i=1}^{l} \alpha_i^0 y_i \langle x, x_i \rangle + b \tag{3.37}$$

whereas b can be obtained by the KKT condition

$$0 = \alpha_k^0 \left(\frac{A}{\sqrt{\sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i^0 \alpha_j^0 y_i y_j \langle x_i, x_j \rangle}} \sum_{i=1}^{l} \alpha_i^0 y_i \langle x_k, x_i \rangle + b \right), \qquad k = 1, \ldots, l. \tag{3.38}$$

By introducing a constant C that equals the optimal value γ_0 for maximizing (3.32), the whole concept of soft margin hyperplanes can be simplified. Thus the generalized optimal hyperplane is determined by ψ that minimizes the functional

$$\Phi(\psi, \xi) = \frac{1}{2} \langle \psi, \psi \rangle + C \left(\sum_{i=1}^{l} \xi_i \right) \tag{3.39}$$

subject to (3.26). By applying the same techniques as above, this leads to the regularizing term

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \tag{3.40}$$

with respect to similar constraints to (3.36)

$$\sum_{i=1}^{l} y_i \alpha_i = 0, \qquad 0 \le \alpha_i \le C, \qquad i = 1, \ldots, l. \tag{3.41}$$

3.3 Support Vector Machine

The idea of a support vector machine (SVM) [14, 36] is based on the principle of structural risk minimization (cf. Section 3.1.2). Basically, a support vector machine tries to learn a linear decision rule (3.2). By finding the hyperplane that maximizes the margin (3.12), however, a SVM chooses the function out of a set of functions for which the smallest error probability is guaranteed.

By mapping the input vectors x into a higher-dimensional *feature space* $\mathcal{H}$, a nonlinear decision rule can be learned [10]. Then the optimal separating hyperplane is found in the new space. The feature space, however, does not need to be considered explicitly. Moreover, only the inner products between the support vectors x_i of which $\alpha_i \neq 0$ and the input vectors has to be calculated in order to learn nonlinear decision rules.

Suppose that we map the vector $x \in \mathbb{R}^n$ into a Hilbert space $\mathcal{H}$ with coordinates $z_1(x), \ldots, z_k(x), \ldots$. Then the inner product in a Hilbert space has an equivalent form,

$$\langle z_1, z_2 \rangle = \sum_{r=1}^{\infty} a_r z_r(x_1) z_r(x_2) \Longleftrightarrow K(x_1, x_2), \qquad a_r \ge 0, \tag{3.42}$$

where $K(x_1, x_2)$ is a symmetric function that satisfies Mercer's condition [24]. Hence for any valid kernel $K(u, v)$ satisfying this condition, there exists a feature space $z_1(u), \ldots, z_k(u), \ldots$ where this function generates the inner product. This leads to the decision function in the input space

$$f(x,\alpha) = \operatorname{sgn}\left(\sum_{i=1,\alpha_i^0 \neq 0}^{l} y_i \alpha_i^0 K(x, x_i) + b\right) \tag{3.43}$$

and the decision function in the feature space $z_1(x), \ldots, z_k(x), \ldots, \ldots$

$$f(x,\alpha) = \operatorname{sgn}\left(\sum_{i=1,\alpha_i^0 \neq 0}^{l} y_i \alpha_i^0 \sum_{r=1}^{\infty} z_r(x_i) z_r(x) + b\right), \tag{3.44}$$

whereas (3.43) can be obtained by constructing a linear separating hyperplane of which inner product is substituted by the kernel function $K(x, x_i)$ into (3.35) and (3.40), respectively.

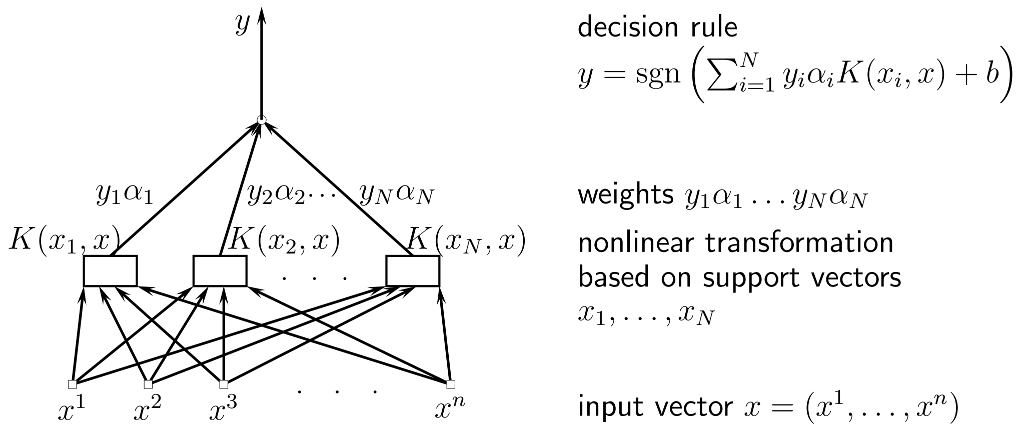

y
decision rule
$y = \operatorname{sgn}\left(\sum_{i=1}^{N} y_i \alpha_i K(x_i, x) + b\right)$
$y_1\alpha_1$
$y_2\alpha_2 \cdots$
$y_N\alpha_N$
weights $y_1\alpha_1 \ldots y_N\alpha_N$
$K(x_1, x)$
$K(x_2, x)$
$K(x_N, x)$
nonlinear transformation based on support vectors $x_1, \ldots, x_N$
x^1
x^2
x^3
x^n
input vector $x = (x^1, \ldots, x^n)$

Figure 3.3: Structure of a support vector machine [36]. The SVM consists of two layers. The first layer is constructed during the learning step. It contains all support vectors x_i, $i = 1, \ldots, N$ after learning. In this first layer, the basic similarities of the input vector compared to each SV are computed in the feature space chosen by the kernel $K(x, x_i)$. The second layer builds the linear function in this space.

Different support vector machines can be defined by different kernel functions (see Figure 3.3) as we already introduced some kernels in Chapter 2, Section 2.1.1. Special kernel functions for time series analysis were presented in Chapter 2, Section 2.4.2.

4 Implementation

In the present chapter the implementation of the thesis is revealed. All ideas are based on the fundamentals and on the related work introduced in the chapters before.

First, the overall ideas of the implementation are briefly outlined in Section 4.1. The arguments given in this section shall justify the following algorithm decisions and lay out the requirements that an algorithm has to fulfill in order to be accepted as solution to the problem of crash discrimination.

Then the determination of conflicts is presented in Section 4.2. The term conflict is precisely defined with respect to the formalism of separating hyperplanes as well. Having a formal definition of them, the search for conflicts can be automated.

Section 4.3 includes all contributions of this thesis toward conflict analysis and the discrimination of crash events. Two improvements based on SV pruning and crash weighting are considered in Sections 4.3.1 and 4.3.2, respectively. Finally, an approach toward time series analysis by SVM based on Fisher scores is presented in Section 4.3.3.

4.1 Overall Ideas

This section shall outline the basic approaches of this thesis and justify universal design decisions. Thereto general demands of the learning machine in terms of ability of interpretation e.g., linearity of the classifier, are explained in Section 4.1.1. Most of the classification problems, however, might force the learning machine to select a quite complex decision function that is not easy to interpret and justify. Hence the learning machine shall have the ability of local complexity in order to relax restrictive requests. Such a concept for local complexity is introduced in Section 4.1.2.

4.1.1 The Request of Ability of Interpretation

The responsibility of the deployment algorithm of an automobile is immense. Suppose that a fire event is not recognized by the algorithm before the belt tensioner and airbag have to be deployed, respectively. In this case the crash might cause fatal injuries to driver and passengers. Contrarily, it is clear that a wrongly classified no-fire event will cause injuries to the passengers due to the forces that belt tensioner and airbag apply to the passengers. In any case, the fire decision has to be interpretable. Thus the requirements for an algorithm that has to solve such a real-world problem are very restrictive. The probably three most important requirements are underlined in the next paragraphs.

At any point of time, *the deployment algorithm for a restraint system shall be transparent*. This fact enables a very high ability of interpretation, especially in the case where both crash data and information about the fire decision stored inside of the ACU has to be assessed after a crash. The correctness of every fire decision can then be approved by human experts.

For instance, a fuzzy-rule based fire decision can be expressed by natural language easily. Its sequence of commands for the algorithmic decision can simply be understood by humans. On the contrary, black box algorithms like artificial neural networks do not offer many insights into neither the learning process nor the fire decision. Unfortunately, the support vector method behaves similar to an ANN since it is obtained by solving a convex optimization problem. Insights how and why certain points become support vectors which would enable an interpretation would be nice to have but are not available. This is one reason why the usage of the SV method is not popular for security critical systems like an airbag deployment algorithm.

The second requirement can be formulated as follows. *The algorithm shall be rather simple than complex*. Complex algorithms might suffer from overfitting which is fatal for the deployment algorithm. This becomes clear when having a look at the training set of crashes. Crash tests are normed and take place in standardized environments. It is understood that not all different road surfaces, climate zones, loadings and number of passengers can be performed as crash tests since their procedure is extremely expensive. Complex algorithms that separately cover nearly every fire and no-fire event might cause non-interpretable fire decisions in an unknown situation. As a consequence, simple and flexible algorithms do not suffer from this kind of overfitting.

Simple algorithms will be also much more convincing to the customers due to the following main reasons. The management who finally decides about the success of a project usually does have a dissimilar knowledge about the formalisms that are applied to discriminate fire events from no-fire ones. They will appreciate concepts that are reasonable to interpret. Therefore algorithms that are easy to understand are preferred to complex ones.

Another very important point that demands simplicity are the expenses for the ACU that contains the deployment algorithm. Complex algorithms usually requires many more system resources and embedded systems inside of the ACU than simple ones. In order to have a fast working ACU that is still inexpensive, simple deployment procedures are a necessity due to economics.

With respect to the SVM principle as introduced in Chapter 3, Section 3.3, simplicity can be expressed by several characteristics of the SV models. Some important characteristics of support vector machines are discussed below.

The third requirement follows from the fact that all crashes are physical processes (cf. Chapter 1, Section 1.2.1) that have to be understood. Every single crash has to be interpretable by means of physical properties. Moreover, every fire or no-fire decision needs to be based on the physics of crashes. As a consequence, *the decision making algorithm shall be founded on the physical nature of crashes*, no matter if calibrated by human experts or found heuristically by a machine.

The ability of physical interpretation is implemented into the algorithm by the variables of the input space. Raw data coming from the automobile sensors (see Chapter 1, Section 1.2.2) is rather hard to interpret. Therefore raw sensor data are transformed to physical properties, so-called features. A selection of real-world features can be found in the following subsection.

Selection of Given Features

The usage of features instead of raw data for decision making algorithms is manifold and was already explained. This subsection names the features that were used to train the learning machines that are elucidated in the remainder of this chapter.

Table 4.1: Selection of stable features for calibration. The acceleration sensors in both directions x and y are located inside the ACU.

feature	description
`feature01`	direction of crash
`feature02`	twice integrated acceleration (length)
`feature03`	signal progress property (crash time counter after crash beginning)
`feature04`	symmetry of x-sensor signal to zero line
`feature05`	velocity of vehicle in x-direction
`feature06`	slope of velocity in x-direction

The selection of features for the learning step is crucial since they express physical properties that might be very important for the processes during a crash. Human experts usually select only a handful out of many different features for the calibration process on account of performance reasons and the ability of interpretation.

Table 4.1 gives an example of smooth features that are used quite often for the calibration. Most of them lead to good performances during the calibration process. All of these features are low-pass filtered and consequently do not suffer from oscillation so much like others. Since we are interested in rather smooth features for our learning machines, we only take subsets of the presented list of features for their evaluations.

Simplicity of Support Vector Machines

With respect to the SVM principle as introduced in Chapter 3, Section 3.3, simplicity of a SV machine can be expressed by the capacity of the function chosen by the principle of structural risk minimization. This principle of minimizing the expected risk (see Chapter 3, Section 3.1.2) controls the capacity such that the chosen hyperplane will guarantee the lowest error on unseen patterns. Thus it heavily influences the complexity of the SVM.

Problems that are hard to classify will lead to a rather high fraction of support vectors with big weights α_i close to the constant $C > 0$. The sum of all positive (or negative) weights α_i and the number of support vectors are reasonable measures for simplicity of a SVM. The complexer the discriminant function has to be, the more support vectors are needed.

Fortunately, the classification problem we deal with does not demand to correctly classify all of the fire samples of a crash[1]. On the contrary, a fire crash will be discriminated by only one correctly classified fire patterns (or two patterns to have more confidence). Thence a pruning of fire examples that are hard to classify before the actual training might be a striking idea to simplify the decision function that will be selected by the support vector machine. An approach to prune the "worst" fire examples is introduced in Section 4.3.1.

Another measure of simplicity is motivated as follows. A priori, the user has to define an appropriate kernel function for the given learning problem. The linear kernel (2.6) will force the SVM to find a linear separating hyperplane in the input space. Hence the decision boundary will be linear and easy to interpret. Other, more sophisticated kernels (cf. Chapter 2, Section 2.1.1) enable the SV machine to choose more complex decision boundaries in order to solve the problem linear in an unknown very high-dimensional feature space.

Note that only kernels generating radial basis functions (2.11) are based on the Euclidean distance metric in the input space. This is important to remark since the DDO tool uses a distance heuristic to determine the crash class of an unseen pattern (see Chapter 2, Section 2.3). So the following question arises: How can we determine the "linearity" of kernels e.g., Gaussian kernels $K(x, x') = \exp\left(-\gamma \|x - x'\|^2\right)$? In [32], it can be found that the linearity of this type of kernel depends on γ. For small values of γ, the resulting feature space corresponds geometrically to the input space. The SV method determines a nearly linear discriminant function. On the other hand, a large γ causes narrow kernels that lead to complex nonlinear functions in the input space. By choosing a smaller γ meanwhile the grid search of tuples (C, γ), the decision boundary will be less complex and more interesting for interpretable models.

[1]On the other hand, recall that all no-fire patterns have to be correctly recognized.

4.1.2 Simplicity versus Complexity

The simplicity of a SV model might be necessary in order to accept and approve such a model. By decreasing the number of possible functions that a SVM can choose, the potential solution probably becomes simpler. The overall risk (3.2), however, will increase in general. As a consequence, the actual pattern recognition problem of crash events with respect to its specifics of discrimination (cf. Chapter 2, Section 2.1.2) may not be solved properly enough anymore. In other words, the classification of partly non-separable crashes that overlap or superimpose is not that simple usually.

A priori, there does not exist any information for the SVM if a crash is hard to classify or critical. A solution to that problem is that the user defines certain weights for those crashes who impose classification problems, so-called conflicts[2]. These weights will determine the importance of those crashes and influence the decision making of the SVM. With the objective of minimizing both the capacity of the chosen function and the expected risk, there has to be a trade-off between *the global simplicity and the local complexity* of the decision function. Such a concept is explained thoroughly in Section 4.3.2.

Before the ideas for incorporating knowledge into the learning process are introduced, the next section will finally define the term conflict with regard to the principle of support vector machines. This will be useful since the search for conflicts is the basis for the evaluation results in the next chapter.

4.2 The Determination of Conflicts

So far we were talking about crashes that were hard to classify, so-called conflicts. Let us examine conflicts more precisely with respect to non-linear separating hyperplanes that are chosen by support vector machines. Thus in this section, the term conflict will be defined.

The definition of a conflict will enable us to search for them when classification has been applied to all crashes. Found conflicts have to be interpreted in order to solve or remove them. The analysis of conflicts plays an important role in the calibration process. Conflicts can occur in every stage and every scaling of a crash.

It is understood that a misclassified crash, either fire or no-fire, imposes a conflict. The only difference between misclassified fire and no-fire crash is the fact that all fire samples have to be classified wrongly (located on the wrong side of the separating hyperplane), whereas just one wrongly discriminated no-fire pattern will change the complete crash into a conflict. Such misclassified crashes are named *critical*.

[2]The term conflict will be defined in the next section.

Critical crashes are only one type of conflict. The following sort of conflict is only applicable to fire crashes. Therefore, recall how the TTF for every fire pattern is predicted as it was explained in Chapter 2, Section 2.3.3. Suppose that a complete fire crash is already classified. Since the classification implies the assigning of fire times to the complete sample of the crash, the earliest fire time of the classified crash determines the TTF t_i^* for this crash and stage, respectively whereas $i \in \{1, \dots, S\}$.

Out of this procedure, a conflict can arise since the TTF t_i^* of the crash can be longer than the required TTF (RTTF) t_i. Such a conflict is called *late fire* since the restraint system would fire too late or later than the RTTF. If there exists a late fire and its difference between predicted TTF and RTTF is small (couple of milliseconds), then the importance of such a conflict is rather low since the restraint protection of driver and passengers will still be deployed early enough.

As yet these two different kinds of conflicts, critical and late-fire crashes, are completely independent from the type of learning machine. Such conflicts can be determined by applying any type of machine learning algorithm that would predict a TTF for every crash. The following sort of conflict is especially related to learning machines that are based on separating hyperplanes and kernels defining an Euclidean distance metric.

The Euclidean distance

$$d(x, \alpha) = \sum_{i=1, \alpha_i^0 \neq 0}^{l} y_i \alpha_i^0 K(x, x_i) + b \tag{4.1}$$

to the hyperplane in the feature space defined by the kernel K, the weighted support vectors $\alpha_i y_i$ and the constant b can be interpreted as a kind of confidence in the discrimination of pattern x. Instances which are geometrically situated very close to the separating hyperplane might not be relevant enough for a proper decision due to their similarity to the other corresponding class. On the contrary, a distance $d \geq 1$ between an instance and the hyperplane coincides with a confident discrimination.

One might argue that a pattern x with a distance $d < 0.5$ to the decision boundary has little confidence. In our case, we say that a complete crash imposes a conflict if one of its patterns is closer than 0.5 to the hyperplane. A crash is almost surely correctly classified when the minimum distance over the crash sample is at least 0.5. Finally, we can state our definition of a conflict by taking into account three different types of conflicts.

Definition 4.1 (Definition of a conflict) *A conflicting crash or (short) conflict* $\mathcal{C} = (k, T, \mathcal{X}, \mathcal{T}, \omega)$ *meets at least one of the following conditions.*

- $\min_{x \in \mathcal{X}} d(x, \alpha) < 0.5$ *(little confidence)*

- *for a fire crash*
  - *All patterns are misclassified (critical fire).*
  - $\exists i : \; t_i^* > t_i \in \mathcal{T}, \; 1 \leq i \leq S$ *(late fire).*
- *for a no-fire crash*
  - *At least one pattern is misclassified (critical no-fire).*

By precisely defining the terminus of a conflict, we are able to search for conflicting crashes in order to perform a conflict analysis. This concept will be a necessity in the next chapter when we will evaluate the methods that can be found below.

4.3 Improvements and Extensions of DDO

In the present section, the contributions to the DDO tool are laid out and explained in detail. Based on the general ideas given in Section 4.1 of this chapter, the following algorithms establish an approach to both simplicity of the model and local complexity, respectively.

First of all, a promising concept for linearizing a subsequent hyperplane classifier by pruning of fire support vectors is presented in Section 4.3.1. This pruning algorithm named *SV pruning* is based on linear support vector machines such that the remainder of fire patterns is easier to linearly separate from the no-fire patterns.

The pruning guarantees less complex discriminant functions, however, the separating hyperplane might not generalize well on some unseen crashes. Thus the hyperplane shall locally become more complex in order to ensure a better generalization. An approach toward local complexity is proposed in Section 4.3.2 as trade-off to the SV pruning method.

Whilst both methods are based on binary classification of multivariate patterns, a different approach is considered in the rest of this section. As an alternative to standard pattern recognition which demands an i.i.d. sample, a promising algorithm toward time series analysis of crashes is presented in Section 4.3.3. Based on a parametrized probability model that tries to explain the data, multivariate feature vectors, so-called *Fisher scores*, are computed for all time series. These new vectors are then discriminated by an arbitrary SV machine.

4.3.1 Pruning of Support Vectors

As we already sketched in Section 4.1.1, a learning machine that shall discriminate crash events has to be interpretable in order to be approved due to security standards. Easier machines (models) are favored instead of models that are hard to understand. Support vector machines might be theoretically very well motivated, however, the feature space is never expressed explicitly. If we can construct a feature space that is geometrically similar to the input space without using the linear kernel (2.6), then it is possible to interpret and understand the resulting machine to a higher degree.

The idea is to favor linear classifiers by pruning a certain amount of patterns that are very hard to classify linearly. Since we deal with a highly unbalanced pattern recognition problem (see Chapter 2, Section 2.1.2), a pruning of no-fire patterns is strictly forbidden. On the other hand, we only need one fire pattern before the RTTF in order to correctly deploy the restraint system. Therefore, pruning is only possible for fire patterns.

The pruning shall remove candidates for critical fire instances from the dataset. It shall not remove complete fire crashes since all crashes have to be considered for the discrimination. The subsequent SVM learned during training without these critical fires should have an easier decision boundary than with pruned fire instances. Thus the C in the grid search of the model selection step should be rather small or around one. The margin should become wider as well since a better linear separation of fire and no-fire samples can be achieved for unseen patterns. Moreover, the pruning serves the analysis of conflicts due to the fact that an easier discriminant function may reveal conflicts more easily than a complex model.

The pruning is realized in two steps: while the selection of the training set and in the method of model selection. The next two subsections will present the algorithms to prune fire patterns before and during the training process.

Training Set Selection by Pruning of Support Vectors

The first pruning process is motivated by the fact that a linear discriminant function of fires and no-fires is geometrically the easiest one. Furthermore, misclassified points, no matter if fire or no-fire patterns, will automatically become support vectors of the linear machine. The farthest fire support vectors[3] on the no-fire side of the hyperplane are the ones that will have a big influence on the model selection step since it is very probable that they will become support vectors again even with a more sophisticated kernel.

This kind of support vectors are the reason for two major influences that may occur after training set selection during the actual training process. First of all, there might be an undesired shift of the hyperplane. Second, it is possible to obtain so-called *island solutions*

[3]The distance to the hyperplane is given by Equation (4.1).

by applying a non-linear kernel. The farthest fire support vectors "pull" and shift the decision boundary locally by their weights such that undesired complex structures of the hyperplane can occur. On the other hand, their distance to the next no-fire support vector might be big compared to the distances among the no-fire support vectors. Their weights may radiate very much so that islands of fire areas next to their positions can occur. Since such a complex decision boundary would form fire islands surrounded by no-fire areas, we refer to such a discriminant function as island solution. An island solution is very hard to interpret and justify since there does not exist one unique linear functional dependency between the chosen input variables.

It is natural to remove those fire patterns that might cause conflicts[4]. It shall be guaranteed, however, that every fire crash is represented by at least m fire patterns in the final training set. This is an important prerequisite for the subsequent SVM training since a minimal number of training instances for every crash ensures the generalization performance.

The pruning can be briefly explained by the following 4 procedures:

1. Train a linear SV machine with all fire and no-fire patterns.

2. Identify misclassified fire support vectors.

3. Create a training set without these fire samples.

4. Start training again and repeat this process until a stable model is obtained.

The third procedure has to assure that none of the crashes is totally pruned. This is done by pruning every crash separately. Only the farthest wrong fire support vectors of every crash are pruned such that the remainder of fire points of that crash is at least m. After all fire crashes are pruned for the first time, a new linear classifier is trained and the procedure begins again until the number of pruned support vectors equals zero.

Note that this iterative procedure of SV pruning is somehow related to boosting [30, 31] where a weak classifier is used to find hard to classify instances. Applying boosting, their weights are then increased whereas we drastically set these weights to zero by removing them from the training set. We do not apply boosting since a fire crash event will be recognized if one single fire instance is correctly classified.

The pruning method is explained thoroughly in Algorithm 1. Heuristically, $m = 10$ in line 1 which ensures that at least 10 patterns of every fire crash will join into the training set. The linear SVM is trained with a constant value of $C = 10$ (line 5). This parameter controls the amount of pruned fire examples. The smaller C, the more wrong fire support vectors will be pruned. A value of 10 performed well in practice.

[4]See Section 4.2 for a definition of conflicts.

Algorithm 1 Training set selection by pruning of support vectors.

```
Input: crash data $(x_1, y_1), \dots, (x_l, y_l) \in \mathcal{X} \times \{\pm 1\}$ of all crashes $\mathcal{C}_k,\ 1 \le k \le K$
Output: training set $\mathcal{X}^* \times \{\pm 1\} \subseteq \mathcal{X} \times \{\pm 1\}$
 1: $m \leftarrow 10$                                  // minimum number of fire samples per crash
 2: $\mathcal{X}^* \times \{\pm 1\} \leftarrow \mathcal{X} \times \{\pm 1\}$          // initial $\mathcal{X}^*$ contains all no-fires and fires
 3: $\mathcal{M} \leftarrow \emptyset$                                  // initialize the linear support vector machine
 4: repeat
 5:    $\mathcal{M} = \mathcal{M}(\alpha, \mathcal{S} \times \{\pm 1\} \subset \mathcal{X}^* \times \{\pm 1\}, b) \leftarrow$ train lin. $C$-SVM, $\mathcal{X}^* \times \{\pm 1\}, C = 10$
 6:    predict labels $\{\pm 1\} \equiv \{\text{fire}, \text{no-fire}\}$ and distances $\mathcal{D}$ to the hyperplane of $x \in \mathcal{S}$
 7:    for every training set $\mathcal{X}_k^* \subset \mathcal{X}^*,\ 1 \le k \le K$ do
 8:       sort every wrong fire SV $x \in \mathcal{S} \setminus \mathcal{X}_k^*$ in descending order according to $\mathcal{D}$
 9:       prune every wrong fire SV $x \in \mathcal{S} \setminus \mathcal{X}_k^*$ of $\mathcal{X}^*$ with biggest distance s.t. $|\mathcal{X}_k^*| \ge m$
10:    end for
11: until no crash pruned
12: return $\mathcal{X}^*$
```

In line 6, the predicted class and distance to the hyperplane (4.1) for every $x \in \mathcal{S}$ of the linear SV machine $\mathcal{M}$ are computed whereas $\mathcal{S}$ is the set of support vectors. Thus it is straightforward to find the farthest wrongly classified fire support vectors.

Since the complete DDO tool is realized in MATLAB, the pruning of fires is implemented in this script language as well. Instead of the standard method for the selection of the training set[5], the user can choose the pruning algorithm to construct a training set[6].

Model Selection by Grid Search and Support Vector Pruning

Whereas most of the fire patterns are pruned in the selection of the training set by applying Algorithm 1, the following algorithm only prunes few fire patterns in the step of model selection. Recall that model selection in DDO is based on grid search as it was outlined in Chapter 2, Section 2.3.2. Moreover, the SV machines that are trained during model selection are based on the Gaussian RBF kernel (2.12).

If pruning shall be applied to the support vectors of a constructed machine, it has to be done for every tuple (C, γ) in the iteration method. A nested loop iterates over selected γ and C values, respectively. It is straightforward to add the pruning method into this nested loop by nesting another loop into it that runs as long as there were pruned fire patterns in the last iteration. The standard training method where the model selection is applied has been extended according to the described SV pruning[7].

[5]The standard algorithm for training set selection is implemented in `selectTrainingSetScaled.m`.

[6]The training set selection based on pruning can be found in the file named `selectTrainingSetPruned.m`

[7]The SVM training algorithm based on model selection and pruning is implemented in the method `modelSelectionSVM()` in the MATLAB file `svmFeatureBackwardSelection.m`. An optional boolean

Since the Gaussian kernel can construct much more complex discriminant functions than the linear one, it is understood that barely any fire pattern is pruned compared to the SV pruning during training set selection. Nevertheless every reduction of the number of support vectors without inducing more critical crashes and errors leads to a simpler model which is preferred to a SV machine with many support vectors.

Algorithm 2 Model selection by grid search and support vector pruning

```
Input: training set $\mathcal{X}^* \times \{\pm 1\}$, test set $\mathcal{V} \times \{\pm 1\}$
Output: SV machine $\mathcal{M} = \mathcal{M}(\alpha,\ \mathcal{S} \times \{\pm 1\} \subset \mathcal{X}^* \times \{\pm 1\},\ b, \gamma)$
 1: $m \leftarrow 2$                                  // minimum number of fire samples per crash
 2: $\mathcal{X}_0^* \times \{\pm 1\} \leftarrow \mathcal{X}^* \times \{\pm 1\}$                   // backup of the original training set
 3: $\mathcal{M}^* \leftarrow \emptyset$; $n_{CC}^* \leftarrow \infty$; $n_{CE}^* \leftarrow \infty$  // init. SVM, number of crit. crashes and errors
 4: for all $\gamma_i \in \{0.1, 0.2, \ldots, 0.9\}$ do
 5:   for all $C_i \in \{1, \ldots, 5\}$ do
 6:     $\mathcal{X}^* \times \{\pm 1\} \leftarrow \mathcal{X}_0^* \times \{\pm 1\}$                // restore the original training set
 7:     repeat
 8:       $\mathcal{M} \leftarrow$ train Gaussian $C$-SVM with $\mathcal{T} \times \{\pm 1\}$, $C = C_i$, $\gamma = \gamma_i$
 9:       $(n_{CC}, n_{CE}) \leftarrow$ test $\mathcal{M}$ on $\mathcal{V} \times \{\pm 1\}$
10:       if $((n_{CC} < n_{CC}^*) \vee ((n_{CC} = n_{CC}^*) \wedge (n_{CE} < n_{CE}^*)))$ then
11:         $\mathcal{M}^* \leftarrow \mathcal{M}$; $n_{CC}^* \leftarrow n_{CC}$; $n_{CE}^* \leftarrow n_{CE}$   // a better model has been found
12:       end if
13:       if $(n_{CE}^* = 0)$ then
14:         return $\mathcal{M}^*$
15:       end if
16:       $\forall \mathcal{X}_k^* \in \mathcal{X}^*, 1 \le k \le K$ : prune farthest wrong fire SV of $\mathcal{X}_k^*$ s.t. $|\mathcal{X}_k^*| \ge m$
17:     until $(n_{CE}^* = 0) \vee$ no SV pruned
18:     if $(n_{CE}^* = 0)$ then
19:       return $\mathcal{M}^*$
20:     end if
21:   end for
22:   if $(n_{CE}^* = 0)$ then
23:     return $\mathcal{M}^*$
24:   end if
25: end for
26: return $\mathcal{M}^*$
```

The pseudocode of this pruning method can be found in Algorithm 2. Note that in line 1 the number of minimum patterns per fire crash m differs dramatically from the one that was used in Algorithm 1, line 1. In the training set selection, we wanted to ensure that at least 10 patterns of every fire crash come into the training sample. Since the SVM for the training step uses the Gaussian kernel, the machine does not find the optimal separating hyperplane in the input space like the linear kernel. Regarding the training, a restraint system is deployed when one single fire is recognized. The recognition of two instead of one pattern is due to

variable PRUNING is set to true if pruning shall be applied during model selection.

the higher confidence that results when a second fire is recognized. Thus m is set to two during training.

In line 2, a copy of the original training set is created. This is needed since for every new tuple (C, γ), the pruning has to start with the original training sample which probably has been reduced before (see line 6). The actual training of the SVM is done in line 8. Then the model $\mathcal{M}$ is validated in line 9 with regard to the number of critical crashes n_{CC} and critical errors n_{CE} (cf. Chapter 2, Section 2.3.2). If the current model $\mathcal{M}$ is better than the best on so far, named $\mathcal{M}^*$, then $\mathcal{M}^*$ is replaced by the current one in line 11. If this model $\mathcal{M}^*$ does not impose any critical errors $n_{CE} = 0$, grid search comes to an end and the model is returned.

If there is at least one critical error, the pruning starts another loop. Since the procedure for the pruning is the same as in Algorithm 1, lines 7–10, it is shortened in Algorithm 2, line 16. The pruning is performed until $n_{CE} = 0$ or no SV was pruned. Then the next tuple (C, γ) is considered and the pruning eventually starts again[8].

If there does not exist any tuple (C, γ) such that $n_{CE} = 0$, then the model $\mathcal{M}^*$ with the lowest n_{CC} and n_{CE} will be returned. In worst case, grid search will select the most complex SVM with $C = 5$ and a Gaussian kernel parameter $\gamma = 0.9$. This is highly undesired since high γ values coincide with a big geometrical dissimilarity between input and feature space. Moreover, the higher C, the farther the distance of a wrongly classified SV to the hyperplane. With regard to the ability of interpretation, we are looking for a tuple (C, γ) with both $C \approx 1$ and positive $\gamma \approx 0$.

The next section will introduce another contribution to the DDO tool. Instead of simplifying the SV machine, the following procedure will locally complicate the discriminant function obtained by the SVM. The best quality of the undermentioned idea is that it can be combined with the pruning algorithms introduced above.

4.3.2 Priority of Crashes by Weighting

The simplicity of both the SV machine and its discriminant function is very nice to have. On the contrary, the model shall also deploy all fire crashes and prevent deployment of all no-fire crashes. As a consequence, a trade-off between simplicity and complexity has to be found. Whereas the last section already presented an approach to global simplicity, this section will introduce a slight modification of the standard C-SV machine which enables us to locally complicate the discriminant function.

Before the modification is dealt with, let us motivate adoptions of SVM toward the calibration

[8]Note that the standard grid search procedure can be obtained from Algorithm 2 by removing lines 7, 13–17.

process. Since we want to emulate and support this process with the principle of the SV method, strategies of calibration have to be modeled into the SVM. One of this strategy is the concept of prioritized crashes.

Customer requirements like

- "Ensure not deploying for all no-fires!"
- "Ensure always deploying all fires!"
- "AZT Crash with 16 km/h should not deploy under any circumstances!"

result from the fact that some crashes are more important than others since they are more critical or very crucial. These demands can be modeled by assigning weights to prioritized crashes. The higher the weight of a crash, the more its misclassification has to be prevented by all means. If such a knowledge is available, it can be incorporated into our learning machine.

To have an understanding of what is changed compared to the standard SVM, we have to recall the problem of minimizing the functional

$$\Phi(\psi, \xi) = \frac{1}{2} \langle \psi, \psi \rangle + C \left(\sum_{i=1}^{l} \xi_i \right) \tag{4.2}$$

subject to

$$F(\xi) = \sum_{i=1}^{l} \xi_i. \tag{4.3}$$

In general, the global parameter C expresses the costs for a misclassification for every pattern. It is set to one by default. As a consequence, a priori all patterns are treated the same without any information of priority. By minimizing

$$\Phi(\psi, \xi) = \frac{1}{2} \langle \psi, \psi \rangle + C \left(\sum_{i=1}^{l} C_i \xi_i \right) \tag{4.4}$$

subject to (4.3), we can define weights C_i for the training instances $x_i,\ 1 \leq i \leq l$. Thus it is straightforward to assign weights to complete crashes.

Therefore changes of the software package that provides the methodology of SV machines have been performed. The DDO tool (cf. Chapter 2, Section 2.3) uses the SVM package LIBSVM [12]. Fortunately there exists already a LIBSVM tool that enables applying a different weight to every training instance [13]. Hence the changes of the code for DDO are based on that tool.

In combination with the pruning method that was introduced in the last section, the assignment of priorities by weighting of single training crashes can be a powerful tool to ensure both an interpretable model and the fulfillment of customer requirements. It might be a good procedure for calibration to first apply the pruning methods and then have a look which fire crashes do not fire and which no-fire crashes do fire, respectively. Finally, pruning can be coupled to the assignment of weights in order to solve the conflicts that arose before.

4.3.3 Time Series Analysis based on Fisher Scores

The general pattern recognition problem as it was formally introduced in Chapter 3, Section 3.1 requires that all patterns are i.i.d. and thence do not have any coherence despite the unknown probability distribution where they were drawn from. In the case of crash events, however, there exists a temporal coherence since a crash test is a series of points in time. The naive assumption of i.i.d. patterns does not hold in our case and might already imply conceptual mistakes.

Therefore another, more profound approach is regarded in this section. The goal is to represent every crash by a multivariate vector. The computation of this vector is based on a probabilistic generative model $P(X|\theta)$ that tries to explain the data (cf. Chapter 2, Section 2.4.3 and its subsections). The advantage of parametrized generative models e.g., Gaussian mixture models, is their ability to handle arbitrarily long sequences of multivariate patterns (e.g, time series). By computing Fisher scores based on Equation (2.30), every sequence is represented by a vector of fixed length and can be discriminated by a support vector machine afterward.

The biggest problem is the training of the parametrized model $P(X|\theta)$. Whereas a SVM is based on a convex optimization problem with a unique optimal solution, the gradient based approaching to a local minimum does not guarantee the globally best model that describes data X in a good way. Another problem is the number of Gaussian mixtures m of the parameter $\theta = (\theta_1, \ldots, \theta_m)$ that has to be chosen by the user in advance of the training process. Different values of m drastically change the performance of the model $P(X|\theta)$ and also the subsequent discrimination by SVM.

First of all, the model $P(X|\theta)$ is only trained on either fire or no-fire patterns. This raises the probability of the class that has been considered for training and vice versa. Assume that the parametrized model has been trained with a subset of fire patterns. Then the model is trained to determined if a pattern is a fire or not. Presented no-fire patterns are unknow to the model and thus probably do not belong to the fire class. This already establishes a profound basis for the subsequently applied discriminative model. However, it has to be stressed again that the generative performance of the probabilistic model is highly unstable.

The training routine for the GMM is based on a MATLAB package developed by the Naval Undersea Warfare Center [8]. It provides a powerful and fast training method for GMMs and HMMs, respectively. Moreover, this tool enables a visualization of the learned multivariate distributions by plotting of two-dimensional subspaces.

After training $P(X|\theta)$, one has to determine the parameter θ which can be any model parameter e.g., mean μ, standard deviation σ, a priori probability p or covariance matrix Σ. Say, we choose $\theta \equiv \mu = (\mu_1, \ldots, \mu_m)$. Furthermore, the training set including fires and no-fires for computing the Fisher scores has to be transformed into sliding windows (see Chapter 2, Section 2.4.1) to ensure an early fire decision before the RTTF.

For every sliding window[9] $X = (x_1, x_2, \ldots, x_w)$, we have to compute its Fisher score. Based on Equation (2.30), the Fisher score for every Gaussian mixture $f_i,\ i \in \{1, \ldots, m\}$ can be written as

$$\nabla_{\mu_i} \log P(X|\mu_i) = \sum_{t=1}^{w} p_i(x_t)\Sigma_i^{-1}(x_t - \mu_i) \tag{4.5}$$

whereas $p_i(x_t)$ corresponds to the a posteriori probability of the distribution f_i given the observation x_t. The complete Fisher score

$$U_\theta(X) = U_\mu(X) = (\nabla_{\mu_1} \log P(X|\mu_1), \ldots, \nabla_{\mu_m} \log P(X|\mu_m)) \tag{4.6}$$

of the sequence X can then be utilized as input for the SVM. In fact, every discriminative classifier can handle the problem after transformation of the sequences to Fisher scores.

[9]Note that we omitted all indices of a sliding window in order to facilitate the notation.

5 Evaluation

In the present chapter, the suggested methods of the previous chapter are evaluated by experimentation based on a real-word crash database. The utilized crash dataset and its characteristics are discussed in Section 5.1. Moreover, insights into the classification results obtained by applying the DDO tool are given in Section 5.1.2.

Many experiments applying different ideas have been carried out on this crash database. The experimental results of the proposed methods can be found in the remainder of this chapter. First of all, the evaluation of DDO in combination with weighted crashes is presented in Section 5.2. Then the results regarding support vector pruning for training set and model selection are demonstrated in Section 5.3.

Section 5.4 elegantly combines both methods of SV pruning and crash weighting and gives evaluations based on the real-world database. Whereas the learning machines so far considered all crash modes together, the combination of both techniques is applied to models regarding pairs of crash modes as well. Since we deal with discrimination of no-fires and fires, the pairs coincide with model for AZT vs. wall crashes (Section 5.4.1), AZT vs. angular crashes (Section 5.4.2) and AZT vs. ODB crashes (Section 5.4.3).

Section 5.5 concludes with the evaluation of a hybrid approach of a probabilistic generative model in combination with a SV machine. This idea shall be a first step toward time series analysis of crash events.

5.1 A Real-World Application

In the present section, the real-world dataset that has been used for the evaluation of the proposed methods is presented and examined first. Moreover, the DDO tool (cf. Chapter 2, Section 2.3) is then used to train a SVM with a Gaussian kernel (2.12) generating radial basis functions. The tests include all 5 stages and 11 crash scalings of the dataset. Results with respect to conflicts, generalization performance and model complexity are listed in tabular form.

5.1.1 Characteristics of the Crash Database

Let us start with the characteristics and particulars of the given dataset. The specific vehicle platform which was used for crash tests will be available in the near future. The crash database contains nearly 80 different crash tests including AZT, wall, angular, ODB and misuse crashes. Since we only consider the first 4 types of crashes as outlined in Chapter 1, Section 1.3.1, misuses are omitted from all tests. There exist 5 stages of severity as they have been listed in Chapter 2, Table 2.1. Thence the used automobile platform has a five-stage restraint system with belt tensioner and two-stage airbag in various combinations. Hence there have to be 5 RTTFs, one for every stage.

Every raw signal was scaled according to the percentages $\{0, \pm 5, \pm 10, \pm 15, \pm 20, \pm 25\}$. Thus every crash signal exists in 11 different versions. In addition, the features that are computed from raw data differ from every scaling as well. All in all, more than 50 different features are available in the crash database. However, for this thesis we will only consider the most stable for calibration. These features can be found in Chapter 4, Table 4.1.

The remaining crashes without misuse crashes amount 63 in total which originate from 5 AZT, 22 wall, 13 angular and 23 ODB crashes. A description of every crash is given in Table 5.1 including key, relative velocity[1] and the RTTFs for all stages.

Table 5.1: Description of the real-world crash database used for evaluation. A stage of which RTTF is zero corresponds to the non-deployment of this stage. All AZT and two wall crashes are no-fires whereas the remaining crashes are fires.

crash key	speed [km/h]	RTTF for stage [ms]				
		1	2	3	4	5
wall01	56.6	13	13	14	18	19
wall02	56.6	13	13	14	18	19
wall03	56.0	13	13	14	18	19
wall04	56.0	13	13	14	18	19
wall05	56.0	13	13	14	18	19
wall06	50.0	14	14	15	19	20
wall07	50.0	14	15	15	19	20
wall08	50.0	14	15	15	19	20
wall09	50.0	14	15	15	19	20
wall10	50.0	14	15	15	19	20
wall11	40.4	15	15	16	20	21
wall12	40.0	15	15	16	20	21
wall13	40.5	22	22	23	27	28
wall14	32.0	27	27	28	32	33
wall15	32.0	27	27	28	32	33
wall16	32.0	27	27	28	32	33

[1]In the remainder of this thesis, we will refer to the relative velocity v_0 as speed due to simplicity.

Table 5.1: (continued)

crash key	speed [km/h]	RTTF for stage [ms]				
		1	2	3	4	5
wall17	27.0	32	32	33	0	0
wall18	27.0	32	32	33	0	0
wall19	27.0	32	32	33	0	0
wall20	27.0	32	32	33	0	0
wall21	16.0	0	0	0	0	0
wall22	15.8	0	0	0	0	0
angular01	50.0	18	18	19	23	24
angular02	40.0	20	20	21	25	26
angular03	40.0	20	20	21	25	26
angular04	40.0	20	20	21	25	26
angular05	40.0	20	20	21	25	26
angular06	40.0	20	20	21	25	26
angular07	40.0	20	20	21	25	26
angular08	40.0	20	20	21	25	26
angular09	40.0	20	20	21	25	26
angular10	32.0	22	22	23	27	28
angular11	32.0	22	22	23	27	28
angular12	32.0	22	22	23	27	28
angular13	32.0	22	22	23	27	28
ODB01	64.0	24	24	25	29	30
ODB02	64.0	24	24	25	29	30
ODB03	64.0	24	24	25	29	30
ODB04	64.0	24	24	25	29	30
ODB05	64.0	24	24	25	29	30
ODB06	64.0	24	24	25	29	30
ODB07	64.0	24	24	25	29	30
ODB08	64.0	24	24	25	29	30
ODB09	64.0	24	24	25	29	30
ODB10	64.0	24	24	25	29	30
ODB11	64.0	24	24	25	29	30
ODB12	64.0	24	24	25	29	30
ODB13	64.0	24	24	25	29	30
ODB14	60.0	30	30	32	35	37
ODB15	60.0	30	30	32	35	37
ODB16	60.0	30	30	32	35	37
ODB17	60.0	30	30	32	35	37
ODB18	48.0	40	40	42	45	47
ODB19	40.0	44	44	45	49	50
ODB20	40.0	44	44	45	49	50
ODB21	40.0	44	44	45	49	50

Table 5.1: (continued)

crash key	speed [km/h]	RTTF for stage [ms]				
		1	2	3	4	5
ODB22	40.0	44	44	45	49	50
ODB23	40.0	44	44	45	49	50
AZT01	15.0	0	0	0	0	0
AZT02	15.0	0	0	0	0	0
AZT03	15.0	0	0	0	0	0
AZT04	15.0	0	0	0	0	0
AZT05	15.0	0	0	0	0	0

Examining the table more in detail, there are some particulars that have to be analyzed. Every stage of a no-fire crash corresponds to a RTTF of zero. This expresses that this stage of the restraint system shall not be deployed. According to Table 5.1, there are 7 no-fire crashes (two wall and 5 AZT crashes) where all stages have a RTTF of zero. Moreover, there exist both high-speed crashes up to 64 km/h and low-speed crashes around 15 km/h.

It is fundamental to note that unfortunately some crash tests were performed with different car platforms. For example, some of these automobiles have had a less stiffer front than cars of other crash tests. Some vehicles have had other engines which is also one cause of a different behavior during a crash. Due to these discrepancies in the crash database, one cannot expect to fulfill all customer requirements. It is hard, for instance to keep the same low RTTF for a car with a smaller stiffness since an inelastic front enables an earlier crash detection.

To have an idea about the influence of different platforms in one database and how features look like, the Figures 5.1 to 5.4 show crash signals of two crashes each with the same speed. The presented features[2] (see Chapter 4, Table 4.1 for elucidations) are normalized to the unit interval $[0, 1]$ and plotted over time with an offset such that every curve can be shown in one plot. In addition, the RTTF is plotted as well as vertical dashed line. Only the RTTF for the first stage $t_{s=1} = t_1$ is plotted. This time t_1 is always the earliest among all fire times and thus usually the hardest task for our learning machine.

First of all, two 27 km/h wall crashes are shown in Figure 5.1. The plot on the left side represents the same type of crash as on the right side. However, the automobile used for the crash on the right side was equipped differently e.g., with a less stiffer front. As a consequence to this fact, the behaviors of both cars in crash scenarios differ a lot from each other. For instance, the curve of the feature `X_Symmetry` has a huge peek around the RTTF on the left-hand side whereas such a peek occurs approximately 70 ms later on the right-hand

[2]The plotted features are based on unscaled raw data. The other 10 scalings $\{\pm 5, \pm 10, \pm 15, \pm 20, \pm 25\}$ are omitted due to simplicity.

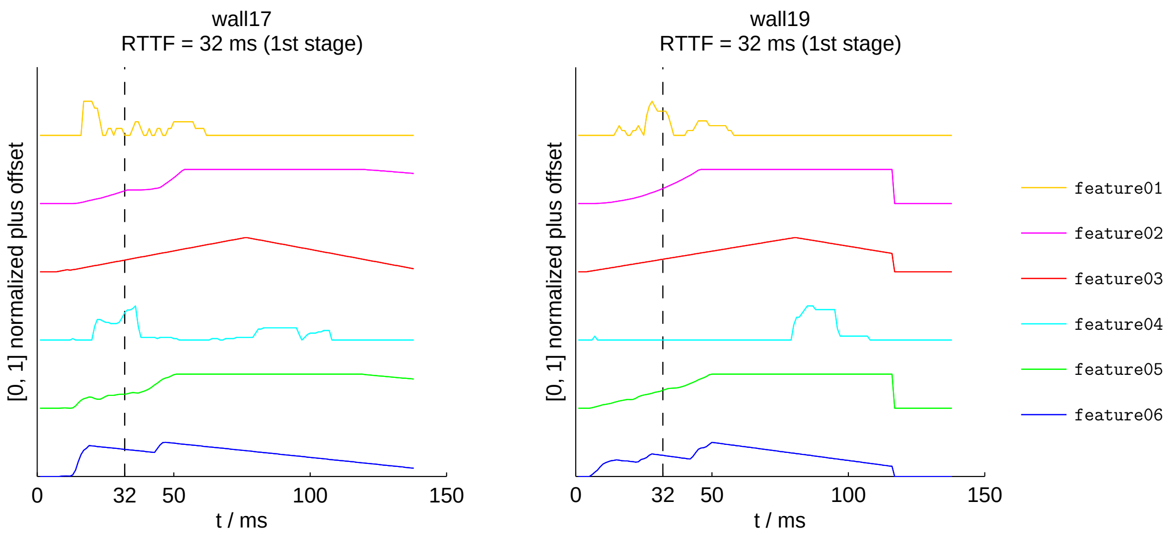


Figure 5.1: Crash signals of two 27 km/h wall crashes. The left-hand plot originates from a car with a different setup than the plot on the right-hand side.

side. Further differences can be found which are not discussed here. We only intend to show the impurity of the database that the learning machine has to deal with.

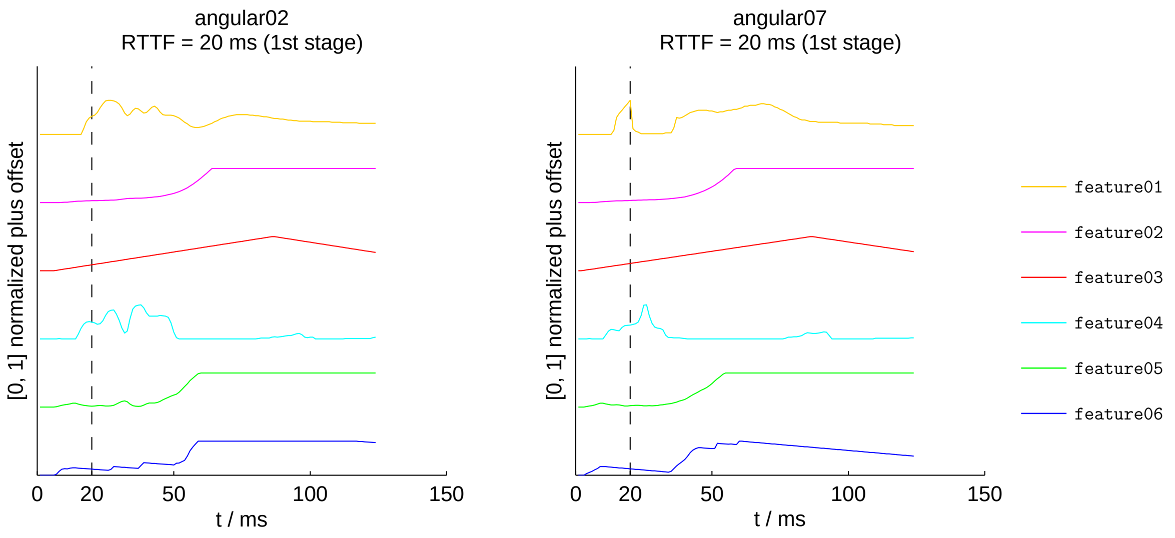


Figure 5.2: Crash signals of two 40 km/h left angular crashes. The car setups again differ between the left and right plot.

Let us have a look at some signals coming from angular crashes. Figure 5.2 shows two of them. Both angular crashes have been performed at the same speed of 40 km/h. In addition, these two crashes demand a deployment of the belt tensioner at 20 ms each, although the automobile setups used in these crashes differ again from each other. Hence some features are quite different in their characteristics e.g., `X_AngleSin` or `X_Symmetry`.

Two candidates of ODB crashes are shown in Figure 5.3. As we already presented quite

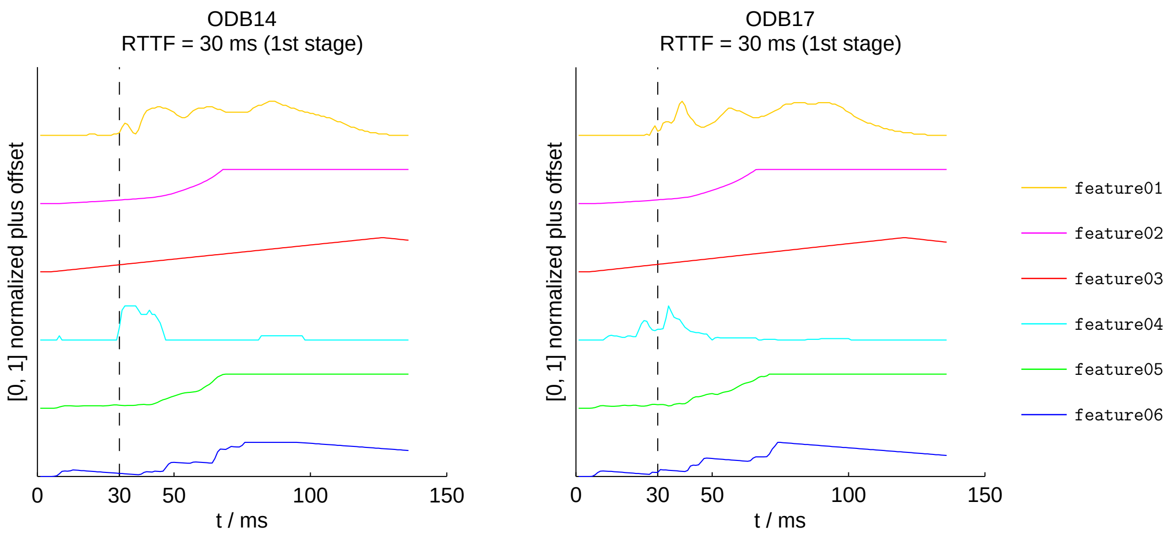


Figure 5.3: Crash signals of two left ODB crashes performed at 60 km/h. Again different car platforms were used in both crash tests.

diverse signals for the wall and angular crash mode, this figure again gives an example how various crash signals may look like although the corresponding crashes were executed at the same speed.

To conclude the overview of crash mode signals, two AZT crashes are presented in Figure 5.4. Whereas a learning machine does not need any crash signal after a fire decision is made, here the "complete" signal plays a fundamental role. A false detection of a fire crash meanwhile a no-fire crash may cause severe injuries. Therefore these signals are crucial for the performance of a learning machine. The plots might suggest the deployment of the belt tensioner after approximately 25 ms since the features resemble wall crashes very much.

Based on 63 crashes partly introduced in this section, classifiers have been trained in order to solve the discrimination of these crash events correctly. The next section starts with the results that can be obtained by applying the standard version of the DDO tool.

5.1.2 Results Obtained by Applying Standard DDO

Now we inspect the classification results based on the standard DDO setting. A support vector machine with Gaussian RBF kernel (2.12) has to be trained separately on every stage. The initial training data consists of all crashes as shown in Table 5.1 but only the scalings $\{-15, 0, +15\}$. The remaining scalings $\{\pm 5, \pm 10, \pm 20, \pm 25\}$ are unknown to the SVM.

We used 5 out of the presented features in Chapter 4, Table 4.1 as input variables. Doing so, we guarantee that all chosen features are stable (low-pass filtered). They shall reflect the physical properties of all used crash modes to a certain degree. Of course, the performance

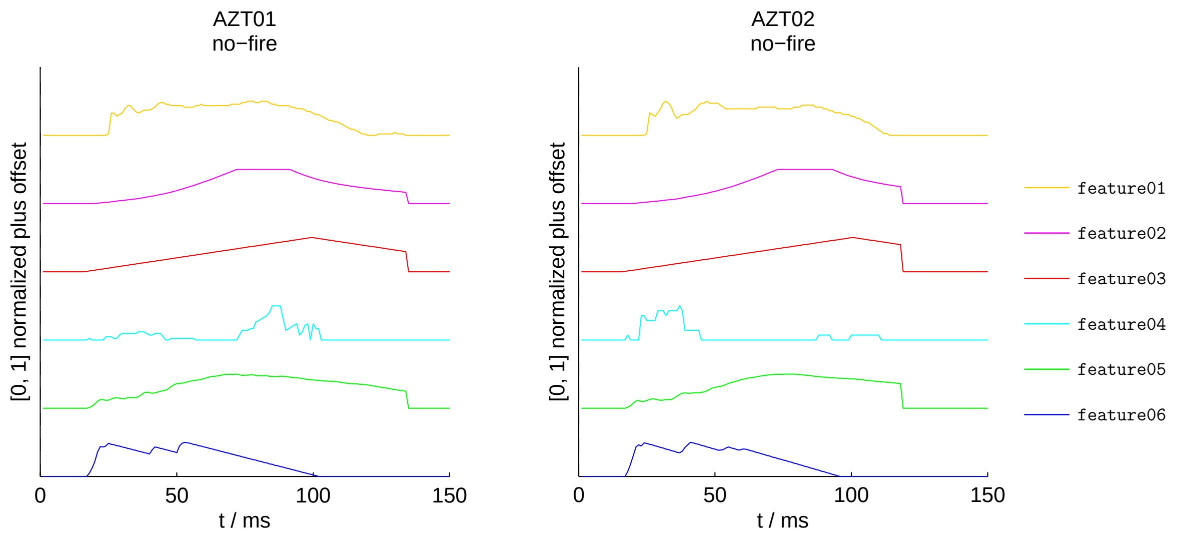


Figure 5.4: Crash signals of two AZT crashes performed at 15 km/h. It is important to note that although the signals are normalized to the unit interval, the magnitude of fire and no-fire signals does not differ to a big degree.

of the SVM massively depends on the selection of features. Nevertheless, we will use this set of features for the following tests.

As we already stated out in Chapter 2, Section 2.3.1, we can further cut fire and no-fire signals, respectively. Therefore the initial settings have to be specified by the user. Since we want a prediction of the fire crashes earlier or around the RTTF, the settings are defined as follows. The parameters $t^{<}$ and $t^{>}$ are both set to zero whereas $t^{-} = -10$ and $t^{+} = 1$. Thus the fire range $\{t_s^{-}, \dots, t_s, \dots, t_s^{+}\} = \{t_s - 10, \dots, t_s, \dots, t_s + 1\}$ which ensures a training set consisting of instances around the RTTF t_s.

This initial set of instances is further reduced by the standard methods presented in Chapter 2, Section 2.3.1. The no-fire patterns of every crash are cutoff at 250 ms where all signals correspond to a flat line anyway. The fires of the computed fire range with the highest stress are then taken away from the training set as well. Finally, after cutting all no-fire crashes and selecting the best fires of every fire crash around the RTTF, the training set is constructed.

During the training procedure (cf. Chapter 2, Section 2.3.2), the no-fire patterns are weighted by the SVM with 40. In general, every pattern has got the weight of one for training a SVM. However, LIBSVM [12] enables to assign different weights to every class[3]. This weight shall prevent the deployment of the restraint system since no-fire patterns are then very hard to classify wrongly.

Performing the grid search procedure for SV model selection, an appropriate machine has been chosen for prediction. The conflicts after classification of the test set (all crashes with

[3]Recall that our problem is a binary classification since we deal with the discrimination of fires from no-fires.

Table 5.2: Conflicting crashes predicted by Gaussian RBF SVM based on grid search. Every no-fire instance is weighted with 40. From all 11 scalings per crash, only the worst conflict of all scalings is shown whereas a critical fire *F* or no-fire *NF* is a bigger conflict than little confidence *lc* or a late fire *LF*.

crash key	stage 1	stage 2	stage 3	stage 4	stage 5
wall01	lcF	lcF	LF		
wall02	lcF	lcF	LF	F	F
wall03	lcF		lcF	lcF	lcF
wall04				lcF	lcF
wall05		LF		F	F
wall06	lcF	lcF		F	F
wall07	lcF		lcF	F	F
wall08	F				
wall09	lcF			F	F
wall10	F			F	F
wall11				F	F
wall12	F	F	lcF		
wall13				lcF	lcF
wall14				F	F
wall15				F	F
wall16				F	F
wall17				NF	NF
wall19			lcF		
AZT01	lcNF	lcNF	NF		
AZT02	lcNF	lcNF	lcNF		
AZT03	lcNF	lcNF	lcNF		
AZT04	lcNF	lcNF	lcNF	lcNF	lcNF
AZT05	NF	NF	NF	NF	lcNF

all scalings) are shown in Table 5.2. The worst conflicts e.g., critical fires (F) and critical no-fires (NF), are highlighted.

Table 5.3: Solved crashes and model complexity of RBF SVM found by grid search. Although only 63 crashes exist in the database, many more crashes arise due to the different scalings. The last three columns represent the complexity of the found SVM. The higher any of the values C, γ or #SVs, the more complex is the discriminant function in general.

stage	fire crashes		no-fire crashes		model complexity		
	solved	total	solved	total	C	γ	#SVs
1	612	616	76	77	5	0.8	559
2	615	616	76	77	5	0.8	560
3	616	616	75	77	5	0.6	555
4	546	572	119	121	5	0.9	575
5	548	572	120	121	5	0.9	550

Table 5.3 finally gives insight into the overall performance of the RBF SVM obtained by applying grid search. It shows the number of crashes that are correctly classified. We will refer to this type of crashes as *solved crashes*. The SVM performs well on all stages as one can see immediately. At most two no-fire crashes have not been recognized correctly. However, many fire crashes (up to 26) have been completely misclassified especially in stage 4 and 5.

Another disadvantage is the obtained SV machine itself since it results in a complex model in every stage after model selection based on (C, γ) grid search. The complexity parameters for all 5 stages are listed in Table 5.3 as well. For all stages, the returned $C = 5$ and thus reaches the maximum of possible values. Recall that we demand a small $C \approx 1$ for a simple model. Moreover, $\gamma = 0.6$ for the third stage is the smallest of all 5 stages[4]. The high values for C and γ lead to a high number of support vectors which lies around 560.

Two goals can be formulated after inspecting the classification of DDO. First of all, we have to incorporate more knowledge in order to classify all non-conflicting crashes correctly. If the conflicts remain and all other crashes have been recognized in the right way, then we have obtained an optimal solution to our problem of crash discrimination and conflict analysis. Second, the model parameters in terms of C, γ and the number of support vectors have to be reduced to ensure an easier model. Therefore we will start applying a different training method to the same training set in the next section in order to force the SVM to classify the remaining non-critical crashes correctly.

[4]Note that $\gamma = 0.1$ is the smallest possible value in the DDO grid search procedure.

5.2 Results for Weighted Crashes

In the present section, we apply the weighting technique that has been introduced in Chapter 4, Section 4.3.2 in order to solve the left over non-conflicting crashes (especially the ones that are critical after the last tests). The following tests have been performed under the same conditions like the ones in the last section concerning chosen features, kernel, class weight of 40 for no-fires, the training set and the model selection step based on grid search.

We simply introduced additional weights for these crashes who had been critical after testing. Of course, both the remaining crashes that have been recognized as critical and the weights depend on the stage as already shown in Table 5.2. This is due to the fact that a different stage may define a different RTTF for the same crash. Every crash weight was set to $C_i = 4$ for all instances indexed by i of former conflicting crashes. *The additional weights C_i are multiplied by the class weight w_{nf} for no-fires and w_f for fires, respectively.* Here and in the last section[5], the tests were performed using $w_{nf} = 40$ and $w_f = 1$. The global weight C that is chosen whilst model selection is still applied to every instance. As a consequence, the weight of an instance multiplicative arises from $C \cdot w_{nf} \cdot C_i$ and $C \cdot w_f \cdot C_i$, respectively[6].

Table 5.4: Conflicts with RBF SVM found by grid search and weighted crashes. Every no-fire instance is weighted with 40. In addition, every former conflicting crash has got a multiplicative weight of 4. From all 11 scalings per crash, only the worst conflict of all scalings is shown whereas a critical fire *F* or no-fire *NF* is a bigger conflict than little confidence *lc*. Weighted crashes that still impose conflicts are marked by *w*.

crash key	stage 1	stage 2	stage 3	stage 4	stage 5
wall17				NF	NF
wall18				NF	NF
wall19				NF	NF
wall20				NF	NF
wall21	NF	NF	NF	lcNF	lcNF
wall22	NF	NF	NF	NF	NF
AZT01	NF	NF	w,NF	NF	lcNF
AZT02	NF	NF	NF	NF	NF
AZT03	NF	NF	NF	lcNF	lcNF
AZT04	NF	NF	NF	NF	NF
AZT05	w,NF	w,NF	w,NF	NF	NF

The results with respect to conflicts of the weighted crashes are found in Table 5.4. The positive remark is the fact that there is not any conflicting fire crash in all stages. However,

[5]Recall that the class weight of 40 for no-fires results from the fact that we want to ensure the non-deployment of no-fire crashes.

[6]Note that the class weights w_{nf} and w_f can be set by LIBSVM through the training parameters `-w0` and `-w1`, respectively

the conflicts somehow "shifted" toward no-fire crashes. The number of misclassified no-fire crashes must be high due to the high occurrence of critical no-fire crashes although only the conflict of the worst conflicting scaling is shown in the table. Note that the first 4 crashes listed in Table 5.4 have a positive RTTF in stage 1 to 3 and a RTTF of zero in stage 4 and 5, respectively.

Table 5.5: Solved crashes and complexity of Gaussian SVM by grid search and weighted crashes. All fire crashes have been solved with a huge drawback. Less than 50% of the no-fires crashes of 1st and 2nd stage are correctly classified.

stage	fire crashes		no-fire crashes		model complexity		
	solved	total	solved	total	C	γ	#SVs
1	616	616	35	77	5	0.9	492
2	616	616	35	77	5	0.9	493
3	616	616	47	77	5	0.8	472
4	572	572	65	121	3	0.3	691
5	572	572	68	121	5	0.4	608

Finally, Table 5.5 concludes the tests of weighted crashes as input for the SV model selection based on grid search. As it was already outlined before, the generalization performance of the SV machines for all stages is very poor with regard to the no-fire crashes. Whereas all fire crashes are correctly labeled due to a higher weight, the classification of the no-fire crashes totally fails. Especially for the first and second stage, less than 50% of the no-fire crashes are classified right.

As we can further resolve from the overall results given in Table 5.5, the complexity of the obtained SV machines is still very high. All stages except the 4th one lead to $C = 5$. Moreover, a very high $\gamma \geq 0.8$ has been found for the first three stages. The number of support vectors even exceeds the former results at more than 50 for stages 4 and 5.

As a consequence, this does not imply that the method for weighting as introduced in Chapter 4, Section 4.3.2 does not have any benefit. It is rather the set of training instances that does not enable a simple model which still generalizes well on the test set. We already gained complex models in the standard DDO tool. It is logical that applying weights in order to introduce a higher local complexity does not conclude in easier models.

The preference of fire crashes by weighting them has shifted the decision boundary in favor of fire patterns and against no-fire patterns, respectively. Hence we drastically have to change the way a training set is selected in order to give predilection to no-fire crashes since they are most essential for the success of our learning machine.

5.3 Evaluation of Support Vector Pruning

The results presented in the last two sections formally ask for more sophisticated methods for selecting an appropriate training set. This section presents the evaluation of such an approach toward simple and still powerful models. The method for pruning of support vectors as proposed in Chapter 4, Section 4.3.1 has been evaluated as follows.

Again we used the same set of features as suggested in Section 5.1.2. Moreover, the initial settings controlling the range of fire instances for every crash remained the same. We only changed the procedure for training set and model selection, respectively.

The training set has been constructed using Algorithm 1 as suggested in the last chapter. The SV machine that is chosen by model selection now follows the procedure of Algorithm 2. Thus we still utilized the Gaussian kernel (2.12) generating radial basis functions. It is significant to note that we neglected the a priori weighting of no-fire instances although particularly this type of crashes impose the biggest problems.

By pruning fire instances that are surrounded or even superimposed by no-fires, many conflicts may be removable from the training set. As a consequence, a much more relaxed SV machine and discriminant function may be found by subsequent model selection. A SVM with a globally simple model e.g., a quasilinear decision boundary, then might perform poor on the test set. However, we still have scope to develop locally complex discriminant function by introducing weights for the remaining wrongly classified crashes. This procedure might fail if the model is already complex (cf. last section).

The test results with regard to left over conflicts in all 5 stages are listed in Table 5.6. It is easy to realize that there are many more remaining conflicts than with standard DDO (cf. Table 5.2) concerning both fire and no-fire crashes. However, only certain crash modes conflict when applying pruning. For instance, the former conflicting wall crashes wall01, wall02, wall04 to wall10 and wall14 do not impose any conflict with pruning.

This observation implies that standard DDO is not optimal for solving wall crashes since the SV pruning solves all but the following three wall crashes: Crash wall18 does not deploy the 3rd and 4th stage and the wall crashes wall21 and wall22 (both no-fires at approx. 16 km/h with less stiffer front) nearly fire in every stage.

Unfortunately, the pruning method totally fails classifying angular crashes. Altogether, 8 out of 13 angular crashes impose conflicts. The classifier found by SV pruning, however, performs well regarding ODB crashes. Only two crashes, ODB20 and ODB21, are critical whereas the former one does not fire in stage 4 and 5 and the latter crash does not fire in any stage which again might result from the fact that the used automobile has a less stiffer front[7].

[7]Note that the ODB crashes ODB18 and ODB22 only impose small conflicts. They have a low confidence in fire decision for the last two stages

Table 5.6: Conflicts resulting from SV pruning for training set and model selection. Every crash has the same weight of one. From all 11 scalings per crash, only the worst conflict of all scalings is shown whereas a critical fire *F* or no-fire *NF* is a bigger conflict than little confidence *lc*.

crash key	stage 1	stage 2	stage 3	stage 4	stage 5
wall03				lcF	lcF
wall11				lcF	lcF
wall12			lcF		
wall13					lcF
wall15				lcF	F
wall16				F	F
wall17				F	F
wall18				F	F
wall19	F	F	F		
wall21	NF	NF	lcNF		
wall22	NF	NF	NF	NF	lcNF
angular01	lcF	lcF			
angular03	F	F	F		
angular04	lcF	lcF			
angular05			lcF		
angular10	F	F	lcF		
angular11	F	F	lcF		
angular12	lcF	lcF	F		
angular13	F	F	F		
ODB18				lcF	lcF
ODB20			lcF	F	F
ODB21	F	F	F	F	F
ODB22					lcF
AZT01	NF	NF	lcNF		
AZT02	NF	NF	lcNF		
AZT03	lcNF	lcNF	lcNF		
AZT04	NF	NF	NF	lcNF	
AZT05	NF	NF	NF		

Regarding the AZT crashes, the pruning method alone does not solve the no-fires sufficiently. Only one of 5 AZT crashes (AZT03) does not impose any critical conflict but has still low confidence in the first three stages. Although the pruning gives big favor to no-fire crashes, it seems to be clear that most problems are too complex to be solved by such an easy model. Recall hereto that we do not apply any class weight to the no-fire instances in advance[8].

Table 5.7: Solved crashes and complexity of RBF SVM by SV pruning. Although no weights have been applied, the classification is only slightly worse than the standard DDO tool. The major advantage is the low number of support vectors (around 300) which is nearly half compared to previous tests.

stage	fire crashes		no-fire crashes		model complexity		
	solved	total	solved	total	C	γ	#SVs
1	589	616	45	77	5	0.5	306
2	588	616	47	77	5	0.5	306
3	600	616	67	77	5	0.9	277
4	558	572	104	121	5	0.9	300
5	557	572	107	121	5	0.9	286

The overall performance and model complexity are listed in Table 5.7. As already expected, the number of solved fire and no-fire crashes is less than standard DDO. However, the number of support vectors is reduced in every stage to nearly 50%. The SV machine created by SV pruning performs nearly as good as the complex SVM from DDO but does not need any a priori information e.g., the class weighting of no-fire instances.

The remaining unrecognized crashes can still be solved by introducing local complexity where it might be necessary to do so. The low number of support vectors reflects that the pruned SVM can further incorporate knowledge without loosing generalization performance[9]. The incorporation of knowledge in terms of weights is evaluated in the next section.

5.4 Combination of SV Pruning and Crash Weighting

Combining the methods of support vector pruning and the weighting of crashes is straightforward. Since the training set which is returned after pruning strongly favors nearly linear models (see Chapter 4, Section 4.1.1 for linearity of SV machines), the introduction of weights for some crashes shall complicate the discriminant function in order to solve remaining unrecognized crashes excluding true conflicts. Recall that we are not interested in obtaining a classifier that solves all crashes. The search for conflicts is the main task for DDO under the condition that all other non-conflicting crashes shall be solved.

[8]The tests before were performed with a class weight of $w_{nf} = 40$ for every no-fire instance.

[9]Recall that the contrary happened in Section 5.2 due to a high number of SVs.

Remember furthermore that the pruning method only prunes fire instances and thus gives strong preference to the no-fire crashes. The subsequent weighting of (basically fire) crashes that have not been correctly classified after pruning (so-called critical crashes) is the trade-off between global simplicity of a linear classifier and the local complexity which is necessary to solve our discrimination problem.

For the sake of simplicity, we applied a weight $C_i = 4$ to every instances of a conflicting crash whereas i is the running index over the complete sample. The value for the weight was found heuristically by merely testing different values. Higher values may force the SVM to finally discriminant hard conflicts which did not even resolve after pruning and weighting. Smaller values may not make any change big enough to solve conflicting crashes. Of course, there might be crashes that completely resemble no-fire crashes. Such conflicts have to be found and isolated by SV pruning and weighting. These remaining conflicts will then be further analyzed by experts. Recall that this is the goal of conflict analysis.

So, let us have a look at the classification results regarding the conflicts that remain after pruning and weighting. Table 5.8 gives an overview of the worst conflicts of all scalings. It is remarkable to see that the biggest fraction of the remaining conflicts are low confidence (lc) fire and no-fire decisions, respectively. Therefore, the SVM would still make correct decisions although the confidence in them is low.

In comparison to the exclusive SV pruning, the combined method does not lead to a model that performs well for wall crashes. The former SV pruning without weights solves all but three wall crashes. We therefore introduced weights for them and as a consequence all stages of these former conflicting crashes perform well. However, the introduction of weights created 9 new critical wall crashes that have at least one misclassified stage. The reason for this phenomenon is simple. Since we basically introduced weights for angular, only a few wall and two ODB crashes, the non-weighted fire crashes do not get any preference. So we basically have many more problems concerning wall crashes.

Fortunately, three out of 8 critical angular crashes could be classified as fires. Hence 5 angular crashes remain unsolved. The two former critical ODB crashes do deploy in every stage anymore. Therefore, the ODB crash ODB21 (less stiffer front) is critical regarding the 3rd and 4th stage. The classification of AZT crashes improved much compared to exclusive SV pruning. However, 4 out of 5 AZT crashes have been misclassified in at least one stage.

The generalization performance and model complexity of the SVM gained by the combined method are presented in Table 5.9. Compared to the standard DDO (cf. Table 5.2), our proposed method does not outperform the old SVM approach. However, the number of support vectors has been reduced to more than 50% for some stages. Model parameters C and γ are similar in all stages.

The reason why our proposed method of constructing an easy SV machine has not outperformed standard DDO might be the following. A training set containing all crash modes already imposes conflicts due to the fact that there exist several modes. Training sets includ-

Table 5.8: Conflicts resulting from SV pruning and crash weighting in combination. Conflicting crashes that occurred after exclusive SV pruning have been applied to a crash weight of 4. From all 11 scalings per crash, only the worst conflict of all scalings is shown whereas a critical fire *F* or no-fire *NF* is a bigger conflict than little confidence *lc*.

crash key	stage 1	stage 2	stage 3	stage 4	stage 5
wall01	lcF	lcF			
wall03				F	F
wall06				F	lcF
wall07				lcF	
wall09				lcF	
wall10				lcF	lcF
wall11				F	F
wall12	F	F	lcF		
wall13				lcF	lcF
wall14				lcF	lcF
wall15				F	w,F
wall16				w,F	w,F
wall17				w,NF	w,NF
wall18				w,lcNF	w,lcNF
wall19	w,F	w,F	w,F		lcNF
wall20	lcF	lcF	F	lcNF	
wall21	w	w	lcNF	lcNF	
wall22	w,lcNF	w,lcNF	w,lcNF	w,lcNF	lcNF
angular02	lcF	lcF			
angular03	w,lcF	w,lcF	w		
angular05	F	lcF			
angular06	lcF	lcF		F	
angular07	lcF	lcF			
angular08	lcF	lcF			
angular10	w,lcF	w,lcF			
angular11	w,lcF	w,lcF	lcF	F	
angular12	F	F	w	lcF	
angular13	w,F	w,F	w		
ODB21	w	w	w,F	w,lcF	w,F
AZT01	w,lcNF	w,lcNF	lcNF		
AZT02	w,lcNF	w,lcNF	lcNF	NF	NF
AZT03	lcNF	lcNF	lcNF	NF	
AZT04	w,NF	w,NF	w,lcNF	lcNF	
AZT05	w,NF	w,NF	w		

Table 5.9: Solved crashes and complexity of RBF SVM by SV pruning and weighting. In comparison to exclusive SV pruning, the combination of both methods results in a slightly higher #SVs for the 1st and 2nd stage whereas the other stages come up with even less support vectors. Many more no-fires and fires are correctly classified. In stage 3, all no-fire crashes are solved correctly.

stage	fire crashes		no-fire crashes		model complexity		
	solved	total	solved	total	C	γ	#SVs
1	601	616	73	77	4	0.8	314
2	602	616	73	77	4	0.8	313
3	607	616	77	77	5	0.6	239
4	562	572	106	121	5	0.8	259
5	561	572	108	121	5	0.9	262

ing only two or three crash modes may solve this problem. The following three sections will evaluate our method of SV pruning combined with crash weighting again. However, we only consider two crash modes in one training set: AZT crashes versus wall, angular and ODB crashes, respectively.

In addition, a dimension reduction is possible since the signals of a crash mode usually differ much compared to another ones. Thus we only consider two-dimensional problems which enables us to plot data points, support vectors and the decision boundary.

5.4.1 AZT vs. Wall

In the present and the next two sections, we will consider crash discrimination of data containing only two crash modes. The task is to find a model that distinguishes between two types of crashes. Since we are interested in separating no-fire crashes from fires, we will regard AZT crashes against wall, angular and ODB crashes, respectively.

The usage of two crash modes as input data instead of all modes has several advantages.

- Less data points have to be discriminated since the database is smaller.
- Less features are needed in order to separate fires from no-fires since the similarity of crashes of the same mode is higher than the one of crashes of different modes.
- The resulting model is hence easier to interpret.
- If the dimensionality of the learning machine's input space is less than 3, the result of

the classifier can be shown as plot.

Therefore we will only study two-dimensional models in the following. Moreover, the SV model that consists of weighted training instances (support vectors) is visualized in order to inspect the classification results, margin of the hyperplane and the linearity of the decision boundary, respectively.

In this section, we start with the discrimination of AZT crashes versus wall crashes. These tests were carried out using two stable features. Again we want to compare the generalization performances of the standard DDO tool with support vector pruning and SV pruning combined with crash weighting. The former class weight of no-fire instances w_{nf} is set back to one since SV pruning and crash weighting do operate with the same setting. Furthermore we also take a bigger fire range $\{t_s^-, \dots, t_s, \dots, t_s^+\}$ into consideration by setting $t^< = 35,\ t^> = 20$ and $t^- = -100,\ t^+ = 5$, respectively. So we take all fire samples 35% before and 20% after the RTTF t_s. Furthermore we substract 100 milliseconds from the left border $t_s^- = 0.65 t_s$ and add 5 milliseconds to the right one $t_s^+ = 1.05 t_s$. In general one can say that we construct a fire range containing every fire instance from the crash beginning[10] at $t = 0$ up to around 5 milliseconds after the RTTF.

The following figures present qualitative results applying standard DDO (on the left), exclusive SV pruning (in the middle) and combination of SV pruning and crash weighting (on the right-hand side) to the discrimination of crashes at a certain stage. The overall results for all stages regarding solved crashes are given in Table 5.10 afterward. The weights $C_i = 4$ have been assigned to crashes that had not been correctly classified before. Although tests for all 5 stages have been performed, we only present results of certain stages since the remaining figures are similar to their representatives.

The features have been normalized for plotting to the normal distribution $\mathcal{N}(0, 1)$. Their original standard deviations and mean values can be found at the axes markings. The pictures itself contain all fire points of the fire range and all no-fire points as red crosses and green dots, respectively. Every support vector is surrounded by a rectangle. The decision boundary is shown as black line whereas the red (blue) line indicates the margin of one on the fire (no-fire) side. These unit margins are shown since the fire or no-fire decision rates as absolutely sure beyond them. One finds minor confidence in this decision between the unit margins of fire and no-fire. The intensity of the shading (cyan for fire region and yellow for no-fire region) corresponds to the value of the discriminant function (4.1) without applying the sign function.

Figure 5.5 shows the decision boundaries of the three different methods resulting from the crash modes AZT and wall at the first stage. One can easily see that the margin in the figure on the left hand side (standard DDO) is small compared to the one in middle and right figure, respectively. The complexity of the decision boundary in the middle is low compared to the figure on the right-hand side. Applying all three methods at the first three stages,

[10]Note that there does not exist any RTTF bigger than 100 milliseconds.

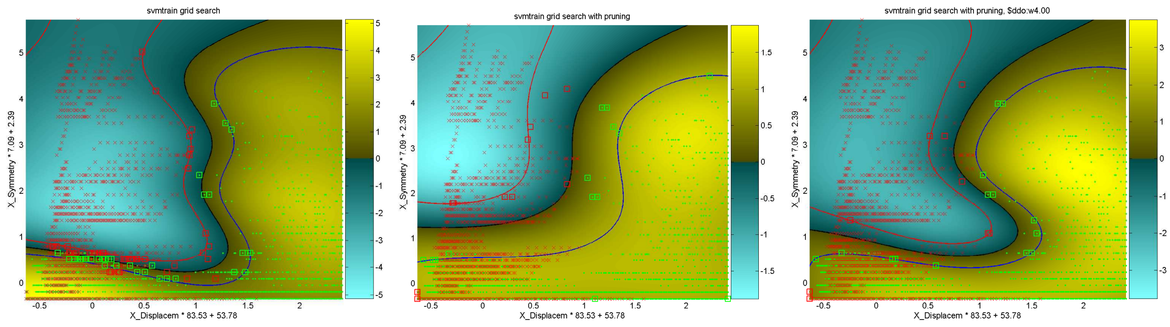


Figure 5.5: AZT crashes vs. wall crashes at 1st stage.

there exists a critical fire crash in the lower left corner of the plots. This conflicting crash is completely located on the wrong side since SV pruning could not remove it and all three models do not recognize any fire pattern. This wall crash (wall19 at 27.0 km/h) shall only fire at the first three stages and not surprisingly was performed with an automobile with a less stiffer front.

Additionally, after SV pruning (middle figure) there is at least one fire crash that is partly located on the no-fire side. By introducing weights for this crash and among others (right figure), the decision boundary takes the conflicting crash into account as well. Note that the figures for the first stage can be found for the second and third stage with only slight differences.

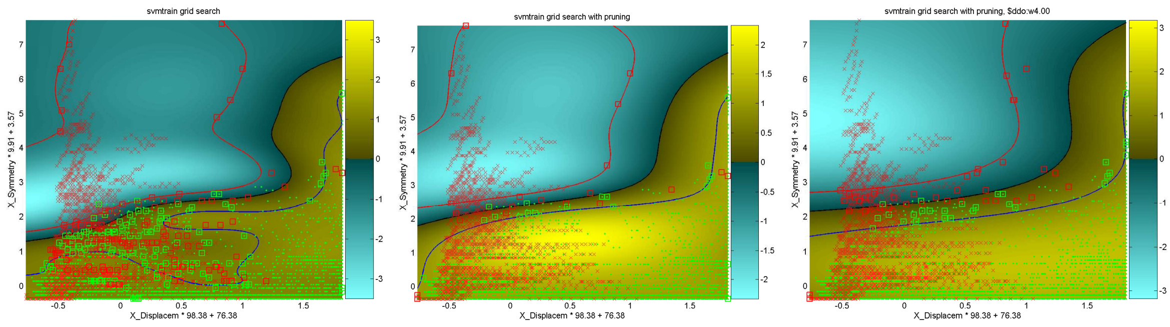


Figure 5.6: AZT crashes vs. wall crashes at 5th stage.

The decision boundaries induced by the SV models for the 4th and 5th stage look similar. Figure 5.6 shows the visualization of the SVM in input space for the 5th stage. Due to the fact that the 4 crashes wall17 to wall20 become no-fires at the 4th and 5th stage, the region of no-fire decision becomes visually bigger compared to the first three stages.

Qualitatively evaluating standard DDO (left plot of Figure 5.6), we find that the unit no-fire margin (blue line) contains a complex bulge which results from the fact that there are many conflicting fire support vectors geometrically far away from the decision boundary. So every no-fire decision inside this huge bulge would lead to a diffident no-fire decision. The rather rough discriminant functions come from high values of $\gamma \geq 0.8$ (cf. Table 5.10) which create

Gaussian distributions with small standard deviation σ at the location of all support vectors since $\gamma = 1/\sigma^2$.

Table 5.10: Solved crashes and complexity of RBF SVM for AZT vs. wall crashes. Standard DDO already performs very well in all stages. Its model complexity, however, is too high. SV pruning and subsequent weighting nearly performs as well as standard DDO. Note that exclusive SV pruning leads to very good results for no-fire crashes.

application	stage	fire crashes		no-fire crashes		model complexity		
		solved	total	solved	total	C	γ	#SVs
standard	1	209	220	65	77	4	0.5	86
pruning	1	176	220	77	77	1	0.4	116
pruning & weighting	1	205	220	77	77	4	0.5	211
standard	2	209	220	65	77	4	0.5	86
pruning	2	176	220	77	77	1	0.3	136
pruning & weighting	2	205	220	77	77	4	0.5	212
standard	3	209	220	64	77	4	0.5	85
pruning	3	176	220	77	77	1	0.4	102
pruning & weighting	3	205	220	77	77	5	0.5	208
standard	4	146	176	108	121	5	0.9	244
pruning	4	138	176	113	121	2	0.4	114
pruning & weighting	4	139	176	120	121	3	0.1	211
standard	5	143	176	108	121	5	0.8	260
pruning	5	138	176	114	121	3	0.5	102
pruning & weighting	5	138	176	115	121	1	0.1	215

Support vector pruning which is shown in the center plot of Figure 5.6 obviously smooths the discrimination function especially on the no-fire side. The bulge disappears due to the high linearity of the kernel ($\gamma = 0.4, 0.5$ for 4th and 5th stage, respectively). Additionally there are whole crashes situated on the wrong side of the hyperplane which naturally impose conflicts, SV pruning finds a much simpler decision function with less support vectors (less than 50%) than the SVM found by standard DDO.

Assigning weights of 4 to the remaining critical crashes does not change the situation dramatically as it can be seen in the right plot. The margin becomes bigger and fire support vectors situated far away from the decision boundary disappear. Only one crash, a no-fire, profits from these changes (cf. Table 5.10) since the slope on the left side of the middle plot disappears by weighting.

To conclude the wall versus AZT crashes, Table 5.10 summarizes the generalization performance and values for model complexity. It becomes clear that the combination of pruning and weighting outperforms standard DDO with respect to the no-fire crashes in all stages.

The quality of classification for the fire crashes nearly reaches the very good results of DDO. One can see that exclusive SV pruning outperforms DDO in the case of the no-fires whereas subsequent weighting is more or less applied to fire crashes.

5.4.2 AZT vs. Angular

The tests for AZT against angular crashes have been carried out using two features of Chapter 4, Table 4.1. It is important to note that another feature combination that has been used for the crash modes AZT versus wall would lead to a perfect classification. Since we are looking for conflicts and methods how to solve every crash but the conflicting ones, this two-dimensional feature combination does not allow us to evaluate our algorithms.

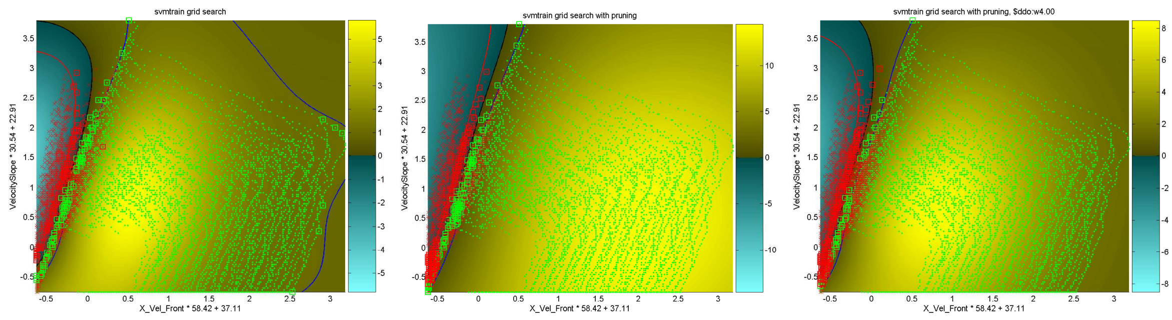


Figure 5.7: AZT crashes vs. angular crashes at 1st stage.

Figure 5.7 shows the labeled input data and the SV model for the 1st stage which is representative for all remaining 4 stages as well. Standard DDO already leads to a somehow quasilinear decision function as it can be seen in the left plot. An almost linear discriminant function in input space can be found by support vector pruning shown in the middle plot. Here the intensity of the no-fire region increases with decreasing distance from the hyperplane which is easier to interpret than the DDO model. There the intensity is high only close to the margin. The result of the combination of SV pruning and crash weighting is shown in the right plot. The found decision functions resembles more the one obtained by DDO, however, its intensity is much higher.

Regarding the generalization performance of all three methods for this combination of crash modes, Table 5.11 shows the classification results and SVM parameters. Surprisingly, the SV pruning performs very bad with respect to no-fire crashes, especially at the 4th and 5th stage. Due to this, SV machines found by SV pruning have a very high number of SVs, even higher than standard DDO. This is due to the fact that the decision boundary between fires and no-fires has a very small margin as it is shown in Figure 5.7. Fortunately, the subsequent crash weighting resolves all unrecognized crashes by assigning weights of 4 to no-fire crashes exclusively. In comparison to standard DDO, the combined method performs very well with similar values for C and γ at all stages. The number of support vectors, however, is higher than with standard DDO.

Table 5.11: Solved crashes and complexity of RBF SVM for AZT vs. angular crashes. Standard DDO only misclassified one fire and two no-fire crashes in stage 1 and 2 and at least two no-fire crashes in the other stages. Its model complexity, however, is too high. SV pruning and subsequent weighting nearly performs as well as standard DDO. Note that exclusive SV pruning leads to very good results for no-fire crashes.

application	stage	fire crashes		no-fire crashes		model complexity		
		solved	total	solved	total	C	γ	#SVs
standard	1	142	143	53	55	1	0.7	187
pruning	1	143	143	26	55	4	0.1	310
pruning & weighting	1	143	143	55	55	1	0.4	268
standard	2	142	143	53	55	1	0.7	187
pruning	2	143	143	26	55	4	0.1	310
pruning & weighting	2	143	143	55	55	1	0.4	268
standard	3	143	143	52	55	1	0.5	210
pruning	3	143	143	24	55	5	0.1	303
pruning & weighting	3	143	143	55	55	4	0.3	189
standard	4	143	143	51	55	2	0.3	219
pruning	4	143	143	5	55	3	0.1	393
pruning & weighting	4	143	143	55	55	1	0.3	311
standard	5	143	143	51	55	2	0.3	221
pruning	5	143	143	5	55	5	0.1	342
pruning & weighting	5	143	143	55	55	1	0.3	317

5.4.3 AZT vs. ODB

To conclude the training of only two different crash modes, we are interested in separating AZT crashes from ODB crashes. Therefore again two different features have been taken as input variables. Before we present the results of these tests, note that ODB crashes resemble AZT crashes at most since an automobile drives into a deformable barrier which usually results in a late crash detection. In addition to this, this classification problem of separating AZT crashes from ODB crashes cannot be solved with a two-dimensional input space given the features that we consider in this thesis (cf. Chapter 4, Table 4.1).

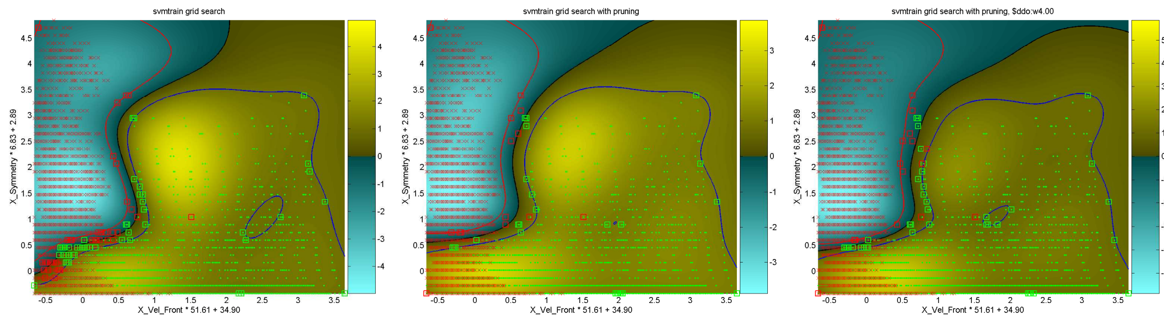


Figure 5.8: AZT crashes vs. ODB crashes at 1st stage.

As we can see in Figure 5.8, many fire patterns are situated in the no-fire region. Without even performing any tests, it is clear that a complete fire-crash in this area will lead to critical conflicts. Standard DDO (left plot) creates a model that appears to be reasonable since the best separation is somehow orthogonal to the x-axis. There exists a fire SV in the no-fire area, however, that introduces a coherent area of a confidence lower than 1 (so-called "island") far away from the decision boundary. A high confidence away from the hyperplane is demanded due to simplicity.

Even SV pruning is not able to prune this fire SV as it can be found in the center plot of Figure 5.8. The island shifts and becomes smaller by applying pruning. The biggest advantage opposite to DDO is the fact that less support vectors are need to construct a discriminant function with nearly same complexity. Crash weighting of non-deployed fire and deployed no-fire crashes just tightens the margin in the critical area. Comparing all solutions together, it becomes clear that there does not exist any quasilinear decision function since the no-fires are located in an area that is upper bounded to some value of the dimension in x-direction. Again, stages 1 to 3 are similar in their obtained SV models.

The SV machines that have been found for the 4th and 5th stage resemble each other. Figure 5.9 gives a qualitative representation of the model for the 5th stage. The standard configuration of the DDO tool finds a model with many SVs with a fair distribution of intensity for the fire patterns. SV pruning results in a similar model with less support vectors and a better intensity distribution of the no-fires. The weighting of unsolved crashes leads to an enlargement of the fire region with a high confidence although not so many fire instances

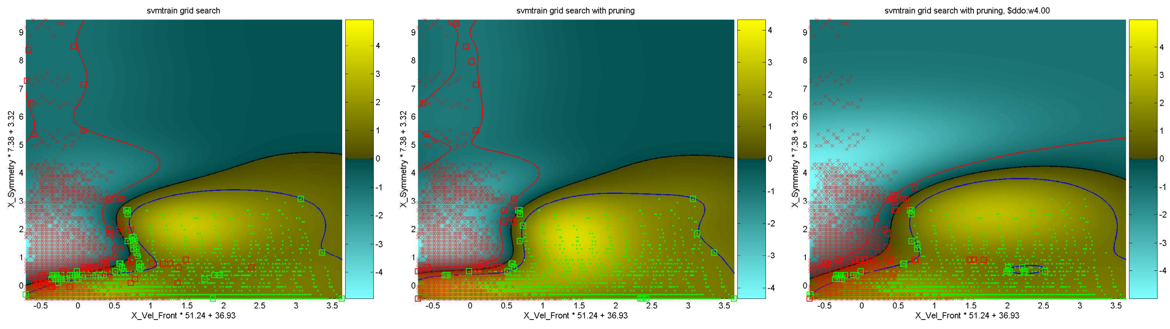


Figure 5.9: AZT crashes vs. ODB crashes at 5th stage.

can be found there. This can be seen by the "slop over" of the fire margin of 1 toward the decision boundary[11]. Thus SV pruning and crash weighting in combination do not lead to a model that is easy to interpret. Moreover, the application of this algorithmic combination creates an island solution which is not accepted from the physical behavior of the system. This area of low confidence in a no-fire decision can be found in the lower center of the middle and right plot.

Table 5.12 substantiates these statements quantitatively once more. Applying only SV pruning nearly recognizes all no-fire crashes at all stages (just one missing crash at the last three stages). Both methods, SV pruning and weighting in combination, only lead to a better performance for the recognition of fire crashes at stages 1 and 2, respectively whereas this quality of classification is even worse than without weighting for the stages 3 to 5.

5.5 Results for Hybrid Approach to Time Series Classification

In the last section of this chapter, we want to evaluate an approach toward time series analysis as it was introduced in Chapter 4, Section 4.3.3. Therefore we map every time series with a possibly variable length to a vector of fixed length (the Fisher score) by a parametrized probabilistic model. We used the GMM implementation of Baggenstoss [8] in order to obtain such a model. The Fisher scores were then discriminated by a support vector machine applying a Gaussian RBF kernel.

The tests of this hybrid approach have had a much longer runtime than all other tests that have been performed. This is due to the computation of the fisher scores which have a complexity of $O(l \cdot m \cdot w)$ where l is the number of temporal sequences. Since we used the

[11]Note the difference of the mean and standard deviation of the normalized feature `X_Symmetry` between the 1st and 5th stage. This is due to the temporal shift of the fire range between these stages. If a crash $\mathcal{C} = (k, T, \mathcal{X}, \mathcal{T}, \omega)$ has to deploy the restraint system in all stages $S = \{1, \dots, 5\}$, then $t_1 > t_5$ whereas $\mathcal{T} = \{t_1, \dots, t_5\}$ is the set of RTTFs.

Table 5.12: Solved crashes and complexity of RBF SVM for AZT vs. ODB crashes. Standard DDO has problems recognizing some no-fire crashes. By only applying SV pruning, one obtains very good results for no-fire crashes. A subsequent weighting of unrecognized crashes impairs the recognition of fire crashes at stages 3 to 5.

application	stage	fire crashes		no-fire crashes		model complexity		
		solved	total	solved	total	C	γ	#SVs
standard	1	239	253	49	55	5	0.8	123
pruning	1	227	253	55	55	5	0.8	113
pruning & weighting	1	236	253	51	55	5	0.8	157
standard	2	239	253	49	55	5	0.8	123
pruning	2	227	253	55	55	5	0.8	113
pruning & weighting	2	236	253	51	55	5	0.8	157
standard	3	239	253	49	55	5	0.8	123
pruning	3	231	253	54	55	4	0.8	105
pruning & weighting	3	216	253	55	55	4	0.3	243
standard	4	239	253	48	55	2	0.4	143
pruning	4	229	253	54	55	5	0.6	60
pruning & weighting	4	222	253	55	55	4	0.2	194
standard	5	237	253	49	55	3	0.2	143
pruning	5	229	253	54	55	5	0.7	60
pruning & weighting	5	221	253	55	55	3	0.2	185

sliding window technique (cf. Chapter 2, Section 2.4.1), every crash is split into many small time series. This and the discrimination for the SVM in the very high dimensional feature space increase the runtime drastically.

Utilizing the gradient space of the Fisher scores as vector space that is based on a GMM, many more parameters come into play which complicate the search for an appropriate model. First of all, the number of Gaussian distributions (modes) and an initialization of the multivariate width of Gaussians has to be chosen. Second, we have to decide if only fire or only no-fire instances are taken for the training of the GMM. This decision is based on the fact that we are interested in the a posteriori probability $p_i(x)$ if a given instance x is a fire (or no-fire) for every Gaussian $f_i,\ i \in \{1, \ldots, m\}$. It should be clear that the performance of the GMM also depends on this choice.

In addition to the GMM settings, the computation of the Fisher scores depends on the window length w and the window lag Δw. The latter parameter has been set to $\Delta w = 1$ for simplicity. Furthermore, the parameter that is used to derive probabilistic model has to be specified (cf. Equation (2.30)). This can be any parameter of the model like mean, standard deviation, covariance, etc. For our tests we set the parameter θ to the mean value $\mu = (\mu_1, \ldots, \mu_m)$ (cf. Chapter 4, Section 4.3.3).

Selected results of this hybrid approach are listed in Table 5.13. The same set of 5 features as in Sections 5.1.2 have been considered as input variables. We only concentrated on the 1st stage since its fire times are the earliest compared to the other stages. Thus stage 1 is usually the hardest problem to discriminate. Moreover, we just performed tests with one and two Gaussian distributions ($m = 1, 2$) since a higher number of modes would increase the runtime enormously. Regarding the window length w, tests with $w = 6, \ldots, 10$ lead much better results than $w = 1, \ldots, 5$. The latter results are omitted since they did not show any reasonable performance.

Unfortunately, the GMM and the Fisher kernel with basic settings do not lead to good solutions concerning the fire crashes. Note that a small window length of $w \leq 5$ does so, however, with extremely poor performance regarding the no-fire crashes. The main problem is the GMM which has to learn a generative model for the data. This learning process is an ill-posed problem that is approximated by a very instable algorithm (cf. Chapter 2, Section 2.4.3). A probabilistic model that generalizes the data well and consequently assigns reasonable a posteriori probabilities to an instance will for sure improve the performance of the SVM based on the Fisher kernel.

Table 5.13: Solved crashes at 1st stage by applying GMM and subsequent RBF SVM. All tests have been performed on an Intel Pentium 4 CPU with 3.6 GHz and 1 GB RAM. The runtime of every test is given in the last column.

GMM settings		Fisher scores		fire crashes		no-fire crashes		runtime
#modes	instances	w	parameter	solved	total	solved	total	[min]
1	no-fires	6	mean	285	616	70	77	19.0
1	no-fires	7	mean	251	616	73	77	9.8
1	no-fires	8	mean	254	616	70	77	3.7
1	no-fires	9	mean	220	616	70	77	1.9
1	no-fires	10	mean	245	616	67	77	1.0
1	fires	6	mean	419	616	66	77	3.1
1	fires	7	mean	355	616	68	77	2.9
1	fires	8	mean	372	616	67	77	0.9
1	fires	9	mean	353	616	70	77	1.2
1	fires	10	mean	196	616	75	77	0.9
2	no-fires	6	mean	293	616	71	77	17.8
2	no-fires	7	mean	275	616	77	77	18.1
2	no-fires	8	mean	161	616	22	77	4.4
2	no-fires	9	mean	258	616	71	77	3.0
2	no-fires	10	mean	245	616	67	77	3.0
2	fires	6	mean	303	616	75	77	1.7
2	fires	7	mean	240	616	76	77	1.8
2	fires	8	mean	199	616	77	77	1.9
2	fires	9	mean	180	616	77	77	2.0
2	fires	10	mean	170	616	77	77	2.0

6 Conclusions and Future Work

In this final chapter of the thesis, the main lessons that were learned based on the work described before are summarized. Open issues and possible improvements for future research are pointed out as well.

6.1 Discussion of Contribution

The present thesis has focused on designing methods for the discrimination of vehicle crash events by the support vector method. On the one hand, the proposed methods can be utilized before the actual calibration process in order to search for conflicts e.g., by using the combination of SV pruning and crash weighting. On the other hand, the suggested SV methods shall be an approach to a predictive model for the discrimination of fires and no-fires.

The predictions are based on different realizations of SV machines. An already established system called DDO which is currently used at Siemens VDO has been extended in multiple ways to especially improve its performance for conflict analysis. The implemented contributions of this thesis have been motivated and explained thoroughly. A broad literature research preceded the design decisions.

The algorithm for conflict analysis named SV pruning that is a first step toward an interpretable model with good generalization performance is based on two steps. The first step is user-independent and works as follows. All no-fires are considered for training. The initial set of fires is pruned by linear SV machines which only tolerate fire patterns that are easy to classify. A subsequent SVM using a kernel that generates Gaussian radial basis functions tries to discriminate the fires from the no-fires. The second stage is optional and requires user information concerning the priority of single crashes. If a crash has not been recognized correctly by SV pruning, then the user can define a higher weight for this crash in order to force the learning machine to classify it correctly. After incorporating the knowledge about priorities, SV pruning is run again with weighted crashes. A subsequent Gaussian RBF SVM is constructed in the training step again.

The experimental approach toward time series analysis on the basis of support vector machines has been implemented by a hybrid model that is motivated by [19]. It consists of a Gaussian mixture model and a support vector machine. After training the GMM, this probabilistic model is used to explain the data by Gaussian mixtures. By transforming the temporal sequences in input space into a gradient spaces based on a chosen parameter of the GMM, the sequences are represented by a new feature vector, the Fisher score. The discriminative power of the SV method is then used to classify the Fisher scores.

The presented ideas have been evaluated by means of a real-world crash database. The results concerning the two-stage approach for gaining a quasilinear model show very clearly that SV pruning and crash weighting almost always lead to a model that generalizes as good as the standard SVM implemented in DDO. In comparison to standard DDO, its advantages are the much lower complexity of the obtained model and the high smoothness of the discriminant function.

The evaluation of the hybrid approach regarding the generalization performance is poor compared to both methods DDO and the combination of SV pruning and crash weighting. Nevertheless it should be stressed out that due to shortage of time not enough parameter tests have been performed to gain a reasonable GMM. As a conclusion, we can say that a more sophisticated approach different toward time series analysis with a better similarity measure than the Fisher scores might enable a well performing tool to discriminate fire from no-fire events.

6.2 Future Work

In the field of crash discrimination and conflict analysis there exist many open issues that persist to be addressed. In the following we take a short look at selected problems and possible extensions that have not been considered in the present thesis. These ideas shall indicate to challenging ways of both future research and performance improvement.

First of all, the combined method of SV pruning and crash weighting has to be applied to different real-world crash databases in order to validate the method for the calibration process. Since Siemens VDO does have many customers all over the world, this broader evaluation should not be very difficult.

Second, the crash weighting that is applied by the user due to a wrong classification may be performed automatically. Simply increase the weights of these crashes which have been misclassified. This does not prevent the user to apply different weights to other crashes or change the automatically applied weights. In point of fact, it is a straightforward simplification to automate the search for conflicts.

So far we applied the same value of weight to every instance of a crash. Since an early fire crash discrimination is preferred, one can also implement the following approach to favor these crash instances that occurred in an earlier point of time. The weight of every instance may be determined by a monotonic decreasing function that starts with an initial weight of e.g., 4. Asymptotically the weights applied to the fire instances become one at either the end of the crash or at $t = t_s$ whereas the RTTF is given by t_s. Thus the SVM will consider an earlier deployment of fire crashes automatically by incorporating decreasing weights.

As we already outlined before, the distance heuristic based on the Gaussian kernel to classify every unseen pattern into one specific type crash is not the best. Since many instances from different crashes resemble each other, a better heuristic has to be found. Having a misclassified crash, for conflict analysis it is important to know details about this decision e.g., the type of crash that has been assigned to the unseen crash. Therefore learning classifiers for different classes might be a key to that problem. The one-against-one approach results in too many classifiers. Considering other approaches likes one-against-all may lead to a better understanding of the conflicts.

Although the SV pruning is already a striking mean against island solutions in the training set selection, the occurrence of these solutions cannot always be prevented. An approach toward a further improvement might be to run sequentially three different filter methods. First, remove all fire patterns that are closer than a certain value to a no-fire pattern. Second, separate every fire crash locally from all no-fire crashes by linear SV machines. Finally, take the rest of instances for the SV pruning which globally separates all fires from all no-fires.

Another open issue is the design decision that we want to solve a binary classification problem. For conflict analysis, this only allows us to take RBF kernels since they preserve the Euclidean distances between the patterns. Modeling our problem with multiple classes, we can apply any Mercer kernel as similarity measure that may result in a learning machine with better performance. Especially kernels for time series analysis as discussed in [29, 36] have to be considered since the assumption of a random distribution does not hold for instances of one crash.

Regarding DDO and conflict analysis, we already state out that the discrimination with only two crash modes as input has many advantages over the complex classification with all 4 crash modes. It is straightforward to extend DDO in order to combine all pairs of crash modes. This would reduce the time for conflict analysis and would give visual insights into the SVM by two-dimensional plots.

The design of an appropriate kernel for our highly unbalanced problem remains an open problem as well. If one could define a Mercer kernel e.g., that measures the similarity between a no-fire trajectory in time with a fire instance, then we could include a priori knowledge about the problem directly by applying such a special kernel function. Hence the generalization performance of this SVM should be better than the presented machines.

Concerning the hybrid approach of GMM and SVM, we already noted that several improvements have to be done to obtain a model with a high generalization performance. The learning step of probabilistic distribution of the fire or no-fire data might be even performed by the SV method itself [36]. Using a probabilistic model to explain the data, one has to determine the number of Gaussian distributions for the GMM. This can be done e.g., by applying clustering methods several times with a different number of clusters.

The ability of the SV model's interpretation was not touched in this thesis. Discriminant functions in d-dimensional input space with $d > 2$ cannot be shown in a plot. Therefore dimension reduction methods e.g., self-organizing maps [21], that preserve the neighborhood of instances might be a key to this problem.

6.3 Conclusions

The primary subject of the present work has been the application of support vector machines to discriminate vehicle crash events. We have motivated the importance of vehicle safety and the utilization of the SVM as learning algorithm to tackle the problem of crash discrimination due to its flexibility and extensibility. We have extended the existing SVM of the tool DDO in order to construct quasilinear and simple SV models which still ensure a good generalization performance. The selection of fire instances for the learning step has become more sophisticated by SV pruning. One may enforce the classification of unrecognized crashes by crash weighting which is directly incorporated into the SV method. We have shown that the combination of SV pruning and crash weighting is a powerful tool for the calibration since it can be used to search for conflicts in the crash database.

With regard to time series analysis, we have proposed the application of a hybrid approach combining the generative power of a parametrized probabilistic model like a GMM and the discriminative power of a SVM. The presented idea has not brought any further improvement or insight into the learning process. However, the analysis of time series instead of single multivariate patterns should be still considered as a way of modeling the problem.

The main message of this thesis is that the SV method represents a powerful and flexible tool to incorporate a priori knowledge in order to search for conflicts and to solve the classification problem of fire and no-fire events effectively. We have demonstrated that the proposed methods can be successfully applied to our problem of crash discrimination in order to achieve both interpretable solutions and low model complexity.

Bibliography

[1] European new car assessment program, 2003. Retrieved August 14, 2007, from http://www.euroncap.com.

[2] Transport safety performance in the EU: A statistical overview. Technical report, European Transport Safety Council, 2003. Retrieved August 14, 2007, from http://www.etsc.be/oldsite/statoverv.pdf.

[3] Road fatalities in europe 2005. Technical report, DG Tren, 2006. Retrieved August 14, 2007, from http://ec.europa.eu/transport/roadsafety_library/care/doc/gis_eu25.pdf.

[4] Materials for presentations. Internal document server. Siemens VDO Automotive AG – SV C RS CC22, January 2007.

[5] National Highway Traffic Safety Administration. Traffic safety facts 2005 - overview. Technical report, U.S. Department of Transportation, 2005. Retrieved August 14, 2007, from http://www-nrd.nhtsa.dot.gov/pdf/nrd-30/NCSA/TSFAnn/TSF2005.pdf.

[6] Franz L. Alt. Archeology of computers: Reminiscences, 1945–1947. *Communications of the ACM*, 15(7):694, July 1972.

[7] R. Anitha, D. Srikrishna Satish, and C. Chandra Sekhar. Outerproduct of trajectory matrix for acoustic modelling using support vector machines. In *IEEE International Workshop on Machine Learning for Signal Processing*, June 2004.

[8] Paul M. Baggenstoss. Statistical modeling using gaussian mixtures and HMMs with MATLAB. Technical report, Naval Undersea Warfare Center, Newport, RI, 2002. Retrieved September 3, 2007, from https://www.npt.nuwc.navy.mil/Csf/htmldoc/pdf/pdf.html.

[9] Claus Bahlmann, Bernard Haasdonk, and Hans Burkhardt. On-line handwriting recognition with support vector machines—a kernel approach. In *Proc. 8th Int. Workshop Front. Handwriting Recognition (IWFHR)*, pages 49–54, 2002.

[10] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin

classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

[11] Klaus Brinker. *Active Learning with Kernel Machines*. PhD thesis, Faculty of Electrical Engineering, Computer Science and Mathematics, University of Paderborn, 2004.

[12] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[13] Ming-Wei Chang and Hsuan-Tien Lin. Weighted training instances for LIBSVM, 2005. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/libsvm-weight-2.81.zip`.

[14] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[16] Fabian Duddeck. Car body design – crash tests, 2007. Lecture Notes. Retrieved August 14, 2007, from `http://www.bm.bv.tum.de/app/de/lehre/vertiefung/app/media/user-files/1168855535663_3-Crash_Tests.pdf`.

[17] I. Guyon, B. Boser, and V. Vapnik. Automatic capacity tuning of very large VC-dimension classifiers. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 147–155. Morgan Kaufmann, San Mateo, CA, 1993.

[18] Paul Herzog. Einsatz von Support Vector Machine Methoden zur Crash Diskriminierung. Diplomarbeit, Fachhochschule Regensburg, November 2006.

[19] Tommi S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 487–493, Cambridge, MA, 1999. MIT Press.

[20] Thorsten Joachims. Estimating the generalization performance of a SVM efficiently. In Pat Langley, editor, *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 431–438, Stanford, US, 2000. Morgan Kaufmann Publishers, San Francisco, US.

[21] Teuvo Kohonen. *Self-Organizing Maps, 3rd edition*. Springer, New York, 2001.

[22] M. Mackay, A. M. Hassan, and J. R. Hill. Observational studies of car occupants' posi-

tions. In *Proceedings of the 16th International Technical Conference on the Enhanced Safety of Vehicles*, Windsor, Canada, May 1998.

[23] J. M. Marin, K. Mengersen, and C. P. Robert. Bayesian modelling and inference on mixtures of distributions. In Dipak Dey and C. R. Rao, editors, *Handbook of Statistics*, volume 25, North Holland, November 2005. Elsevier.

[24] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A 209:415–446, 1909.

[25] C. S. Myers and L. R. Rabiner. A comparative study of several dynamic time-warping algorithms for connected word recognition. *The Bell System Technical Journal*, 60(7):1389–1409, September 1981.

[26] Clemens Otte. *Data-Driven Optimization of Sensor Configurations*, May 2006. Siemens AG – Corporate Technology – Information and Communications.

[27] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[28] F. Rosenblatt. *Principles of Neurodynamics: Perceptron and Theory of Brain Mechanisms*. Spartan Books, Washington, DC, 1962.

[29] Stefan Rüping. SVM kernels for time series analysis. In Ralf Klinkenberg, Stefan Rüping, Andreas Fick, Nicola Henze Christian Herzog, Ralf Molitor, and Olaf Schrüder, editors, *LLWA 01 – Tagungsband der GI-Workshop-Woche Lernen – Lehren – Wissen – Adaptivität*, pages 43–50, Dortmund, Germany, 2001.

[30] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[31] Robert E. Schapire. The boosting approach to machine learning: An overview. In D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, and B. Yu, editors, *Nonlinear Estimation and Classification*. Springer, 2003.

[32] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[33] G. Scholpp, W. Dübel, and B. Steeger. Development of adaptive restraint components by use of DoE methods. In *New Advances in Body Engineering*, pages 1–7, December 1999.

[34] H. Shimodaira, K.-I. Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In T. G. Dietterich, S. Becker, and Z. Ghahramani,

editors, *Advances in Neural Information Processing Systems 14*, volume 2, pages 921–928, Cambridge, MA, 2001. MIT Press.

[35] V. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian]*. Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).

[36] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Diplomarbeit eigenhändig und selbständig verfasst, keine als die angegebenen Hilfsmittel und Quellen verwendet, und direkt oder indirekt übernommene Gedanken als solche gekennzeichnet zu haben.

Magdeburg, 16. Oktober 2007 Christian Moewes