An Intelligent Energy Allocation Method for Hybrid Energy Storage Systems for Electrified Vehicles

by

Xing Zhang
B.Sc., Tongji University, 2007
M.Sc., Tongji University, 2009

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Mechanical Engineering

An Intelligent Energy Allocation Method for Hybrid Energy Storage Systems for Electrified Vehicles

by

Xing Zhang
B.Sc., Tongji University, 2007
M.Sc., Tongji University, 2009

Supervisory Committee

Dr. Zuomin Dong, Co-Supervisor
(Department of Mechanical Engineering)

Dr. Curran Crawford, Co-Supervisor
(Department of Mechanical Engineering)

Dr. Yang Shi, Academic Unit Member
(Department of Mechanical Engineering)

Dr. Adel Guitouni, UVic Non-unit Member
(Peter B. Gustavson School of Business)

Supervisory Committee

Dr. Zuomin Dong, Co-Supervisor
(Department of Mechanical Engineering)

Dr. Curran Crawford, Co-Supervisor
(Department of Mechanical Engineering)

Dr. Yang Shi, Academic Unit Member
(Department of Mechanical Engineering)

Dr. Adel Guitouni, UVic Non-unit Member
(Peter B. Gustavson School of Business)

ABSTRACT

Electrified vehicles (EVs) with a large electric energy storage system (ESS), including Plug-in Hybrid Electric Vehicles (PHEVs) and Pure Electric Vehicles (PEVs), provide a promising solution to utilize clean grid energy that can be generated from renewable sources and to address the increasing environmental concerns. Effectively extending the operation life of the large and costly ESS, thus lowering the lifecycle cost of EVs presents a major technical challenge at present. A hybrid energy storage system (HESS) that combines batteries and ultracapacitors (UCs) presents unique energy storage capability over traditional ESS made of pure batteries or UCs. With optimal energy management system (EMS) techniques, the HESS can considerably reduce the frequent charges and discharges on the batteries, extending their life, and fully utilizing their high energy density advantage. In this work, an intelligent energy allocation (IEA) algorithm that is based on Q-learning has been introduced. The new IEA method dynamically generate sub-optimal energy allocation strategy for the HESS based on each recognized trip of the EV. In each repeated trip, the

self-learning IEA algorithm generates the optimal control schemes to distribute required current between the batteries and UCs according to the learned Q values. A RBF neural networks is trained and updated to approximate the Q values during the trip. This new method provides continuously improved energy sharing solutions better suited to each trip made by the EV, outperforming the present passive HESS and fixed-cutoff-frequency method.

To efficiently recognize the repeated trips, an extended Support Vector Machine (e-SVM) method has been developed to extract significant features for classification. Comparing with the standard 2-norm SVM and linear 1-norm SVM, the new e-SVM provides a better balance between quality of classification and feature numbers, and measures feature observability. The e-SVM method is thus able to replace features with bad observability with other more observable features. Moreover, a novel pattern classification algorithm, Inertial Matching Pursuit Classification (IMPC), has been introduced for recognizing vehicle driving patterns within a shorter period of time, allowing timely update of energy management strategies, leading to improved Driver Performance Record (DPR) system resolution and accuracy. Simulation results proved that the new IMPC method is able to correctly recognize driving patterns with incomplete and inaccurate vehicle signal sample data.

The combination of intelligent energy allocation (IEA) with improved e-SVM feature extraction and IMPC pattern classification techniques allowed the best characteristics of batteries and UCs in the integrated HESS to be fully utilized, while overcoming their inherent drawbacks, leading to optimal EMS for EVs with improved energy efficiency, performance, battery life, and lifecycle cost.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to express my special appreciation to my advisors, Dr. Zuomin Dong and Dr. Curran Crawford, for your generous supporting, warm encouragement, and priceless advices on both my research and on my career. Thank you so much for being tremendous mentors for me.

A special thanks to my family. Words cannot express how grateful I am to my parents and in-laws for all of the sacrifices that they have made on my behalf and for the patience and guidance at the difficult moments during my life. I would also like to thank my daughter Lynnette Zhang who came to my life last year. Thank you for giving me the opportunity to be a father and for bringing me so many unforgettable moments. Lastly, I would like express heartfelt appreciation to my beloved wife Yue Hu who spent sleepless nights with me and always supported me in the moments when there was no one to answer my queries.

It seems probable that once the machine thinking method had started, it would not take long to outstrip our feeble powers. They would be able to converse with each other to sharpen their wits. At some stage therefore, we should have to expect the machines to take control.

Alan Turing

DEDICATION

In memory of my beloved grandmother, YU Yuelian.
You left fingerprints of grace on my life.

Chapter 1

Introduction

1.1 Background

Today, increasing environmental concerns, dwindling petroleum reserves, and increasing gas price demand alternative energy solutions for the dominating energy consuming transportation sector, especially ground vehicles. Meanwhile the total number of vehicles globally will increase from 700 million to 2.5 billion over the next 50 years [1]. Further improvement of fuel economy and finding alternative or renewable energy sources for vehicle propulsion has been the focus of vehicle technology development worldwide.

There are a bunch of advanced technologies that help to reduce petroleum consumption and tailpipe emissions. One straightforward approach is to enhance powertrain efficiency and lower vehicle resistance forces. This technical path can be further divided into four major technical categories: engine, transmission, vehicle techniques and hybrid techniques. A comprehensive survey of those techniques can be found in [2]. Hybridization of powertrain is widely considered as a practical and effective solution to remarkably improve ICE efficiency and emissions in near future [3]. Hybrid vehicle (HV) is defined as a vehicle with two or more energy storage system (ESS), both of which must provide propulsion power–either together or independently [4]. Specifically, in addition to conventional fuel tank, the secondary ESS could be flywheel, compressed air tank, battery, ultracapacitor as well as combination of battery-ultracapacitor, as summarized in right-bottom block of Figure 1.1 [5] [6] [7]. These types of HVs differ from each other greatly from operation principle, performance and FE benefits as well as costs. Those HVs equipped with battery as

ESS are in a monopoly position from aspects of both count and type, in comparison with other competitors. The second strategy of reducing petroleum consumption is to shift use of petroleum to other energy sources. Various alternative energy sources and corresponding powertrains are summarized at two sides of Figure 1.1, according to energy sources, on-board energy and propulsion systems. Primary solutions include flexible-fuel vehicle (FFV) and electrified vehicles (EV).

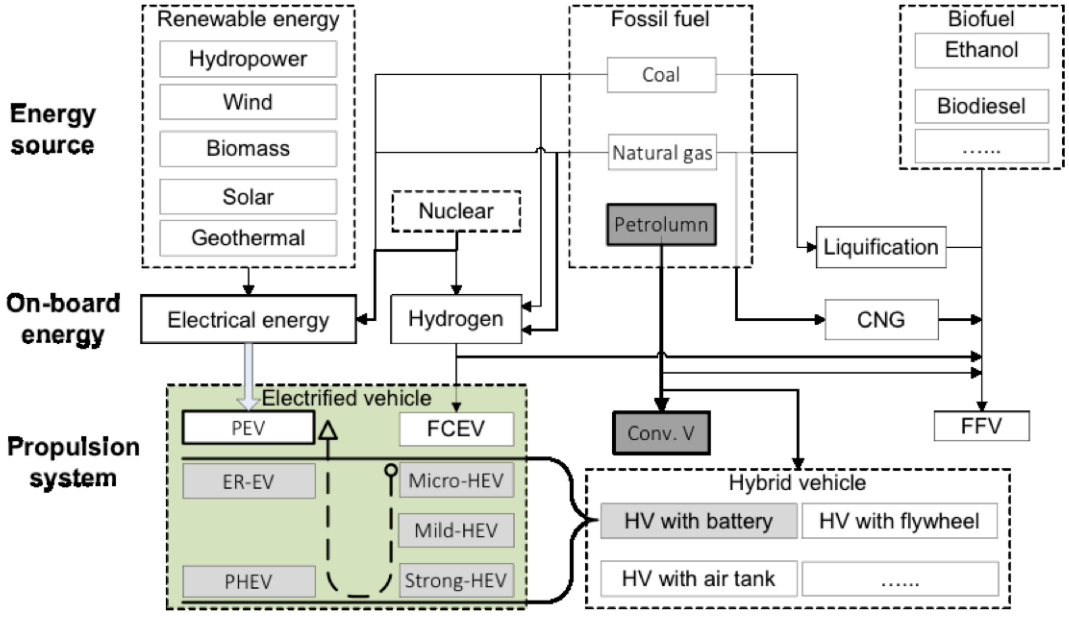


Figure 1.1: Alternative fuel and powertrain solutions

1.1.1 Electrified Vehicles

Although flexible-fuel vehicle (FFV) will continue expanding market penetration, electrified vehicles (EV) will be the most practical and influential choice in the following decades for a couple of reasons: a) electric energy is pivotal element for diversification of energy sources, beneficial for energy security; b) petroleum will continue to be primary fuel of on-land vehicles in decades, so hybridization of vehicle will play a critical role in improving mass-production vehicle efficiency and reducing emissions; c) hybrid electric vehicles shown at left-bottom corner of Figure 1.1 is intersection of electrification and hybridization approaches, providing a wide range of technical solutions [3, 2]. Electrified vehicle, especially hybrid vehicle, combines hybridization and electrification, possessing special potential.

Powertrain architecture, which refers to layout and energy flow paths among powertrain components, is an important index of xEV powertrain. Many scholars have categorized xEV into Series Hybrid, Parallel Hybrid and Power-split Hybrid, with emphasizes on EM, power electronics or modeling, respectively [5] [8] [9] [10] [11]. Meanwhile, electrification level, which has great impact on architecture design and selection, breaks up electrified vehicle into five categories of hybrid vehicle (micro Hybrid EV, mild Hybrid EV and strong Hybrid EV, Plug-in Hybrid Electric Vehicle (PHEV), Extended Range Electric Vehicle (ER-EV)), Pure Electric Vehicle (PEV) and Fuel Cell Electric Vehicle (FCEV).

The first appearance of EVs dates back to the early 1830. These EVs were not commercial vehicles as they used non-rechargeable batteries. It will take an additional half a century before batteries are developed sufficiently to be used in commercial vehicles [12]. In 1989 Ferdinand Porsche, an employee of the Austrian company Jacob Lohner & Co, developed a drive system based on fitting an electric motor to each front wheel, without using a transmission [13]. During the 20th century petroleum powered vehicles showed absolute dominance over the EVs. The reasons are easily understood when the specific energy of petroleum fuel is compared to that of batteries. For example, the specific energy of diesel, i.e. energy stored per kilogram, is about 12600 Wh/kg, while the highest reported specific energy of Lithium-air batteries is about 360 Wh/kg [14, 15]. Moreover, the diesel is much cheaper with 0.15 e/kWh, compared to the optimistic price of about 180 e/kWh for energy optimized batteries projected by the United States Advanced Battery Consortium [16].

Nevertheless, the electrification of vehicles has increased again in the 21st century motivated by the air pollution, global warming and rapid depletion of the Earth's petroleum resources. Obviously, in order to develop efficient and cost effective EVs, one of the key technical challenges and bottle necks needed to be improved or solved in vehicle electrification is the high performance, reliable, long-lasting and low-cost on board electrical energy storage system (ESS).

1.1.2 Electrical energy storage system

As mentioned above, an electrified vehicle could be categorized into Series Hybrid EV, Parallel Hybrid EV or Power-split Hybrid EV depending upon how the power from its ICE and electric motors/generators (M/Gs) are blended. Also, an electrified vehicle can be classified into micro Hybrid EV, mild Hybrid EV and strong Hybrid

EV, Plug-in Hybrid EV, Extended Range EV, Pure EV or even Fuel Cell EV based on its electrification level. Obviously, no matter in what category an EV is, it always needs a reliable and efficient electrical energy storage system to store the electrical energy.

The most common energy storage device used in electrified vehicles is the battery. Batteries have been the technology of choice for most electrified vehicle applications, because they can store large amounts of energy in a relatively small volume and weight and provide suitable levels of power for many applications. Shelf and cycle life have been a problem/concern with most types of batteries, but people have learned to tolerate this shortcoming due to the lack of an alternative. In recent times, due to the increasing electrification level, power requirements in a number of electrified vehicle applications have increased markedly and have exceeded the capability of batteries of standard design.[17] This has led to the design of special high power, pulse batteries often with the sacrifice of energy density and cycle life. Ultracapacitor have been developed as an alternative to pulse batteries. As an attractive alternative, ultracapacitors enjoy much longer shelf and cycle life than batteries. By 'much' is meant about one order of magnitude higher.[17]

In order to improve the efficiency, performance, cost and life of on-board electrical energy storage system, appropriate integration of two or more energy sources has been researched to allow the best characteristics of each type to be fully utilized, leading to a hybrid energy storage system (HESS). As discussed above that battery alone cannot serve as the optimal energy source for electrified vehicles due to their power/energy trade-offs, as shown in the Ragone plot of Figure 1.2 [18]. Combining batteries and ultracapacitors can create an ESS with both high peak power and high energy density.

1.2 Research contributions

As mentioned above, a hybrid energy storage system (HESS) is able to provide Electrified Vehicles with both high peak power and high energy density. However, this new hybrid structure also raises a question on how to control battery and ultracapacitor together properly or even optimally.

This work mainly focuses on developing an intelligent control algorithm which coordinates the battery and ultracapacitor inside a HESS by optimizing their the energy flow. This new proposed algorithm is able to adapt itself to new driving trips. Together with this intelligent control algorithm, this research also contributes with

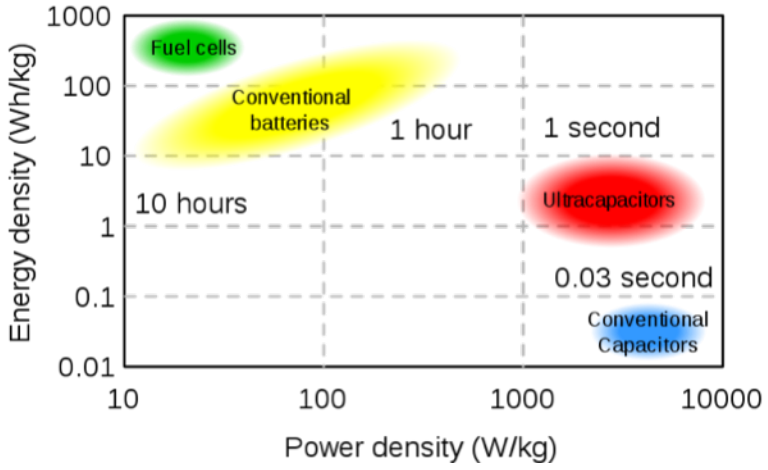


Figure 1.2: Ragone plot

a new joint simulation method, two new driving pattern recognition algorithms focusing on automatic feature selection and better dynamic property separately. These two new pattern recognition algorithms are used by the proposed intelligent control algorithm, since it allocates the energy between battery and ultracapacitor according to the current driving trip estimated by the recognized driving patterns. The proposed new joint simulation method is used to simulate the new intelligent control algorithm and validate its performance. Detailed contributions are listed as follows:

1. Both battery and ultracapacitor have complicated electrochemical and electrothermal processes. In order to carry out Reinforcement Learning (RL) from the HESS data and simulate/validate the proposed algorithm, a Matlab-Python combined simulation method is proposed in this work. By using this method, we are able to generate training data set for training the value network inside the proposed Intelligent Energy Allocation algorithm. Also, by using this method, we are able to validate this work.

2. The system efficiency of three major topologies of battery-ultracapacitor HESS are calculated and compared by using a typical pulsed current load in this manuscript. This work was published on the Proceedings of the ASME 2011 International Mechanical Engineering Congress & Exposition, IMECE2011 [19]. Based on this comparison analysis work, the semi-active topology with an ideal DC-DC converter in the ultracapacitor side is selected as the HESS structure in this research work

3. For the selected HESS structure, we proposed an intelligent energy allocation algorithm (IEA) by employing a reinforcement learning. This proposed algorithm is able to adapt itself to different daily driving modes. It keeps learning to generate optimal policies to prolong the battery life and improve the HESS efficiency from the data collected during the daily driving. The more the car is driven, the higher the HESS efficiency would be. The fact, that the proposed IEA algorithm is based on the driving modes recognized from the daily driving, leads to the following two contributions.

4. A more efficient and robust driving pattern recognition technique, extended Support Vector Machine (SVM) with embedded feature selection ability, is proposed in this work. Besides statistical significance, this proposed SVM also takes into account the accessibility and reliability of features during feature selection, so as to enable the driving condition discrimination system to achieve higher recognition efficiency and robustness. This work was published in the Journal of Franklin Institute [20]. This work helps to obtain a higher accuracy of the driving pattern recognition, hence the better performance of the IEA algorithm.

5. A novel classification algorithm, Inertial Matching Pursuit Classification (IMPC), is also proposed in this work. Compared with the traditional methods using SVM or Neural Networks, IMPC can recognize the diving patterns by using vehicle velocity data sampled in less sampling time, so that the accuracy of estimating the overall driving conditions for entire driving trip is improved. This work has been submitted to the IEEE Transaction on Intelligent Transportation Systems. This work enables the IEA to recognize the driving pattern within a shorter time and therefore helps the IEA to have a better dynamic performance.

1.3 Outline of the dissertation

This dissertation is organized in seven chapters. After introducing the research background and reviewing some related topics in Chapter 1 and Chapter 2, we firstly compared the three different topologies for the battery-ultracapacitor hybrid storage system in Chapter 3. Also in that chapter, a combined simulation method is proposed by employing both Matlab and Python. At the end of that chapter, we will have the vehicle model and the simulation platform ready for the following chapters.

Chapter 4 and Chapter 5 are both focusing on the pattern recognition problem. In Chapter 4, we are focusing on the feature selection problem, while in Chapter 5, we are focusing on improving the dynamic property of the pattern recognition system. Feature selection approaches can be divided into three categories: filters, wrappers and embedded approaches. Among them, embedded approaches can simultaneously determine features and classifier during the training process and hence enjoys the highest efficiency. [21] In spite of the lack of oracle property, 1-norm SVM, as a very standard embedded approach, has both good performance in feature selection and classification. [22]. In Chapter 4, the extended Support Vector Machine (SVM) is developed from the standard 1-norm SVM. This proposed method is able select less features with better observabilities. In Chapter 5, the Inertial Matching Pursuit Classification (IMPC), is proposed especially for recognizing vehicle driving patterns in a shorter time period. Earlier research [23] has shown that if the length of the sampling time of the vehicle speed reaches or exceeds 3 minutes, the characteristic of the current driving pattern can be effectively recognized by common pattern recognition algorithms, such as Support Vector Machine (SVM) and Neural Networks. However, a Driving Pattern Recognition system with 3 minutes sampling time has really bad resolution in daily driving. Therefore, the Inertial Matching Pursuit Classification is developed from the Compressed Sensing algorithm because of its ability of extracting high-level abstract information (features) from sparse data. After Chapter 4 and Chapter 5, the accurately recognized driving pattern is fed to the Intelligent Energy Allocation algorithm, which is finally proposed in Chapter 6. This IEA algorithm needs current driving pattern as input, so that the Trip Mode can be accurately recognized in time. After each driving pattern is recognized, this algorithm generates the optimal control actions to separate the required current between battery and ultracapacitor according to the learned Q network. In Chapter 7, we conclude the entire thesis and discuss the future work. More details about the structure can be found in the following list and Figure 1.3

Chapter 1 Introduction and claims of this research

Chapter 2 Review on the HESS design/control problems and related topics

Chapter 3 Compare the HESS topologies and build the object models and simulation platform. Propose the combined simulation method. Parts of this work was presented at the 2011 ASME International Mechanical Engineering Congress and Exposition as a conference paper

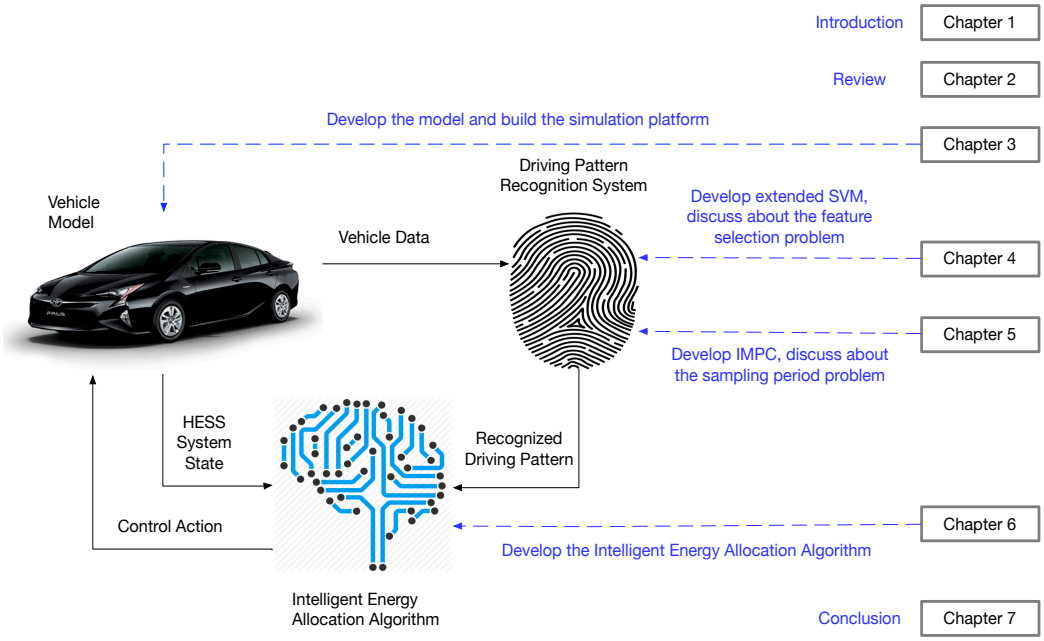


Figure 1.3: Outline of the dissertation

Chapter 4 Present the new extended SVM for driving pattern recognition and discuss about the feature selection problem of the driving pattern recognition system. An early version of the work was published on the Journal of the Franklin Institute in 2015.

Chapter 5 Present the Inertial Matching Pursuit Classification algorithm for driving pattern recognition and discuss about the sampling period problem. This work was submitted to the IEEE transactions on intelligent transportation systems as a journal paper.

Chapter 6 Develop the intelligent energy allocation algorithm (IEA)

Chapter 7 Summarize the research work and discuss the future work.

Chapter 2

Related Topics Review

This chapter gives a background on Li-ion battery, ultracapacitor and hybrid energy storage system, and based on that, formulates the intelligent HESS design/control problem.

2.1 Li-ion battery in EVs

Li-ion batteries have an unmatchable combination of high energy and power density, making it the technology of choice for portable electronics, power tools, and electrified vehicles (EVs) [24]. Many kinds of li-ion batteries are employed in EV. If EVs replace the majority of gasoline powered transportation, Li-ion batteries will significantly reduce greenhouse gas emissions [25]. Some of the current EV and the employed batteries are listed in Table 2.1 [26]. Li-ion batteries are of intense interest from both industry and government funding agencies, and research in this field has abounded in the recent years. Yet looking to the future, there are many who doubt that Li-ion batteries will be able to power the world's needs for portable energy storage in the long run [27]. For some applications (such as transportation and grid) Li-ion batteries are costly at present, and a shortage of Li and some of the transition metals currently used in Li-ion batteries may one day become an issue [28].

2.1.1 Temperature and voltage restrictions

Li-ion batteries on EVs have high capacity and large serial-parallel numbers, which, coupled with such problems as safety, durability, uniformity and cost, imposes limitations on the wide application of li-ion batteries in the vehicle. Li-ion batteries must

Table 2.1: Some EVs and their employed li-ion batteries

Vehicle	Bettery Supplier	Positive electrode	Negative electrode
Nissan Leaf EV	Nissan NEC JV	LMO	C
Chevrolet Volt	Subsidiary of LG Chem	LMO	C
Renault Fluence	Nissan NEC JV	LMO	C
Tesla Roadster	Panasonic Energy	NCA	C
Tesla Model S	Panasonic Energy	Nickel-type	C
BYD E6	BYD	LFP	C
Subaru G4e	Subary	LVP	C
Honda Fit EV	Toshiba Corporation	NCM	LTO

operate within the safe and reliable operating area, which is restricted by temperature and voltage windows. Exceeding the restrictions of these windows will lead to rapid attenuation of battery performance and even result in safety problem.

According to the instructions of most battery manufacturers, the reliable operating temperatures required by a majority of current automotive li-ion batteries (graphite/$LiMn_2O_4$ or by acronyms C/LMO, C/$LiCo_xNi_yMn_zO_2$ or C/NCM, C/LFP or C/$LiFePO_4$, C/$LiNi_{0.8}Co_{0.15}Al_{0.05}O_2$ or C/NCA) are: discharging at -20 to 55°C and charging at $0-45$°C and for li-ion battery with $Li_4Ti_5O_{12}$ or LTO negative electrode, the minimum charge temperature can be -30°C. Usually, the operating voltage of lithium-ion batteries is between 1.5V and 4.2V (C/LCO, C/NCA, C/NCM and C/LMO about $2.5-4.2$V, LTO/LMO about $1.5-2.7$V and C/LFP about $2.0-3.7$V). Normally when the temperature is $90-120$°C, the Solid Electrolyte Interface (SEI) film will start exothermic decomposition [29, 30, 31], but some electrolyte systems will decompose at a lower temperature of about 69°C [26]. When the temperature exceeds 120°C, the SEI film after decomposition is unable to protect negative carbon electrode from side reactions with the organic electrolyte and combustible gas would be produced [31]. When the temperature is about 130°C, the separator will start melting and shutting the cell down [32]. When the temperature becomes higher, the positive material will start decomposition ($LiCoO_2$ will start decomposition at temperature of about 150°C [26], $LiNi_{0.8}Co_{0.15}Al_{0.05}O_2$ at about 160°C , $LiNi_xCo_yMn_zO_2$ at about 210°C [33], $LiMn_2O_4$ at about 265°C [29] and $LiFePO_4$ at about 310°C [26]) and produce oxygen. When the temperature is above 200°C, the electrolyte will decompose and produce combustible gas [31], and it will have violent reaction with

the oxygen produced by the decomposition of the positive electrode [33] and start to catch fire and lead to thermal runaway. To charge li-ion batteries below 0°C will lead the metallic lithium to deposit on the carbon negative electrode surface and therefore reduce the cycle life of batteries [34]. At an extremely low temperature, the cathode of batteries will break down, and result in short circuit [35]. If the voltage is too low or the batteries are overdischarged, the phase change will lead the lattice to collapse and therefore the performance of the batteries is influenced [36]. Moreover, it will lead the negative copper collector to dissolve in the electrolyte (For this reaction, the thermodynamic equilibrium potential is 0.521V vs. SHE (Standard Hydrogen Electrode) or 3.566V vs. Li/Li+ under standard condition). When the batteries are recharged, the copper dendrite will be formed at the negative electrode, which, consequently, will result in short circuits within the batteries [36, 29]. An extremely low voltage or overdischarge will also lead to the reduction of the electrolyte, produce combustible gas [36] and therefore pose potential security risks. An extremely high voltage or overcharge will lead the positive electrode to compose and therefore a great amount of heat is produced [37, 36] . It will also lead the metallic lithium to be deposited on the surface of negative electrode, which will accelerate the capacity fade, result in internal short circuits and safety problem [36],as well as the decomposition of the electrolyte (the common electrolyte will decompose if the voltage is higher than 4.5V [36])

2.1.2 Battery Management System (BMS) in EV

Usually, the capacity and voltage of the battery cell used in the EV are relatively small. So first the single battery cells should be packed and integrated to a battery module, and the battery system in the EV often contains one or more module according to the requirement. The battery system usually consists of hundreds or thousands of single cells. To manage so many cells, the battery management system (BMS) is very important. The system could manage the battery by monitoring the battery, estimating the battery state, protecting the battery, reporting the data, balancing it, etc. BMS in vehicles is comprised of kinds of sensors, actuators, controllers which have various algorithms and signal wires. Three main tasks of the BMS in vehicles are as follows: [26]

- To protect the cells and battery packs from being damaged.

- To make the batteries operate within the proper voltage and temperature interval, guarantee the safety and prolong their service life as long as possible.
- To maintain the batteries to operate in a state that the batteries could fulfill the vehicles' requirements.

At present, the key issues or difficulties of BMS are precise measurement of cell voltage, estimation of battery states, battery uniformity and equalization, and battery fault diagnosis. Among them, estimation of battery states is the most challenger issue. Battery states include state of charge, state of health and state of function.

State of charge

State of charge (SOC) means the ratio of the remaining charge of the battery and the total charge while the battery is fully charged at the same specific standard condition [38]. And the SOC is often expressed in percent, 100% means fully charged and 0% means fully discharged. SOC is defined mathematically as follows:

$$SOC = [SOC_0 - \frac{1}{C_N}\int_{t_0}^{t} I \cdot d\tau] \cdot 100\% \tag{2.1}$$

State of health

There is still no consensus in the industry on what SOH is and how SOH should be determined. State of health (SOH) is a figure of merit of the present condition of a battery cell (or a battery module, or a battery system), compared to its ideal conditions [39]. The unit of SOH is percent, and 100% means it is a fresh battery. The SOH could be derived by capacity and the internal resistance, and it could also be derived by other battery parameters like AC impedance, self-discharge rate, and power density. Take the capacity as an example, SOH could be defined as the ratio of the current capacity and the rated capacity given by the manufacture [40]. See Equation 2.2. Or SOH can be defined by the internal resistance as shown in Equation 2.3, where R_{eol} is the internal resistance at the end of battery life; R_{new} represents the internal resistance of new battery; R indicates the current internal resistance of battery. [41]

$$SOH = C_k / C_{rated} \cdot 100\% \tag{2.2}$$

$$SOH = \frac{R_{eol} - R}{R_{eol} - R_{new}} \cdot 100\% \tag{2.3}$$

State of function

The SOC describes how the battery differs from a fully charged battery, and the SOH describes how the battery differs from a fresh battery. The state of function (SOF) is used to describe while the battery is employed, how the battery performance meets the real demands. The SOF is determined by the SOC, SOH, operating temperature and the charge/discharge history if needed. For the battery used in the system which requires specific supplied power, the SOF should describe how the battery meets the power demands. Thus, the SOF could be defined as a yes/no logical variable [42], while the SOF equals 1 means the battery could meets the demands and SOF equals 0 means could not. However, it would be more preferred to define the SOF as this equation:

$$SOF = \frac{P - P_{demands}}{P_{max} - P_{demands}} \cdot 100\% \tag{2.4}$$

where P means the possible power the battery could supply, the $P_{demands}$ means the demands of the power, and the P_{max} means the maximum possible supplied power of the battery.[26]

2.1.3 Battery remaining useful life and state of health estimation

According to Equation 2.3, SOH can be calculated by using the battery internal resistance. In fact, battery internal resistance is one of the important factors in determining battery performance and battery remaining useful life (RUL). The main factor which leads to the increase of internal resistance is the formation of the solid electrolyte interface (SEI). This is a chemical process between electrolyte and anode, which leads to the deposit of a thin layer, as shown in Figure 2.1, it resists the flow of current during both charge/discharge [43]. The thin layer of SEI is useful to prevent intercalation of impurity lithium-ion. But, with battery cycling and temperature effect, this lead to increase SEI layer and slow down the current flow. Which reflect on the capability of the battery to hold capacity and supply it. [44]

As discussed in the previous section, SOH is a primary indicator that quantita-

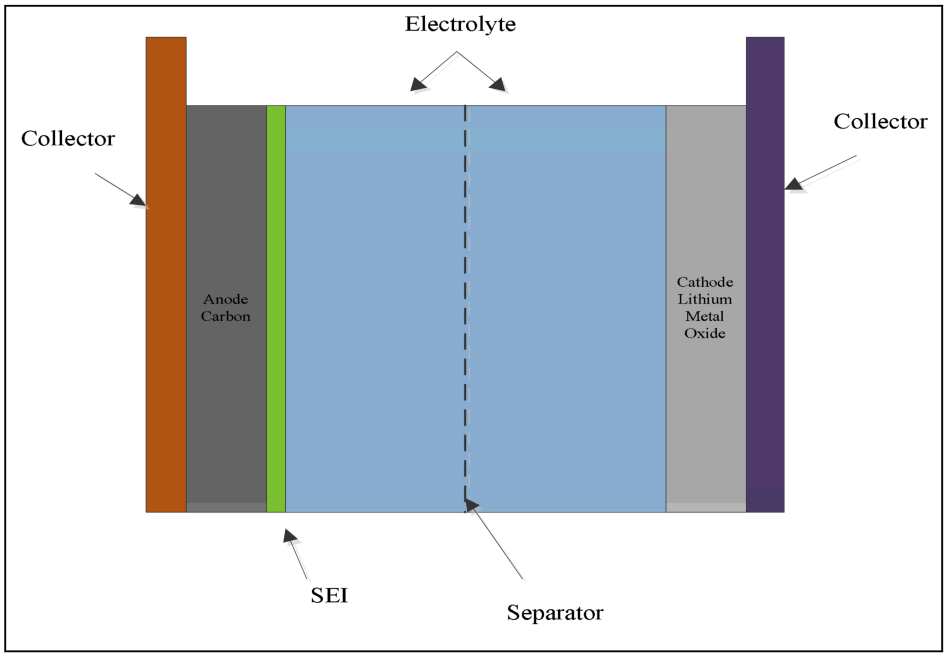


Figure 2.1: SEI on the Anode

tively illustrates battery aging state. With the obtained battery aging state, RUL of battery can be accurately estimated, as well as the very concerned data, like remaining mileages provided by electrical vehicles. An accurate estimation of battery SOH plays a crucial role in guaranteeing the reliability and security for devices or systems. One of main ideas of SOH estimation for single cell is that the RUL of battery is estimated combining historical monitoring or state data with battery model or empirical formulas of capacity fade. The commonly used SOH estimation methods are listed as follows:

- Data-based method [45, 46, 47, 48, 49]
  - **Strengths**: easily comprehensible with uncomplex mechanism.
  - **Weaknesses**: large amount of data, larger errors, poor applicability
- Feature-based method [50, 51]
  - **Strengths**: easily comprehensible with uncomplex mechanism.
  - **Weaknesses**: large amount of data, lack of physical significance of some feature parameters, larger errors
- Model-based method [52, 53]

- **Strengths**: precise description of degradation mechanism,
- **Weaknesses**: complicated mechanism, large amount of model parameters, difficult for on-line application

The former two methods can be categorized as data-based method with no need to understand battery internal mechanism. Poor prediction performances will occur if historical data is inadequate, or the actual working operation and aging condition is inconsistent. Based on a better understanding of battery internal mechanism, model parameters can be selectively used as indicators to assess battery SOH with the knowledge of their variations. Due to numerous model parameters and numerical coupling relations between parts of model parameters, much time is needed to thoroughly and quantitatively estimate them by intellectual algorithm, like genetic algorithm [54].

The SOH prediction methods mentioned above all have their own limits. As battery SOH is affected by numerous factors, like charge and discharge rate, ambient temperature, cycle times, and cut-off voltage, a well-performed SOH prediction method with accurate precision, wide applicability, and less computing time has not been reported before, which leaves room for challenging. [55]

2.2 Ultracapacitor

The ultracapacitor also known as a supercapacitor or electric double layer capacitor are large capacitance devices, with capacitances upto of several thousand farads. The first patent for a capacitor based on high surface area carbon dates back to 1957 [56]. In 1969 the SOHIO Corporation made the first attempt to commercialize ultracapacitors [57].

However it was not until the nineties that the interest in ultracapacitors was renewed in the context of hybrid electric vehicles. An ever increasing power requirement for automotive applications have rendered the standard battery design obsolete leading to the design of pulsed batteries and battery-ultracapacitor hybrid systems for high power applications [58, 59]. However with the increasing penetration of renewable energy technologies that require energy storage, the usage of ultracapacitors in these systems has also been investigated by few authors [60, 61].

Ultracapacitors are high power density devices. It can be seen from the Figure 1.2 that ultracapacitors fills the gap between batteries and conventional capacitors in

terms of specific energy and specific power and due to this it lends itself very well as a complementary device to the battery. By combining ultracapacitors with batteries, which are typically low power devices, the battery performance can be improved in terms of the power density. Besides the high power density, ultracapacitor also enjoys following superiorities [62]:

- Very High Efficiency

  Ultracapacitors are highly efficient components. Their coulombic efficiency (defined as the total charge removed divided by the total charge added to replenish the charge removed) is greater than 99%, even at very high currents, meaning that little charge is lost when charging and discharging the ultracapacitor. Round-trip efficiency is also very high, due to the low equivalent series resistance (ESR).

- High Current Capability

  Ultracapacitors are designed with a very low equivalent series resistance (ESR), allowing them to deliver and absorb very high current. The low ESR of ultracapacitors allows them to be charged very quickly, making them well suited for regenerative braking applications and other quick-charge scenarios. The inherent characteristics of the ultracapacitor allow it to be charged and discharged at the same rates, something no battery can tolerate. If the energy storage device needs to be quickly charged (in applications like regenerative braking and quick-charge toys), the ultracapacitor can be charged as quickly as the system will allow, within reasonable limits based only on simple resistive heating. In battery-based systems, systems designer can only charge as fast as the battery will accept the charge. This limits the system to only low to moderate charging rates, and may also limit how frequently one can charge, a significant issue in braking systems. Furthermore, the battery does not self-limit this charging rate, therefore the systems designer must manage this charging. In some cases, systems designer may need the extra energy he gets with a battery. In these cases, systems designer can combine an ultracapacitor and a battery to get the best of both, optimizing the system design. Examples include consumer electronics such as digital cameras, in which an inexpensive alkaline battery is combined with an ultracapacitor, and automotive applications such as hybrid power trains. In both examples, the high power pulses are provided by the ultracapacitor, while the large energy requirement is provided by the battery.

- Wide Voltage Range

  Ultracapacitators are not confined to a narrow voltage window. Designers need only consider the voltage range of the system, which can be much wider than the narrow voltage range required by a battery. The ultracapacitor can operate at any voltage below its maximum continuous operating voltage. To achieve higher voltages, multiple cells are placed in series, and are operated at or below their total series maximum voltage. There is no risk of over-discharging the ultracapacitor, and in fact there is additional safety for service personnel, who can fully discharge an ultracapacitor system before servicing, reducing the electrical hazard. In some systems such as fuel cells, the ability of the ultracapacitor to track with the fuel cell's voltage is a significant benefit over battery/fuel cell systems, where the fuel cell wants to operate over a voltage range that is wider than that tolerated by batteries.

- Wide Temperature Range

  Since ultracapacitors operate without relying on chemical reactions, they can operate over a wide range of temperatures. On the high side, they can operate up to 65°C, and withstand storage up to 85°C, without risk of thermal runaway. On the low side, they can deliver power (with slightly increased resistive losses) as cold as −40°C, well below the cold performance threshold of batteries. The excellent cold performance of ultracapacitors is an excellent fit for engine-starting applications. When combined with batteries, systems designer can implement a system that meets the energy requirements with a battery (such as powering lights and stereos while the engine is off) and the power requirements with the ultracapacitor (such as turning the engine over when it is cold, or when the battery may be discharged from powering lights and stereos while the engine is off).

- Condition Monitoring (SOC and SOH)

  Determining battery state of charge (SOC) and state of health (SOH) is a significant factor in designing robust battery systems, requiring sophisticated data acquisition, complex algorithms, and long-term data integration. In comparison, it is very simple to determine the SOC and SOH of ultracapacitors. Since the energy stored in a capacitor is a function only of capacitance and voltage, and the capacitance is constant (relatively speaking), a simple open-circuit volt-

age measurement defines state of charge. Since capacitance is relatively stable, voltage alone effectively determines SOC. Because of the relatively slow change in capacitance and equivalent series resistance over time, occasional calculations of capacitance and ESR can be used to determine SOH.

- Long Cycle Life

  The energy storage mechanism of an ultracapacitor is a highly reversible process. The process moves charge and ions only. It does not make or break chemical bonds. It therefore is capable of hundreds of thousands of complete cycles with minimal change in performance. Cycle depth is also not an issue, so ultracapacitors can be micro-cycled (cycled less than 5% of their total energy) or full cycled (cycled greater than 80% of their total energy) with the same long life. They can be cycled infrequently, such as in an uninterruptible power supply system where they may only be discharged a few times a year, or they may be cycled very frequently, as in a hybrid vehicle.

- Long Operational Life

  Since there are no chemical reactions, the energy storage mechanism of an ultracapacitor is a highly stable process. It is therefore capable of many years of continuous duty with minimal change in performance. Long-term storage is not an issue, since the ultracapacitor can (and should) be stored completely discharged. The long cycle life and long operational life make the ultracapacitor a lifetime component for most applications. Battery replacement is considered normal routine maintenance, costing time and money. In most cases, ultracapacitors are installed for the life of the system.

- Life Extension for Other Energy Sources

  Energy sources such as batteries, specialty engines, and fuel cells don't perform well in transient conditions. For some components, transients can significantly shorten life. Coupling an ultracapacitor with these energy sources off-loads many of these transients from the main energy source. The benefits are a smaller main energy source, and one that has potentially much longer life. The life cycle cost of the battery associated with an ultracapacitor-battery system may be much lower than that of a battery-only system.

- Ease of Maintenance

Ultracapacitors require basically no maintenance. They have no memory effects, cannot be over-discharged, and can be held at any voltage at or below their rating. If kept within their wide operating ranges of voltage and temperature, there is no recommended maintenance.

2.2.1 Principle of operation

The storage of electric charge and energy in an ultracapacitor is electrostatic i.e non-faradaic. An electrode when immersed in an electrolyte results in the formation of an electrochemical double layer at the solid/electrolyte interface. Ultracapacitors store the electric energy in this electro- chemical double layer also known as the Helmholtz Layer. The double layer capacitance is about $16 - 50\ \mu F/cm^2$ [63] for an electrode in concentrated electrolyte solution and the corresponding electric field in the electrochemical double layer is very high and assumes values of up to 106 V/cm [64]. In order to achieve a higher capacitance the electrode surface area is increased by using porous electrodes with an extremely large internal effective surface(1000 to 2000 m^2/g) [63]. A single cell of an ultracapacitor (shown in figure 2.2) consists of two electrodes immersed in an electrolyte. The electrodes in the system are separated by a porous separator containing the same electrolyte. The energy stored in an ultracapacitor is given by,

$$E = \frac{CV^2}{2} \tag{2.5}$$

where E is the energy stored, C is the capacitance and V is the voltage. However the calculation of capacitance for an ultracapacitor is very complex. For an ideal double-layer capacitor there should not be any faradaic reactions between the electrode and electrolyte. The capacitance for such a capacitor is independent of the voltage. Another mode of storage has also been utilized by the ultracapacitors that involves faradaic reactions. Capacitance in such cases is termed pseudo-capacitance. Charge transferred in such cases is voltage dependent subsequently leading the capacitance to also be voltage dependent.

2.2.2 Ultracapacitor cost considerations

It must be noted that ultracapacitors are currently not in high volume production and hence the costs can be prohibitive for implementation in some of the applications

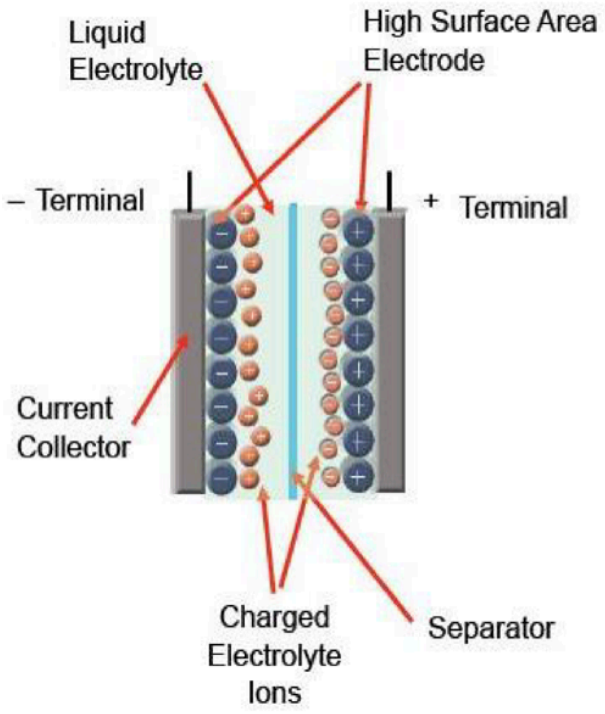


Figure 2.2: Ultracapacitor [65]

mentioned above. With a steady increase in demand for ultracapacitors, automating the production facilities is a way to reduce production costs. However the cost of manufacturing also depends on material costs which are currently high for ultracapacitors. The major material costs for double layer capacitors are the carbon, the organic electrolyte, and the salt added to the electrolyte to provide the ions. The cost of carbon, the material used for electrodes, can be as high as $\$100/kg$ with an average in the range of $\$30 - 50/kg$ [58].

Ultracapacitors can not compete with batteries in terms of $\$/Wh$, but they can compete in terms of $\$/kW$ and $\$/unit$ to satisfy a particular vehicle application.[66] Both energy storage technologies must provide the same power and cycle life and sufficient energy (Wh) for the application. The weight of the battery is usually set by the system power requirement and cycle life and not the minimum energy storage requirement. Satisfying only the minimum energy storage requirement would result in a much smaller, lighter battery than is needed to meet the other requirements. On the other hand, the weight of the ultracapacitor is determined by the minimum energy storage requirement. The power and cycle life requirements are usually easily satisfied. Hence the unit can be a more optimum solution for many applications and

its weight can be less than that of the battery even though its energy density is less than one-tenth that of the battery.

Currently, the $\$/W$ costs of the ultracapacitor unit are about one-fourth those of the batteries. The present price of ultracapacitors is in the range of $1-2 cents/F$ for small devices and $0.25-0.5 cents/F$ for large devices with automated production and reduced material costs, but with high volume production and increases in energy density, the price of ultracapacitors will continue to decrease.[66] In addition, high power batteries, being more expensive than high energy density lithium batteries, are likely priced at $\$1000/Wh$ or higher. Hence in the near future, it is likely that ultracapacitor energy storage units for hybrid vehicle applications can be cost competitive with lithium battery units.[66]

2.3 Control strategies for HESS

Passive hybrid topology is the simplest one and does not need a controller. However, as to semi-active and active hybrid topology, many variations of control strategies have been proposed. Among these strategies, rules or reference curves and tables based, fuzzy logic control and closed-loop control are the three most common control methods.

2.3.1 Rules and reference tables

The common method based on rules and reference tables is to calculate the total power demand first. Then use a set of rules to divide the power between battery and ultracapacitor. For example, in a given situation, all the power demand exceeding a threshold would be supplied by the ultracapacitor [67]. Another method can be found in [68], the ultracapacitor state of charge (SOC) is determined by the speed of the vehicle and the battery SOC. In this way, the ultracapacitor is discharged as the vehicle accelerates (and vice-versa), so that the peaks in power demand related to acceleration and braking is reduced. In [69] the different rules (for example, battery supplies power to the load and to recharge the ultracapacitor) are selected by the use of a flowchart that takes into consideration the state of charge of the sources and the load demand.

2.3.2 Fuzzy logic control

Fuzzy logic control does not demand a precise model of the plant because it is based on designer's knowledge on it, what is an important advantage when a model is not available. In [64], fuzzy logic control was used to the specific problem of controlling a hybrid energy storage system with good results. Fuzzy logic control can also be applied together with management methodology to the problem of controlling a battery/ultracapacitor HESS [70].

2.3.3 Closed-loop control

The traditional feedback control can also used to control HESS. In reference [66], two loops are used to control the current and voltage of the battery. The inner loop is the current loop and the outer loop is to control the voltage. In the outer loop, the load current and the battery converter output current are treated as perturbations. In reference [71], ultracapacitor semi-active hybrid topology is used and a filter is applied to generate reference signal for the control loop.

Chapter 3

HESS Topologies Comparison and Modeling Strategy

Parts of the work presented in this chapter was presented at the 2011 ASME International Mechanical Engineering Congress and Exposition:

Citation[19]: Z. Xing, D. Zuomin, and C. Curran, "Hybrid energy storage system for hybrid and electric vehicles: Review and a new control strategy.," in *ASME International Mechanical Engineering Congress and Exposition*, vol. 4, pp. 91–101, Sept 2011

In this chapter, the superiorities of a hybrid energy storage system are discussed first. Then three standard HESS topologies are compared by using the typical pulsed current load. Based on the comparison, the semi-active topology with an ideal DC-DC converter in the ultracapacitor end is selected as the HESS structure in this research work. Finally, a semi-active HESS model is built and simulated by using Matlab and Python. This model is then used to simulate the proposed intelligent energy allocation algorithm in Chapter 6.

3.1 Hybrid Energy Storage System

In order to improve energy storage system (ESS), the possibility of the integration of two (or more) energy sources should be researched, with the objective of utilizing the best characteristics of each, producing a hybrid energy storage system (HESS). Nowadays, hybridization of high-energy batteries with ultracapacitors is a common choice, since, from the analysis above, they have complementary characteristics that

make them attractive for a HESS. Combining batteries and ultracapacitors can create an energy storage system with both high peak power and high energy density. Besides, a HESS has following superiorities

- Higher Energy Efficiency

  "Delivering high power for a short period of time is deadly to batteries, but it is the ultracapacitor strongest suit [72]." The relationship of the discharge time and discharge current in a battery can be modeled by Peukert capacity.[73]

$$C_p = I^k \cdot T \tag{3.1}$$

  In which C_p is the Peukert capacity, I is discharge current, T is discharge time, and k is the Peukert coefficient, which is usually 1.1-1.3 for lead acid, and 1.05-1.2 for nickel metal hydride and lithium ion [74]. Battery delivers less charge (the integral of current) when discharged faster. Reference [75] shows that pulsed discharge profile results in increased cell temperature, considering the same average current. Because of the system losses, the efficiency for battery is lower when the frequency of the current is higher.

  Since the ultracapacitor is able to deliver or receive energy in peak power situations, if a battery is hybridized with an ultracapacitor, its demand would become closer to the average power demand. Therefore, the system losses are reduced and the efficiency is improved.

- Longer Battery Life

  As the battery cost is a significant part in the price of the whole car, the life of batteries is very important to customer acceptance of EVs. However, high charge or discharge rates shorten the battery life, even including high current-rate lithium-ion batteries [76, 77]. Reference [78] analyses the life reduction of cobalt-based lithium-ion cells for high charge or discharge current. As to a HESS, the batteries' better working condition created by ultracapacitor makes the battery life longer. Here additional remarks should be made that ultracapacitors have a very long life, significantly higher than batteries.

- Better Thermal Management

  Ultracapacitors can operate under a wider temperature range than batteries[79]. When used together, ultracapacitors can attenuate the reduction in the power

available from batteries in extreme temperature conditions. Furthermore, if the weather is so cold that batteries fail to work, ultracapacitors can supply energy to the thermal management system in a pure electric vehicle to preheat the batteries.

- Lower System Cost

  As discussed previously, the rapid-developing ultracapacitor technology allows achieving power density of several thousand W/kg at a relatively low cost. Some Li-ion polymer batteries reach the same power density, but at much higher prices. Therefore, HESS is a better choice than pure batteries to achieve high power density.

3.2 Three topologies of battery ultracapacitor hybrids

3.2.1 Typical pulsed current load

The majority of electric and hybrid vehicles possess certain load profile characteristics, described by relatively high peak-to-average power requirements. Such loads can be closely represented by a typical pulsed current load [80], which is illustrated in Figure 3.1.

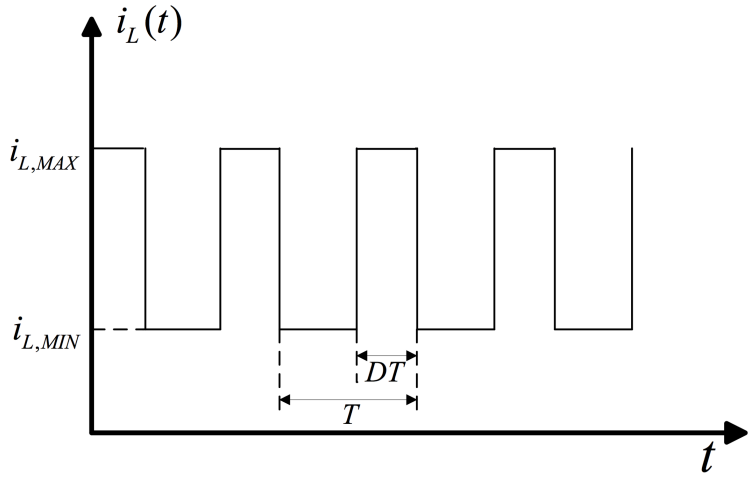


Figure 3.1: Pulsed Current Load

The consumption profile is characterized by a periodic rectangular pulse train,

alternating between two current levels, $i_{L,MIN}$ and $i_{L,MAX}$ with period T and duty cycle D,

$$i_L(t) = i_{L,MIN}u(t) + \sum_{k=0}^{N}(i_{L,MAX} - i_{L,MIN})[u(t - kT) - u(t - DT - kT)] \quad (3.2)$$

where $u(t)$ is a unit step function and N is the number of operation periods. Also, $i(t)$ can be written as:

$$i_L(t) = i_{L,AVE}(t) + i_{L,DYN}(t) \quad (3.3)$$

where, $i_{L,AVE}(t)$ is the average current and $i_{L,DYN}(t)$ is the dynamic part.

$$i_{L,AVE}(t) = \frac{1}{T}\int_0^T i_L(t)dt = D \cdot i_{L,MAX} + (1 - D) \cdot i_{L,MIN} = I_{L,AVE} \quad (3.4)$$

Furthermore, the load current can also be represented by Fourier series as:

$$i_L(t) = I_{L,AVE} + \sum_{n=1}^{\inf} I_{L,n} \cdot cos(n\frac{2\pi}{T}t + \phi_n(jn\frac{2\pi}{T})) \quad (3.5)$$

where the current harmonics magnitude is

$$i_{L,n} = \pi D(i_{L,MAX} - i_{L,MIN})[sin(n\pi D)] \quad (3.6)$$

and ϕ is the current harmonics phase.

3.2.2 Passive hybrid

Till now, the most common battery-ultracapacitor hybrid topology is the passive hybrid, which has been studied by many researchers [81, 82, 83, 84] and employed in commercial products [85, 86, 87]. In a passive topology, as shown in Figure 3.2, the batteries and ultracapacitors are connected in parallel with each other and the load. The advantages of passive topology are the simplicity and the absence of power electronics and control circuitries, which reduces the cost and increasing reliability. However, the main disadvantage is that the load current is distributed in a nearly uncontrolled manner between the battery and the ultracapacitor.

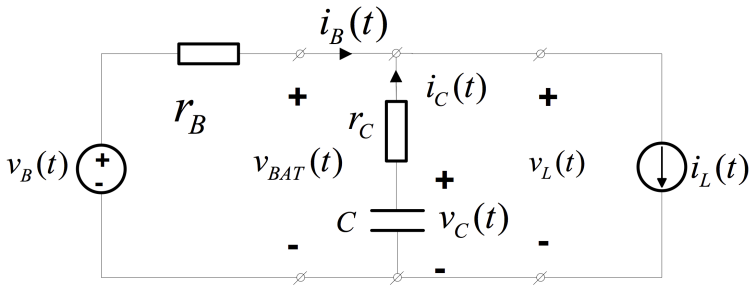


Figure 3.2: Passive HESS

The ultracapacitor is represented by the nominal capacitance C and the internal resistance r_C. r_B is the internal resistance of the battery. The load current is Typical Pulsed Current (TPC), which is represented by Equation 3.5.

From [88], we obtain the system power loss is:

$$P_{LOSS} = P_{LOSS,BAT} + P_{LOSS,C} = r_B I_{L,AVE}^2 + \frac{1}{2}\sum_n I_{L,n}^2 \cdot r_{P,n} \tag{3.7}$$

where,

$$r_{P,n} = r_B \mid H_C(jn\frac{2\pi}{T}) \mid^2 + r_C(1- \mid H_C(jn\frac{2\pi}{T}) \mid)^2 \tag{3.8}$$

and,

$$H_C(j\omega) = \frac{1 + j\omega C r_C}{1 + j\omega C(r_B + r_C)} = \mid H_C(j\omega) \mid e^{j\theta C^{(j\omega)}} = \sqrt{\frac{1 + (\omega C r_C)^2}{1 + (\omega C(r_B + r_C)^2}} e^{j\theta C^{(j\omega)}} \tag{3.9}$$

If only batteries drive the load without ultracapacitors, the system losses are:

$$P_{LOSS} = P_{LOSS,BAT} = r_B I_{BAT,RMS}^2 = (I_{L,AVE}^2 + \frac{1}{2}\sum_n I_{L,n}^2) \cdot r_B \tag{3.10}$$

Since both $\mid H_C(j\omega) \mid$ and $1- \mid H_C(j\omega) \mid$ are less than unity as well as $r_C \ll r_B$, it can be obtained from Equation 3.8 that $r_{P,n} < r_B$. Therefore, the system losses are reduced as a result of hybridization. According to [89, 81, 82, 83], battery current in time domain is:

$$\begin{aligned} i_B(t) =& i_{L,MIN} + (i_{L,MAX} - i_{L,MIN}) \times \sum_k ([1 - \frac{r_B}{r_B + r_C} e^{-\omega_B (t-kT)}] u(t - kT) \\ & - [1 - \frac{r_B}{r_B + r_C} e^{-\omega_B (t-DT-kT)}] u(t - DT - kT)) \end{aligned} \tag{3.11}$$

where $\omega_B = \frac{1}{(r_B + r_C)C}$

From Equation 3.11, conclusion can be drawn that during the high load demand, both the battery and the ultracapacitor supply charge to the load. During the low load demand, the battery supplies both the load and the capacitor. Furthermore, battery current ripple reduces, and battery terminal voltage dips become lower than in the battery-only case. Hence, the battery is more efficiently utilized. Increasing the capacitance will force the maximum and minimum values of the battery currents to become closer to each other. The discharge curve of a passive hybrid converges towards the discharge at $I_{L,AVE}$ curve as the capacitance is increased. As a result, either more energy can be drawn from the same battery or a battery with lower rating can be utilized. However, a negative byproduct of capacitance increase by connecting capacitors in parallel is weight/volume/price increase. On the other hand, the internal resistance of the capacitor pack is decreased, and as a result the losses are decreased. If one of the negative consequences of capacitance increase cannot be tolerated, semi-active or fully active hybrid should be considered. In addition, there is a trade-off between the allowed load voltage ripple and capacitor utilization [88].

3.2.3 Semi-active hybrid

If a DC–DC converter is employed to the battery and ultracapacitor banks, a semi-active hybrid is composed [90, 91, 92, 93]. The semi-active topology enhances the performance of the passive hybrid at the price of an additional DC–DC converter and control circuitry. Three different ways of creating a semi-active hybrid are considered, battery semi-active, capacitor semi-active and load semi-active

- Battery Semi-active Hybrid

  In this topology, the DC-DC converter is connected between the battery and the load, as illustrated in Figure 3.3 [94, 95, 96]. The output current of the DC-DC converter is controlled to follow the current $I_{L,AVE}$. The main advantage of such a topology is the ability to control the battery current at a near constant value

despite the load current variations. This allows significant battery performance improving in lifetime, energy efficiency and operating temperature. In addition, voltage matching between the battery and the load is no longer required.

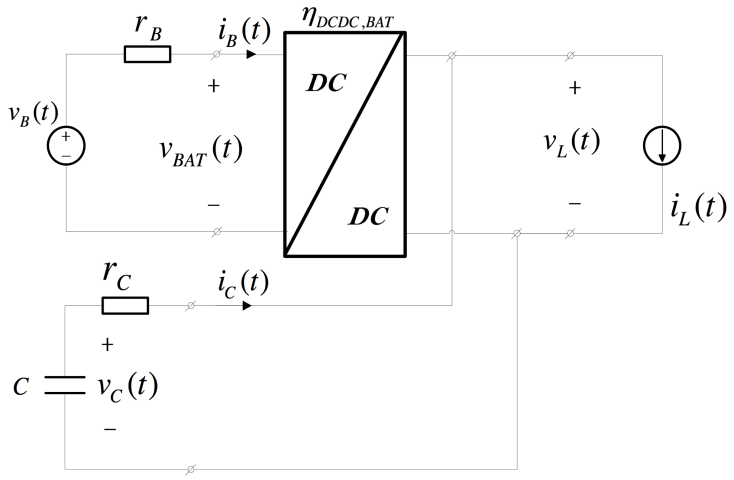


Figure 3.3: Battery semi-active hybrid topology

The batter voltage and current are

$$v_{BAT} = \frac{v_L}{K_{BAT}(t)}, \quad i_B = K_{BAT}(t) \cdot \frac{I_{L,AVE}}{\eta_{DCDC,BAT}} \tag{3.12}$$

where $K_{BAT}(t)$ and $\eta_{DCDC,BAT}$ are the battery converter voltage conversion ratio and efficiency, the system losses are

$$P_{LOSS} = P_{LOSS,BAT} + P_{LOSS,C} = r_B(K_{BAT}(t) \cdot \frac{I_{L,AVE}}{\eta_{DCDC,BAT}})^2 + \frac{1}{2} r_C \sum_n I_{L,n}^2 \tag{3.13}$$

The main disadvantage of the topology is the variations of the load voltage during capacitor charging/discharging.

- Capacitor Semi-active Hybrid

  In the capacitor semi-active topology, a DC-DC converter is located between the capacitor and the load, as shown in Figure 3.4 [97, 98, 99, 100, 101]. Such a topology allows controlling of the capacitor current.

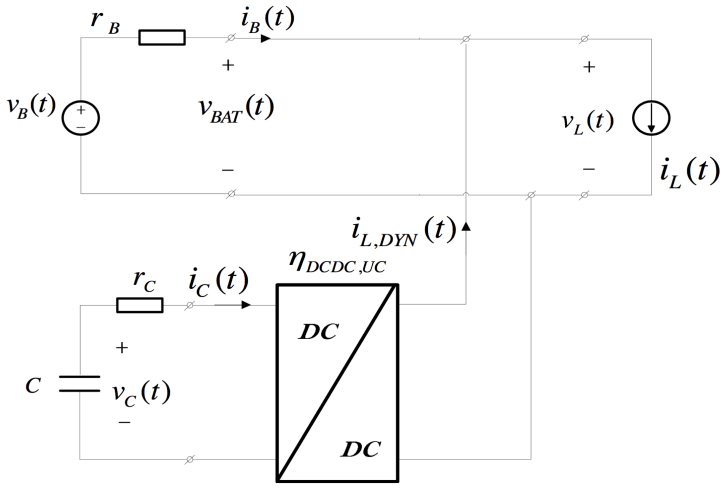


Figure 3.4: Capacitor semi-active hybrid topology

As a result of decoupling between the ultracapacitor and the battery, the utilization of the ultracapacitor energy is improved. If the DC- DC converter output current is controlled to follow the dynamic part of the load current $i_{L,DYN}(t)$, current of the battery and the ultracapacitor are:

$$i_B = i_L - i_{L,DYN} = i_{L.AVE}, \quad i_C = K_{UC}(t) \cdot \frac{i_{L,DYN}}{\eta_{DCDC,UC}} \tag{3.14}$$

where $K_{UC}(t)$ and $\eta_{DCDC,UC}$ are the ultracapacitor converter voltage conversion ratio and efficiency, respectively. Therefore, the system losses are formulated as

$$P_{LOSS} = P_{LOSS,BAT} + P_{LOSS,C} = r_B(I_{L,AVE}^2 + \frac{1}{2} r_C \sum_n (K_{UC}(t) \cdot \frac{I_{L,n}}{\eta_{DCDC,UC}})^2 \tag{3.15}$$

Note that the in the capacitor semi-active configuration, the load voltage possesses no ripple (since a nearly constant current is drawn from the battery) but is unregulated, decreasing as the battery is depleted according to the battery discharge curve at $I_{L,AVE}$.

- Load Semi-active Hybrid

  As to a load semi-active configuration, a DC-DC converter is placed between the load and electrical sources (parallel branch of battery/ ultracapacitor), as

shown in Figure 3.5. [102, 103, 104, 105] This topology is developed from the passive hybrid topology. It allows a mismatch between the battery voltage (and hence the ultracapacitor voltage rating) and the load. According to Figure 6

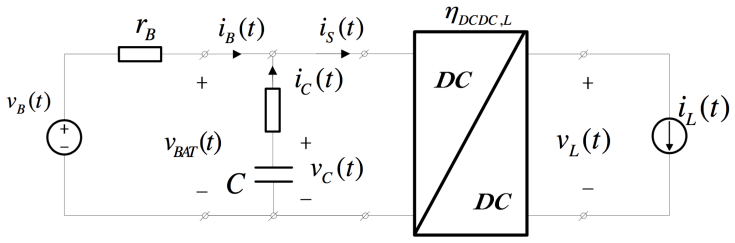


Figure 3.5: Load semi-active hybrid topology

$$v_L = K_L(t) \cdot v_{BAT}, \quad i_S = K_L(t) \cdot \frac{i_L}{\eta_{DCDC,L}} \tag{3.16}$$

Where, i_S is the current supplied by the battery ultracapacitor parallel branch to the DC-DC converter inputs. Substituting (16) in to (5),

$$i_S(t) = K_L(t) \cdot \frac{i_{L,AVE}}{\eta_{DCDC,L}} + \sum_n K_L(t) \cdot \frac{i_{L,n}}{\eta_{DCDC}} cos(n\frac{2\pi}{T}t + \phi_n(jn\frac{2\pi}{T})) \tag{3.17}$$

Furthermore, the current of battery and ultracapacitor is:

$$I_{B,n}(t) = K_L(t) \cdot \frac{I_{L,n}}{\eta_{DCDC,L}} \cdot \mid H_C(jn\frac{2\pi}{T}) \mid, \quad I_{C,n}(t) = K_L(t) \cdot \frac{I_{L,n}}{\eta_{DCDC,L}} \cdot \mid 1 - H_C(jn\frac{2\pi}{T}) \mid \tag{3.18}$$

And the system losses are

$$P_{LOSS} = P_{LOSS,BAT} + P_{LOSS,C} = (\frac{K_L(t)}{\eta_{DCDC,L}})^2 \times \{r_B I_{L,AVE}^2 + \frac{1}{2}\sum_n I_{L,n}^2 \cdot r_{P,n}\} \tag{3.19}$$

However, it does not change the fact that the battery supplies part of the dynamic current and the ultracapacitor available charge is still limited.

3.2.4 Active hybrid

As to an active hybrid, two DC–DC converters are employed [101]. There are three topologies: battery series active, capacitor series active and parallel active. The fully active hybrid brings the system to an ultimate performance; however, as a result, the control complexity is increased.

- Battery Series Active Hybrid

This topology is an enhancement of the battery semi-active hybrid, as illustrated in Figure 3.6. It solves the disadvantages of ultracapacitor voltage variations and matching by placing an additional DC-DC converter between the ultracapacitor and the load. However, it comes at the price of an extra full rating DC-DC converter and the reduced efficiency, since there are two conversion stages between the battery and the load.

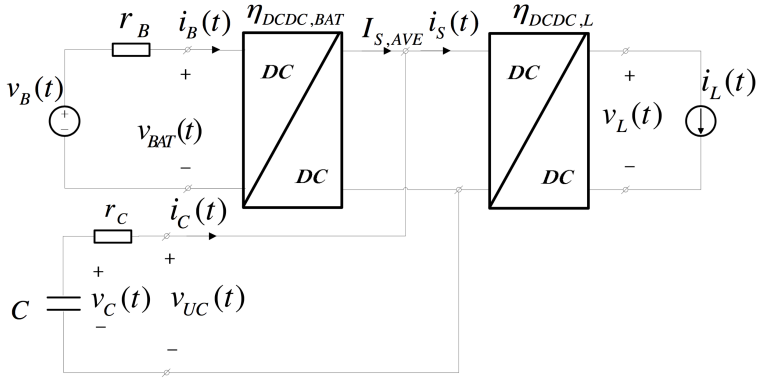


Figure 3.6: Battery series active hybrid

The system losses are:

$$P_{LOSS} = P_{LOSS,BAT} + P_{LOSS,C} = r_B(K_{BAT}(t) \cdot K_L(t) \cdot \frac{I_{L,AVE}}{\eta_{DCDC,BAT} \cdot \eta_{DCDC,L}})^2 + \frac{1}{2} r_C \sum_n (K_L(t) \cdot \frac{I_{L,n}}{\eta_{DCDC,L}})^2 \tag{3.20}$$

- Capacitor Series Active Hybrid

This topology is an enhancement of the capacitor semi-active hybrid, as shown in Figure 3.7. It solves the disadvantages of battery voltage reduction and matching by

placing an additional DC-DC converter between the battery and the load. However, it again comes at the price of an extra full rating DC–DC converter and the reduced efficiency, since there are two conversion stages between the ultracapacitor and the load.

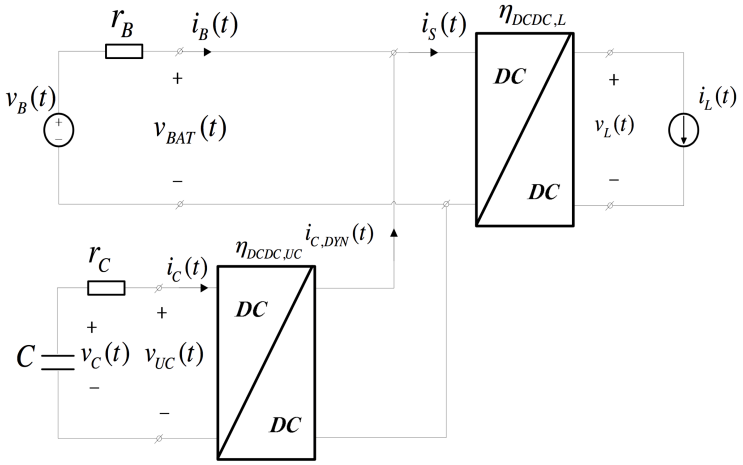


Figure 3.7: Ultracapacitor series active hybrid topology

The losses of the topology are given by

$$P_{LOSS} = P_{LOSS,BAT} + P_{LOSS,C} = r_B(K_L(t) \cdot \frac{I_{L,AVE}}{\eta_{DCDC,L}})^2 + \frac{1}{2} r_C \sum_n (K_{UC}(t) \cdot K_L(t) \cdot \frac{I_{L,n}}{\eta_{DCDC,UC} \cdot \eta_{DCDC,L}})^2 \tag{3.21}$$

- Parallel Active Hybrid

This topology is the optimal active hybrid. It solves the disadvantages of ultracapacitor voltage variations, achieves a nearly constant current flow from the battery and voltage mismatch between the battery and the load by placing two DC-DC converters. The topology, which combines the advantages of battery and ultracapacitor semi-hybrids, is shown in Figure 3.8.

The system losses are

$$P_{LOSS} = P_{LOSS,BAT} + P_{LOSS,C} = r_B(K_{BAT}(t) \cdot \frac{I_{L,AVE}}{\eta_{DCDC,BAT}})^2 + \frac{1}{2} r_C \sum_n (K_{UC}(t) \frac{I_{L,n}}{\eta_{DCDC,UC}})^2 \tag{3.22}$$

The losses are the lowest among the three active hybrid topologies.

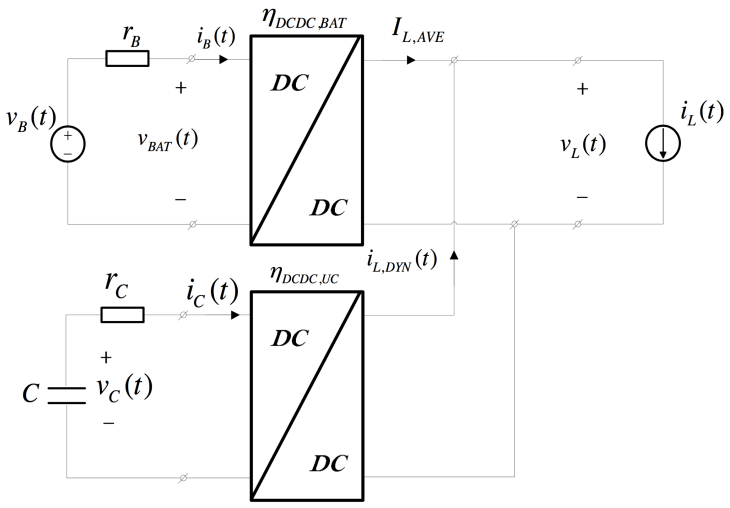


Figure 3.8: Parallel active hybrid topology

3.2.5 Conclusions for topologies of battery-ultracapacitor hybrids

The passive hybrid is the most simple and cheap topology, however, the utilization of battery and ultracapacitor is relatively lower and the load regulation is not good enough. On the other hand, the fully active hybrid attains the best performance, however, the cost is high and the control strategy is complicated. The semi-active hybrid might be a good trade-off between the performance and the circuit complexity and price.

3.3 Matlab-Python joint simulation framework

As we know, there exist a lot of useful tools in Python, such as the TensorFlow by Google, facilitating the machine learning research. On the other hand, Matlab is super powerful in simulating vehicle and power electronics models. In order to enjoy the superiorities of both Python and Matlab, we propose a novel Matlab(Autonomie)-Python joint simulation framework in this work. The vehicle model is created in Autonomie and simulated in Matlab. The HESS model and the proposed IEA algorithm is developed in Python. Details about this joint framework can be found in Figure 3.9. The Matlab model and Python model are connected by vehicle electrical load data in *json* format. In fact, these models are not executed simultaneously. The vehicle model in Matlab is firstly run, and the current load data is generated and

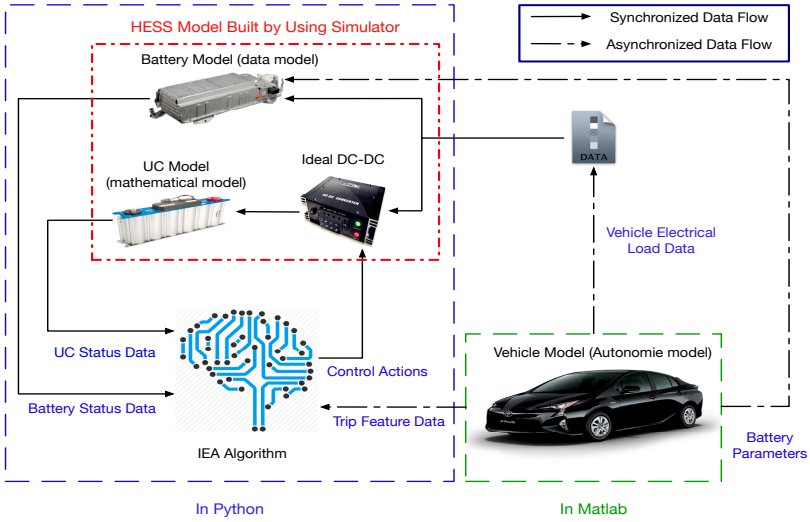


Figure 3.9: Matlab(Autonomie)-Python joint simulation framework

saved into a *json* file. Then python loads those data from the *json* file and executes the HESS model.

In order to build the HESS model in Python, a simulation package, called Simulator is developed in Python. More details about this package and the simulation platform hosting can be found in the Appendix B.

3.4 HESS modeling

This work is financially supported by Canadian Natural Resources and Applied Sciences Endowment Fund. The vehicle model was exported into Matlab/Simulink from Autonomie, which is a simulation software provided by the Argonne National Laboratory. In Autonomie, 2004 Prius is the only vehicle model built and validated from the real car by the Autonomie group.[106] Since Toyota does not open all parameters of its cars for modeling, 2004 Prius is our only option.

The battery in the HESS is exactly the same battery in the Prius, while the ultracapacitor is the Maxwell HTM Power Series 390v BOOSTCAP Ultracapacitor Modules. More detailed parameters about these two devices can be found in the following sections.

3.4.1 Battery model

As introduced in the previous section, the model in Autonomie is a data-driven model. Detailed battery parameters can be found in Table. 3.1.

Table 3.1: Pruis battery parameters

Parallel module	1	Series module	28
Cells in each module (series)	6	Cell nominal voltage	$1.2V$
Cell max/min voltage	$0.95V/1.55V$	Cell mass	$0.17kg$
Module mass	$1.02kg$	Pack mass	$28.56kg$
Packaging mass	$7.14kg$	Module heat capacity	$521J/K$
Max charging current	$-170.8864A$	Max discharging current	$184.5218A$
Capacity	$6.5Ah$	Pack Nornimal Voltage	$201.6V$
Power Density	$1.3kW/kg$	Energy Density	$46Wh/kg$

Note that, the battery internal resistor is not constant, it changes with the battery SOC. In Figure3.10, relation between the resistor and SOC can be found. Moreover, Toyota provides a 8-year/100,000 miles warranty for this battery and the replacement cost of this one is $\$3,939$. [107]

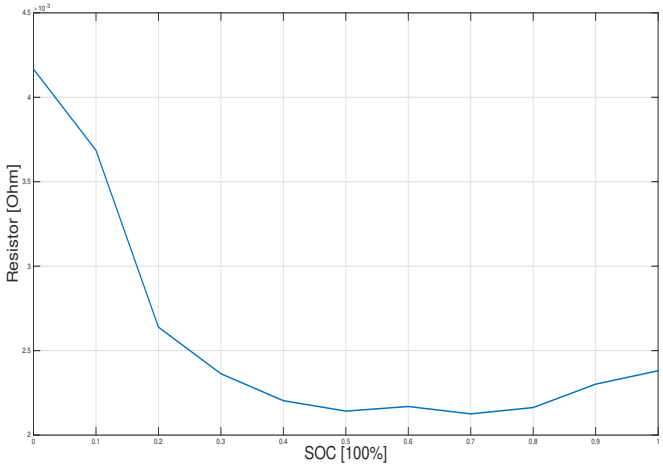


Figure 3.10: Internal Resisotr - SOC relation

In this work, we are trying to rebuild the Prius battery model into a mathematically model in Python. According to [108], the Rint battery model is shown in Figure 3.11

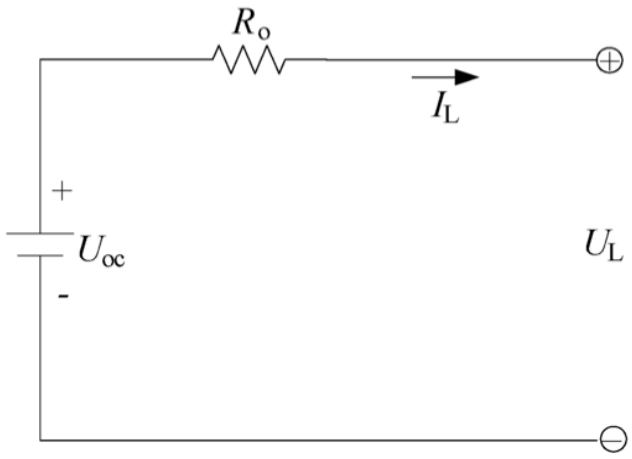


Figure 3.11: Battery Model

Note that the voltage source U_{oc} is a controlled-source, which is controlled by the battery SOC. The relation between the SOC and the cell voltage exported from Autonomie is plotted. Given a current load, the system output can be calculated as follows:

$$U_L = U_{oc}(SOC) - I_L R_o \tag{3.23}$$

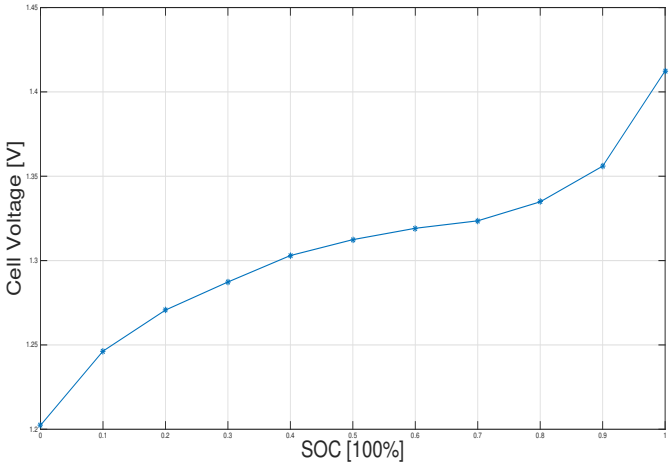


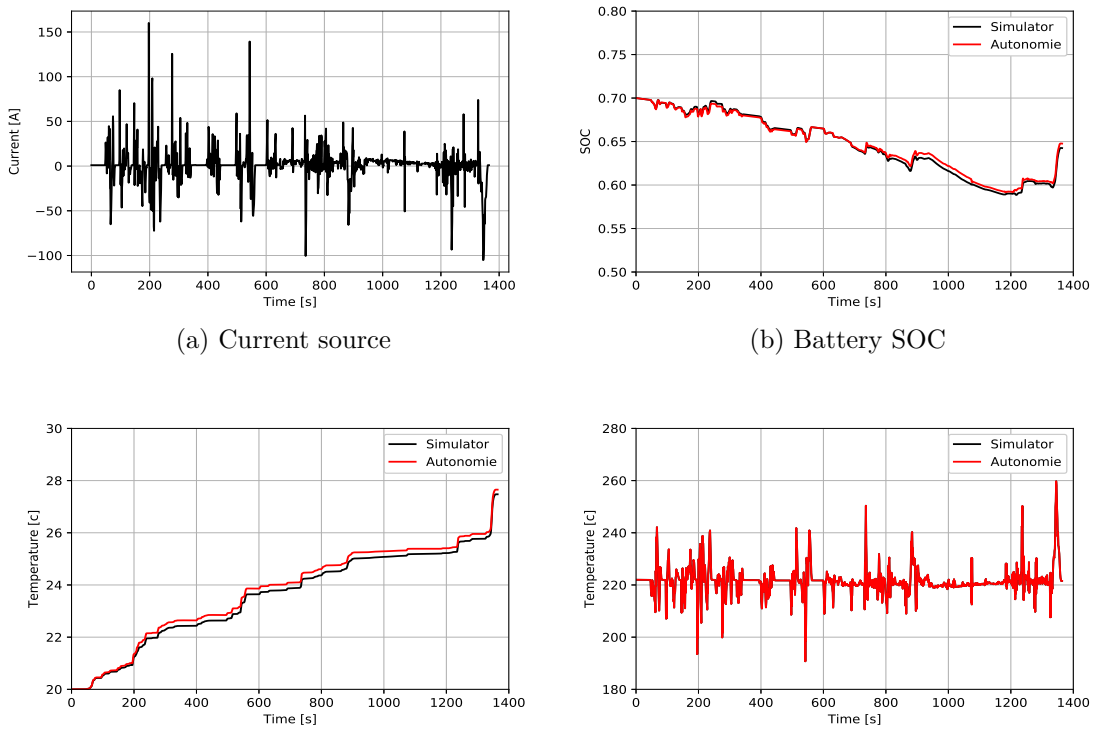
Figure 3.12: OCV - SOC relation

where $U_{oc}(SOC)$ is a function of battery SOC and battery SOC can be calculated by using Equation 2.1. Moreover, we also add the battery thermal performance simulation in the resistor R_o. For each module in the battery pack, the temperature

changing can be estimated by Equation 3.24.

$$\Delta Temp = \frac{I_L^2 R_o N_{cell}}{C_{mod} \cdot m_{mod}} \tag{3.24}$$

where, N_{cell} is the cell number in a module, C_{mod} is the module heat capacity and m_{mod} is the module mass. The battery model is then compared with the model in Autonomie. The results can be checked in the following plots.



(a) Current source

(b) Battery SOC

(c) Battery temperature

(d) Battery voltage

Figure 3.13: Simulation results compare: battery model

3.4.2 Ultracapacitor model

The ultracapacitor used in this work is the Maxwell HTM Power Series 390v BOOSTCAP Ultracapacitor Module. Its detailed parameters can be found in Table 3.2. The cost of this Module is $3332.

The one-order ultracapacitor model (See Figure 3.14) is built in Python. Then experiment is conducted to test the model. In the experiment, the ultracapacitor is

Table 3.2: Maxwell ultracapacitor parameters

Nornimal Voltage	$390V$	Maximal Voltage	$394V$
Surge Voltage	$406V$	Nominal Capacitance	$17.8F$
Tolerance Capacitance	$+20\%/-0\%$	Resistance	$65m\Omega$
Energy	$282Whkg$	Maximal Continuous/Pulse Current	$150A/950A$

charged from 0 SOC by a current load. The results are compared with the model in Matlab and plotted as follows.

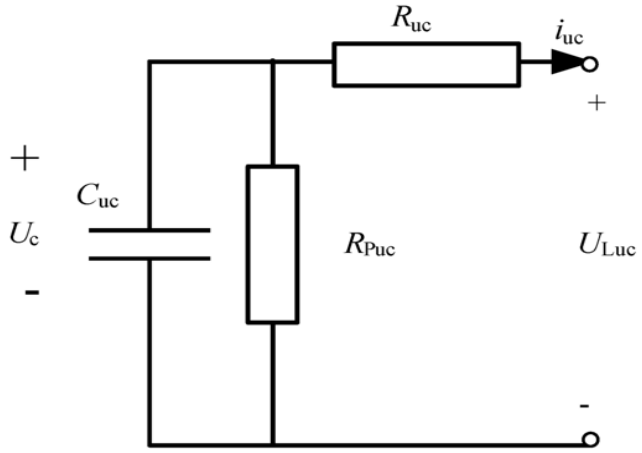


Figure 3.14: One order ultracapacitor model

Note that, since we are using the one-order ultracapacitor model and in order to avoid solving ordinary differential equations (ODEs), we convert the ODE into discrete integral equations. For example, for a passive HESS shown in Figure 3.2, when $v_B(t) > v_C(t)$, we have:

$$v_B(t) - [(i_L(t) + i_C(t)] \cdot r_B = i_C(t) \cdot r_C + v_C(t) \tag{3.25}$$

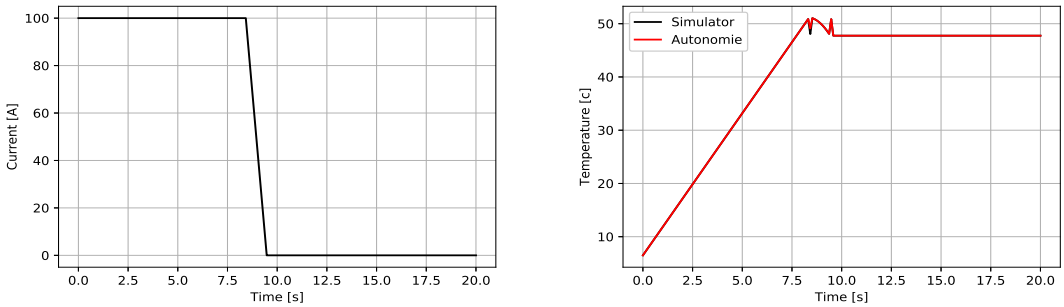
And

$$i_C(t) = \frac{d[v_C(t)]}{dt} \tag{3.26}$$

So we have

$$C \cdot \frac{d[v_C(t)]}{dt} = \frac{v_B(t) - v_C(t) - i_L(t) \cdot r_B}{r_B + r_C} \tag{3.27}$$

We convert this ODE to a discrete integral equation and use Trapezoidal Rule to



(a) Current source (b) Voltage

Figure 3.15: Simulation results compare: ultracapacitor model

solve it. Then we have:

$$i_C(k) = \frac{v_B(k) - v_C(k-1) - \frac{T}{2C} \cdot i_C(k-1) - i_L(k) \cdot r_B}{r_B + r_C + \frac{T}{2C}}, \quad \text{when} \quad k > 0 \qquad (3.28)$$

Chapter 4

Powertrain Feature Selection Based on Extended Support Vector Machine

An early version of the work presented in this chapter was presented on the Journal of the Franklin Institute in 2015:

Citation[20]: X. Zhang, G. Wu, Z. Dong, and C. Crawford, "Embedded feature-selection support vector machine for driving pattern recognition," *Journal of the Franklin Institute*, vol. 352, no. 2, pp. 669 – 685, 2015

As mentioned in Chapter 1, the proposed IEA algorithm needs the recognized driving patterns as input to optimize the energy flow between battery and ultrocapacitor in HESS. Therefore, this chapter and Chapter 5 will focus on the topics related to the driving pattern recognition. This chapter focuses on selecting significant features for driving pattern recognition, while next chapter focuses on recognizing driving pattern in a much shorter time period.

The extended Support Vector Machine (SVM) with embedded feature selection ability is proposed in this chapter. Besides statistical significance, this proposed SVM also takes into account the accessibility and reliability of features during feature selection. This enables the driving pattern recognition system to achieve higher recognition efficiency and robustness, hence the better performance of the IEA algorithm. Note that the algorithm proposed in this chapter is not related to the one in Chapter 5. These two algorithms are independent to each other and dealing with different concerns in driving pattern recognition.

4.1 Introduction

To effectively recognize and distinguish different driving conditions during vehicle operation, representation of different driving conditions by using several key parameters (features) is essential. In HEV applications, a number of state variables of the vehicle powertrain can be regarded as features for driving condition recognition. In order to choose statistically significant features for recognition, research into feature extraction and selection for the pattern recognition in vehicle applications is necessary. Higher recognition efficiency, less time in feature data collection and higher accuracy in classification, can be achieved by using a smaller number of features with higher significance in driving condition recognition. A lot of work has been done in this area. Ericsson has successfully reduced 62 different features to 16 independent driving condition features using Factorial Analysis [109]. In [110], four significant features were found and compared with the features extracted or selected by other common methods. All current work has focused only on selecting statistically significant features [111].

However, the accessibility and reliability of features used to describe vehicle driving conditions are normally quite different. In this work, 'Observability' is introduced to measure the accessibility and reliability of features. Observability of a feature means the difficulty in sampling/collecting the signal/data of this feature. For instance, it is easier to acquire instant vehicle speed using angular velocity sensors mounted on the driving shaft precisely, than to acquire the instantaneous engine power data. So the vehicle speed is therefore has better observability compared to engine power. Moreover, observability here also contains the concept of the reliability of obtained feature data. For instance, the real-time data on the State of Charge (SOC) of the HEV/PHEV's Energy Storage System (ESS) is typically estimated onboard by Kalman filters [112, 113, 114]. So SOC can be regarded as a feature with worse observability compared to vehicle speed. The concept of observability is therefore an important additional concept to statistical significance. However, in past and present research, the observability of data for the chosen features have never been taken into account. All feature data are treated equally, leading to less robust and efficient driving pattern recognition systems. In this work, an extended Support Vector Machine (SVM) with the capability for pattern discrimination and feature selection is developed. This extended SVM can choose features based on both the significance and observability of features.

4.2 Feature selection for pattern recognition

Two important factors in driving condition recognition are the feature selection, and the classification algorithm. The study in [109] aims at finding independent features to describe the dimensions of urban driving conditions and to investigate which properties have main effect on emissions and fuel consumption. In [110], common methods for feature extraction, the Fast Fourier Transform (FFT), the Discrete Cosine Transform (DCT), and the Principal Component Analysis (PCA), were analyzed and compared against the four features found in this work. A detailed summary of features used to recognize driving cycles in the past several years can be found in [115].

In pattern recognition scenarios, Feature Subset Selection (FSS) typically picks out several significant features from the original n-dimension inputs (features), in order to facilitate data collection, reduce storage space and classification time and improve the accuracy of classification [21]. FSS mainly requires two parts, a feature search strategy to select candidate feature subsets and an objective function to evaluate the quality of the subset candidates.

Feature selection approaches can be divided into three categories: filters, wrappers and embedded approaches [21]. Filters are the most known approaches, acting as a preprocessing step independently of the final classifier [116]. Filters usually evaluate feature subsets by their information content, typically using interclass distance, statistical dependence or information-theoretic measures. Filters generally involve a non-iterative computation on the original n-dimension features, which can execute very fast. However, since the filter objective functions are generally monotonic, the filter tends to select the full feature set as the optimal solution. In contrast, wrappers generally achieve better results than filters since they are tuned to the specific interactions between the classifier and the original feature set. In fact, wrappers take the classifier into account as a black box [117]. However, since the wrapper must train a classifier for each feature subset (or several classifiers if cross-validation is used), the method can become unfeasible for computationally intensive methods. Finally, unlike filters and wrappers, embedded approaches simultaneously determine features and classifier during the training process [21]. The embedded methods in [118] are based on a linear classifier. Another embedded approach for SVM was suggested in [119]. More detailed discussion about the embedded approached for SVM can be found in the following section.

4.3 Support Vector Machine with embedded feature selection

Support Vector Machines (SVM), proposed by Vapnik and his group at AT&T Bell Laboratories, are among the best "off-the-shelf" supervised learning algorithms. Given a two-class data set:

$$\boldsymbol{T} = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_p, y_p)\} \tag{4.1}$$

where, $\boldsymbol{x}_i \in \boldsymbol{R}^n, i = 1, 2 \cdots p$ are n-dimension feature vectors, $y_i \in \{-1, 1\}, i = 1, 2, \cdots, p$ are class levels. Then the linear classifier with maximum margin, $y_i = sgn(\boldsymbol{\omega}^T \cdot \boldsymbol{x}_i + b), i = 1, 2, \cdots, p$, can be found by solving the optimization problem as follows:

$$\begin{aligned} \min_{\boldsymbol{\omega}, b} \quad & \frac{1}{2}\|\boldsymbol{\omega}\|_2^2 \\ \text{s.t.} \quad & y_i \cdot (\boldsymbol{\omega}^T \cdot \boldsymbol{x}_i + b) \geq 1, i = 1, 2, \ldots, p \end{aligned} \tag{4.2}$$

To make the algorithm work for non-linearly separable data sets as well as be less sensitive to outliers, the above optimization problem can be stated (using $l1$ regularization) as follows:

$$\begin{aligned} \min_{\boldsymbol{\omega}, b} \quad & \frac{1}{2}\|\boldsymbol{\omega}\|_2^2 + C \cdot \sum_{i=1}^{p} \xi_i \\ \text{s.t.} \quad & y_i \cdot (\boldsymbol{\omega}^T \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i, i = 1, 2, \ldots, p \\ & \xi_i \geq 0, i = 1, 2, \ldots, p \end{aligned} \tag{4.3}$$

The parameter C controls the relative weighting between the twin goals of making the $\|\boldsymbol{\omega}\|_2^2$ small and of ensuring that most training data have functional margin at least 1. After solving this optimization problem, the optimal $\boldsymbol{\omega}^\star, b^\star$ can be found. For a new data point $\hat{\boldsymbol{x}}$, which is not in the training data set and is called testing data in this work, its group level $\hat{y}$ can be determined by just plugging this data point $\hat{\boldsymbol{x}}$ into $\hat{y} = sgn(\boldsymbol{\omega}^{\star T} \cdot \hat{\boldsymbol{x}} + b^\star)$, where $sgn(\star)$ is a sign function, defined as:

$$sgn(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases} \tag{4.4}$$

In this work, the optimization problem (4.3) is called the standard 2-norm SVM

(with soft margin). Furthermore, according to [120], it (4.3) can be also stated as:

$$\{\boldsymbol{\omega}^\star, b^\star\} = arg\min_{\boldsymbol{\omega},b} \sum_{i=1}^{p} [1 - y_i \cdot (\boldsymbol{\omega}^T \cdot \boldsymbol{x}_i + b)]_+ + \lambda \|\boldsymbol{\omega}\|_2^2, i = 1, 2, \ldots, p \tag{4.5}$$

where the subscript "+" means the positive part ($z_+ = max(z, 0)$).

In 1998, a linear SVM with embedded feature selection was firstly proposed in [118] by measuring the margin distance using different vector norms. In 2004, another SVM with embedded feature selection was developed in [121]. This SVM was originally stated as

$$\begin{aligned} \min_{\boldsymbol{\omega},b} \quad & \|\boldsymbol{\omega}\|_1 \\ \text{s.t.} \quad & y_i \cdot (\boldsymbol{\omega}^T \cdot \boldsymbol{x}_i + b) \geq 1, i = 1, 2, \ldots, p \end{aligned} \tag{4.6}$$

Then in order to deal with the cases of non-linearly separable data sets, this optimization problem was manipulated into the form as follows:

$$\begin{aligned} \min_{\boldsymbol{\omega},b} \quad & \|\boldsymbol{\omega}\|_1 \\ \text{s.t.} \quad & Prob(\boldsymbol{x}_1 \in \mathcal{H}_1) \geq \eta \\ & Prob(\boldsymbol{x}_2 \in \mathcal{H}_2) \geq \eta \\ & \boldsymbol{x}_1 \sim (\mu_1, \sigma_1), \boldsymbol{x}_2 \sim (\mu_2, \sigma_2) \end{aligned} \tag{4.7}$$

where, $\boldsymbol{x}_i \sim (\mu_i, \sigma_i)$ denotes a class of distributions that have mean μ_i, and covariance σ_i, The discriminating hyperplane of the SVM tries to place class 1 in the half space $\mathcal{H}_1(\boldsymbol{\omega}, b) = \{\boldsymbol{x} | \boldsymbol{\omega}^T \cdot \boldsymbol{x} > -b\}$ and class 2 in the other half space $\mathcal{H}_2(\boldsymbol{\omega}, b) = \{\boldsymbol{x} | \boldsymbol{\omega}^T \cdot \boldsymbol{x} < -b\}$. To ensure this, proper $\{\boldsymbol{\omega}, b\}$ has to be found, such that $Prob(\boldsymbol{x}_1 \in \mathcal{H}_1)$ and $Prob(\boldsymbol{x}_2 \in \mathcal{H}_2)$ are both high. Finally, this optimization problem can be posed as a second-order cone programming (SOCP) by using a Chebyshev-Cantelli inequality and then solved. In the same year, [119] proposed the 1-norm SVM by replacing the 2-norm penalty in Equation 4.5 with the 1-norm penalty.

$$\{\boldsymbol{\omega}^\star, b^\star\} = arg\min_{\boldsymbol{\omega},b} \sum_{i=1}^{p} [1 - y_i \cdot (\boldsymbol{\omega}^T \cdot \boldsymbol{x}_i + b)]_+ + \lambda \|\boldsymbol{\omega}\|_1, i = 1, 2, \ldots, p \tag{4.8}$$

In spite of the lack of oracle property [22], 1-norm SVM has both good performance in feature selection and classification. In the next year, [21] summarized former works and used the Difference of Convex functions Algorithm (DCA) to solve the l_2-l_0-SVM,

quadratic FSV and the kernel-target alignment approach. In 2007, an improved 1-norm SVM was proposed in [122]. In 2010, [123] introduced a 0-1 vector into the classification model of SVM

$$\begin{aligned} \min_{\boldsymbol{\omega},b} \quad & \frac{1}{2}\|\boldsymbol{\omega}\|_2^2 + C \cdot \sum_{i=1}^{p} \xi_i \\ \text{s.t.} \quad & y_i \cdot (\boldsymbol{\omega}^T \cdot \phi(\boldsymbol{z} * \boldsymbol{x}_i) + b) \geq 1 - \xi_i, i = 1, 2, \ldots, p \\ & \xi_i \geq 0, i = 1, 2, \ldots, p \end{aligned} \tag{4.9}$$

where $\boldsymbol{z} * \boldsymbol{x}_i = [z_1 x_i^1, \cdots, z_j x_i^j, \cdots, z_n x_i^n]^T$ is a vector and $z_i = 0$ or 1. Although this is not a convex problem, it can be stated as a SOCP. However, all these works were not designed especially for driving pattern recognition in HEV applications and did not take observability of features into account. So an SVM with embedded feature selection and with respect to both statistical significance and observability of features is proposed in this work.

4.4 Proposed algorithm

4.4.1 Embedded Feature-selection Support Vector Machine

In order to embed feature selection into SVM, SVM is expected to use as few features as possible during classification. This can be expressed as follows:

$$\begin{aligned} \min_{\boldsymbol{\omega},b} \quad & \frac{1}{2}\|\boldsymbol{\omega}\|_2^2 + C \cdot \sum_{i=1}^{p} \xi_i + T \cdot \|\boldsymbol{\omega}\|_0 \\ \text{s.t.} \quad & y_i \cdot (\boldsymbol{\omega}^T \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i, i = 1, 2, \ldots, p \\ & \xi_i \geq 0, i = 1, 2, \ldots, p \end{aligned} \tag{4.10}$$

where $\|\boldsymbol{\omega}\|_0$ is the number of nonzero entries in $\boldsymbol{\omega}$. T is the tuning parameter controlling the tradeoff between the performance of the SVM and the sparsity of $\|\boldsymbol{\omega}\|$. According to [124], adding this $L0$-norm penalty to the objective function is equivalent to adding an $L1$-norm LASSO penalty (least absolute shrinkage and selection operator) [125]. Moreover, in order to take observability of each feature into account, a weight vector is introduced into the LASSO penalty of the objective function. After slight manipulation, the SVM with embedded feature selection proposed in this work

can be finally stated as :

$$\begin{aligned} \min_{\boldsymbol{\omega},b} \quad & K \cdot \|\boldsymbol{\omega}\|_2 + C \cdot \sum_{i=1}^{p} \xi_i + T \cdot \|\boldsymbol{z} * \boldsymbol{\omega}\|_1 \\ \text{s.t.} \quad & y_i \cdot (\boldsymbol{\omega}^T \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i, i = 1, 2, \ldots, p \\ & \xi_i \geq 0, i = 1, 2, \ldots, p \end{aligned} \tag{4.11}$$

here, $\boldsymbol{z} * \boldsymbol{\omega} = [z_1\omega_1, \cdots, z_j\omega_j, \cdots, z_n\omega_n]^T$. $z_i \in (0, 1]$. For a small z_i, the observability of ith feature is considered good. K, C and T are all tuning parameters and K is always set to be 1. If T is zero, then this extended SVM is a standard 2-normSVM (with soft margin). If K are all zeros and $\boldsymbol{z}$ is a vector having all "1" entries, then this extended SVM is:

$$\begin{aligned} \min_{\boldsymbol{\omega},b} \quad & C \cdot \sum_{i=1}^{p} \xi_i + T \cdot \|\boldsymbol{\omega}\|_1 \\ \text{s.t.} \quad & y_i \cdot (\boldsymbol{\omega}^T \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i, i = 1, 2, \ldots, p \\ & \xi_i \geq 0, i = 1, 2, \ldots, p \end{aligned} \tag{4.12}$$

which is equivalent to a 1-norm SVM as shown in Equation 4.8. So the conclusion can be drawn that standard 2-norm SVM (with soft margin) and 1-norm SVM are both special cases of this extended SVM, when the tuning parameters and weight vector $\boldsymbol{z}$ have particular values. More detailed comparison among these three SVMs can be found in the simulation section.

4.4.2 SVM optimization approach

The optimization problem of the proposed SVM is solved by using Second-order Cone Programming (SOCP). In a SOCP, a linear function is minimized over the intersection of an affine set and the product of second-order (quadratic) cones. SOCPs are nonlinear convex problems that include linear and (convex) quadratic programs as special cases, but are less general than Semidefinite Programs (SDPs). [126]. The relation among Linear Programming (LP), (convex) Quadratic Programming (QP), SOCP, SDP and Convex Programming (CP) can be found in Figure 4.1 [127].

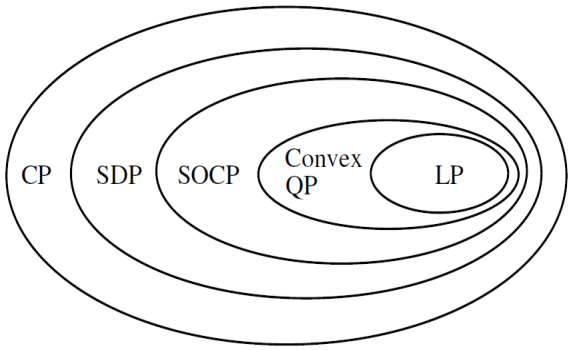


Figure 4.1: Relation among LP, convex QP, SOCP,SDP and CP

A typical SOCP problem can be stated as follows:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{f}^T \cdot \boldsymbol{x} \\ \text{s.t.} \quad & \|\boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i\| \leq \boldsymbol{c}_i^T \boldsymbol{x} + d_i, \quad i = 1, 2, \ldots, M \end{aligned} \tag{4.13}$$

where $\boldsymbol{x} \in \boldsymbol{R}^n$ is the optimization variable. The problem parameters are $\boldsymbol{f} \in \boldsymbol{R}^n$, $\boldsymbol{A}_i \in \boldsymbol{R}^{(n_i-1)\times n}$, $\boldsymbol{b}_i \in \boldsymbol{R}^{n_i-1}$, $\boldsymbol{c}_i \in \boldsymbol{R}^n$, and $d_i \in \boldsymbol{R}$. The norm appearing in the constraints is the standard Euclidean norm.

The extended SVM, Equation 4.11, can be proved to be a standard SOCP problem as follows:

Proof. Two auxiliary variables δ and $\boldsymbol{\eta}$ are introduced into Equation 4.11. δ is the upper bounds of $\|\boldsymbol{\omega}\|^2$ and $\boldsymbol{\eta}$ is a n-dimensional vector, $|\omega^i| \leq \eta^i$ and ω^i means the ith entry of vector $\boldsymbol{\omega}$ and η^i means the ith entry of vector $\boldsymbol{\eta}$. After introducing these two auxiliary variables, the optimization variable in Equation 4.11 can be changed to $\hat{\boldsymbol{\omega}} = \begin{bmatrix} \boldsymbol{\omega}^T & \boldsymbol{\eta}^T & \xi_1 \cdots \xi_p & b & \boldsymbol{\delta}^T \end{bmatrix}^T$. By using the new variable $\hat{\boldsymbol{\omega}}$, the constraints in Equation 4.11 can be stated as follows:

$$\begin{bmatrix} y_1\boldsymbol{x}_1 & \boldsymbol{0}_{(1\times n)} & \boldsymbol{u}_{(1\times p)}(1) & y_1 & 0 \\ y_2\boldsymbol{x}_2 & \boldsymbol{0}_{(1\times n)} & \boldsymbol{u}_{(1\times p)}(2) & y_2 & 0 \\ & & \vdots & & \\ y_p\boldsymbol{x}_p & \boldsymbol{0}_{(1\times n)} & \boldsymbol{u}_{(1\times p)}(p) & y_p & 0 \end{bmatrix} \cdot \hat{\boldsymbol{\omega}} - \boldsymbol{e}_{(p\times 1)} \geq \boldsymbol{0}_{p\times 1}$$
$$\begin{bmatrix} \boldsymbol{0}_{(p\times n)} & \boldsymbol{0}_{(p\times n)} & \boldsymbol{I}_{(p\times p)} & \boldsymbol{0}_{(p\times 1)} & \boldsymbol{0}_{(p\times 1)} \end{bmatrix} \cdot \hat{\boldsymbol{\omega}} - \boldsymbol{0}_{(p\times 1)} \geq \boldsymbol{0}_{p\times 1} \tag{4.14}$$

here, $\boldsymbol{u}_{(1\times p)}(i)$ is a $1 \times p$ vector, whose ith entry is $+1$ and all other entries are 0, $\boldsymbol{I}_{(p\times p)}$ is a $p \times p$ identity matrix, and $\boldsymbol{e}_{(p\times 1)}$ is a $p \times 1$ vector having all 1 entries. Moreover, these two constraints can be combined and manipulated into a standard line cone as follows:

$$\boldsymbol{D}^T \hat{\boldsymbol{\omega}} + \boldsymbol{f} \geq \boldsymbol{0}_{2p\times 1} \tag{4.15}$$

where

$$\begin{aligned}
\boldsymbol{D}^T &= \left[\begin{array}{c} \boldsymbol{D}_1 \\ \boldsymbol{D}_2 \end{array}\right]_{(2p\times(2n+p+2))} \\
\boldsymbol{D}_1 &= \left[\begin{array}{ccccc} y_1\boldsymbol{x}_1 & \boldsymbol{0}_{(1\times n)} & \boldsymbol{u}_{(1\times p)}(1) & y_1 & 0 \\ y_2\boldsymbol{x}_2 & \boldsymbol{0}_{(1\times n)} & \boldsymbol{u}_{(1\times p)}(2) & y_2 & 0 \\ & & \vdots & & \\ y_p\boldsymbol{x}_p & \boldsymbol{0}_{(1\times n)} & \boldsymbol{u}_{(1\times p)}(p) & y_p & 0 \end{array}\right]_{p\times(2n+p+2)} \\
\boldsymbol{D}_2 &= \left[\begin{array}{ccccc} \boldsymbol{0}_{(p\times n)} & \boldsymbol{0}_{(p\times n)} & \boldsymbol{I}_{(p\times p)} & \boldsymbol{0}_{(p\times 1)} & \boldsymbol{0}_{(p\times 1)} \end{array}\right]_{p\times(2n+p+2)} \\
\boldsymbol{f} &= \left[\begin{array}{c} -\boldsymbol{e}_{(p\times 1)} \\ \boldsymbol{0}_{(p\times 1)} \end{array}\right]_{2p\times 1}
\end{aligned} \tag{4.16}$$

However, because of introducing two auxiliary variables, two more sets of constraints should be added to Equation 4.11. They are:

$$\|\boldsymbol{\omega}\|_2 \leq \delta \tag{4.17}$$

$$|\omega^i| \leq \eta^i, i = 1, 2, \ldots, n \tag{4.18}$$

These two sets of constraints can be manipulated into standard second-order cones. For Equation 4.17, this constraint can be stated as follows:

$$\|\boldsymbol{A}_1^T\hat{\boldsymbol{\omega}} + \boldsymbol{c}_1\|_2 \leq \boldsymbol{b}_1^T\hat{\boldsymbol{\omega}} + d_1 \tag{4.19}$$

where

$$\begin{aligned}
\boldsymbol{A}_1^T &= \left[\begin{array}{ccccc} \boldsymbol{I}_{(n\times n)} & \boldsymbol{0}_{(n\times n)} & \boldsymbol{0}_{(n\times p)} & \boldsymbol{0}_{(n\times 1)} & \boldsymbol{0}_{(n\times 1)} \end{array}\right]_{n\times(2n+p+2)} \\
\boldsymbol{b}_1 &= \left[\begin{array}{c} \boldsymbol{0}_{((2n+p+1)\times 1)} \\ 1 \end{array}\right] \\
\boldsymbol{c}_1 &= \boldsymbol{0}_{n\times 1}, \quad d_1 = 0
\end{aligned} \tag{4.20}$$

For Equation 4.18:

$$\|\boldsymbol{A}_{i+1}^T\hat{\boldsymbol{\omega}} + \boldsymbol{c}_{i+1}\|_2 \leq \boldsymbol{b}_{i+1}^T\hat{\boldsymbol{\omega}} + d_{i+1}, i = 1, 2, \ldots, n \tag{4.21}$$

where

$$\begin{aligned}
&\boldsymbol{A}_{i+1}^T = \begin{bmatrix} \boldsymbol{u}_{(1\times n)}(i) & \mathbf{0}_{(1\times n)} & \mathbf{0}_{(1\times p)} & 0 & 0 \end{bmatrix}_{1\times(2n+p+2)}, i = 1, 2, \ldots, n \\
&\boldsymbol{b}_1^T = \begin{bmatrix} \mathbf{0}_{1\times n} & \boldsymbol{u}_{(1\times n)}(i) & \mathbf{0}_{(1\times p)} & 0 & 0 \end{bmatrix}_{1\times(2n+p+2)}, i = 1, 2, \ldots, n \\
&c_{i+1} = d_{i+1} = 0, i = 1, 2, \ldots, n
\end{aligned} \tag{4.22}$$

Here, $\boldsymbol{u}_{(1\times n)}(i)$ is a $1 \times n$ vector, whose ith entry is $+1$ and all other entries are 0. Finally, Equation 4.11 can be stated as a standard SOCP problem as follows:

$$\begin{aligned}
&\min_{\hat{\boldsymbol{\omega}}} \quad \boldsymbol{b}^T\hat{\boldsymbol{\omega}} \\
&\text{s.t.} \quad \boldsymbol{D}^T\hat{\boldsymbol{\omega}} + \boldsymbol{f} \geq \mathbf{0}_{p\times 1} \\
&\quad \|\boldsymbol{A}_1^T\hat{\boldsymbol{\omega}} + \boldsymbol{c}_1\|_2 \leq \boldsymbol{b}_1^T\hat{\boldsymbol{\omega}} + d_1 \\
&\quad \|\boldsymbol{A}_{i+1}^T\hat{\boldsymbol{\omega}} + \boldsymbol{c}_{i+1}\|_2 \leq \boldsymbol{b}_{i+1}^T\hat{\boldsymbol{\omega}} + d_{i+1} \\
&\quad i = 1, 2, \ldots, n
\end{aligned} \tag{4.23}$$

where, $\boldsymbol{b} = \begin{bmatrix} \mathbf{0}_{(1\times n)} & T \cdot \boldsymbol{z}_{(1\times n)} & c \cdot \boldsymbol{e}_{(1\times p)} & 0 & 1 \end{bmatrix}^T$, $\boldsymbol{z}_{(1\times n)}$ here is the weight vector. All other detailed optimization parameters can be found from Equations 4.15-4.22. □

Moreover, all the equations above are written in accordance with the notation of SeDuMi, an MATLAB toolbox for optimization written by Jos F. Sturm in 1999, so that this extended SVM can be solved efficiently by SeDuMi.

4.5 Simulation results

4.5.1 Comparison with standard 2-norm SVM and 1-norm SVM by using "Ionosphere" data set

In this section, the proposed SVM, standard 2-norm SVM and 1-norm SVM are compared by using the widely adopted data set "Ionosphere", provided by UCI machine learning repository(http://archive.ics.uci.edu/ml /datasets.html) [128]. Each pattern in this data set has 34 features, which means adequate different feature combinations can be provided. Therefore, the performance of feature selection can be compared by using this data set. Note that in order to keep the fairness of the comparison among

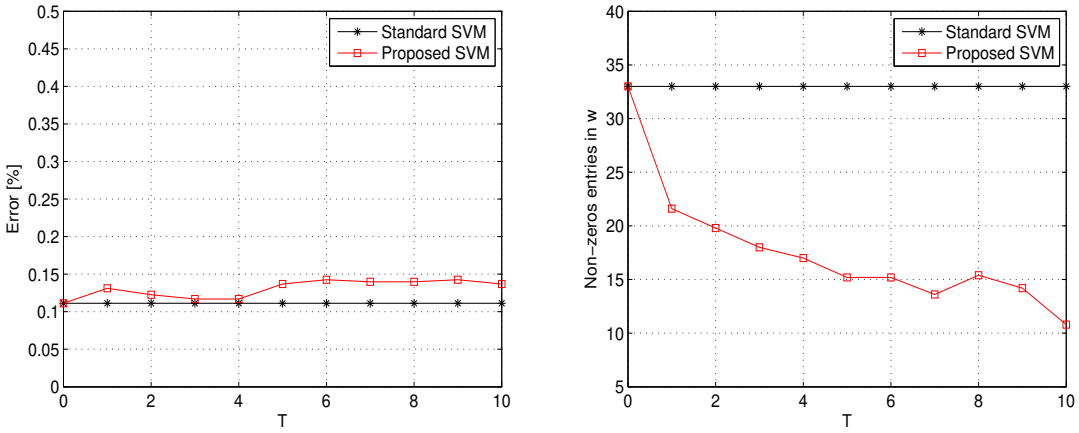
different SVMs, the weight vector $\boldsymbol{z}$ (in the proposed SVM) of the observability of each feature is set to have all 1 entries in this section. Finally, to give a more realistic estimation of generalization, a five-fold cross-validation is used to compare the performance of all SVMs. The classification error and the number of non-zero entries in $\boldsymbol{\omega}$, which are shown in the following figures, are the mean values of the results of the five-fold cross-validation.

First, the comparison between the proposed SVM and standard 2-norm SVM can be found in Figure 4.2. The tuning parameter K in the proposed SVM is always set to be 1, and Cs of both SVMs keep constant at 1, too. With increasing tuning parameter T, the proposed SVM succeeds in selecting fewer features for classification. However, the classification error also goes up. When T increases to 10, only 11 features are used for classification and the classification error only rises to 14% from 11%. When T is zero, the performance of the proposed SVM and standard 2-norm SVM are identical. Compared to standard 2-norm SVM, the proposed SVM has obviously higher efficiency by using many fewer features for classification at the cost of acceptable decrease of classification accuracy. In Figure 4.3, standard deviation of both non-zeros entries in $\boldsymbol{w}$ and classification error of these two types of SVMs are illustrated. An important note should be made here: since the standard 2-norm SVM does not have parameter T, tuning T has no effect on the performance of standard 2-norm SVM. Therefore the performance curves of standard 2-norm SVM are all constant horizontal lines in all of the figures with horizontal axis of parameter T.

Next, the proposed SVM is compared with 1-norm SVM. As shown in Figure 4.4, the performance of these two SVMs are almost the same. However, from Figure 4.6, the proposed SVM can provide more tuning options during the design of pattern recognition system by tuning C and T simultaneously.

4.5.2 Testing by using the data collected from the simulation results of Toyota Prius

In this section, the proposed SVM is tested and compared with standard 2-norm SVM and 1-norm SVM by using the data collected from the simulation results of a Toyota Prius. The model of the Toyota Prius HEV is built in Autonomie, a vehicle simulation tool developed by US DOE Argonne National Laboratory. The Toyota Prius is a hybrid electric mid-size hatchback with power split powertrain architecture. There are two electric motors and one engine in its powertrain and the M/G 1 is coupled



(a) (b)

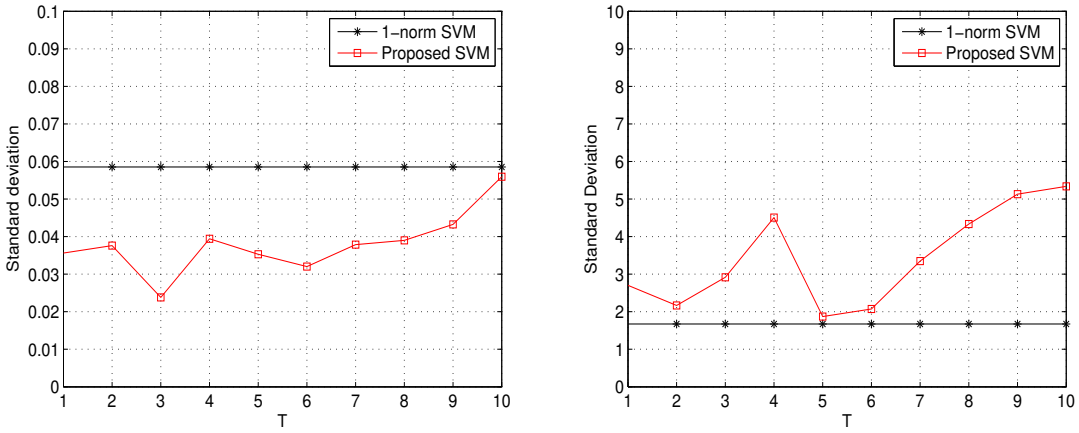
Figure 4.2: Comparison of the proposed SVM and standard 2-norm SVM

with the final drive directly.

First of all, in order to generate the data set for training and testing the SVMs, the model of the Prius is run in four different types of driving cycles. These cycles are highway, congested urban road, flowing urban road and country road driving cycles, which can be found in Figure 4.7. Based on [115], 12 real-time vehicle state variables (features) are collected from the simulation results to form four data subsets, $Data_1$ collected from highway simulation, $Data_2$ from congested urban, $Data_3$ from flowing urban and $Data_4$ from country road. These 12 features are listed in Table (4.1). So the entire data set in this section is $DATA = Data_1 \cup Data_2 \cup Data_3 \cup Data_4$. Furthermore, the observability values, $z_i \in (0, 1]$ are assigned to each feature as shown in Table (4.1). According to [23], the characteristic of the current driving condition can be effectively recognized, if the length of the sampling time reaches or exceeds 3 minutes. So all these features are sampled and calculated over every 3 minutes. After collecting the data set $DATA = Data_1 \cup Data_2 \cup Data_3 \cup Data_4$ from the simulation, a five-fold cross-validation is also used to test the performance of the proposed SVM. All the results shown in the following figures are also the mean values of the results of the five-fold cross-validation.

Two-class driving condition recognition

In this section, the data subsets: $Data_1$, $Data_2$, $Data_3$ and $Data_4$ are combined in various pairs for training and testing the proposed SVM. The combinations used in



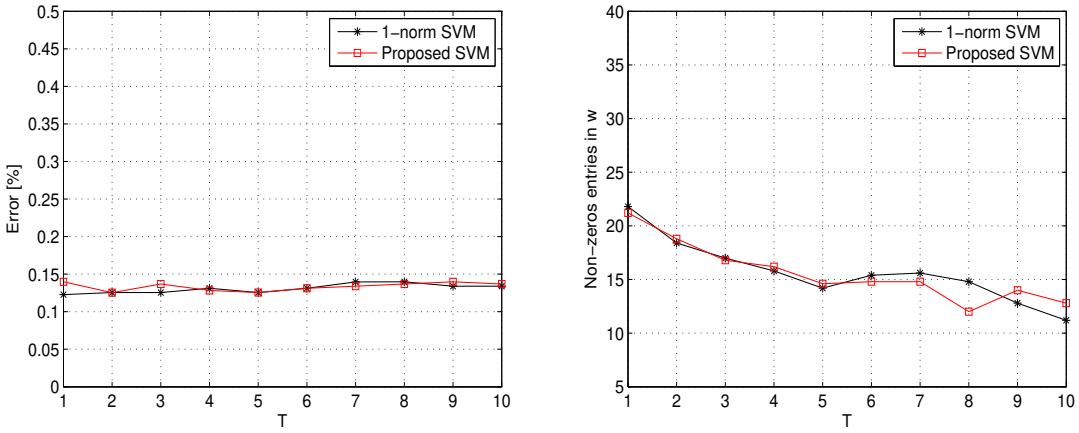
(a) Standard deviation of classification error (b) Standard deviation of non-zeros entries in $\boldsymbol{w}$

Figure 4.3: Standard deviation of the five-fold cross-validation: 2-norm SVM vs. the proposed SVM

Table 4.1: Feature set: "OBS" means the Observability value for that feature

FN	Name	OBS	#	Name	OBS
1	Average vehicle speed	0.1	7	Average engine speed	0.15
2	Maximal vehicle acceleration	0.3	8	Average engine torque	0.15
3	Maximal vehicle deceleration	0.3	9	Average current in Motor 1	0.35
4	Idle rate of vehicle speed	0.2	10	Average speed of Motor 1	0.15
5	Change of battery SOC	0.5	11	Average current in Motor 2	0.35
6	Average driver demanded power	0.4	12	Average speed of Motor 2	0.15

this section are highway vs. congested urban road, flowing urban road vs. congested urban road, flowing urban road vs. country road, and highway vs. country road. Standard 2-norm SVM, 1-norm SVM and the proposed SVM are trained and tested by these four 2-class data set. Moreover, the proposed SVM are trained and tested in two modes, *Mode 1*: taking observability into account and *Mode 2*: without consideration of observability (by setting observability of each feature as 1). The results can be found in Figure 4.8. As shown in these figures, the proposed SVM can find a relatively better balance between the number of selected features and the accuracy of classification. Standard 2-norm SVM has the best accuracy of classification, however, it does not have the ability for feature selection. On the other hand, 1-norm SVM



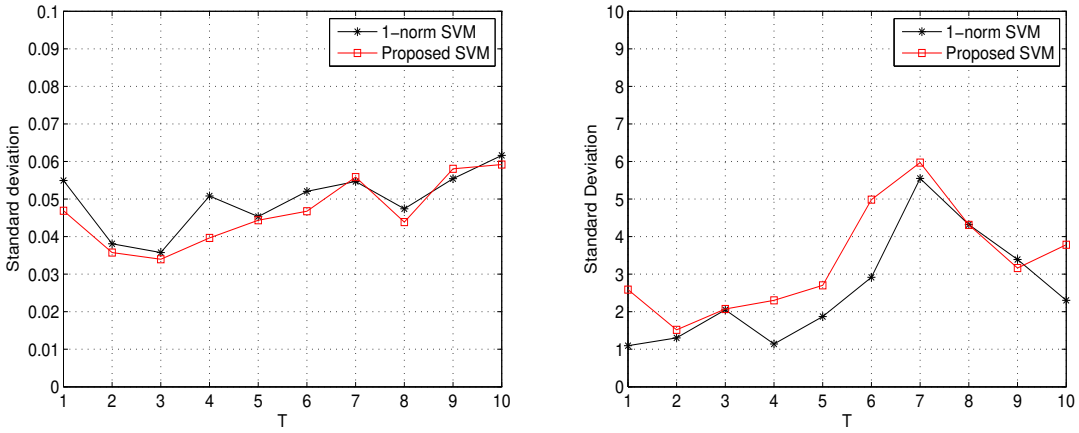
(a) (b)

Figure 4.4: Comparison of the proposed SVM and 1-norm SVM

can select the fewest features, while the classification accuracy is the worst. Note should be made here that the parameter T in 1-norm SVM cannot be set to 0, so the performance curves of 1-norm SVM in these figures start from $T = 1$. Moreover, in order to demonstrate the superiority of taking observability into account, the features selected by the proposed SVM in Mode 1 (taking observability into account) are compared with those features selected by the proposed SVM in Mode 2 (without consideration of observability). The selected features by both modes are listed in Table (4.2). The symbol '$\star$' denotes a feature selected by the SVM in Mode 1 and $\circ$ means this feature selected by Mode 2 . Furthermore, the total observability value of the features selected by both modes are calculated and listed in the right side in Table (4.2). Obviously, the proposed SVM with observability consideration can select more reliable and accessible features so that higher recognition efficiency can be achieved.

Multi-class driving condition recognition

In this section, the entire 4-class data set, $DATA = Data_1 \cup Data_2 \cup Data_3 \cup Data_4$, is used to train and test the proposed SVM for multi-class pattern recognition. In this work, the multi-class SVM is based on "Max WIN Voting" strategy (MWV-SVM). For the four driving conditions scenario, 6 ($C(4, 2)$) the proposed SVMs are used for classifying current driving pattern into each of the two driving conditions. If an SVM classifies the current driving pattern into the ith driving condition, then the vote for the ith driving condition is incremented by one. Finally, the current driving pattern



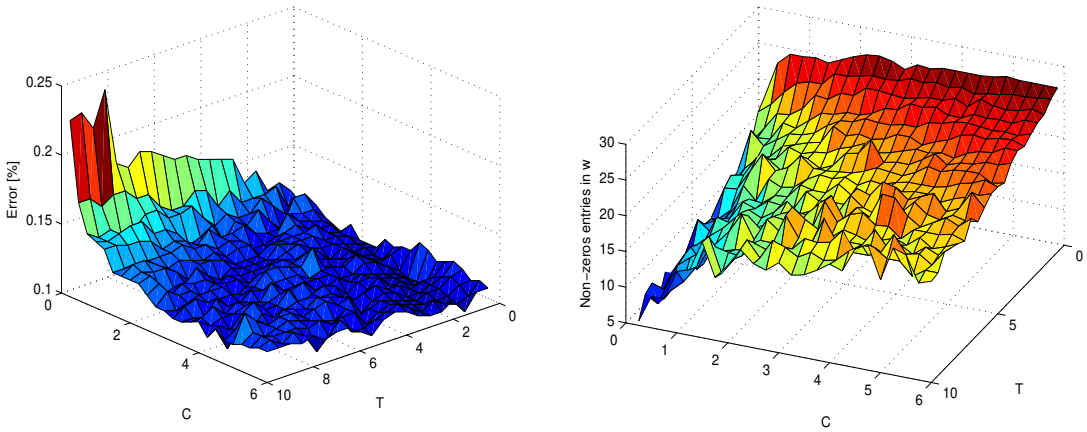
(a) Standard deviation of classification error (b) Standard deviation of non-zeros entries in $\boldsymbol{w}$

Figure 4.5: Standard deviation of the five-fold cross-validation: 1-norm SVM vs. the proposed SVM

is predicted to be the driving condition with largest vote. The selected feature set of $\mathscr{F}$ in the multi-class SVM is determined as follows:

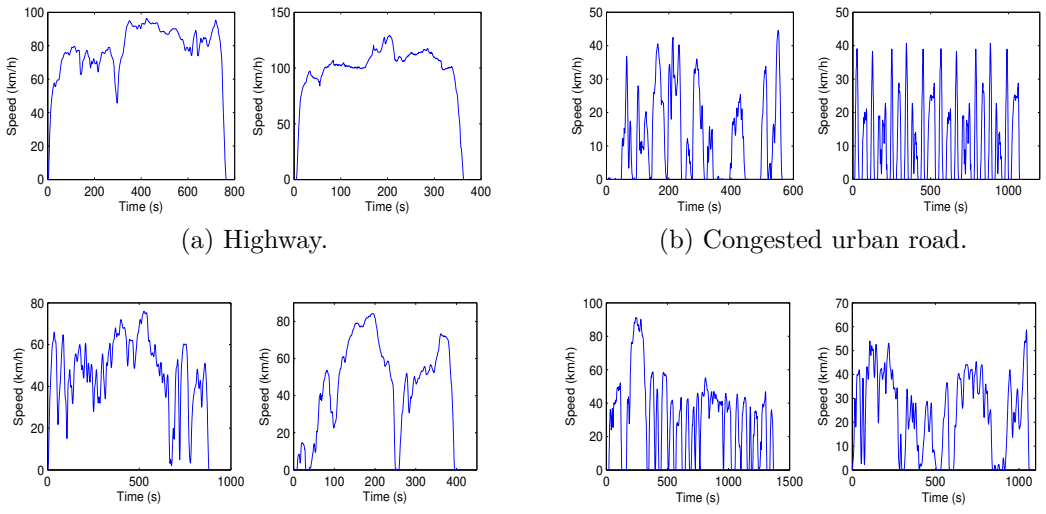
$$\mathscr{F} = \bigcup_{i=1}^{6} f_i \tag{4.24}$$

where, f_i is the feature subset selected by each of the six proposed SVM. The experiment results can be found in Figure 4.9, where the proposed SVM are also compared with two other SVMs and obviously also has a better balance between the amount of selected features and the accuracy of classification. Note should be made here that the parameter T in 1-norm SVM cannot be set as 0, so the curves of 1-norm SVM in these figures start from $T = 1$.



(a) (b)

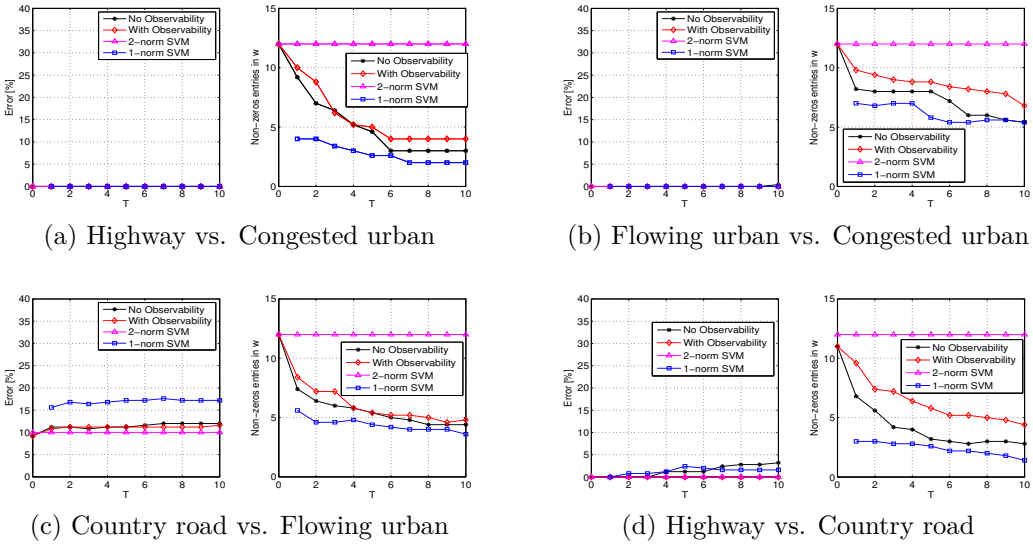
Figure 4.6: Performance of the proposed SVM



(a) Highway. (b) Congested urban road.

(c) Country road. (d) Flowing urban road.

Figure 4.7: Speed signals for four different driving conditions, sampled at $1Hz$



(a) Highway vs. Congested urban

(b) Flowing urban vs. Congested urban

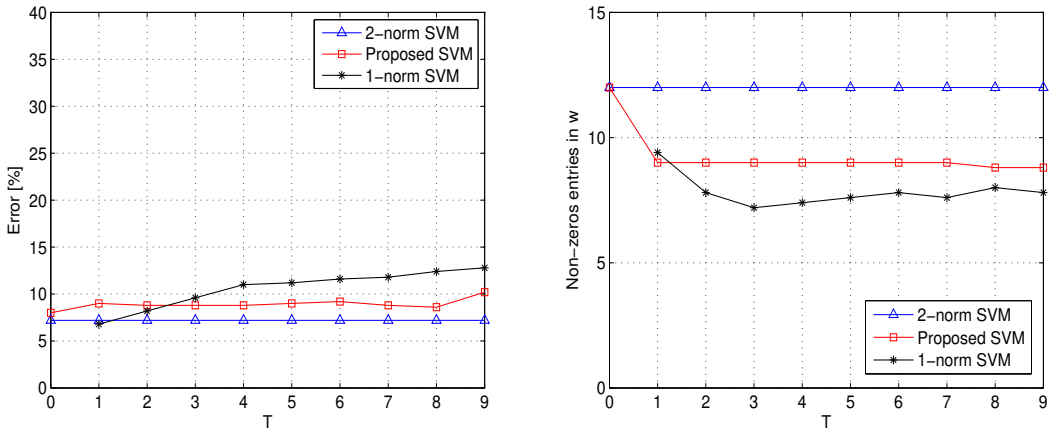
(c) Country road vs. Flowing urban

(d) Highway vs. Country road

Figure 4.8: Comparison between with observability consideration and no observability consideration

Table 4.2: Comparison of selected features: FN-Feature Number; OBS-Observability Value; ⋆ is the feature selected by the proposed SVM in mode 1 (with observability consideration); ∘ is the feature selected by the proposed SVM in mode 2 (without observability consideration)

FN	1	2	3	4	5	6	7	8	9	10	11	12	Total OBS	
OBS	0.1	0.3	0.3	0.2	0.5	0.4	0.15	0.15	0.3	0.15	0.3	0.15	⋆	∘
Highway vs. Congested Urban Road														
T=0	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	3	3
T=5	⋆ ∘				∘		⋆	⋆ ∘		⋆ ∘		⋆ ∘	0.7	1.05
T=10	⋆				∘			⋆ ∘		⋆		⋆ ∘	0.55	0.8
Flow Urban Road vs. Congested Urban Road														
T=0	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	3	3
T=5	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	∘		⋆ ∘	⋆ ∘		⋆ ∘		⋆	1.5	1.85
T=10	⋆	⋆ ∘	⋆ ∘	⋆ ∘	∘		⋆ ∘	⋆		⋆			1.35	1.45
Flow Urban Road vs. Country Road														
T=0	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	3	3
T=5	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘				⋆ ∘		∘			1.05	1.2
T=10	⋆ ∘		⋆ ∘	⋆ ∘				⋆		∘			0.75	0.75
Highway vs. Country Road														
T=0	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	⋆ ∘	3	3
T=5	⋆	⋆			∘			⋆ ∘	∘	⋆		⋆ ∘	0.85	1.1
T=10	⋆				⋆ ∘			⋆ ∘	∘			⋆	0.9	0.95



(a) (b)

Figure 4.9: Comparison among three SVMs on multi-class data set

Chapter 5

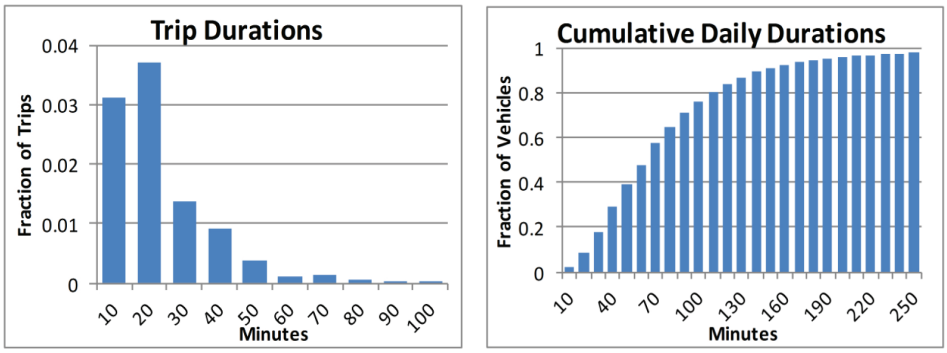
Inertial Matching Pursuit Classification for Driving Pattern Recognition

An early version of the work presented in this chapter was submitted to the IEEE transactions on intelligent transportation systems in 2017.

In the previous chapter, we presented the new extended SVM for driving pattern recognition and discuss about the feature selection problem of the driving pattern recognition system. In this chapter, we will focus on the sampling time of the pattern recognition. After the introduction and discussion, we will propose the Inertial Matching Pursuit Classification algorithm. This algorithm is able to recognize the driving pattern within a much shorter time period. This would be able to improve the dynamic property of the driving pattern recognition system. As mentioned in Chapter 1, the IEA algorithm proposed in Chapter 6 needs the current driving pattern as input to recognize the entire Trip Mode. Therefore, a driving pattern recognition with better dynamic property means better performance of the IEA algorithm. Note that the Inertial Matching Pursuit Classification algorithm is not based on the extended SVM proposed in previous chapter. These two algorithms are dealing with different concerns in driving pattern recognition.

5.1 Introduction

Common real-time data available for determining driving patterns can be of two main types: images and telemetry. An image or a video supplies detailed information to distinguish the driving conditions [129, 130], but the DPR system using image or video data requires relatively complicated image processing to obtain useful results. The second type, more commonly used system, relies on a time series of vehicle lacation, velocity, and acceleration that are easily collected. Although this type of data normally cannot supply enough detailed information to discriminate complicated driving conditions, it is a stable and efficient choice to recognize common driving patterns for the control system of HEV. Several works have used velocity and/or acceleration data to determine the driving pattern for HEV control system [110, 131, 132, 133]. Note that Geographical Information Systems (GIS) and Global Positioning Systems (GPS) can also help to determine the driving conditions. However, only vehicle position data is not sufficient for DPR. For example, the traffic congestion level of highway in rush hours can be even worse than that of city in the midnight. In fact, Traffic Flow Pattern is a key fact for recognizing driving conditions for vehicles [134, 135, 136]. Traffic flow has three main variables: speed, density and flow. Obviously, among them, vehicle speed (and acceleration) is the most important variable to determine driving conditions for an individual vehicle. A lot of research has been conducted to recognize the driving conditions by using vehicle speed (and acceleration) [137, 138, 139].



(a) Trip durations. (b) Cumulative daily duration.

Figure 5.1: Distribution of driving durations

Earlier research [23] has shown that if the length of the sampling time of the vehicle speed reaches or exceeds 3 minutes, the characteristic of the current driv-

ing pattern can be effectively recognized by common pattern recognition algorithms, such as Support Vector Machine (SVM) and Neural Networks. In vehicle applications, with the consideration of dynamic property of the driving control system and the computational power of controller, sampling length is usually slightly less than 3 minutes [133]. (In both [110] and [133], the sampling time is 150s.) However, according to [140], the average duration of each driving trip in U.S is 18.1 minutes and the daily average driving duration is 74.9 minutes. Detailed distribution of driving duration can be found in Figure 5.1 [140]. According to these data, a DPR system with 3 minutes sampling time (which means its recognition period is also 3 minutes) only updates the driving pattern around 7 times for each driving trip and totally about 25 times for the entire daily driving. This actually leads to a low accuracy of estimating the overall driving conditions for entire driving trips or even for daily driving. For instance, suppose we have a 18-minute trip mixed by 58% city and 42% highway conditions. A DPR system with 3-minute recognition period can only estimate the city percentage of the entire trip as $n \cdot 3\text{minutes}/18\text{minutes}$. ($n$ here is the number of the driving patterns recognized as city condition in the trip) This actually means that the error of the percentage estimation could be around 8%, even when the recognition results are all correct (when $n = 3$, the estimated percentage is 50.01%, or when $n = 4$, the estimated percentage is 66.68%). We defined the resolution of a DPR system as follows:

$$Re = \frac{\Delta T}{2D} \cdot 100\% \tag{5.1}$$

here D is the duration of the entire trip and ΔT is recognition period which equals to the speed signal sampling time. So the resolution of the DPR system in the above example is 8.33%. In order to improve the resolution and increase the accuracy of DPR systems, a novel pattern recognition algorithm, Inertial Matching Pursuit Classification (IMPC), is proposed in this manuscript.

In traditional DPR system, the vehicle speed is completely sampled following the basic principle of the Nyquist-Shannon sampling theorem. However, according to the theory of Compressed Sensing (CS), it is possible to reconstruct signals of scientific interest accurately and sometimes even exactly from a number of samples which is far smaller than the sampling theorem requires [141]. Furthermore, unlike image processing, speed signals in driving pattern recognition are not required to be exactly reconstructed. Instead, DPR system usually only extracts or selects some features from the speed signals for recognition, therefore speed signals used for driving

pattern recognition are only required to be statistically significant. It seems that the number of speed samples can be even smaller than CS required. In this manuscript, investigation is conducted to reveal the relationship between the recognition accuracy and the number of speed samples. In fact, the proposed algorithm, IMPC, does not require the vehicle speed to be fully sampled or accurately reconstructed.

5.1.1 Driving Condition Recognition

Currently, no uniformly accepted standard exists for classification of driving conditions. Based on practical requirements, various classification standards have been adopted based on the intended application and data availability. For example, based on the types of road, driving conditions could be classified into highway, urban road, and extra urban road [137, 142]. Similarly, the level of congestion is also a very important factor in classifying the real-time driving conditions [143]. Moreover, under contract with the Environmental Protection Agency (EPA), Sierra Research Inc. has developed a set of 11 drive cycles that represent passenger car and light truck operations over a range of facilities and congestion levels in urban areas. [144] In other work, six Levels of Service (LOS) were defined for each type of facility, where LOS is defined as a qualitative measure describing operational conditions within a traffic stream, based on service measures such as speed, travel time, freedom to maneuver, traffic interruptions, comfort, and convenience. [145].

Two important factors in DPR system are the feature selection method and the classification algorithm. In vehicle applications, a number of state variables of vehicle powertrain can be used to generate or even directly regarded as features for driving pattern recognition. The study in [109] has successfully reduced 62 different features to 16 independent driving condition features using Factorial Analysis. A detailed summary of features used to recognize driving patterns in the past several years can be found in [115]. As discussed in Section I, vehicle speed (and/or acceleration) is very useful to determine driving patterns for an individual vehicle. A lot of work has been done in using the vehicle speed to recognize the driving pattern [110, 131, 132, 133, 137, 138, 139]. In [110], four features are extracted from real-time vehicle speed and then SVM are used to recognize current driving pattern. In [131, 133], vehicle speed is employed to extract features for predicting the roadway type and traffic congestion level. A multilayered and multi-class neural network is developed for prediction. Several standard driving cycles are utilized to test the performance

of this neural network. In [137, 138, 139], vehicle speed is used to access or detect driving conditions. In [132], vehicle speed is one of the features used for driving pattern recognition. The framework of the DPR system using vehicle speed is shown in Figure 5.2 [110, 131, 133]

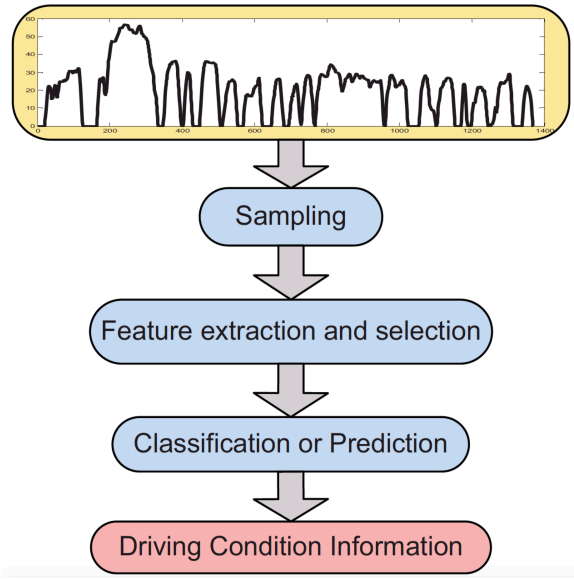


Figure 5.2: Framework of DPR system using vehicle speed

5.1.2 Compressed Sensing

It is generally agreed that the foundation of current compressed sensing (CS) theory (also known as compressive sampling or compressive sensing) was laid down by the three papers [146, 147, 148] in 2006. These and several other papers, have inspired a burst of intensive research on CS in the past several years. CS acquires a signal of interest indirectly by correcting a relatively small number of its projections (defined later) rather than the sample at the regular Nyquist rate. In a way, this new signal acquisition paradigm has fundamentally changed the traditional approach with which real-time data are acquired [149].

For sensing efficiency, one wishes to collect a relatively much smaller number of measurements.[141]. The essence of Compressed Sensing is to acquire signal data more efficiently as follows. Acquisition of the signal vector, $\boldsymbol{x}$ is carried out by measuring m projections of $\boldsymbol{x}$ onto the sensing vectors $\boldsymbol{\varphi}_i^T, i = 1, 2, \ldots, m$ in the form of $y_i = \boldsymbol{\varphi}_i^T \cdot \boldsymbol{x}$

for $i = 1, 2, \ldots, m$. Using matrix notation, this sensing process is described as

$$\boldsymbol{y} = \hat{\boldsymbol{\Phi}} \cdot \boldsymbol{x} \tag{5.2}$$

Where

$$\hat{\boldsymbol{\Phi}} = \begin{bmatrix} \boldsymbol{\varphi}_1^T \\ \boldsymbol{\varphi}_2^T \\ \vdots \\ \boldsymbol{\varphi}_m^T \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \tag{5.3}$$

It is often assumed that the lengths of the projection vectors $\boldsymbol{\varphi}_i^T, i = 1, 2, \ldots, m$ are unity. In this signal model, signal vector, $\boldsymbol{x}$ is unknown and is recovered from the measurements $y_i, i = 1, 2, \ldots, m$. Furthermore, $\boldsymbol{x}$ is assumed to be r-sparse (or approximately sparse) in an appropriate dictionary $\boldsymbol{D} \in \boldsymbol{R}^{(n \times p)}$. That is,

$$\boldsymbol{x} = \boldsymbol{D} \cdot \boldsymbol{a} \tag{5.4}$$

where $p \times 1$ column vector $\boldsymbol{a}$ is the coordinates of $\boldsymbol{x}$ under the dictionary $\boldsymbol{D}$, which is r-sparse ($\boldsymbol{a}$ has r non-zero entries) or approximately sparse.

The signal $\boldsymbol{x}$ can be reconstructed by solving Equation 5.5 to find $\boldsymbol{a}$. Then $\boldsymbol{x}$ can be obtained by Equation 5.4.

$$\begin{aligned} &\min \|\boldsymbol{a}\|_1 \\ s.t. \quad &\hat{\boldsymbol{\Phi}} \cdot \boldsymbol{D} \cdot \boldsymbol{a} = \boldsymbol{y} \end{aligned} \tag{5.5}$$

Here $\hat{\boldsymbol{\Phi}}$ is given by Equation 5.3. In Equation 5.5, let $\hat{\boldsymbol{\Phi}} \cdot \boldsymbol{D} = \hat{\boldsymbol{D}}$, then it can be transferred into a sparse representation problem. Several algorithms can be used to solve this sparse representation problem to find coordinates $\boldsymbol{a}$. Generally, these algorithms can be classified in two main groups [150]. The first group includes greedy algorithms such as Matching Pursuit (MP) [151], Orthogonal MP (OMP) [152] and Subspace Pursuit [153]. The algorithms in the second group are based on convex relaxation methods such as the basis pursuit denoising [124] or least absolute shrinkage and selection operator (LASSO) [154], which changes Equation 5.5 into:

$$\min_{\boldsymbol{a}} \frac{1}{2} \|\boldsymbol{y} - \hat{\boldsymbol{D}} \cdot \boldsymbol{a}\|_2^2 + T \cdot \|\boldsymbol{a}\|_1 \tag{5.6}$$

5.1.3 Greedy Algorithms

The proposed classification algorithm IMPC is developed based on the greedy algorithms. Greedy algorithms, such as Matching Pursuit (MP) [151], Orthogonal MP (OMP) [152], Subspace Pursuit (SP) [153] and Regularized Orthogonal Matching Pursuit (ROMP) [155], are efficient approaches to solve the sparse representation problems, which are NP-hard. The greedy algorithms find the supports of the signal $\boldsymbol{x}$ iteratively, and reconstruct the signal using the pseudoinverse [156]. Let $\mathcal{D} = \{\boldsymbol{d}_i\}_{i\in\Gamma}$ be a dictionary of s unit-norm vectors (atoms) with $s > r$ in r-dimensional Hilbert space $\boldsymbol{C}^r$. $\boldsymbol{x}$ is the signal of interest to be represented by $\mathcal{D}$ sparsely. Suppose K atoms from the dictionary $\mathcal{D}$ are required to represent the signal vector $\boldsymbol{x}$. The MP and OMP algorithms augment the support set by one index at each iteration until K atoms are selected or the approximation error is within a preset threshold. More specifically, the standard Matching Pursuit iteration finds a sequence of dictionary vectors $\{\boldsymbol{d}_{i_k}|k = 0, 1, \cdots, K-1\}$ that satisfy

$$|\langle \boldsymbol{R}^k\boldsymbol{x}, \boldsymbol{d}_{i_k}\rangle| \geq \alpha \sup_{i\in\Gamma} |\langle \boldsymbol{R}^k\boldsymbol{x}, \boldsymbol{d}_i\rangle| \tag{5.7}$$

where, $\alpha \in (0, 1]$ is a relaxation factor. $\langle\cdot,\cdot\rangle$ is the inner product. $\boldsymbol{R}^k\boldsymbol{x}$ is the residue after the kth iteration, satisfying

$$\begin{cases} \boldsymbol{R}^k\boldsymbol{x} = \boldsymbol{R}^{k-1}\boldsymbol{x} - \langle \boldsymbol{R}^{k-1}\boldsymbol{x}, \boldsymbol{d}_{i_{k-1}}\rangle \boldsymbol{d}_{i_{k-1}} \\ \boldsymbol{R}^0\boldsymbol{x} = \boldsymbol{x} \end{cases} \tag{5.8}$$

Finally, signal $\boldsymbol{x}$ can be expressed sparsely as follows:

$$\boldsymbol{x} = \sum_{k=0}^{K-1} \langle \boldsymbol{R}^k\boldsymbol{x}, \boldsymbol{d}_{i_k}\rangle \boldsymbol{d}_{i_k} + \boldsymbol{R}^K\boldsymbol{x} \tag{5.9}$$

The OMP algorithm improves MP method by orthogonalizing the direction of projection in each iteration by a Gram-Schmidt procedure. The SP algorithm maintains a set of K indices. At each iteration, the index set is refined by adding K new candidates to the current list and then discarding K insignificant ones from the list of $2K$ candidates [157]. The MP, OMP and SP algorithms are all quite fast and they are all provable. However, they all lack the strong guarantees for sparse recovery [156]. The ROMP algorithm is a stable greedy algorithm providing uniform guarantees for

sparse recovery [155].

The manuscript is organized as follows: the Inertial Matching Pursuit Classification (IMPC) algorithm is introduced in Section III. In Section IV, experiments are conducted to demonstrate the superiority of this algorithm. Finally, Section V summarizes the work and suggests future research.

5.2 The Inertial Matching Pursuit Classification

Given L types of different driving conditions, the DPR system classifies current driving pattern in to one of these L types. In the traditional DPR system, as shown in Figure 5.2, the vehicle speed in ΔT seconds are usually completely sampled at the sampling frequency of $1HZ$ to form an ΔT dimensional speed vector $\boldsymbol{x}$ [110]. In this work, the Compressed Sensing is used to sample these ΔT-second speed signals. The sampling matrix $\hat{\mathbf{\Phi}}$ is constructed by m rows randomly selected from an $\Delta T \times \Delta T$ identity matrix. ($m < \Delta T$) The sampling model can be expressed by using Equation 5.2. In this signal model, speed vector, $\boldsymbol{x}$ is unknown and can be recovered from the m measurements $y_i, i = 1, 2, \ldots, m$ by using Equation 5.4 and Equation 5.5. Note that dictionary $\boldsymbol{D}$ is given before the Compressed Sensing (CS).

After measurements $\boldsymbol{y}$ is obtained, one straightforward approach of DPR is as follows: Firstly, the measurements $\boldsymbol{y}$ are used to recover the speed signal $\boldsymbol{x}$; Then SVMs or Neural Networks are employed to recognize current driving pattern by using the features extracted or selected from this recovered speed signal $\hat{\boldsymbol{x}}$. However, in spite of the improvement of the sensing efficiency of vehicle speed, this approach has no difference with traditional DPR method. The CS algorithm is still required to sample and recover the speed signal for about every 3 minutes ($\Delta T \approx 180$), so that the accuracy of classification can be guaranteed. More detailed discussion and comparison about this approach can be found in Section 5.3. Moreover, in order to recover $\boldsymbol{x}$ with overwhelming probability, the number of sampling points m in CS is required to satisfy the four-to-one practical rule which says that for exact reconstruction, one needs about four incoherent measurements per unknown nonzero term in coordinates $\boldsymbol{a}$, i.e.

$$m \geq 4 \cdot r \tag{5.10}$$

regardless of the signal's dimension n [141]. Here, r is the number of nonzero terms

in coordinate vector $\boldsymbol{a}$ ($\boldsymbol{a}$ can be found in Equation 5.5).

5.2.1 Inertial Matching Pursuit Classification

Instead of recovering the vehicle speed $\boldsymbol{x}$ from the CS measurements $\boldsymbol{y}$, IMPC recognizes current driving pattern directly from the coordinates $\boldsymbol{a}$. Actually, sparsity of signals has been an extremely powerful tool in many classical signal processing applications, such as compression and denoising [158]. In recent years, applications of sparse representation have been extended to the area of pattern recognition [159], including face recognition [160], iris recognition [161], tumor classification [162], hyperspectral unmixing [163] and target detection [164].

Given L types of different driving conditions, suppose we have N pairs of training data $\boldsymbol{T}_l = \{(\boldsymbol{x}_1^l, l), \cdots, (\boldsymbol{x}_N^l, l)\}$ in each type l. And in each pair of training data $(\boldsymbol{x}_n^l, l)$, $\boldsymbol{x}_n^l$ is the vehicle speed sampled from the lth type of driving condition at a sampling rate of $1Hz$ in ΔT seconds. The entire training data set can be also expressed as follows:

$$\begin{aligned}\boldsymbol{T} &= \{\boldsymbol{T}_1, \boldsymbol{T}_2, \cdots, \boldsymbol{T}_L\} \\ &= \{(\boldsymbol{x}_1^1, 1), \cdots, (\boldsymbol{x}_N^1, 1), \\ &\quad (\boldsymbol{x}_1^2, 2), \cdots, (\boldsymbol{x}_N^2, 2), \\ &\quad \vdots \\ &\quad (\boldsymbol{x}_1^L, L), \cdots, (\boldsymbol{x}_N^L, L)\}\end{aligned} \tag{5.11}$$

Since $\boldsymbol{T}$ is training data set, speed vectors $\boldsymbol{x}_n^l$ in it are all known for training. Suppose we choose P speed vectors $\boldsymbol{x}_n^l$ randomly ($P \leq N$) from each type l to construct the sub-dictionary $\boldsymbol{D}_l$. Then the entire dictionary $\boldsymbol{D}$ for IMPC is as follows:

$$\begin{aligned}\boldsymbol{D} &= [\boldsymbol{D}_1, \boldsymbol{D}_2, \cdots, \boldsymbol{D}_L] \\ &= [\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_{L \cdot P}]\end{aligned} \tag{5.12}$$

where, $\boldsymbol{D}$ is a $\Delta T \times (L \cdot P)$ matrix, and its column vectors $\boldsymbol{d}_i$ are the speed vectors from training data set $\boldsymbol{T}$. Obviously, column vectors $\boldsymbol{d}_i, i \in [(l-1) \cdot P + 1, l \cdot P]$ belong to the lth type of driving condition.

Given a dictionary $\boldsymbol{D}$, sampling matrix $\hat{\boldsymbol{\Phi}}$ (constructed by m rows randomly selected from a $\Delta T \times \Delta T$ identity matrix) and a sequence of measurement vectors$\{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_R\}$ from a trip, then the corresponding sequence of driving patterns $\{\ell_1, \ell_2, \cdots, \ell_R\}$ can be recognized by IMPC, which can be found in Algorithm 1. Note that measurements

$\boldsymbol{y}_r$ are sampled according to CS theorem from real-time vehicle speed in practical application. In the experiments of this manuscript, $\boldsymbol{y}_r = \hat{\boldsymbol{\Phi}} \cdot \boldsymbol{x}_r$. Here $\boldsymbol{x}_r$ is the speed vector in the testing data set and unknown to IMPC

In pattern recognition and data clustering, the distance between two points (patterns) is usually employed to measure the similarity between them.[165] Given two points $\boldsymbol{x}$ and $\boldsymbol{d}_i$, we use Euclidean norm to measure their similarity. Without loss of generality, we have:

$$\begin{aligned}(\|\boldsymbol{x} - \boldsymbol{d}_i\|_E)^2 &= \langle \boldsymbol{x}, \boldsymbol{x} \rangle + \langle \boldsymbol{d}_i, \boldsymbol{d}_i \rangle - 2\langle \boldsymbol{x}, \boldsymbol{d}_i \rangle \\ &= \langle \boldsymbol{x}, \boldsymbol{x} \rangle + \langle \boldsymbol{d}_i, \boldsymbol{d}_i \rangle - 2\|\boldsymbol{x}\|_E \cdot \|\boldsymbol{d_i}\|_E \cdot cos(\theta)\end{aligned} \tag{5.18}$$

where, θ is the angel between vector $\boldsymbol{x}$ and $\boldsymbol{d}_i$. Moreover, if $\boldsymbol{d}_i$ is a unit-norm vector, then we have:

$$(\|\boldsymbol{x} - \boldsymbol{d}_i\|)^2 = \langle \boldsymbol{x}, \boldsymbol{x} \rangle + \langle \boldsymbol{d}_i, \boldsymbol{d}_i \rangle - 2 \cdot SP_{\boldsymbol{d}_i}(\boldsymbol{x}) \tag{5.19}$$

where, $SP_{\boldsymbol{d}_i}(\boldsymbol{x}) = \|\boldsymbol{x}\|_E \cdot cos(\theta)$ is the scalar projection of vector $\boldsymbol{x}$ onto $\boldsymbol{d}_i$. From Equation 5.19, the distance between $\boldsymbol{x}$ and $\boldsymbol{d}_i$ is inversely proportional to the scalar projection $SP_{\boldsymbol{d}_i}(\boldsymbol{x})$. Therefore $SP_{\boldsymbol{d}_i}(\boldsymbol{x})$ can also be used to measure the similarity between $\boldsymbol{x}$ and $\boldsymbol{d}_i$.

In Step 3 of Algorithm 1, column vector $\boldsymbol{d}_{i_0}$ is selected according to the length of the projection of the measurement $\boldsymbol{y}_r$ onto vectors $\hat{\boldsymbol{\Phi}}\boldsymbol{d}_i$ in the first iteration ($k = 0$). It is trivial to recognize driving conditions, when the vehicles are moving in reverse. So, obviously, all the entries in the speed vector $\boldsymbol{x}$ are none negative. Therefore, $\boldsymbol{y}$ and $\hat{\boldsymbol{\Phi}}\boldsymbol{d}_i$ are all in the first quadrant of the space spanned by $\hat{\boldsymbol{\Phi}}\boldsymbol{D}$. So:

$$|\langle \boldsymbol{y}_r, \frac{\hat{\boldsymbol{\Phi}}\boldsymbol{d}_i}{\|\hat{\boldsymbol{\Phi}}\boldsymbol{d}_i\|} \rangle| = \langle \boldsymbol{y}_r, \frac{\hat{\boldsymbol{\Phi}}\boldsymbol{d}_i}{\|\hat{\boldsymbol{\Phi}}\boldsymbol{d}_i\|} \rangle \quad = SP_{\frac{\hat{\boldsymbol{\Phi}}\boldsymbol{d}_i}{\|\hat{\boldsymbol{\Phi}}\boldsymbol{d}_i\|}}(\boldsymbol{y}_r) \tag{5.20}$$

So when $k = 0$ in Equation. 5.13, we have:

$$SP_{\frac{\hat{\boldsymbol{\Phi}}\boldsymbol{d}_{i_0}}{\|\hat{\boldsymbol{\Phi}}\boldsymbol{d}_{i_0}\|}}(\boldsymbol{y}_r) \geq \alpha_i \sup_{i \in [1, T \cdot P]} SP_{\frac{\hat{\boldsymbol{\Phi}}\boldsymbol{d}_i}{\|\hat{\boldsymbol{\Phi}}\boldsymbol{d}_i\|}}(\boldsymbol{y}_r) \tag{5.21}$$

which means vector $\hat{\boldsymbol{\Phi}}\boldsymbol{d}_{i_0}$ could be regarded as the most similar column vector to the measurement $\boldsymbol{y}_r$.

Moreover, according to Equation 5.13, Equation 5.14 and Equation 5.15, mea-

Algorithm 1 Inertial Matching Pursuit Classification

Input: measurement vectors $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_R\}$, dictionary $\boldsymbol{D}$, sampling matrix: $\hat{\boldsymbol{\Phi}}$, number of iterations: K

Output: recognized driving patterns $\{\ell_1, \ell_2, \cdots, \ell_R\}$

Step 1: Set relaxation factor $\alpha_i = 1$ for each column vector $\boldsymbol{d}_i$ of dictionary $\boldsymbol{D}$, so $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_{L \cdot P}]^T = [1, 1, \cdots, 1]^T$. Set $r = 1$.

Step 2: Set $a_i = 0$ for each column vector $\boldsymbol{d}_i$, so $\boldsymbol{a} = [a_1, a_2, \cdots a_{L \cdot P}]^T = [0, 0, \cdots, 0]^T$. Set $k = 0$.

Step 3: For the rth measurement vector $\boldsymbol{y}_r$, find kth best matching $\boldsymbol{d}_{\boldsymbol{i_k}}$, which satisfies:

$$|\langle \boldsymbol{R}^k \boldsymbol{y}_r, \frac{\hat{\boldsymbol{\Phi}} \boldsymbol{d}_{i_k}}{\|\hat{\boldsymbol{\Phi}} \boldsymbol{d}_{i_k}\|} \rangle| \geq \sup_{i \in [1, T \cdot P]} \alpha_i \cdot |\langle \boldsymbol{R}^k \boldsymbol{y}_r, \frac{\hat{\boldsymbol{\Phi}} \boldsymbol{d}_i}{\|\hat{\boldsymbol{\Phi}} \boldsymbol{d}_i\|} \rangle| \tag{5.13}$$

where:

$$\begin{cases} \boldsymbol{R}^k \boldsymbol{y}_r = \boldsymbol{R}^{k-1} \boldsymbol{y}_r - \langle \boldsymbol{R}^{k-1} \boldsymbol{y}_r, \frac{\hat{\boldsymbol{\Phi}} \boldsymbol{d}_{i_{k-1}}}{\|\hat{\boldsymbol{\Phi}} \boldsymbol{d}_{i_{k-1}}\|^2} \rangle \hat{\boldsymbol{\Phi}} \boldsymbol{d}_{i_{k-1}} \\ \boldsymbol{R}^0 \boldsymbol{y}_r = \boldsymbol{y}_r \end{cases} \tag{5.14}$$

Step 4: Update a_i:

$$a_i = \langle \boldsymbol{R}^{k-1} \boldsymbol{y}_r, \frac{\hat{\boldsymbol{\Phi}} \boldsymbol{d}_{i_{k-1}}}{\|\hat{\boldsymbol{\Phi}} \boldsymbol{d}_{i_{k-1}}\|^2} \rangle \tag{5.15}$$

Step 5: If $k < K$, then $k = k + 1$ and go to Step 3, else go to Step 6.

Step 6: Calculate distance dis_l for each type of driving condition.

$$dis_l = \|\boldsymbol{D}\boldsymbol{a} - \sum_{i=(l-1) \cdot P+1}^{l \cdot P} \boldsymbol{d}_i \cdot a_i\|, l = 1, 2, \cdots, L \tag{5.16}$$

Step 7: Update $\boldsymbol{\alpha}$ by using Equation 5.28, and recognize rth driving pattern

$$\ell_r = arg \min_{l=1,2,\cdots,L} dis_l \tag{5.17}$$

Step 8: If $r < R$, $r = r + 1$ and go to Step 2, else Stop

return $\{\ell_1, \ell_2, \cdots, \ell_R\}$

surement $\boldsymbol{y}_r$ could be presented as follows:

$$\boldsymbol{y}_r = \hat{\boldsymbol{\Phi}}\boldsymbol{D} \cdot \boldsymbol{a} + \boldsymbol{R}^{K+1}\boldsymbol{y}_r = \sum_{i=1}^{K} \hat{\boldsymbol{\Phi}}\boldsymbol{d}_{i_k} \cdot a_i + \boldsymbol{R}^{K+1}\boldsymbol{y}_r \tag{5.22}$$

where $\boldsymbol{R}^{K+1}\boldsymbol{y}_r$ is the residue after K iterations. Furthermore, according to Section 5.2.4, we have

$$\lim_{k \to +\infty} \|\boldsymbol{R}^k \boldsymbol{y}_r\| = 0 \tag{5.23}$$

hence,

$$\boldsymbol{y}_r \approx \sum_{i=1}^{K} \hat{\boldsymbol{\Phi}}\boldsymbol{d}_{i_k} a_i \tag{5.24}$$

So measurement $\boldsymbol{y}_r$ could be regarded as the linear combination of vector $\hat{\boldsymbol{\Phi}}\boldsymbol{d}_{i_k}$. On the other hand, $\boldsymbol{d}_{i_k}$ are the speed vectors selected from the training dataset, which could be also regarded as the speed features, if the dictionary is constructed by using some dictionary-learning techniques. So $\hat{\boldsymbol{\Phi}}\boldsymbol{d}_{i_k}$ are the features of speed measurements. Therefore, IMPC is in fact always selecting top K significant features to approximate measurement vector $\boldsymbol{y}_r$.

5.2.2 Dictionary for IMPC

Given data set $\boldsymbol{T}$ (Equation 5.11), instead of randomly selecting P vectors from each type l, we are trying to choose P more representative speed patterns to form our dictionary $\boldsymbol{D}$.

In each data subset $\boldsymbol{T}_l$, the mean of all speed vectors $\boldsymbol{x}_n^l, n = 1 \cdots N$ is calculated as follows:

$$\boldsymbol{\mu}_l = \frac{1}{N} \cdot \sum_{n=1}^{N} \boldsymbol{x}_n^l \tag{5.25}$$

Then the distance between each training point $\boldsymbol{x}_n^l$ and $\boldsymbol{\mu}_l$ is calculated:

$$\begin{aligned} d_M(\boldsymbol{x}_n^l, \boldsymbol{\mu}_l) =& \|\boldsymbol{S}^{-1/2}(\boldsymbol{x}_n^l - \boldsymbol{\mu}_l)\|_E \\ =& [(\boldsymbol{x}_n^l - \boldsymbol{\mu}_l)^T \boldsymbol{S}^{-1} (\boldsymbol{x}_n^l - \boldsymbol{\mu}_l)]^{1/2} \end{aligned} \tag{5.26}$$

The Mahalanobis distance $d_M(\cdot, \cdot)$ is used here to measure the similarity between $\boldsymbol{x}_n^l$ and $\boldsymbol{\mu}_l$. $\|\cdot\|_E$ donates the Euclidean norm and $\boldsymbol{S}$ is the covariance matrix calculated from the data subset $\boldsymbol{T}_l$.

The top P speed vectors $\boldsymbol{x}_n^l$ with smallest distance to the mean vector $\boldsymbol{\mu}_l$ are selected to construct the sub-dictionary $\boldsymbol{D}_l$ in this manuscript and entire dictionary $\boldsymbol{D}$ can be constructed by using Equation 5.12. A more robust dictionary can be also designed by dictionary-learning techniques [166, 167, 168].

5.2.3 Inertial Factor

Since the velocity of vehicles is bounded, the current location of a vehicle should be highly related to or dependent on its previous location. Therefore, the driving condition of measurements $\boldsymbol{y}_r$ should be highly correlated to $\boldsymbol{y}_{r-1}$ and $\boldsymbol{y}_{r+1}$ in most cases, except at the junction of two different driving conditions, such as the intersection of highway and urban. Based on this assumption, current recognized pattern and the uncertainty degree of this recognition could be used to improve the next recognition.

Suppose the measurement vector $\boldsymbol{y}_{r-1}$ has been recognized as the ℓ_{r-1}th driving condition, and the L distances $dis_l, l = 1, 2, \cdots, L$ have also been calculated by using Equation 5.16 in Algorithm 1 for $\boldsymbol{y}_{r-1}$. The uncertainty degree of this $(r-1)$th recognition could be then defined as:

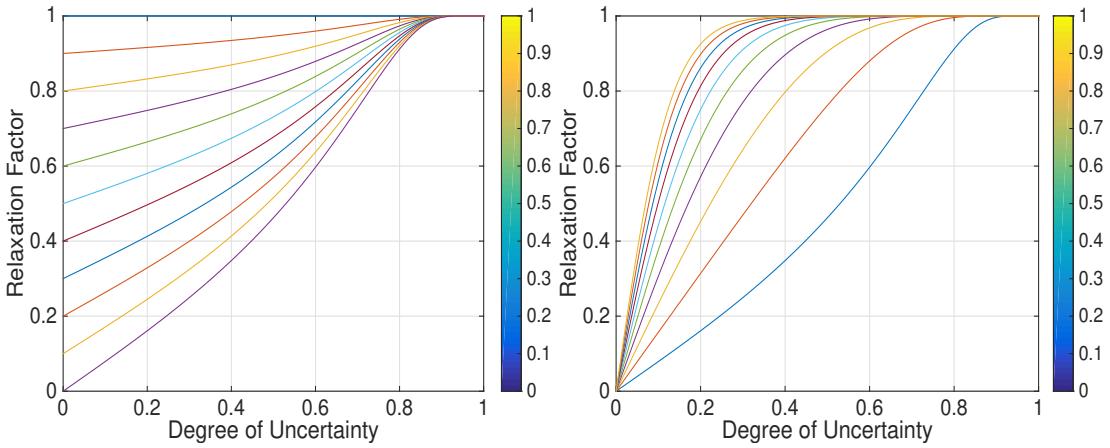
$$u_{r-1} = \frac{dis_{\ell_{r-1}}}{\min_{l \in [1,\ell_{r-1}) \cup (\ell_{r-1},L]} dis_l} \tag{5.27}$$

Then the relaxation factor in Equation 5.13 could be used as an inertial factor in recognizing the rth measurement vector $\boldsymbol{y}_r$. This inertial factor enables the column vectors $\boldsymbol{d}_i$ from the driving condition of type ℓ_{r-1} to be more easily selected by Equation 5.13. The inertial factor α_i, for the lth measurement $\boldsymbol{y}_r$, could be calculated by using the function defined in Equation 5.28.

$$\alpha_i = \begin{cases} 1, & (\ell_{r-1} - 1) \cdot P + 1 \leq i \leq \ell_{r-1} \cdot P \\ \dfrac{2A}{1 + e^{-B \cdot tan(\frac{\pi}{2} \cdot u_{r-1})}} + 1 - 2A, & others \end{cases} \tag{5.28}$$

where, $tan(\frac{\pi}{2} \cdot u_{r-1})$ maps uncertainty u_{r-1} from $[0, 1]$ to $[0, +\infty]$. When $i \in [1, (\ell_{r-1} - 1) \cdot P] \cup [\ell_{r-1} \cdot P + 1, L \cdot p]$, α_i is in the range of $[1 - A, 1]$. Moreover, B here is used to control the shape of the function. More details about parameters A, B and Equation 5.28 could be found in Figure 5.3a and Figure 5.3b

Obviously, this function updates the inertial factors for $\boldsymbol{y}_r$ according to the previous recognition result ℓ_{r-1}. More complicated algorithm could be designed to update



(a) $B = 3$, iterating A from 0 to 1. (b) $A = 1$, iterating B from 1 to 10.

Figure 5.3: Function used to update inertial factors

the inertial factors according to the previous sequence of results $[\ell_1, \ell_2, \cdots \ell_{r-1}]$.

5.2.4 Convergence and Iteration Number K

Although the relaxation factor (inertial factor) α_i is different for each column $\boldsymbol{d}_i$, IMPC could be still proved to be convergent. According to Equation 5.14, we have:

$$\boldsymbol{R}^k\boldsymbol{y}_r = \langle \boldsymbol{R}^k\boldsymbol{y}_r, \frac{\hat{\boldsymbol{\Phi}}\boldsymbol{d}_{i_k}}{\|\hat{\boldsymbol{\Phi}}\boldsymbol{d}_{i_k}\|^2}\rangle \hat{\boldsymbol{\Phi}}\boldsymbol{d}_{i_k} + \boldsymbol{R}^{k+1}\boldsymbol{y}_r \tag{5.29}$$

Then, together with Equation 5.13, we have:

$$\begin{aligned} \frac{\|\boldsymbol{R}^{k+1}\boldsymbol{y}_r\|^2}{\|\boldsymbol{R}^k\boldsymbol{y}_r\|^2} &= 1 - |\langle \frac{\boldsymbol{R}^k\boldsymbol{y}_r}{\|\boldsymbol{R}^k\boldsymbol{y}_r\|}, \frac{\hat{\boldsymbol{\Phi}}\boldsymbol{d}_{i_k}}{\|\hat{\boldsymbol{\Phi}}\boldsymbol{d}_{i_k}\|}\rangle|^2 \\ &\leq 1 - \mu^2(\boldsymbol{R}^k\boldsymbol{y}_r, \boldsymbol{D}) \end{aligned} \tag{5.30}$$

where, $\mu(\boldsymbol{r}, \boldsymbol{D})$ is defined by:

$$\mu(\boldsymbol{r}, \boldsymbol{D}) = \max_i \{\alpha_i \cdot |\langle \frac{\boldsymbol{r}}{\|\boldsymbol{r}\|}, \frac{\hat{\boldsymbol{\Phi}}\boldsymbol{d}_i}{\|\hat{\boldsymbol{\Phi}}\boldsymbol{d}_i\|}\rangle|\} \tag{5.31}$$

Note that $\alpha_i \leq 1$. Therefore, according to [151], $\mu(\boldsymbol{r}, \boldsymbol{D})$ has a strictly positive lower bound, $0 < \mu(\boldsymbol{r}, \boldsymbol{D}) < 1$, hence we have:

$$\lim_{k \to +\infty} \|\boldsymbol{R}^k\boldsymbol{y}_r\| = 0 \tag{5.32}$$

Therefore, when iteration number $K \to +\infty$, measurement vector $\boldsymbol{y}_r$ can be explicitly expressed, and together with Equation 5.4 and Equation 5.22 speed signal $\boldsymbol{x}_r$ could also be accurately reconstructed.

On the other hand, IMPC does not require to explicitly find speed vector $\boldsymbol{x}_r$ and measurement vector $\boldsymbol{y}_r$ for pattern recognition. Therefore, K is only needed to be a relatively big positive number. More investigation about parameter K is given experimentally in the next section.

5.3 Experiments

Numerical experiments are carried out in this Section. As shown in Figure 4.7, these eight different driving cycles $\boldsymbol{C}_u^l, l = 1, 2, 3, 4$ and $u = 1, 2$, are used to simulate four

different types of driving conditions.

For the lth type of driving condition, the combined driving cycle for simulation is $\boldsymbol{C}^l = [(\boldsymbol{C}_1^l)^T, (\boldsymbol{C}_2^l)^T]^T$. Then the nth speed signal $\boldsymbol{x}_n^l$ of this driving condition is generated as follows:

$$\begin{aligned} \boldsymbol{x}_n^l = \{c_{nj}\}, \quad & j = 1, 2, \cdots, \Delta T \\ & n = 1, 2, \cdots, L - \Delta T + 1 \end{aligned} \tag{5.33}$$

where: ΔT is the length of sampling time, L is the length of the combined driving cycle $\boldsymbol{C}^l$. c_{nj} is the $(n-1+j)$th entry of driving cycle $\boldsymbol{C}^l$.

Now we have our experiment dataset $\boldsymbol{T}$ as follows:

$$\boldsymbol{T} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, \boldsymbol{T}_3, \boldsymbol{T}_4\} \tag{5.34}$$

For each type of driving condition:

$$\boldsymbol{T}_l = \{(\boldsymbol{x}_n, l) | 1 \leq n \leq N\}, l = 1, 2, 3, 4 \tag{5.35}$$

where, we have $N = 1000$ pairs of data for each driving condition and 4000 pairs of data totally. Note that, for IMPC, $\boldsymbol{x}_n^l$ is used to simulate the real-time vehicle speed, which will be sampled by Compressed Sensing to generate measurements $\boldsymbol{y}_r$, and $\boldsymbol{x}_n^l$ is unknown to both Compressed Sensing and IMPC. In order to give a realistic estimation of generalization, ten-fold cross-validation is used in this section to compare the performance of all DPR systems.

5.3.1 Comparison between IMPC and other DPR systems

In this part, IMPC is compared with 4 different DPR systems as listed in Table 5.1. In DPR system 1 and 2, four features, mean velocity, maximal acceleration, minimal acceleration, and idle rate are extracted directly from vehicle speed $\boldsymbol{x}_n^l$ according to [110]. In DPR system 3 and 4, the speed signal $\boldsymbol{x}_n^l$ is unknown and used to generate the speed measurements $\boldsymbol{y}_r$ according to Equation 5.2. Same four features are extracted from the reconstructed speed signal $\hat{\boldsymbol{x}}_n^l$, which is reconstructed from $\boldsymbol{y}_r$ by solving the following $l_1 - l_2$ optimization problem:

$$\min_{\boldsymbol{a}} \frac{1}{2} \|\boldsymbol{y}_r - \hat{\boldsymbol{D}} \cdot \boldsymbol{a}\|_2^2 + T \cdot \|\boldsymbol{a}\|_1 \tag{5.36}$$

where $\hat{\boldsymbol{D}} = \hat{\boldsymbol{\Phi}} \cdot \boldsymbol{D}$ and $\hat{\boldsymbol{x}}_n^l$ can be recovered by $\hat{\boldsymbol{x}}_n^l = \boldsymbol{D} \cdot \boldsymbol{a}$. The MFISTA [169, 170] is employed here to solve this $l_1 - l_2$ problem. In DPR system 1 and 3, a Support Vector Machine (SVM) is trained to recognize current driving condition from aforementioned four features, and in DPR system 2 and 4, a Feed-forward Neural Network (FFNN) with one hidden layer is trained. There are 7 nodes in the hidden layer. In DPR system $3-5$, the number of measurements m for compressed sensing is set to be $round(60\% \times \Delta T)$, here ΔT is the recognition period, which equals to the length of vector $\boldsymbol{x}_n^l$. The dictionary $\boldsymbol{D}$ is constructed from the training dataset according to the method in Section 5.2.2. Finally, in IMPC, the iteration number $K = 10$, and the parameters of the inertial factor updating function $A = 0.075$ and $B = 5$.

Results of this comparison can be found in Figure 5.4 and Figure 5.5. In these plots, the x-axis is the recognition period (the length of speed vector $\boldsymbol{x}_n^l$), while the y-axis is the error of driving pattern recognition. First of all, with the increase of the recognition period, the accuracy of each DPR system is improved. Secondly, DPR system 3 and 4 theoretically enjoy a relatively high sampling efficiency due to the employment of compressed sensing. However, their accuracies are worse than system 1 and 2. The reason for the worse performances of system 3 and 4 is that the aforementioned four features are not designed for classification based on compressed sensing. Actually, these four features are very sensitive to the recover accuracy of speed signal $\boldsymbol{x}_n^l$ in the experiment. For example, the forth feature, idle rate ($I = length(\boldsymbol{x}_n^l == 0)/\Delta T$) does not work well on the recovered speed signal $\hat{\boldsymbol{x}}_n^l$, because zero speed is usually recovered to a very small number by compressed sensing. (See Figure 5.6 and Figure 5.7). Finally, IMPC has a relatively better performance compared to the other four systems, even when the recognition period is less than 100 seconds. However, as shown in the plots, the standard deviation of the IMPC accuracy is larger than the other four systems, especially when the recognition period is small. This actually means that the performance of the IMPC is not stable, when the speed measurements are too few to provide enough statistical information for classification.

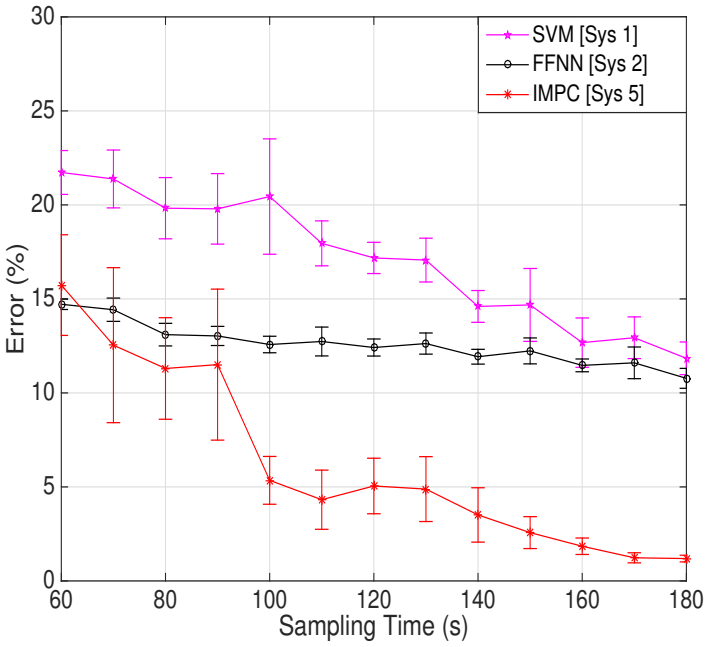


Figure 5.4: Comparison among SVM, FFNN and IMPC

5.3.2 Investigation into the necessity of sampling/recovering speed signal completely

As we mentioned in Section 5.1.2 and 5.2, do we really need to sample and recover speed signal $\boldsymbol{x}_n^l$ completely and accurately for driving pattern recognition? In order to discuss this question, experiments are conducted here on the measurement number m in compressed sensing and IMPC iteration number K. As we know, smaller m and k mean less measurements from the original speed signal and less accuracy of the

Table 5.1: DPR systems comparison

DPR Sys	Classifier	Sampling Method
1	Support Vector Machine	Nyquist–Shannon Sampling
2	Feed-forward Neural Network	Nyquist–Shannon Sampling
3	Support Vector Machine	Compressed Sensing
4	Feed-forward Neural Network	Compressed Sensing
5	IMPC	Compressed Sensing

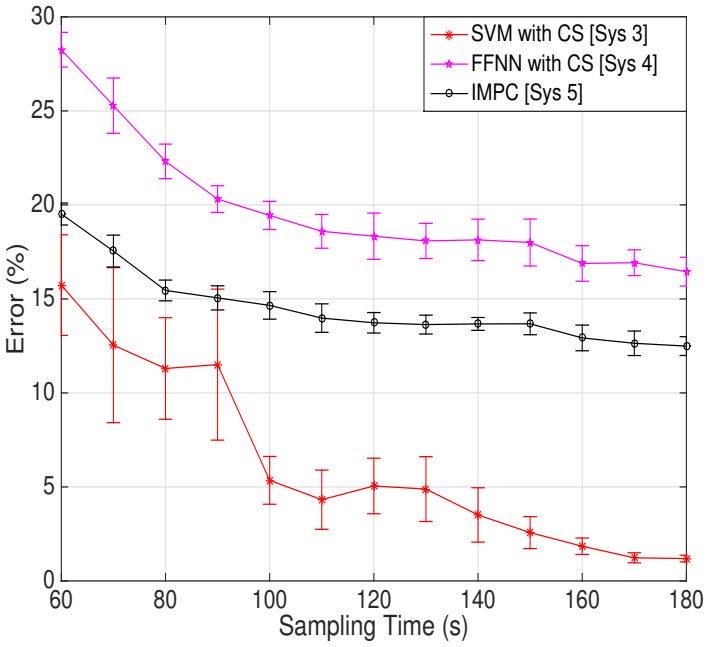


Figure 5.5: Comparison among SVM with CS, FFNN with CS and IMPC

speed reconstruction. For example, in Figure 5.6 and Figure 5.7, a randomly selected speed signal is recovered by using the coordinate vector $\boldsymbol{a}$ (which is generated by IMPC) and dictionary $\boldsymbol{D}$ in Equation 5.16 under different value of parameter m and K. Obviously, larger m and K give us better reconstruction results. More detailed information about the impacts of parameter m and parameter K on the speed signal reconstruction can be found in Figure 5.8. The reconstruction accuracy is calculated as follows:

$$error = \frac{||\boldsymbol{x}_n^l - \hat{\boldsymbol{x}}_n^l||}{||\boldsymbol{x}_n^l||} \tag{5.37}$$

where $\boldsymbol{x}_n^l$ is the raw speed signal and $\hat{\boldsymbol{x}}_n^l$ is its reconstruction.

However, as shown in Figure 5.9, the results of the driving pattern recognition conducted by IMPC seems to be much less sensitive to the variation of both m and K, compared to the results of speed signal reconstruction in Figure 5.8. According to Figure 5.9, parameter K has very tiny impact on the performance of the IMPC, and the IMPC accuracy seems to be stable after $m >= 20$. Based on these experiment results, conclusion can be drawn that the DPR system, at least the IMPC, does not need the speed signal to be fully sampled or accurately recovered. As long as the the

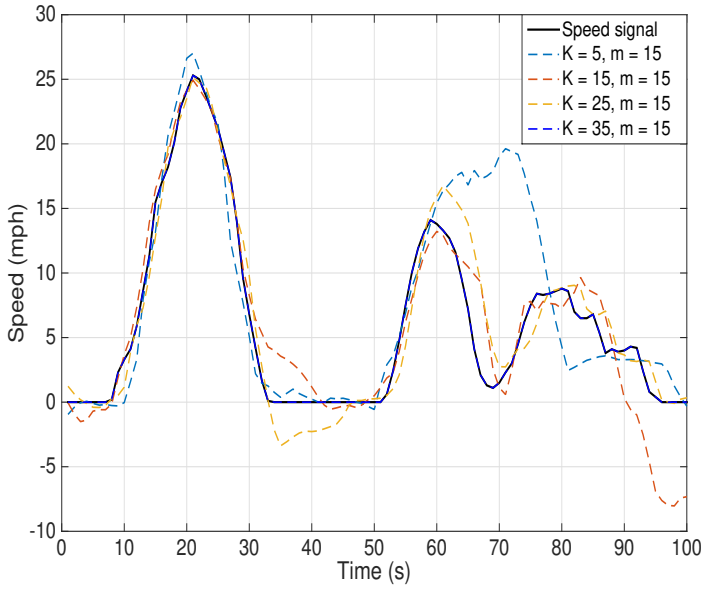


Figure 5.6: Comparison among different iteration number K (m is fixed to 20)

speed signal contains significant statistical information, the IMPC can recognize it.

5.3.3 Experiment by using practical vehicle speed signals

The driving trip started from A: Palo Alto, California, through the highway 280 and ended at B: Sunnyvale, California as show in Figure 5.10. We had firstly a flowing city driving condition, then a highway, and finally another flowing city driving condition. The duration of the entire trip is 23 minutes and 16 seconds and the speed signal was sampled at $1Hz$. So the length of the sampled speed signal $\boldsymbol{x}_{real}$ is $L = 1396$. In this trip, we had totally 881 seconds of flowing city and 515 seconds of highway driving condition. The speed signals can be found in Figure 5.11.

The recognition performance of the IMPC and the system 1 and 2 in Table 5.1 are compared by using the practical vehicle speed $\boldsymbol{x}_{real}$. All these three system are firstly trained by using the speed signal data of the highway and the flowing city driving cycles in Fig 4.7. Given a recognition period ΔT, the top $floor(L/\Delta T) \cdot \Delta T$ entries in the practical speed vector $\boldsymbol{x}_{real}$ are used to form the testing dataset. Here, the function $floor(*)$ maps a real number to the largest previous integer. There are $floor(L/\Delta T)$ data in the testing dataset and the length of each testing data is ΔT. All other parameters in this experiment are same with those in Section 5.3.1.

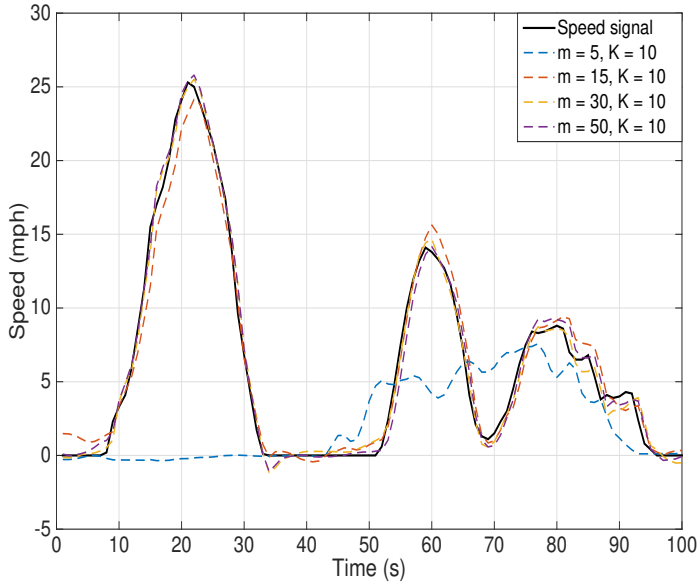


Figure 5.7: Comparison among different measurement number m (K is fixed to 50)

When the recognition period $\Delta T = 180$ seconds, the top $floor(1396/180) \cdot 180 = 1260$ entries are used to generate the 7 testing data. Note that in that 1260 seconds of driving, we have 745 seconds of flowing city and 515 seconds of highway driving. So the percentage of the city driving is 59.13%. The experiment results are illustrated in Figure 5.12. It can be found that all these three systems are able to recognize the driving conditions correctly when the recognition period is 180 seconds. However, these DPR systems only updates the driving condition 7 times during the entire trip. If we have a vehicle energy management system or a control system based on the recognition of driving conditions, then this system only adjusts itself 7 times during the entire trip because of the larger recognition period. This could lead to a bad dynamic property of this energy management system or control system. Moreover, the larger recognition period also leads to a lower DPR system resolution and higher relative error of the city percentage estimation compared to the system with 100-second recognition period. The results of the DPR systems with 100-second recognition period can be found in Figure 5.13. As shown in this plot, only the IMPC is still able to recognize all driving patterns correctly, while SVM and FFNN both have some recognition errors. As shown in Table 5.2, the IMPC with 100-second recognition period also enjoys better system resolution (defined in Equation 5.1) and lower error

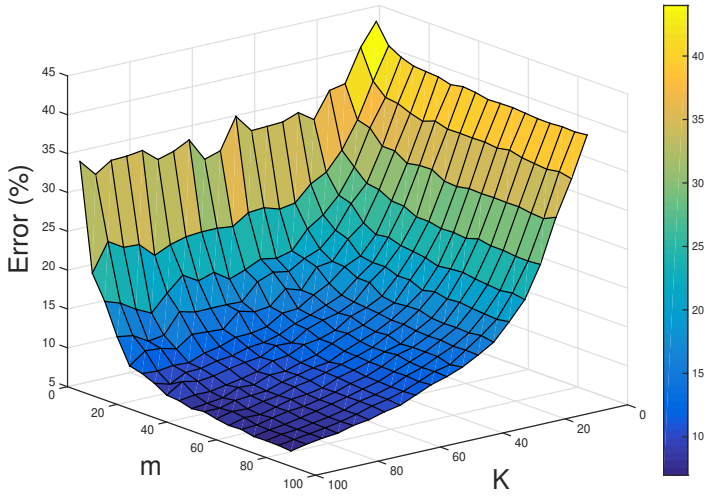


Figure 5.8: The impacts of m and K on the signal reconstruction error ($\Delta T = 100$)

of city percentage estimation.

Table 5.2: IMPC results comparison (180 seconds vs. 100 seconds)

	Ground-truth percentage	**Estimated percentage**	**Relative error**	**System resolution**
180 seconds	59.13%	57.14%	3.37%	7.14%
100 seconds	60.28%	61.54%	2.09%	3.85%

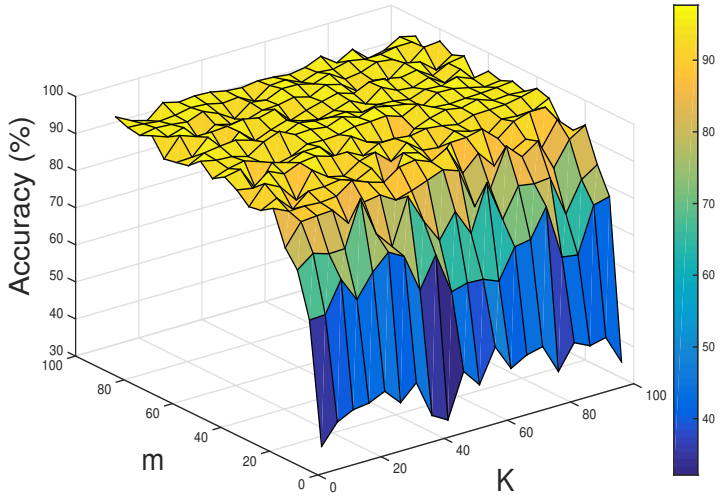


Figure 5.9: The impacts of m and K on the accuracy of IMPC ($\Delta T = 100$)

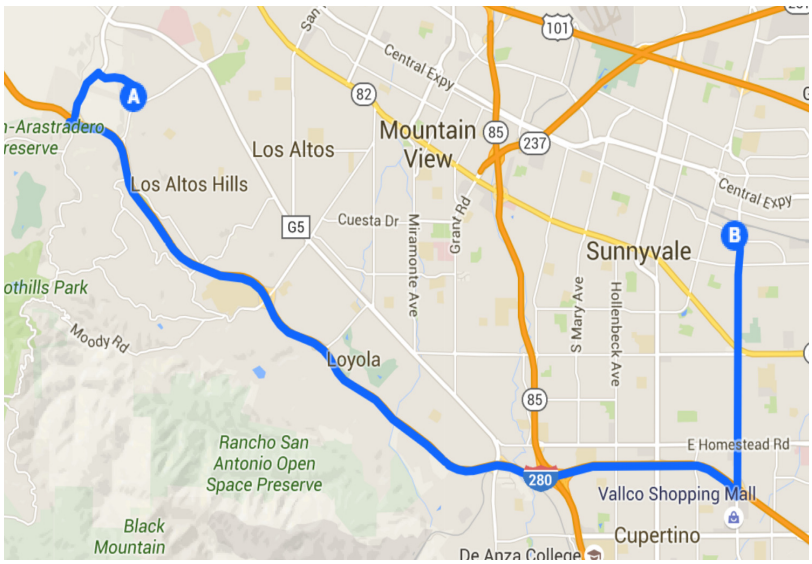


Figure 5.10: The 23-minute experimental trip on map

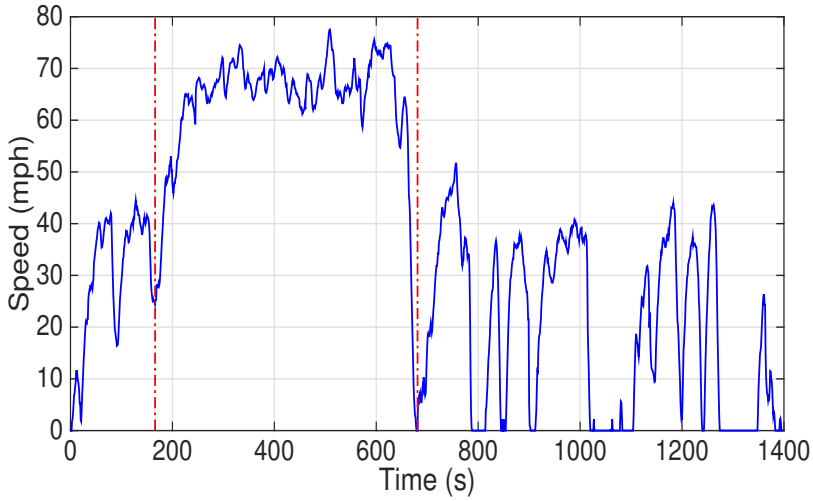


Figure 5.11: Speed $\boldsymbol{x}_{AB}$ of the 23-minute experimental trip

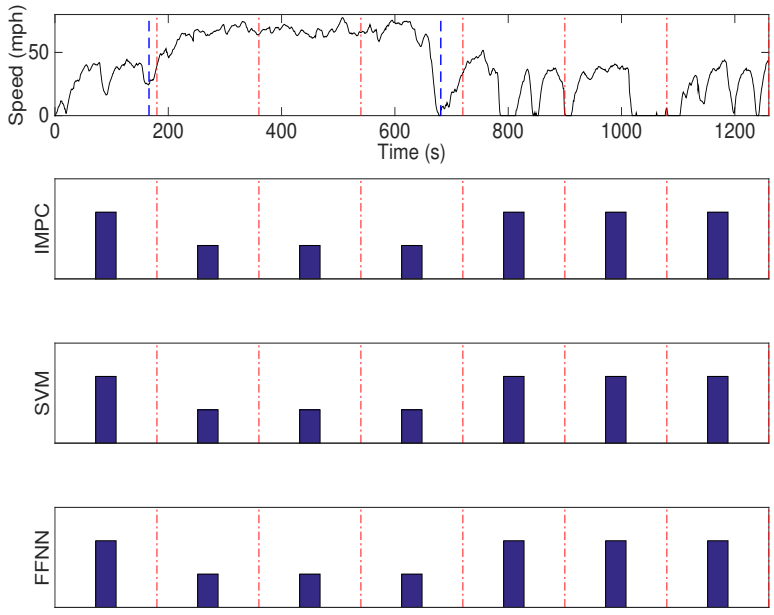


Figure 5.12: Recognition comparison when sample time $\Delta T = 180$ seconds: taller bar - flowing city, shorter bar - highway

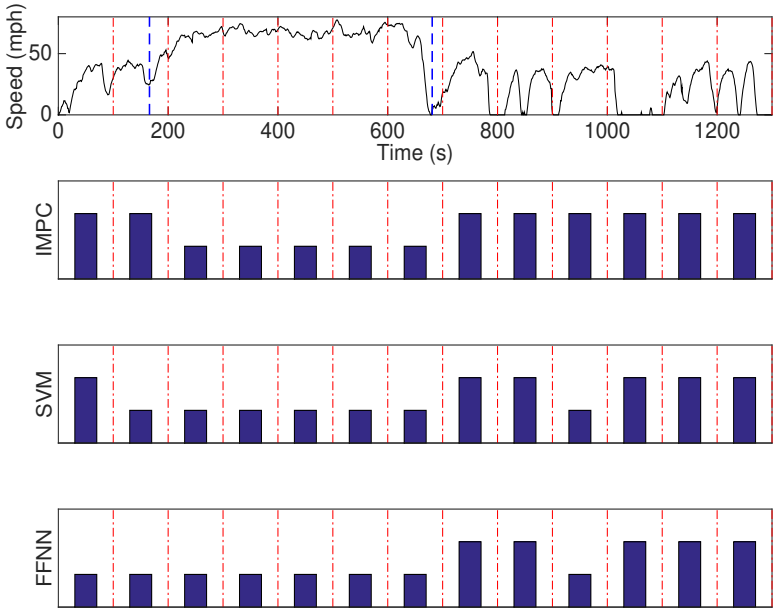


Figure 5.13: Recognition comparison when sample time $\Delta T = 100$ seconds: taller bar - flowing city, shorter bar - highway

Chapter 6

Intelligent Energy Allocation Algorithm for HESS

As discussed in Chapter 2, a higher system efficiency of HESS, as well as a longer battery life could be achieved by optimizing the energy allocation between battery and ultracapacitor in the HESS. Note that, in a battery-only energy storage system, we are not able to fully control the battery. All we can do is to monitor the battery status and to control its thermal performance by actively control the temperature of it, but we are not able to control its current based on a given requested power. Thus, it is not possible for us to improve its efficiency. Only in a hybrid system, we have the freedom to change the battery current, by asking the UC to provide a part of the requested power. Also note that, since the battery voltage is determined by SOC and other factors, including environment temperature, battery aging and etc., optimized energy allocation actually only means separating the required current between battery and ultracapacitor properly. A lot of researches have been conducted in optimizing the battery charing/discharging current, such as [71, 171, 172, 173].

By using the algorithms proposed in Chapter 4 and Chapter 5, we are able to recognize current driving pattern accurately in a small time period by using less features with better observabilities. However, note must be made here that the Intelligent Energy Allocation (IEA) algorithm proposed in chapter is not based on those two algorithms. IEA can still work if the traditional DPR algorithms are used for it. Based on the topology comparison in Chapter 3, we select the semi-active structure with an ideal DC-DC in ultracapacitor side as our target HESS in this chapter. In this chapter, the IEA algorithm is developed based on reinforcement learning. This algorithm

will first recognize the current trip mode based on the series of driving patterns, and then optimizes the allocation of the energy flow between battery and ultracapacitor according to the trip mode. This proposed algorithm is able to adapt itself to different daily trip modes. It keeps learning to generate optimal policies to prolong the battery life and improve the HESS efficiency from the data collected during the daily driving. Finally, this proposed IEA algorithm is simulated and evaluated by using the joint simulation framework proposed in Chapter 3.

6.1 Introduction

Dynamic Programming (DP) and Reinforcement Learning algorithm (RL), including the Monte Carlo method and Temporal-Difference Learning are introduced int his section.

6.1.1 Dynamic Programming

Dynamic programming (DP) is an optimization approach that transforms a complex problem into a sequence of simpler problems; its essential characteristic is the multistage nature of the optimization procedure. DP can be used to solve many problems in time $O(n^2)$ or $O(n^3)$ for which a naive approach would take exponential time. DP provides a general framework for analyzing many problem types. Within this framework a variety of optimization techniques can be employed to solve particular aspects of a more general formulation. Usually creativity is required before we can recognize that a particular problem can be cast effectively as a dynamic program; and often subtle insights are necessary to restructure the formulation so that it can be solved effectively.[174]

Given a complex optimization problem, which could be divided into N subproblems with objective function $R(\boldsymbol{s})$, where $\boldsymbol{s}$ is the state vector. DP is trying to minimize the following value function by figuring out the optimal policy $\pi(\boldsymbol{s})$:

$$V_\pi(\boldsymbol{s}_k) = \sum_{i=k}^{N} \gamma^{i-k} R(\boldsymbol{s}_i, \boldsymbol{a}_i) \tag{6.1}$$

where, $0 < \gamma \leq 1$ a discount factor and $\boldsymbol{a}_i$ is the action vector for the ith subproblem, which is generated by $\boldsymbol{a}_i = \pi(\boldsymbol{s}_i)$. N is the number of the subproblems, which could be a positive integer or even ∞. Note that we have no assumptions on the state and

action sets of the DP problem, $\mathcal{S}$ and $\mathcal{A}(\boldsymbol{s})$, for $\boldsymbol{s} \in \mathcal{S}$. They could be continuous or discreet, finite or even infinite. By writing Equation 6.1 as

$$V_\pi(\boldsymbol{s}_k) = R(\boldsymbol{s}_k, \boldsymbol{a}_k) + \gamma \sum_{i=k+1}^{N} \gamma^{i-k-1} R(\boldsymbol{s}_i, \boldsymbol{a}_i) \tag{6.2}$$

one sees that a difference equation equivalent to is given by

$$V_\pi(\boldsymbol{s}_k) = R(\boldsymbol{s}_k, \boldsymbol{a}_k) + \gamma V_\pi(\boldsymbol{s}_{k+1}) \tag{6.3}$$

That is, instead of evaluating the sum of all steps (see Equation 6.1), one can solve the difference equation to evaluate the value function by using a current policy $\boldsymbol{a}_k = \pi(\boldsymbol{s}_k)$. Bellman's principle [175] is a cornerstone of solving DP problems, and states that "An optimal policy has the property that no matter what the previous decisions (i.e. controls) have been, the remaining decisions must constitute an optimal policy with regard to the state resulting from those previous decisions". In terms of equations, this means that

$$V_\pi^*(\boldsymbol{s}_k) = \min_{\pi(\cdot)} \{R(\boldsymbol{s}_k, \pi(\boldsymbol{s}_k)) + \gamma V_\pi^*(\boldsymbol{s}_{k+1})\} \tag{6.4}$$

This is known as the Bellman optimality equation, and the optimal policy could be expressed as

$$\pi^*(\boldsymbol{s}_k) = \text{argmin}_{\pi(\cdot)} \{R(\boldsymbol{s}_k, \pi(\boldsymbol{s}_k)) + \gamma V_\pi^*(\boldsymbol{s}_{k+1})\} \tag{6.5}$$

Note that $\boldsymbol{s}_{k+1}$ is the system state in next step or next subproblem. If there exists a model $\boldsymbol{F}(\cdot)$ to describe the system's or the object's behavior, then the next-step state can be obtained by $\boldsymbol{s}_{k+1} = \boldsymbol{F}(\boldsymbol{s}_k, \boldsymbol{a}_k)$. Moreover, this model could be in any forms, such as a state-space difference equation $\boldsymbol{s}_{k+1} = f(\boldsymbol{x}_k) + g(\boldsymbol{x}_k) \cdot \boldsymbol{a}_k$ or a Markov decision processes (MDP) model with a set of transition probabilities, $p(s'|s, a) = \boldsymbol{Pr}\{S_{k+1} = s'|S_k = s, A_k = a\}$. As long as the model exists, then the DP problem can be solved by using a backwards-in-time procedure. One must already know the optimal policy at time $k + 1$, so that the optimal policy at time k can be determined according to Equation 6.5. This is the basis for Dynamic Programming algorithms in extensive use in control system theory, Operations Research, and elsewhere. However, this backwards-in-time method is a by nature off-line planning method. It is relatively

slow, inefficient, and cost a lot of computation resources, if the to-be-solved DP problem has large step number N, or large start set $\mathcal{S}$ or large action set $\mathcal{A}(\boldsymbol{s})$. [176]

Policy Evaluation

The evaluation of the value functions $\boldsymbol{V}_{\pi(\cdot)}(\boldsymbol{s}_k), \boldsymbol{s}_k \in \mathcal{S}$ of the current policy using the Bellman Equation amounts to determining the value of using the policy starting in all current states $\boldsymbol{s}_k$. (Note, $\boldsymbol{V}_{\pi(\cdot)}(\boldsymbol{s}_k)$ here is vector of all $V_{\pi(\cdot)}(\boldsymbol{s}_k)$.) This is called a full backup in [176] and can involve significant computation. In fact, it can be shown that the Bellman equation is a fixed point equation. That is, given an admissible policy $\boldsymbol{a}_k = \pi(\boldsymbol{s}_k)$, has a unique fixed point $\boldsymbol{V}_{\pi(\cdot)}(\boldsymbol{s}_k)$, and the following contraction map

$$\boldsymbol{V}^{i+1}(\boldsymbol{s}_k) = R(\boldsymbol{s}_k, \pi(\boldsymbol{s}_k)) + \gamma \boldsymbol{V}^{i}(\boldsymbol{s}_{k+1}) \tag{6.6}$$

can be iterated starting with any value $\boldsymbol{V}^0(\boldsymbol{s}_k)$, and there results in the limit $\boldsymbol{V}^i(\boldsymbol{s}_k) \rightarrow \boldsymbol{V}_{\pi(\cdot)}(\boldsymbol{s}_k)$, where, i is the iteration number, $\boldsymbol{V}_{\pi(\cdot)}(\boldsymbol{s}_k)$ is the values of the given policy $\pi(\boldsymbol{s}_k)$, $\boldsymbol{V}^i(\boldsymbol{s}_k)$ is the values in the ith iteration.

Policy Iteration and Value Iteration

Instead of using the backwards-in-time procedure mentioned previously, there actually exist several other algorithms to solve a DP problem efficiently, such as Policy Iteration (PI) and Value Iteration (VI).

In the PI method, the initial given policy π_0 would be evaluated by using the aforementioned policy elvaluation method, so that the values $\boldsymbol{V}_{\pi_0(\cdot)}(\boldsymbol{s}_k), \boldsymbol{s}_k \in \mathcal{S}$ can be calculated first. Given these values $\boldsymbol{V}_{\pi_0(\cdot)}$ a better policy π_1 could be found out according to

$$\pi_1(\boldsymbol{s}_k) = \text{argmin}_{\pi(\cdot)}\{R(\boldsymbol{s}_k, \pi(\boldsymbol{s}_k)) + \gamma V_{\pi_0}(\boldsymbol{s}_{k+1})\} \tag{6.7}$$

So once a policy, π_j, has been improved using $\boldsymbol{V}_{\pi_j}$ to yield a better policy, π_{j+1}, we can then compute $\boldsymbol{V}_{\pi_{j+1}}$ and improve it again to yield an even better π_{j+2}. We can thus obtain a sequence of monotonically improving policies and value functions:

$$\pi_0 \xrightarrow{E} \boldsymbol{V}_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} \boldsymbol{V}_{\pi_1} \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} \boldsymbol{V}_{\pi_*} \tag{6.8}$$

where $overset E\rightarrow$ denotes a policy evaluation and $\xrightarrow{I}$ denotes a policy improvement. Each policy is guaranteed to be a strict improvement over the previous one

(unless it is already optimal). this process must converge to an optimal policy and optimal value function in a finite number of iterations. [174]

One drawback to policy iteration is that each of its iterations involves policy evaluation, which may itself be a protracted iterative computation requiring multiple sweeps through the state set. If policy evaluation is done iteratively, then convergence exactly to $\boldsymbol{V}_\pi$ occurs only in the limit. In fact, the policy evaluation step of policy iteration can be truncated in several ways without losing the convergence guarantees of policy iteration. One important special case is when policy evaluation is stopped after just one sweep (one backup of each state). This algorithm is called value iteration.

Policy iteration consists of two simultaneous, interacting processes, one making the value function consistent with the current policy (policy evaluation), and the other making the policy greedy with respect to the current value function (policy improvement). In policy iteration, these two processes alternate, each completing before the other begins, but this is not really necessary. In value iteration, for example, only a single iteration of policy evaluation is performed in between each policy improvement. As long as both processes continue to update all states, the ultimate result is typically the same - convergence to the optimal value function and an optimal policy. We use the term generalized policy iteration (GPI) to refer to the general idea of letting policy evaluation and policy improvement processes interact, independent of the granularity and other details of the two processes. Here is a diagram displaying the GPI iterations:

One might also think of the interaction between the evaluation and improvement processes in GPI in terms of two constraints or goals - for example, as two lines in two-dimensional space:

6.1.2 Reinforcement Learning

In the last few years, Reinforcement Learning (RL), also called adaptive (or approximate) dynamic programming (ADP), has emerged as a powerful tool for solving complex sequential decision-making problems. [177]. The power of RL lies in its ability to solve, near-optimally, complex and large-scale problems on which classical DP breaks down. RL emerged as a tool in the artificial intelligence (AI) and neural research communities, where combining DP with derivative-based adaptive function approximations [178] and learning-based methods [179] was advocated in the mid-1980s. The modern science of RL has emerged from a synthesis of notions from

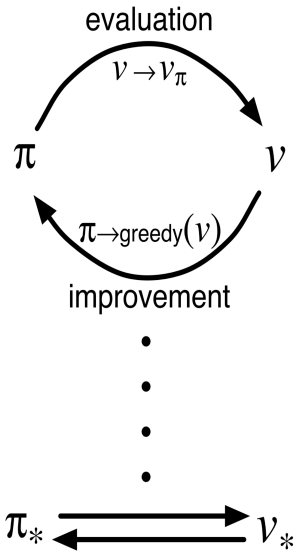


Figure 6.1: Generalized policy iteration

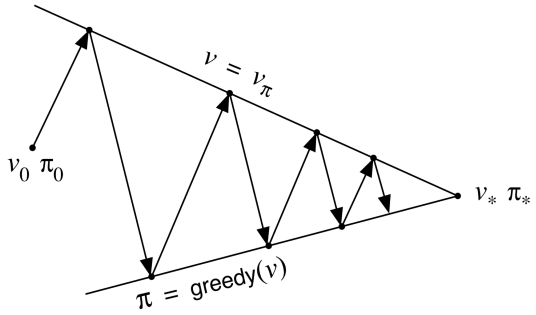


Figure 6.2: Generalized policy iteration converging

four different fields: classical DP, AI (temporal differences), stochastic approximation (simulation), and function approximation (regression, Bellman error, and neural networks).

We call modification of actions based on interactions with the environment reinforcement learning (RL) [180]. There are many types of learning including supervised learning, unsupervised learning, etc. Reinforcement learning refers to an actor or agent that interacts with its environment and modifies its actions, or control policies, based on stimuli received in response to its actions. This is based on evaluative information from the environment and could be called action-based learning. RL implies a cause and effect relationship between actions and reward or punishment. [174]

RL is learning what to do - how to map situations to actions - so as to minimize a

numerical reward signal. It is a means of learning optimal behaviors by observing the response from the environment to non-optimal policies. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These two characteristics, trial-and-error search and delayed reward, are the two most important distinguishing features of reinforcement learning.[176].

The reinforcement learning problem is meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The learner and decision-maker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment. (See Figure 6.3) These interact continually, the agent selecting actions and the environment responding to those actions and presenting new situations to the agent. The environment also gives rise to rewards, special numerical values that the agent tries to minimize over time. A complete specification of an environment defines a task, one instance of the reinforcement learning problem.

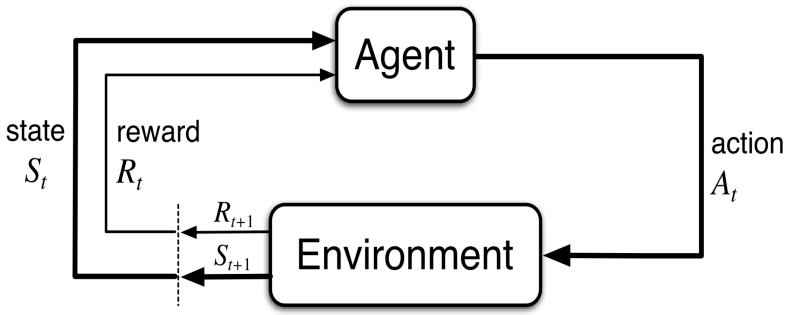


Figure 6.3: The agent-environment interaction in reinforcement learning

Reinforcement Learning Application

RL has been applied in a large number of domains successfully. A few case are enumerated here. Continuous time discounted algorithms were employed for elevator scheduling [181, 182] because the problem structure had a continuous time Markov chain underlying it. The job-shop scheduling problem in [183] had an episodic nature, and hence $TD(\lambda)$ became preferable. Preventive maintenance problems are

usually SMDPs with an un-discounted objective function, which make SMART [184] and R-SMART [185] suitable. Other applications of these algorithms include "voice-over-packet" networks in [186] (R-SMART) and vendor selection for supply chains in [187] (SMART). A well-known "revenue management problem" can be set up as an average-reward SMDP [188]. But it has a unique reward structure with much of the reward concentrated in certain states that makes SMART, which is $TD(0)$, unstable. Hence Q-P-Learning [188] and λ-SMART [189] were applied. The work related to hyperheuristics [190] can be used when RL is to be used dynamically to select a meta-heuristic. Reference [191] employ hierarchical RL, since the AGV scheduling task they consider has controllers at multiple levels.

In 2016, Google announced the Go game machine "AlphaGo" [192], which uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. They also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This was the first time that a computer program has defeated a human professional player in the full-sized game of Go. It is likely that the field of applied RL will explode in the coming years because of RL's ability to solve problems previously considered intractable.

Reinforcement Learning vs. Dynamic Programming

Dynamic Programming is an off-line planning method, and the planning results from which are usually served as a benchmark to evaluate the online control algorithms. DP requires full understanding of the target object or system. It is of limited utility in real-time control both because of its great computational expense and because of its assumption of a perfect model. Here is an example about the problem of great computational expense. As discussed before, in order to solve the DP problem, we need to evaluate the value function $V(\boldsymbol{s})$ for all system state $\boldsymbol{s} \in \mathcal{S}$. For an HESS system, assume that the battery SOC, the ultracapacitor SOC, and the battery temperature are now used as the system states and the percentage of the Nyquist

frequency of the required current is used as the control action. Note that this action a here is set to a number between 0 and 1 (more details about this action a can be found in Section 6.2.5). We also discretize all these signals. The discretization step for both SOCs and the control action a is 0.01. For the battery temperature, we assume that we are only interested in the temperature range from $-20°C$ to $120°C$. And the temperature discretization step is $5°C$. Finally, we will have $100 \cdot 100 \cdot 28 = 280000$ possible system states. Moreover, in order to fully explore the policy space, the DP needs to exam $280000 \cdot 100 = 28000000$ possible combinations. This is obviously impossible for real-time computation. Note that, here we just assume that we only have three dimensional system state and the control action is only one dimension. What if we have more dimensions? The computational expense will be dramatically increased because of the curse of dimensionality. Moreover, what if we need smaller discretization step?

However, even we assume that we are able to finish these computation in real-time or we are able to evaluate and store the value function for all system states before the real-time operation, we are still not able to use DP for real-time control because of its assumption of a perfect model. As we discussed, in order to solve the following optimization problem for DP at time k:

$$\min_{\boldsymbol{a}_k}\{R(\boldsymbol{s}_k, \boldsymbol{a}_k) + \gamma V(\boldsymbol{s}_{k+1})\} \tag{6.9}$$

We need to not only find the minimized the instant reward $R(\boldsymbol{s}_k, \boldsymbol{a}_k)$, but also the minimized value function for next step $k+1$. Evan we know the value function $V(\boldsymbol{s})$ for all system state $\boldsymbol{s} \in \mathcal{S}$, this still won't be possible if we don't have a perfect system model $\boldsymbol{s}_{k+1} = \boldsymbol{F}(\boldsymbol{s}_k, \boldsymbol{a}_k)$ to predict the system state $\boldsymbol{s}_{k+1}$. By using the system model, we are able to convert the optimization problem to Equation 6.10,

$$\min_{\boldsymbol{a}_k}\{R(\boldsymbol{s}_k, \boldsymbol{a}_k) + \gamma V^*(\boldsymbol{F}(\boldsymbol{s}_k, \boldsymbol{a}_k))\} \tag{6.10}$$

Then, we are able to solve this optimization problem according to the pre-defined reward function $R(\boldsymbol{s}_k, \boldsymbol{a}_k)$ and the pre-calculated value function value at state $\boldsymbol{F}(\boldsymbol{s}_k, \boldsymbol{a}_k)$.

Therefore, for a complicated system without a relative accurate model, such as HESS, DP is not a feasible option for real-time control. Note that Dynamic Programming is also different from Static Programming. DP requires the system dynamic model in order to *dynamically* program the policy according to the system state, while SP does not adapts itself dynamically to any changes. Therefore, it is

impossible to conduct DP even offline, if you don't have a accurate system model. In fact, DP has never been used for real-time control. It is just an off-line programming method requiring accurate system model.

In reinforcement learning, the agent, which generates the action policy, is mathematically described by the policy $\boldsymbol{a}_k = \pi(\boldsymbol{s}_k)$. Everything outside the agent is considered to be the environment. Thus, the target system or object is considered as part of the environment, as are all disturbances and extraneous effects. In fact, in standard applications of reinforcement learning, the system dynamics is not even considered, and as part of the environment, no explicit model of the dynamics is even used. Reinforcement learning has enjoyed rather remarkable successes for complex systems with unknown dynamics. RL is usually regarded as an online control algorithm, while the DP is an off-line planning method. Moreover, in DP, the value function and the policy function are usually expressed by tabular format, while RL usually employs neural network to approximate these functions. In fact, the neural network not only learns to predict the functions but also the system dynamics.[192].

In RL, a new kind of value function is defined to associate value function with policy π.

$$Q_{\pi(\cdot)}(\boldsymbol{s}_k, \boldsymbol{a}_k) = R(\boldsymbol{s}_k, \boldsymbol{a}_k) + \gamma V_{\pi(\cdot)}(\boldsymbol{s}_{k+1}) \tag{6.11}$$

Note that the Q function is a function of both the state $\boldsymbol{s}_k$ and the control $\boldsymbol{a}_k$ at time k. It has been called the action value function for policy π. Define the optimal Q function as

$$Q^*(\boldsymbol{s}_k, \boldsymbol{a}_k) = R(\boldsymbol{s}_k, \boldsymbol{a}_k) + \gamma V^*(\boldsymbol{s}_{k+1}) \tag{6.12}$$

In terms of Q^*, one writes the Bellman Optimality equation in the very simple form

$$V^*(\boldsymbol{s}_k) = \min_{\boldsymbol{a}_k} Q^*(\boldsymbol{s}_k, \boldsymbol{a}_k) \tag{6.13}$$

Obviously, according to Figure 6.3, it is much easier to learn the environment's response state and rewards by using this Q function.

Monte Carlo Method

Monte Carlo methods are ways of solving the reinforcement learning problem based on averaging sample returns. To ensure that well-defined returns are available, Monte Carlo methods is defined only for episodic tasks. That is, experience is assumed to

be divided into episodes, and that all episodes eventually terminate no matter what actions are selected. It is only upon the completion of an episode that value estimates and policies are changed. Monte Carlo methods are thus incremental in an episode-by-episode sense, but not in a step-by-step sense. [176] Here is the details of Monte Carlo control algorithm. [174]

Algorithm 2 Monte Carlo control algorithm

Initialize.: for all $\boldsymbol{s} \in \mathcal{S}, \boldsymbol{a} \in \mathcal{A}(\boldsymbol{s})$: $Q(\boldsymbol{s}, \boldsymbol{a}) \leftarrow$ arbitrary
$\pi(\boldsymbol{s}) \leftarrow$ arbitrary
Q values, $q(\boldsymbol{s}, \boldsymbol{a}) \leftarrow$ empty list
Repeat forever:

- Choose $\boldsymbol{s}_0 \in \mathcal{S}$ and $\boldsymbol{a}_0 \in \mathcal{A}(\boldsymbol{s}_0)$. Generate an episode starting from $\boldsymbol{s}_0$, $\boldsymbol{a}_0$, following π
- For each pair $\boldsymbol{s}$, $\boldsymbol{a}$ appearing in the episode:
  $g \leftarrow$ log the Q value following the first occurrence of $\boldsymbol{s}$, $\boldsymbol{a}$.
  Append g to $q(\boldsymbol{s}, \boldsymbol{a})$
  $Q(\boldsymbol{s}, \boldsymbol{a}) \leftarrow \text{average}(q(\boldsymbol{s}, \boldsymbol{a}))$
- For each $\boldsymbol{s}$ in the episode: $\pi(\boldsymbol{s}) \leftarrow \text{argmin}_{\boldsymbol{a}} Q(\boldsymbol{s}, \boldsymbol{a})$

Temporal-Difference Learning

Temporal-Difference (TD) learning is a combination of Monte Carlo ideas and DP ideas. Like Monte Carlo methods, TD methods can learn directly from raw experience without a model of the environment's dynamics. Like DP, TD methods update estimates based in part on other learned estimates, without waiting for a final outcome (they bootstrap). The relationship between TD, DP, and Monte Carlo methods is a recurring theme in the theory of reinforcement learning. Figure 6.4 shows a rough comparison about them three. [176]

Based on the Bellman equation, a time-varying residual equation error for the kth step is defined as follow:

$$e_k = R(\boldsymbol{s}_k, \boldsymbol{a}_k) + \gamma V_\pi(\boldsymbol{s}_{k+1}) - V_\pi(\boldsymbol{s}_k) \tag{6.15}$$

This error is called TD error, which could be used to update value functions $V_\pi(\boldsymbol{s})$ step-by-step. Here is the TD(0) algorithm for estimation V_π.

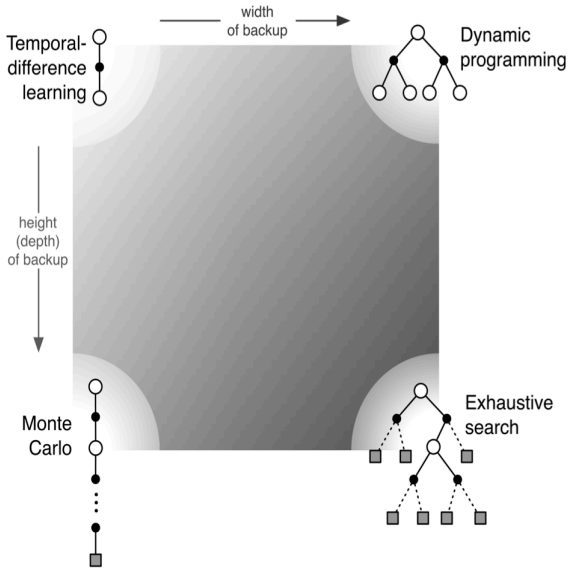


Figure 6.4: A slice of the space of reinforcement learning methods.

What is the space of methods lying between Monte Carlo and TD methods? Consider estimating V_π from sample episodes generated using π. Monte Carlo methods perform a backup for each state based on the entire sequence of observed rewards from that state until the end of the episode. The backup of simple TD methods, on the other hand, is based on just the one next reward, using the value of the state one step later as a proxy for the remaining rewards. One kind of intermediate method, then, would perform a backup based on an intermediate number of rewards: more than one, but less than all of them until termination. This method is TD(n) algorithm. Finally, The TD(λ) algorithm can be understood as one particular way of averaging n-step backups. This average contains all the n-step backups, each weighted proportional to λ^{n-1}, where $0 \leq \lambda \leq 1$. See Figure 6.5. A normalization factor of $1 - \lambda$ ensures that the weight sum to 1. [176]

6.2 Intelligent energy allocation algorithm

6.2.1 Battery degradation model

As we discussed in Chapter 2, battery SOH and Remaining Useful Life (RUL) can be calculated by using the battery internal resistance or expressed as a function of it. The main factor which leads to the increase of internal resistance is the formation of

Algorithm 3 TD(0) learning for estimating value functions

Input: the policy π to be evaluated

Initialize.: $V(\boldsymbol{s}) \leftarrow$ arbitrary for all $\boldsymbol{s} \in \mathcal{S}$

Repeat: for each episode

Repeat: for each step in current episode

- $\boldsymbol{a} \leftarrow$ action given by π for $\boldsymbol{s}$ in this step
- Take action $\boldsymbol{a}$; observe reward R and next state $\boldsymbol{s}'$
- Update $V(\boldsymbol{s})$:

$$V(\boldsymbol{s}) \leftarrow V(\boldsymbol{s}) + \alpha \cdot e \tag{6.14}$$

where $e = R(\boldsymbol{s}, \boldsymbol{a}) + \gamma V(\boldsymbol{s}') - V(\boldsymbol{s})$ and α is the learning rate, which is a positive number smaller than 1.

the solid electrolyte interface (SEI). During the initial cycling of a Lithium-ion cell, a resistive film is formed on the negative and positive electrodes, creating a solid-electrolyte interphase (SEI). Formation of the SEI layer is important for proper cell operation. However, unwanted side reactions increase the thickness and resistance of the SEI layer and consume active Lithium. They also change the degree of Lithium insertion in both electrodes and cause irreversible loss in charge capacity [193]. Therefore, several research works have been conducted to find a proper model to predict the formation of SEI [194, 195]. In [194], battery health degradation is expressed in terms of film resistance given in Equation 6.16. This film resistance R_{film} grows with time and use, and adds to the initial resistance of SEI layer R_{SEI}.

$$R_{film}(x,t) = R_{SEI} + \frac{\delta_{film}(x,t)}{\kappa_p} \tag{6.16}$$

where $\delta_{film}(x,t)$ is the film thickness as a function of spatial location inside the battery (x) and time (t), and κ_p is the conductivity of the film. The growth rate of film thickness is governed by Equation 6.17.

$$\frac{\partial \delta_{film}(x,t)}{\partial t} = -\frac{M_p}{a_n \rho_p F} J_s(x,t) \tag{6.17}$$

Here M_p , a_n, ρ_p and F represent molecular weight, specific surface area, mass density of the reaction product, and Farady's constant, respectively. The local volumetric current density for this side reaction is represented by $J_s(x,t)$. Note that the

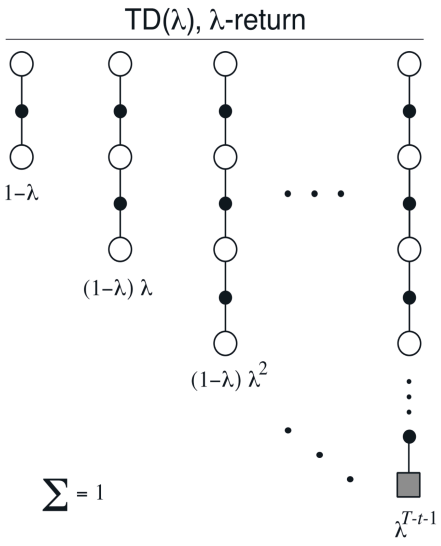


Figure 6.5: The TD(λ) algorithm.

mass density of the reaction product ρ_p is impacted by the battery temperature. So actually, the growth rate can be expressed as a function of battery current i, Depth of Discharge DOD and battery temperature tp at the given time.[196]

$$\frac{\partial \delta_{film}(x,t)}{\partial t} = \tau(i, dod, tp) \tag{6.18}$$

According to Equation 2.3 in Chapter 2, together with Equation 6.16 and Equation 6.18 above, the SOH can be expressed as following:

$$\begin{aligned} SOH &= \frac{R_{eol} - R}{R_{eol} - R_{new}} \cdot 100\% \\ &= \frac{R_{eol} - R_{film}}{R_{eol} - R_{SEI}} \\ &= 1 - \frac{\delta_{film}(x,t)}{\kappa_p \cdot (R_{eol} - R_{SEI})} \end{aligned} \tag{6.19}$$

Assume that R_{eol}, R_{SEI} and the conductivity of the film κ_p are all constants, which won't change by time. So, we have:

$$\begin{aligned} \frac{dSOH}{dt} &= -\frac{1}{\kappa_p \cdot (R_{eol} - R_{SEI})} \cdot \frac{\partial \delta_{film}(x,t)}{\partial t} \\ &= -\frac{1}{\kappa_p \cdot (R_{eol} - R_{SEI})} \cdot \tau(i, dod, tp) \end{aligned} \tag{6.20}$$

Therefore, battery health degradation after a driving trip can be theoretically calculated by Equation 6.21

$$\Delta SOH|_{t_{start}}^{t_{end}} = \int_{t_{start}}^{t_{end}} \frac{dSOH}{dt} dt = \int_{t_{start}}^{t_{end}} -\frac{1}{\kappa_p \cdot (R_{eol} - R_{SEI})} \cdot \tau(i, dod, tp) dt \quad (6.21)$$

Although we don't know the mathematical formula of function $\tau(i, dod, tp)$ and its integral over time, Equation 6.21 still proves that RL algorithm can learn how the battery degrades from battery SOC, temperature and current. In our IEA algorithm, RL agent takes battery SOC and temperature as states and battery current as control output. Moreover, according to Equation 6.21, battery degradation is actually caused by the time-domain cumulative effects of battery current, dod and temperature. Therefore, it is not possible to estimate the battery degradation according to these signals at every instance of time. Or we can say, it is trivial or at least unnecessary to update the energy allocation policy with a high frequency. This conclusion is very important. Because it provides us with the feasibility of applying IEA algorithm on the HESS control problem. In the following sections, we will find that the IEA algorithm is based on driving pattern recognition. As discussed in Chapter 4 and Chapter 5, the driving pattern recognition system usually needs a relatively long time period to finish recognition (averagely around hundred seconds). If the energy allocation policy needs to be updated in a high frequency, then we might not able to recognize current driving pattern correctly, hence the failure of the IEA algorithm.

6.2.2 Trip Mode

As we discussed in previous section, DP cannot solve the HESS energy allocation problem. Because DP needs perfect system model, so that it can predict the system behavior accurately based on current system states and input actions. But it is almost impossible to build an accurate model for the entire HESS because of its complexity. Moreover, even assuming that we have an accurate model of HESS, we still cannot use DP to solve this problem. When DP is used to solve the power management problem for the HEV [197], DP needs to know the speed signal for the entire trip accurately prior to drive the car, so that DP can optimally plan the power allocation for that trip while driving. Similarly, in our HESS energy allocation problem, DP also needs to know the required current for the entire trip prior to start it. This requirement

makes it almost impossible to implement DP in the practical scenario.

In this work, we propose a new concept, Trip Mode (TM), which converts a series of trip speed signals into to a driving pattern sequence. Together with RL, TM provides a sub-optimal solution to the HESS energy allocation problem.

6.2.3 Definition of Trip Mode

Given a series of vehicle speed of a driving trip, the Extended SVM in Chapter 4 will be used to extract useful features and IMPC in Chapter 5 will be used to classify vehicle signal section into different driving patterns. Therefore, we are able to convert the raw vehicle speed signals $\boldsymbol{x}$ to a sequence of driving patterns:

$$\boldsymbol{P} = [p1, p1, p2, p2, p2, p3, p3, ..., p4] \tag{6.22}$$

each pattern p_i could be 'flowing city condition', 'highway', and etc, which is recognized according to the vehicle signal in ΔT seconds. The battery degradation could also be estimated and the energy allocation policy will be updated accordingly for each pattern. Moreover, together with some meta data such as trip start time, current day in a week, locations and etc, similar trips will be recognized and the energy allocation policy generated by RL will be applied to them.

Moreover, let's assume that one repeated common trip will have a fixed Trip Mode. Here "repeated" means same start and destination via identical route at similar time. Of course, the repeated interval could be hours, days, months or even forever. For example, the trip from the work place to home via identical route at around 5:30PM everyday has a fixed trip mode. Or the trip to Costco from home every Saturday afternoon. Or even the trip to Banff from home last summer. Note that the repeated interval for this trip could be infinity.

One inference that can be drawn from this assumption is that the number of the different Trip Modes for a car must be less than the number of its finished trips in its entire life. In fact, our IEA algorithm is only useful to those repeated Trip Modes.

6.2.4 Repeated Trip Mode recognition

Naive Bayesian Classifier is used here to recognize the Trip Modes. Given the Trip Mode of current driving trip $\boldsymbol{P}_{driving}$ and a set of N repeated Trip Modes $\mathcal{P} = \{\boldsymbol{P}_i | i =$

$1, 2, 3..., N\}$, the recognition problem can be stated as follows:

$$\boldsymbol{P}^* = \text{argmax}_{\boldsymbol{P}_i \in \mathcal{P}} Pr(\boldsymbol{P}_i | \boldsymbol{P}_{driving}) \tag{6.23}$$

For each $\boldsymbol{P}_i$, the probability $Pr(\boldsymbol{P}_i|\boldsymbol{P}_{driving})$ can be expressed as Equation 6.24 according to Bayes' rule:

$$Pr(\boldsymbol{P}_i|\boldsymbol{P}_{driving}) = \frac{Pr(\boldsymbol{P}_{driving}|\boldsymbol{P}_i) \cdot Pr(\boldsymbol{P}_i)}{Pr(\boldsymbol{P}_{driving})} \tag{6.24}$$

In this equation, $Pr(\boldsymbol{P}_{driving})$ is unknown, but it is also a constant for all $Pr(\boldsymbol{P}_i)$. So we only need to calculate the numerator for ranking $Pr(\boldsymbol{P}_i|\boldsymbol{P}_{driving})$. $Pr(\boldsymbol{P}_i)$ can be easily estimated according to the following equation:

$$Pr(\boldsymbol{P}_i) \doteq \frac{Num(\boldsymbol{P}_i)}{Num(\text{finished trips})} \tag{6.25}$$

where $Num(\boldsymbol{P}_i)$ is the number of $\boldsymbol{P}_i$ happened and Num(finished trips) is the number of all finished trips in the car history. Moreover, if the meta data previously mentioned is also taken into consideration, then this equation could be:

$$Pr(\boldsymbol{P}_i) \doteq \frac{Num(\boldsymbol{P}_i|\text{day, time, location})}{Num(\text{finished trips}|\text{day, time, location})} \tag{6.26}$$

where $Num(\boldsymbol{P}_i|$day, time, location) is the number of $\boldsymbol{P}_i$ happened under the condition of the given day, start time and the location. Num(finished trips) is the number of all trips finished under those conditions in the car history In order to calculate the numerator of Equation 6.24, we also need to find a way to estimate the probability $Pr(\boldsymbol{P}_{driving}|\boldsymbol{P}_i)$. Obviously, more and more driving patterns would be recognized and added to the Trip Mode of current trip $\boldsymbol{P}_{driving}$. So the length of $\boldsymbol{P}_{driving}$ is increasing, while the car is driving. Then $\forall \boldsymbol{P}_i \in \mathcal{P}$, if $len(\boldsymbol{P}_{driving}) > len(\boldsymbol{P}_i)$, then $Pr(\boldsymbol{P}_{driving}|\boldsymbol{P}_i) = 0$ (where $len(\cdot)$ calculates the length of the given vector). Otherwise, for the Trip Modes in $\{\boldsymbol{P}_i|len(\boldsymbol{P}_{driving}) \leq len(\boldsymbol{P}_i)\}$, we define the probability $Pr(\boldsymbol{P}_{driving}|\boldsymbol{P}_i)$ as follows:

$$Pr(\boldsymbol{P}_{driving}|\boldsymbol{P}_i) = \frac{\xi(\nu(\boldsymbol{P}_i, len(\boldsymbol{P}_{driving})), \boldsymbol{P}_{driving})}{len(\boldsymbol{P}_i)} \tag{6.27}$$

where $\nu(\boldsymbol{x}, n)$ is a select function, which returns a sub-vector of the given vector

$\boldsymbol{x}$, starting from the first entry and ending at the nth entry. $\xi(\boldsymbol{x}, \boldsymbol{y})$ is a compare function, which compare the value of vector $\boldsymbol{x}$ and $\boldsymbol{y}$ in the element-wise fashion, and returns the number of identical elements. Note that, if $Pr(\boldsymbol{P}_{driving}|\boldsymbol{P}_i) \cdot Pr(\boldsymbol{P}_i) < \epsilon$, where ϵ is a small positive number, for all $i = 1, 2, 3, ..., N$, then $\boldsymbol{P}_{driving}$ should be added to the Trip Mode set as a new Trip Mode.

6.2.5 IEA for a given Trip Mode

In this section, the IEA algorithm is developed for a given Trip Mode. Here, "given" means that the Trip Mode is already determined and we do not need to recognize it. The IEA algorithm is based on the Q-learning, which is an off-policy TD control algorithm. (See TD learning algorithm in Section 6.1.2. In the IEA algorithm, the system state $\boldsymbol{s}$ is defined as:

$$\boldsymbol{s} = [SOC_{batt-min}, SOC_{uc-min}, T_{batt-max}, DI_{batt-aver}, idx]^T \tag{6.28}$$

where $SOC_{batt-min}$ is the minimal battery SOC during each driving pattern p_i, SOC_{uc-min} is also the minimal ultracapacitor SOC and $T_{batt-max}$ is the maximal battery temperature within the driving pattern. $DI_{batt-aver}$ is the average value of the battery current difference, which is defined as

$$DI_{batt-aver} = \frac{\sum ||Diff(\boldsymbol{i}_{batt})||}{\Delta T - 1} \tag{6.29}$$

where $Diff(\cdot)$ is a function returns the difference vector of the input vector. ΔT is the duration of the pattern section. $\boldsymbol{i}_{batt}$ is the battery current vector duing the driving pattern. Note that, since the system state is highly impacted by the required current, the pattern index idx is also considered as a system state.

A high-pass filter is employed in this research to separate the required current into two part, low frequency current i_{batt} for the battery and high frequency current i_{uc} for the ultracapacitor. The IEA output actions a is actually the cutoff frequency for the high-pass filter. Note that, without loss of generality, action a here is set to a number between 0 and 1 ($0 \leq a \leq 1$), which is the percent of the Nyquist frequency of the required current.

Note, "converge" in the algorithm could be:

$$\min_a Q(\boldsymbol{s}_0, a) \leq \epsilon \tag{6.31}$$

Algorithm 4 Intelligent Energy Allocation Algorithm for a given Trip Mode

Initialize: the Q-network weights $\boldsymbol{\omega}$ arbitrarily, (the Q-network is used to approximate $Q(s, a)$, for $\forall \boldsymbol{s} \in \mathcal{S}, \forall a \in \mathcal{A}(\boldsymbol{s})$)

Repeat: given trip until converge

Initialize.: state $\boldsymbol{s}$ or using the existing system state as $\boldsymbol{s}$

Repeat: for each pattern section in the given trip

- Choose a according to $\boldsymbol{s}$ using policy derived from the Q-network
- Take action a; observe reward R and next state $\boldsymbol{s}'$
- Update $Q(\boldsymbol{s}, a)$:

$$Q(\boldsymbol{s}, a) \leftarrow Q_{\boldsymbol{\omega}}(\boldsymbol{s}, a) + \alpha \cdot e \tag{6.30}$$

  where $e = R(\boldsymbol{s}, a) + \gamma \max_a Q_{\boldsymbol{\omega}}(\boldsymbol{s}', a) - Q_{\boldsymbol{\omega}}(\boldsymbol{s}, a)$

- use $Q(\boldsymbol{s}, a)$ as target value and $[\boldsymbol{s}^T, a]^T$ as input to train the Q-network to get the new network weights $\boldsymbol{\omega}'$
- $\boldsymbol{s} \leftarrow \boldsymbol{s}'$, $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega}'$

where $\boldsymbol{s}_0$ is the system initial state of the entire trip, so $Q(\boldsymbol{s}_0, a)$ is the overall cost of the trip, which is proportional to the battery degradation based on a properly designed R function. ϵ is here a positive integer.

Or "converge" could also mean that under the condition of Equation 6.31, the validating error of the Q-network is also small enough.

Ideally, the reward R should be the difference of battery internal resistor over time, entire system efficiency, the difference of the battery SOH or RUL or even the combination of these factors. However, in practical scenarios, R could just be a function designed heuristically, as long as it returns a metric measuring how "well" the battery is used according to battery current, temperature and SOC. A detailed diagram about the IEA for a given Trip Mode can be found in Figure 6.6

6.2.6 Searching the optimal action

In the algorithm, we need to choose action a according to the system state $\boldsymbol{s}$ using policy derived from the Q function approximated by the Q-network. This task could be stated as the following mathematically problem:

$$\min_a Q(\boldsymbol{s}, a) \tag{6.32}$$

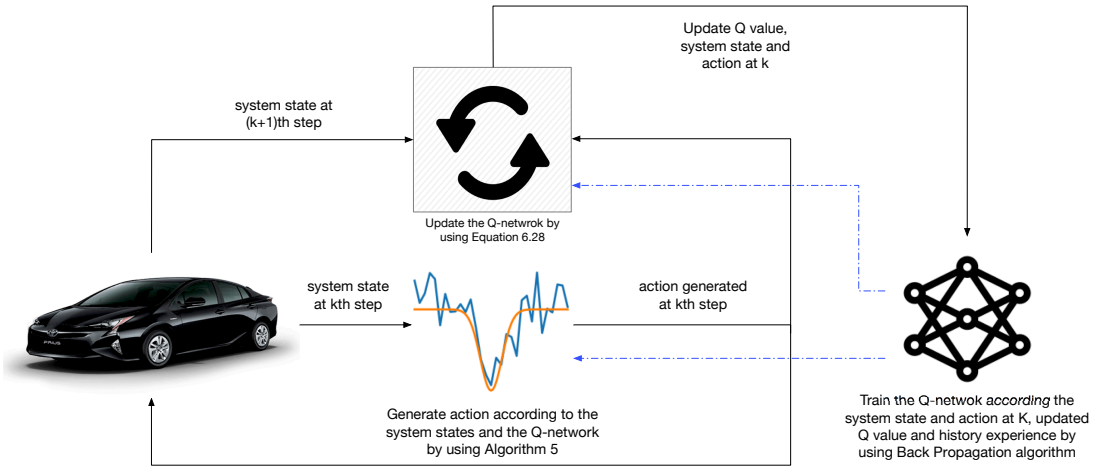


Figure 6.6: IEA for a given Trip Mode

However, since we Q function is approximated by the Q-network, we don't have the explicit mathematically equation for it. Moreover, we have no idea if it is convex, differentiable or not. Therefore, in order to solve this optimization problem, Direct Search (also called Pattern Search) [198, 199] algorithm is employed. Since our action a is just a number between 0 and 1, we use the Quadratic Interpolation Search. [127] In side this algorithm, $q(\boldsymbol{s}, a)$ is the output when input $\boldsymbol{s}$ and a into the the Q-network. Functions $min(\cdot)$ selects the minimal value form its inputs, while $max(\cdot)$ selects the maximal.

6.2.7 Q-network designing and training

Finally, let's introduce how to design and train the Q-network.

Radial-Basis-Function network for function approximation

Basically, the Q-network here is a neural network, which is able to approximate the Q function. According to [200], it is proved that the Radial-Basis-Function (RBF) networks having one hidden layer are capable of universal function approximation. So RBF network is employed in this research. The structure of the one hidden layer network can be found in Figure 6.7. Given an input $\boldsymbol{x}$, the output y can be calculated by using Equation 6.35.

$$y = \sum_{i=1}^{c} \omega_i \cdot \phi_i(\boldsymbol{x}) + b \tag{6.34}$$

Algorithm 5 Quadratic Interpolation Search for action a

Input: ϵ and K

Initialize.:set $k = 1$, $a_1 = 0$, $a_3 = 1$, and $\bar{a}_0 = 10^{99}$. Compute $a_2 = \frac{1}{2}(a_1 + a_3)$ and $q_i = Q(\boldsymbol{s}, a_i)$, $i = 1, 2, 3$, Compute $\bar{a}$

$$\bar{a} = \frac{(a_2^2 - a_3^2)q_1 + (a_3^2 - a_1^2)q_2 + (a_1^2 - a_2^2)q_3}{2[(a_2 - a_3)q_1 + (a_3 - a_1)q_2 + (a_1 - a_2)q_3]} \tag{6.33}$$

Repeat: if $|\bar{a} - \bar{a}_0| \geq \epsilon$ or $k \leq K$

- If $a_1 < \bar{a} < a_2$, then
  - If $\bar{q} \leq q_2$, assign $a_3 = a_2$, $q_3 = q_2$, $a_2 = \bar{a}$, $q_2 = \bar{q}$
  - Otherwise, if $\bar{q} > q_2$, assign $a_1 = \bar{a}$, $q_1 = \bar{q}$
- If $a_2 < \bar{a} < a_3$, then
  - If $\bar{q} \leq q_2$, assign $a_1 = a_2$, $q_1 = q_2$, $a_2 = \bar{a}$, $q_2 = \bar{q}$
  - Otherwise, if $\bar{q} > q_2$, assign $a_3 = \bar{a}$, $q_3 = \bar{q}$
- Set $\bar{a}_0 = \bar{a}$, Compute new $\bar{a}$ according to Equation 6.33

Output: : $a^* = min(1, max(0, \bar{a}))$, $q^* = Q(\boldsymbol{s}, a^*)$

where b is the bias, ω_i is the weights. b and ω_i can be determined through Back-propagation Training [201]. RBF function $\phi_i(\boldsymbol{x})$ is:

$$\phi_i(\boldsymbol{x}) = exp(-\frac{||\boldsymbol{x} - \boldsymbol{v}_i||^2}{2\sigma_i^2}), i = 1, 2, 3..., c \tag{6.35}$$

Note that each $\boldsymbol{v}_i$ can be randomly selected from the training dataset, [202] and σ_i is

$$\sigma_i = \frac{d_{max}}{\sqrt{2c}} \tag{6.36}$$

where d_{max} is $\max_{i,j=1}^{c} ||\boldsymbol{v}_i - \boldsymbol{v}_j||$

Experience Replay

According to Algorithm 4, the Q-network would be trained and updated by using $Q(\boldsymbol{s}, a)$ as target value and $[\boldsymbol{s}^T, a]^T$ as input for each pattern in the given Trip Mode.

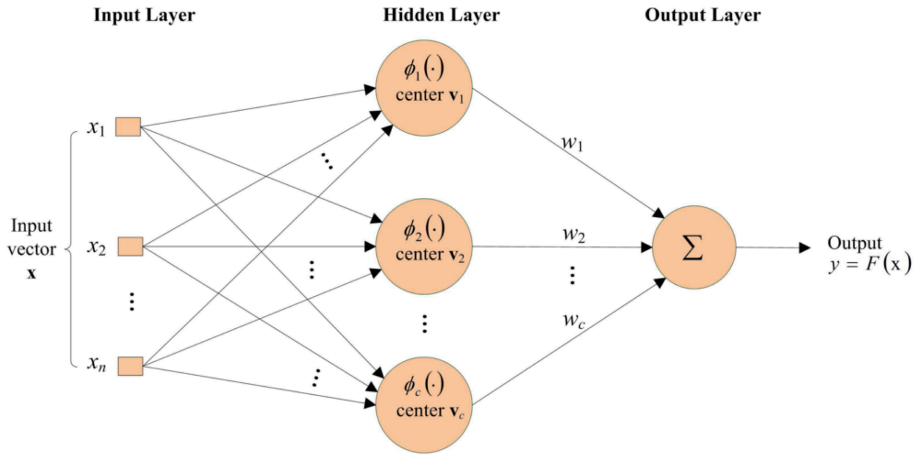


Figure 6.7: One hidden layer RBF Neural Network

However, in order to stabilized the training, as well as the Q-network, Experience Replay (ER) is used in this research. ER is a promising approach for RL control, which enables the RL algorithm to learn quickly from a limited amount of data, by repeatedly presenting these data to it. [203]. ER has already been applied to several interesting works. [204, 205] In this work, the system state $\boldsymbol{s}$, action a and reward R for every step (pattern) are saved to the system experience. So during the training, a small batch of history experience is randomly selected. These experience is then used to generate the data for training by using Equation 6.30. Note that in this equation, $Q_{\boldsymbol{\omega}}(\boldsymbol{s}, a)$ is the Q value approximated by the Q-network based on the old weights $\boldsymbol{\omega}$ and b. The stochastic gradient descent method is finally used to train the Q-network based on the data generated by current step, as well as the history experience.

6.2.8 IEA for an uncertain Trip Mode

In this section, the IEA algorithm for an uncertain Trip Mode is proposed. Assume that the set of repeated Trip Mode has N different Trip Mode $\boldsymbol{P}_i, i = 1, 2, 3, ..., N$. For each Trip Mode $\boldsymbol{P}_i$, there exists a Q-network $Q^i(\boldsymbol{s}, a)$ to approximate the Q values. Of course, this Q-network could be a well trained network, or just be a randomly initialized one. The IEA algorithm for an uncertain Trip Mode is illustrated in Algorithm 6.

Note that the probability $Pr(\boldsymbol{P}_i|\boldsymbol{P}_{driving})$ is estimated by the softmax function

Algorithm 6 Intelligent Energy Allocation Algorithm for uncertain Trip Mode

Input: the set of repeated Trip Mode $\mathcal{P} = \{\boldsymbol{P}_1, \boldsymbol{P}_2, ..., \boldsymbol{P}_N\}$, the set of Q-networks $\mathcal{Q} = \{Q^1(\boldsymbol{x}, a), Q^2(\boldsymbol{x}, a), ..., Q^N(\boldsymbol{x}, a)\}$

Initialize.: state $\boldsymbol{s}$ or using the existing system state as $\boldsymbol{s}$

Repeat: for each pattern section in the given trip

- Choose a according to $\boldsymbol{s}$ using policy derived from the combined Q functions, which is defined as:

$$Q_{combined} = \sum_{i=1}^{N} Pr(\boldsymbol{P}_i|\boldsymbol{P}_{driving}) \cdot Q^i(\boldsymbol{x}, a) \tag{6.37}$$

- After taking the action a, save reward R, next state $\boldsymbol{s}'$, as well as action a itself.

Recognize and learn: Recognize the driving Trip Mode by using the algorithm proposed in Section 6.2.4, use Monte Carlo method described in Section 6.1.2 to generate training data, use the training method introduced in Section 6.2.7 to train and update the Q-nerwork for the recognized Trip Mode.

[176]:

$$Pr(\boldsymbol{P}_i|\boldsymbol{P}_{driving}) = \frac{e^{z_i}}{\sum_{k=1}^{N} e^{z_k}}, i = 1, 2, ..., N \tag{6.38}$$

where $z_i = Pr(\boldsymbol{P}_{driving}|\boldsymbol{P}_i) \cdot Pr(\boldsymbol{P}_i)$, which can be calculated by using Equation 6.26 and Equation 6.27. More detailed information about the IEA for an uncertain Trip Mode can be found in Figure 6.8

Note that in practical scenario, we will always have the uncertain Trip Mode cases. However, if the current driving Trip Mode can be easily recognized in the early stage of the trip, then this control problem can be converted to a given Trip Mode problem.

6.3 Experiment

In this section, experiments are conducted to validate the performance of the proposed IEA algorithm. The required current of the HESS is generated by running the Prius model in Automonieon/Matlab on the trip illustrated in Figure 5.10. Similar to Chapter 5, the sample time ΔT is set to 100 seconds. So, we have 14 pattern sections in this trip. The Hess is in a semi-active structure, which has an ideal DC-DC converter in the ultracapacitor end. So the HESS voltage is actually the battery in-

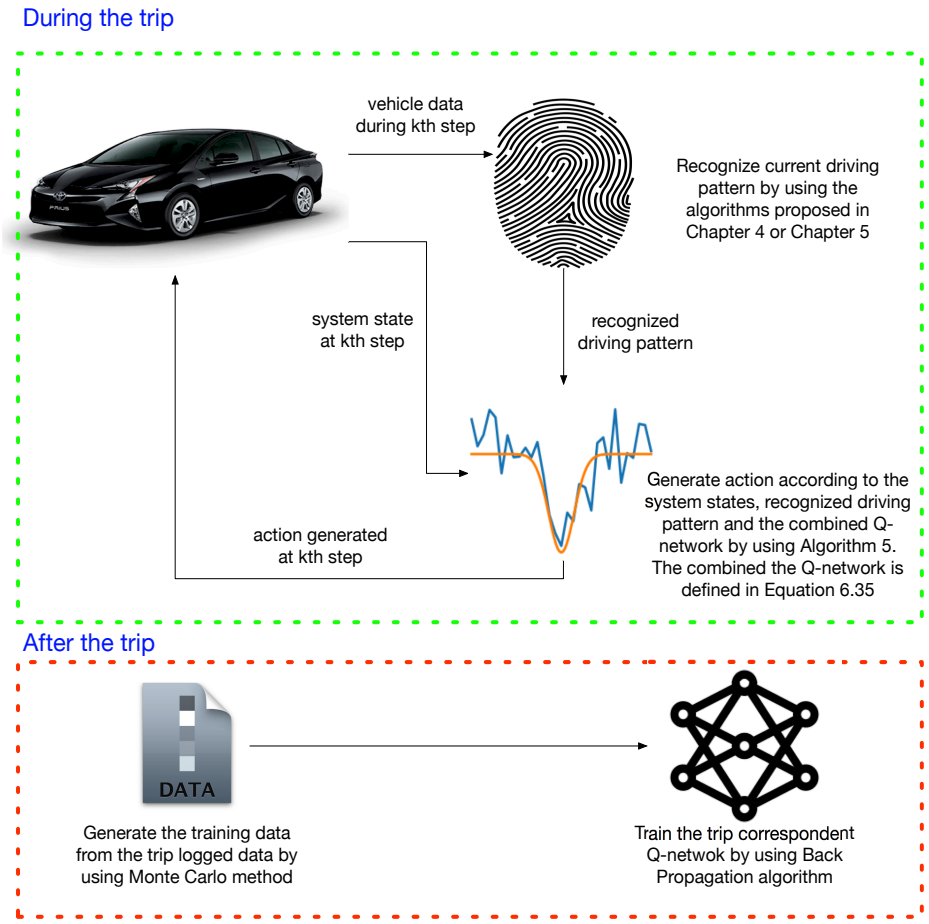


Figure 6.8: IEA for an uncertain Trip Mode

circuit voltage. A high-pass filter is employed to filter out low frequency component of the required current for the ultracapacitor.

6.3.1 Reward function and performance metric

The reward function is firstly defined here according to Equation 6.21. It is consist of four parts as show in Equation 6.39. They are battery SOC reward $r_{batt-SOC}$, ultracapacitor SOC reward r_{uc-SOC}, battery temperature reward $r_{batt-temp}$, and battery current reward $r_{bett-curr}$. k_i is the coefficient to balance these four rewards.

$$R_{all} = k_1 \cdot R_{batt-SOC}(\boldsymbol{s}_k, a_k) + k_2 \cdot R_{uc-SOC}(\boldsymbol{s}_k, a_k) + k_3 \cdot R_{batt-temp}(\boldsymbol{s}_k, a_k) + k_4 \cdot R_{bett-curr}(\boldsymbol{s}_k, a_k) \tag{6.39}$$

where $\boldsymbol{s}_k$ is defined in Equation 6.28 and a_k is the control action.

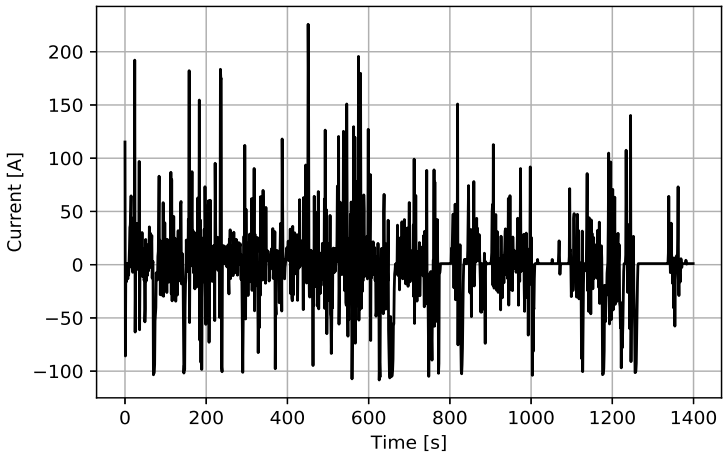


Figure 6.9: Required current for the 23-min trip generated by the Pruis model in Autonomie

Battery SOC reward

The battery SOC reward $R_{batt-SOC}$ is calculated by $1 - SOC_{batt-min}$. $SOC_{batt-min}$ is one of the system states. The Prius vehicle controller monitors the battery SOC and keeps it around 70% at the end of each trip. So, if the SOC of the battery in our HESS is less then 65% at the end of the entire trip, another penalty would be added to the battery SOC cost.

Ultracapacitor SOC reward

Similarly, the ultreacapacitor SOC reward R_{uc-SOC} is calculated by $1 - SOC_{uc-min}$, and the ultracapacitor SOC reward would be increased, if the ultracapacitor SOC is less than 70% at the end of the trip.

Battery temperature and current reward

According to the introduction in Section 2.1.1, the battery temperature reward $R_{batt-temp}$ is calculated as

$$R_{batt-temp} = \frac{T_{batt-max} - T_{amb}}{70 - T_{amb}} \tag{6.40}$$

where T_{amb} is the ambient temperature. Finally, the battery current reward $R_{batt-curr}$ is defined by Equation 6.29.

6.3.2 Performance metric

The overall reward $R_{all}(\boldsymbol{s}_k, a_k)$ is the sum of these four mentioned rewards. Note that we have varies methods to define the overall reward according to the practical needs. Moreover, this overall reward can be expressed by a mathematical function, a piece of code with logic defined or even some practical measurements after the vehicle driving.

In this research work, the sum of overall reward $\sum_{k=1}^{end} R_{all}(\boldsymbol{s}_k, a_k)$ over the entire trip is used as the metric to evaluate and compare the performances of the control algorithms. Moreover, in order to avoid the confusion, we call the sum of the overall reward "trip cost" in the following sections.

6.3.3 Results of passive HESS and fixed-cutoff-frequency control method

As we discussed previously, the action a is the cutoff frequency of the current high-pass filter. In Figure 6.10, we plot the trip cost ($\sum R_i$), when a fixed action is employed for the entire trip and the initial state is $SOC_{battery} = 0.7, SOC_{uc} = 0.7, Temp = 25$. Note that, $a = 0$ means using ultracapacitor only, while $a = 1$ is using battery only. Obviously, using the ultracapacitor only minimizes the trip cost, if the a needs to be constant during the trip. However, the ultracapacitor SOC is smaller then 70% at the end of the trip, which could make the vehicle not useable for next trip in the worst cases.

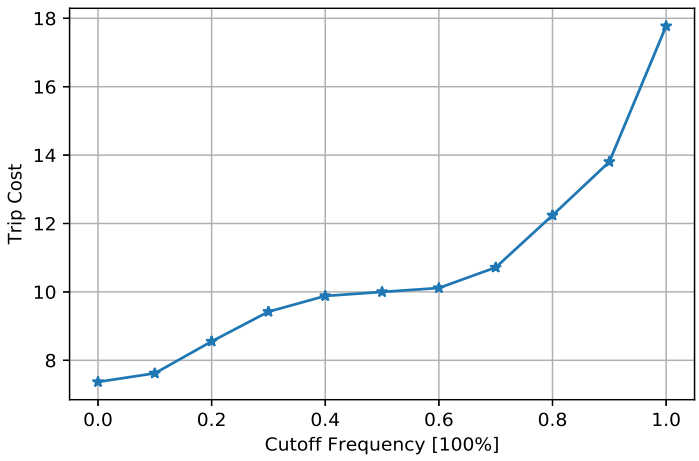


Figure 6.10: Trip cost under a fixed cutoff frequency

Moreover, we also plot the results of the passive HESS under the same application scenario. The overall trip cost of passive HESS is 8.2149

6.3.4 Results for a given trip mode with fixed initial state

In order to compare the policy figured out by the IEA algorithm with the fixed-cutoff-frequency strategy, IEA algorithm is tested under the same application scenario, on the same given trip with the same initial system state. The Q-network learning curve and policy figured out by the IEA are plotted in Figure 6.12. Meanwhile, the detailed results of the HESS are also plotted in Figure 6.13.

The 23-min trip is repeated 200 times. Inside this trip, we have 14 pattern sections. Therefore, the Q-network is trained 2800 times. According to the learning curve, the training of the Q-network converges to a relative small and stable approximation error. Note that, after the training in each pattern section, we also performed test to check the generalization of the Q-network.

According to the policy plot, IEA decides to gently use the battery in the city and highway driving condition at the beginning of the trip. Then it gradually reduces the energy output of the battery. When the car enters the second long city condition, IEA stops using the battery. In the last pattern section, IEA uses the battery only. This strategy actually enjoys a lot of benefits. First of all, in the begin of the trip, gently using the battery leads to the slow rising of the battery temperature. Moreover, use the battery first in the trip can actually preserve the battery energy into vehicle kinetic energy during the highway driving. Secondly, in the long city driving, stopping using the battery also makes a lot of senses. Finally, in order to avoid the ultracapacitor SOC penalty, using the battery only in the last pattern section to boost the ultracapacitor SOC is also smart. The overall trip cost of the entire trip by using this policy is only 5.9288, which is much less than any fixed-cutoff-frequency policies as show in Table 6.1. In that table, battery-only situation is actually equivalent to set $a = 1$ in the fixed-cutoff-frequency method. Also note that the detailed cost of each pattern section can also be found in the Figure 6.13.

At the beginning of the training, IEA randomly picks the cutoff frequency in each pattern section. After 200 trips, IEA converges to this efficient policy. In order to see the process of how IEA solves this problem, we also plot the detailed results in the 100th trip driving in Figure 6.14 and Figure 6.16. The overall cost of this policy is 7.3469, which is already smaller than the fixed-cutoff-frequency policies.

6.3.5 Results for a given trip mode with random initial state

In the practical usage, the initial state of the HESS system won't be always the same. Therefore, we train the IEA in the given 23-min trip with a random initial state. In this experiment, the battery SOC and ultracapacitor SOC are randomly picked from 0.1 to 1 and the ambient temperature is still a constant at 25^oC. Allowing random initial states actually means that the IEA algorithm needs to explore more states in the state space, which increases the difficulties of training the Q-network. Moreover, the policy function can be expressed as:

$$a = \pi(\boldsymbol{s}) = \text{argmin}_{a \in \mathcal{A}(\boldsymbol{s})} Q(\boldsymbol{s}, a) \tag{6.41}$$

Since the initial state is random, now the policy decided by the IEA can not be explicitly plotted. However, we can still find the learning curve in Figure 6.17. As show in this figure, the approximation error is larger than the error in the fixed initial state case.

In order to validate the performance of the IEA under this scenario, the learned policy function is applied to 100 different cases. The initial state of these 100 cases can be stated as $[SOC_{batt}, SOC_{uc}, 25]^T, SOC_{batt} = 0.1, 0.2, ..., 1, SOC_{uc} = 0.1, 0.2, ..., 1$. The trip cost of the IEA is compared with the costs of the fixed-cutoff-frequency method. The results are plotted in Figure 6.18. The trip cost of the IEA is sorted together with the 11 trip costs of the fixed-cutoff-frequency method (cutoff frequency $a = 0, 0.1, 0.2, ..., 1$) from small to large. The z axis in the figure is the rank number of the IEA cost. In most cases, the IEA cost ranks in the top 2 position (top 2 small cost). However, the larger the ultracapacitor initial SOC is, the lower the IEA ranks. In those cases, the IEA policy is worse than using the fixed-cutoff-frequency policy, such as $a = 0$ or even $a = 0.1$.

Note that, random initial state actually means that the IEA algorithm needs to explore much more states in the state space, which leads to increased difficulties in training the Q-network. Because of the limitation of the computation resources, the presented results here could not be the optimal results.

6.3.6 Results for an uncertain Trip Mode

In this section, experiment is conducted on the uncertain Trip Mode. Two highly different Trip Modes can be easily differentiated. So they can be recognized in the

early stage of the trip, which means they won't have a lot of impacts on each other. On the other hand, if we have two similar Trip Modes, then it is hard to differentiate them. In the most extreme case, they could be recognized even at the end of the trip, which means the action a is selected to minimize the expected value of the entire trip cost of these two Trip Modes. Obviously, the more lately the IEA algorithm recognizes the Trip Mode, the more challenging it is to obtain an optimal policy. In this experiment, we test an extreme case. The first Trip Mode $\boldsymbol{P}_1$ is the 23-min trip we used in previous experiments, while the second trip $\boldsymbol{P}_2$ is the first 21 minutes and 40 seconds of the 23-min trip. The practical application scenario of this experiment could be that the user drives home in the first Trip Mode, and visits his/her neighbor living close to him/her in the second one. In order to plot and compare the policy figured out by the IEA algorithm, we fix the initial state to $[0.7, 0.7.25]^T$. Moreover, we also assume that $Pr(\boldsymbol{P}_1) = Pr(\boldsymbol{P}_2) = 0.5$. The required current for the second Trip Mode generated by the Prius model in Autonomie is plotted in Figure 6.19. The policy generated by IEA for first Trip Mode can be found in Figure 6.15. Finally, detailed results of the HESS system and the recognition probabilities calculated according to Algorithm 6 are also illustarted.

The interesting part here is that the IEA seams to decide to boost or maintain the ultracapacitor SOC in the 13th pattern section. By doing this, IEA can keep the ultracapacitor SOC larger than 70% at the end of both trips. However, the cost of doing this is that the entire trip cost for the first Trip Mode is raised to 6.1710.

Table 6.1: Trip cost under fixed initial state

	Given Trip	Uncertain Trip	Passive Hess	Best Fix-freq	Battery-only
Trip Cost	5.9288	6.1710	8.2149	7.4536	17.8821

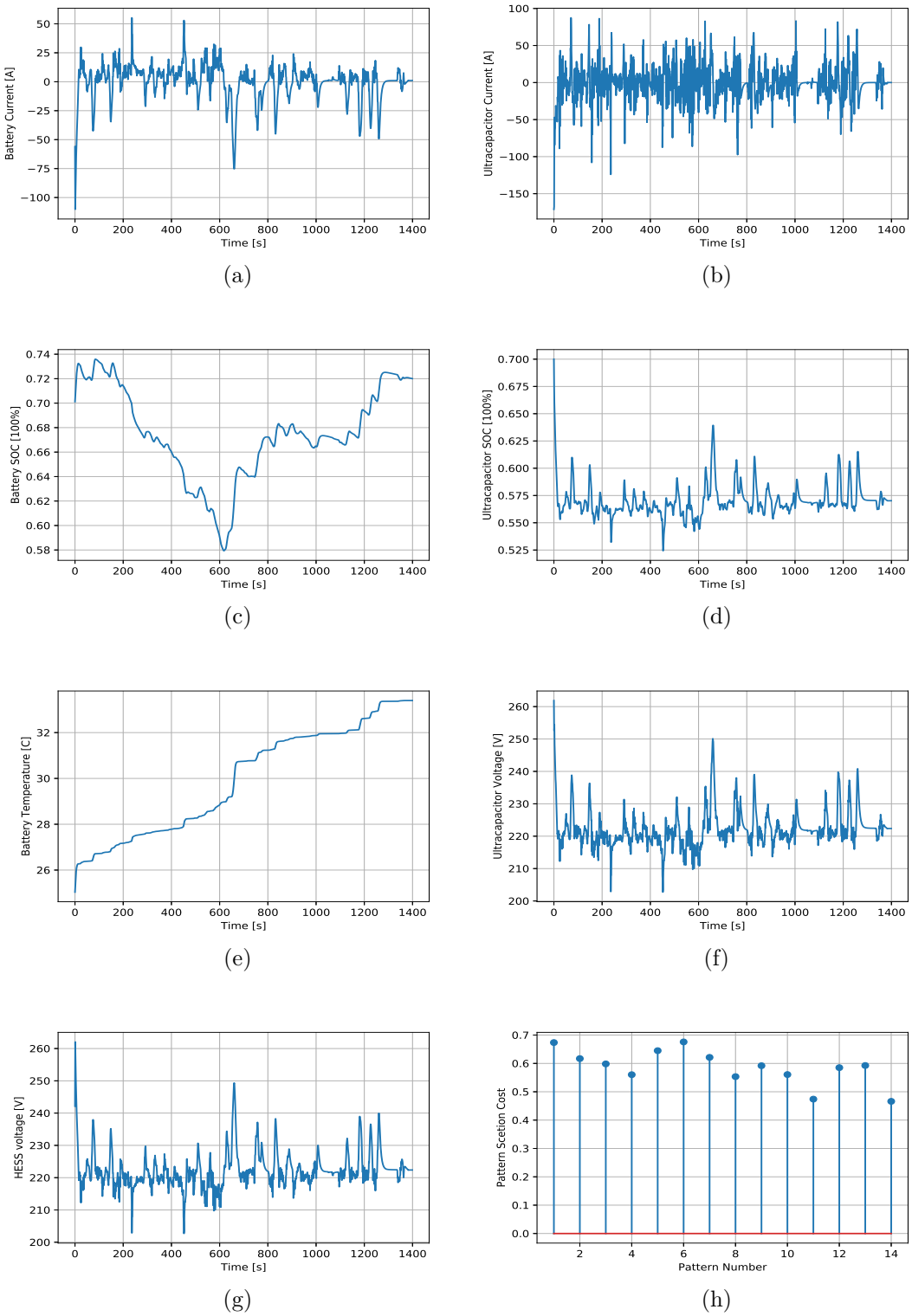
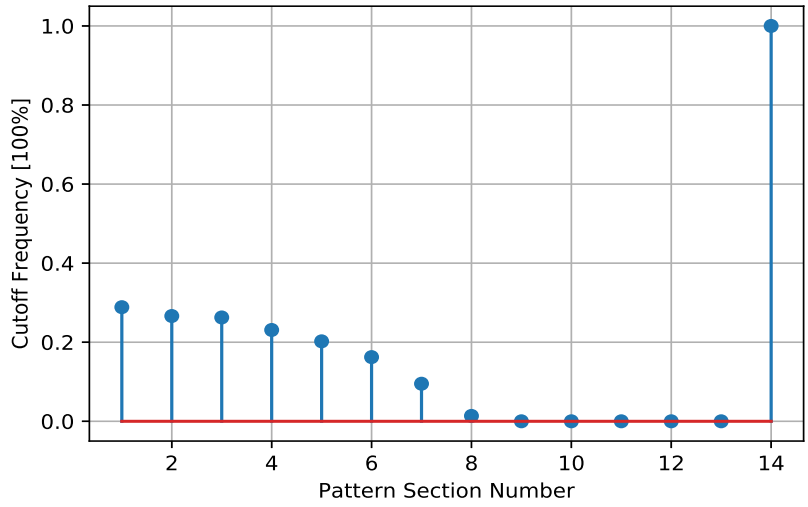
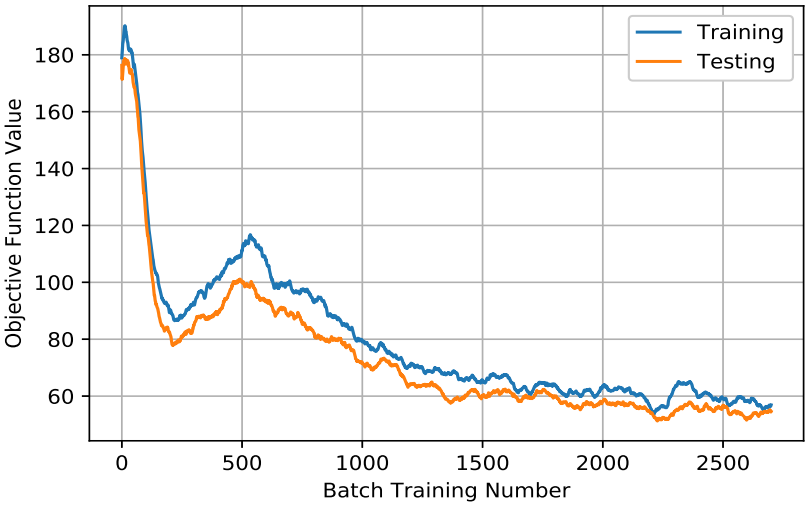


Figure 6.11: Results of the passive HESS



(a)



(b)

Figure 6.12: Policy figured out by IEA and Q-network learning curve

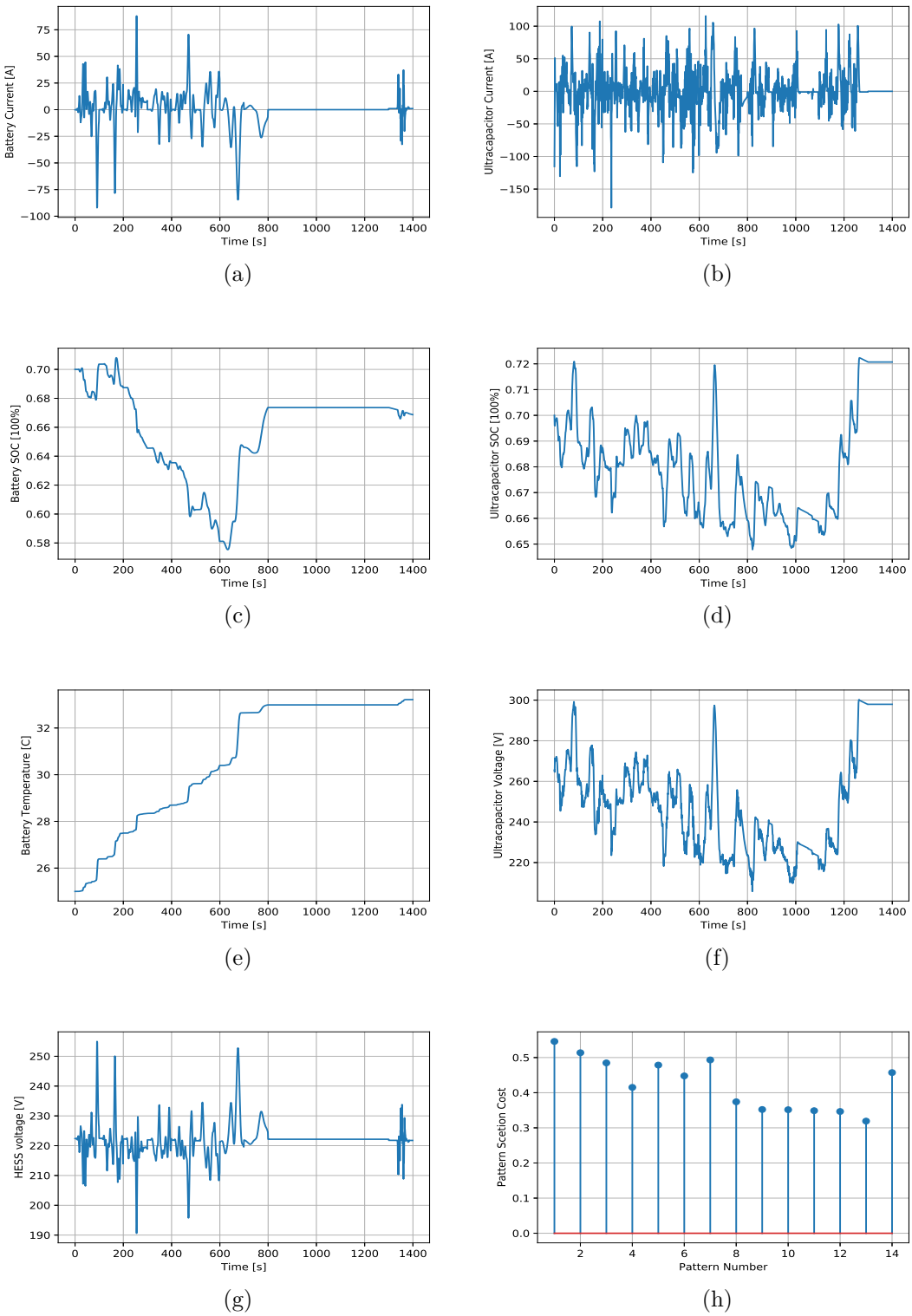


Figure 6.13: Results of the IEA on the 23-min trip

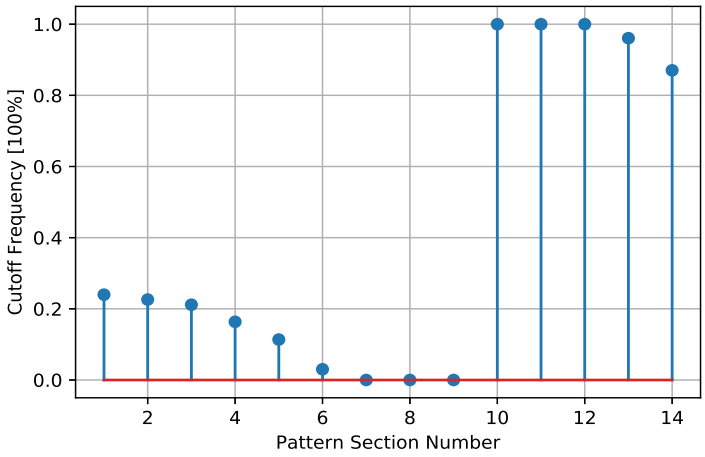


Figure 6.14: Policy figured out by IEA in the 100th trip driving

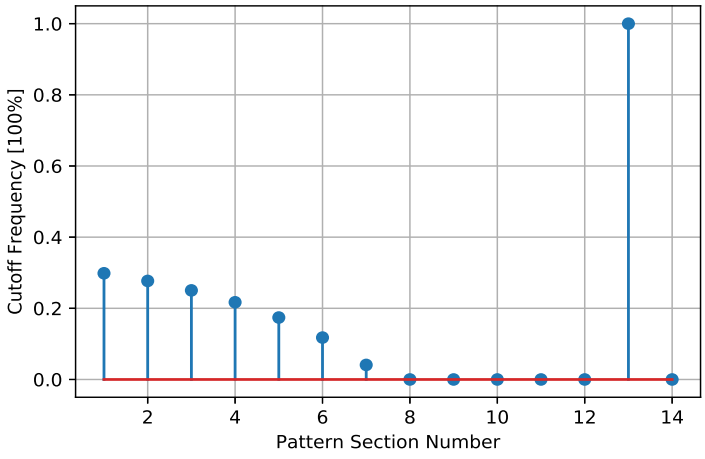


Figure 6.15: Policy figured out by IEA for the uncertain trip

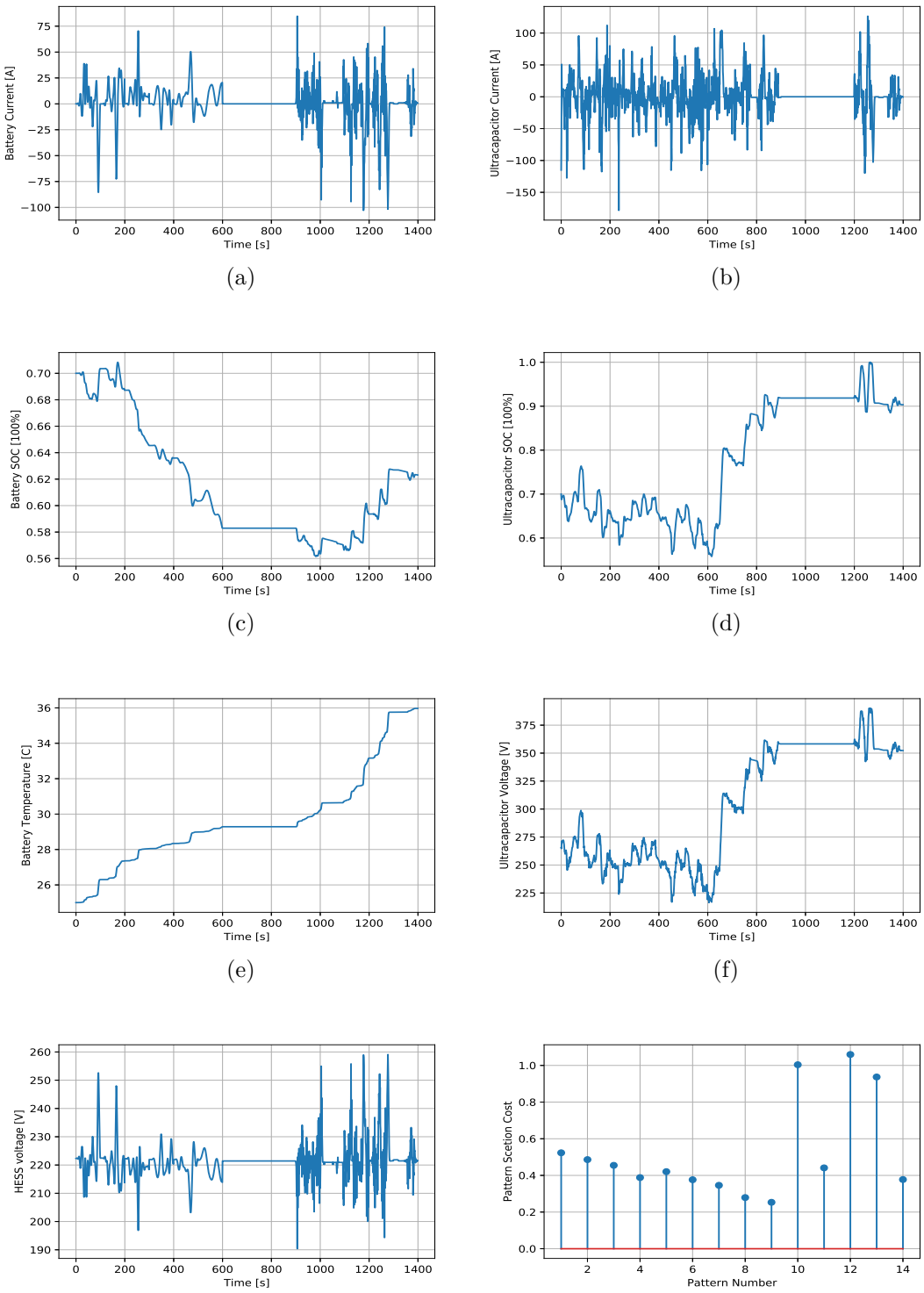


Figure 6.16: Results of the IEA on the 23-min trip in the 100th driving

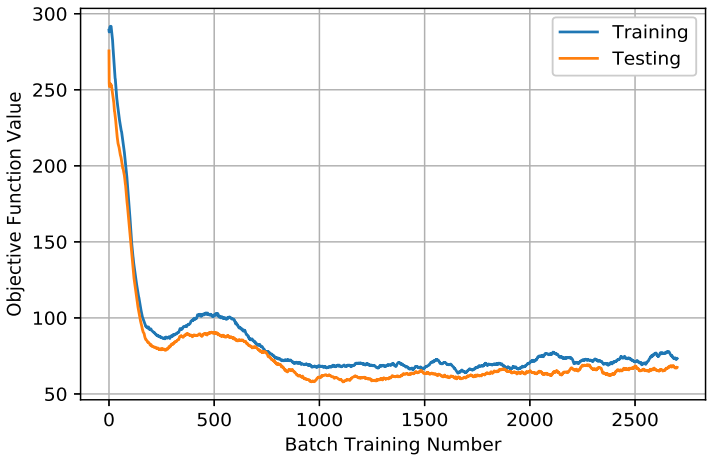


Figure 6.17: Learning curve for the random initial states case

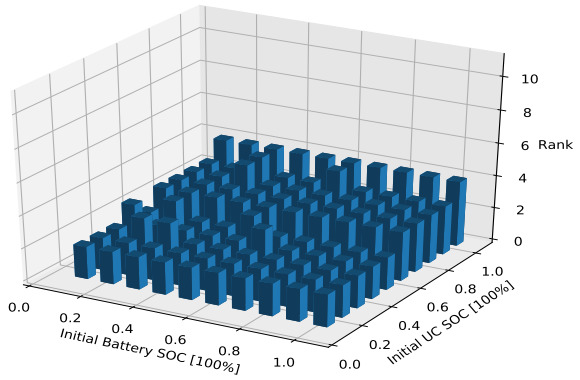


Figure 6.18: Results compare for the random initial states scenario

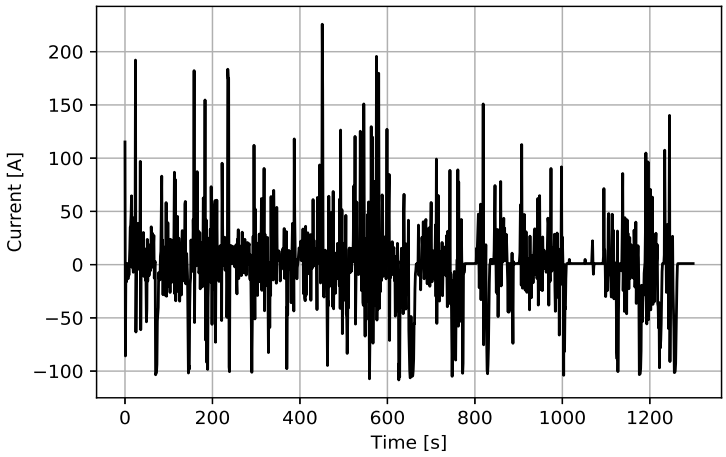


Figure 6.19: The required current of Trip Mode 2

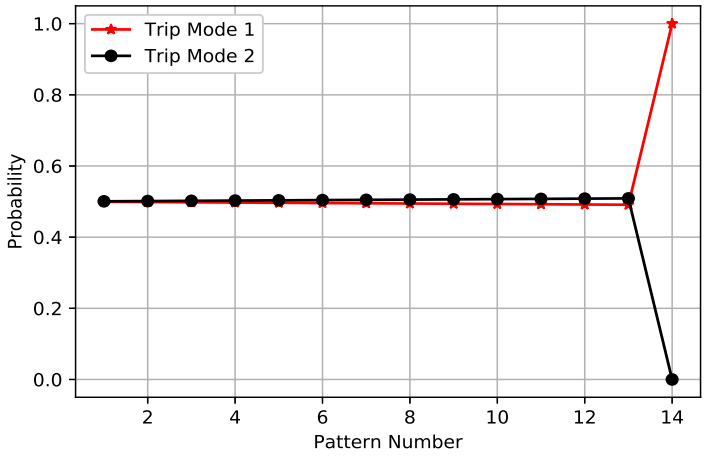


Figure 6.20: The running probabilities for both Trip Modes

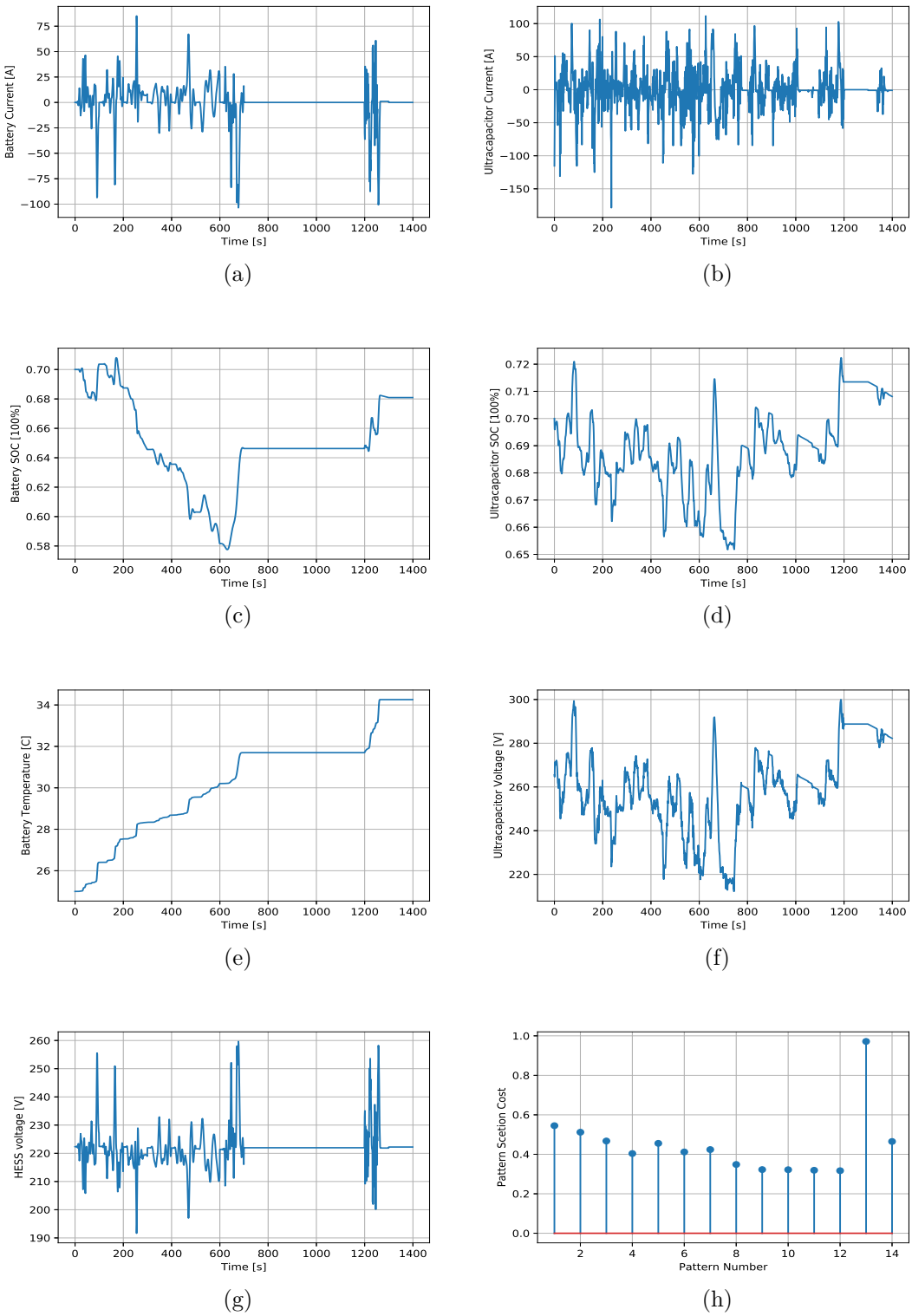


Figure 6.21: Results of the IEA on the uncertain trip

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this work, an intelligent energy allocation method was proposed for the Hybrid Energy Storage System. After introducing the research background and reviewing some related topics, we firstly compared the three different topologies for the battery-ultracapacitor hybrid storage system. A typical pulsed current load was used as the input of the HESS. The system efficiency was then calculated for those three topologies. Then, a combined simulation method was proposed by employing both Matlab and Python. A simulation platform package was developed in Python and hosted on the Pypi server, so that futher research can use, extend and improve this simulation package. The Prius model in Autonomie was exported into Matlab, and then rebuilt in Python. By using this combined simulation method, running complex model and sophisticated reinforcement learning algorithm could be possible and efficient.

In chapter 4, the extended Support Vector Machine (SVM) with an embedded feature selection ability was developed. This proposed SVM takes the observability of features into account, so as to enable the driving condition recognition system to achieve higher recognition efficiency and select significant and easily-accessable features. This algorithm was then proved mathematically to be a standard SOCP optimization problem, and can be solved efficiently. After the comparing with the standard 2-norm SVM and the linear 1-norm SVM, this proposed SVM reaches a better balance between classification and feature numbers, while also taking feature observability into account. Moreover, the proposed SVM is able to replace those features with bad observability by choosing other more observable features.

In Chapter 5, a novel classification algorithm, Inertial Matching Pursuit Classification (IMPC), was proposed especially for recognizing vehicle driving patterns in a shorter time period. This IMPC algorithm is based on the Compressed Sensing and could be solved by a new kind of greedy Algorithm developed in this chapter. Compared with other traditional or general methods, IMPC is able to recognize the diving patterns within a shorter recognition period, which leads to the improvement of the DPR system resolution and accuracy. Moreover, we experimentally show that the IMPC is able to recognize driving patterns correctly without requiring the vehicle speed signal to be fully sampled or accurately recovered.

Finally, in Chapter 6, the intelligent energy allocation algorithm was developed. This IEA algorithm employs the SVM in chapter 4 to select features and uses the IMPC to recognize driving pattern with high resolution, so that the Trip Mode can be recognized accurately. This proposed IEA algorithm is based on Q-learning. After each driving pattern is recognized, this algorithm generates the optimal control actions to separate the required current between battery and ultracapacitor according to the learned Q values. A RBF neural networks is trained and updated to approximate the Q values. Compared with passive HESS and other traditional control methods, this IEA algorithm can give a sub-optimal policy to allocate the energy between battery and ultracapacitor. Unlike other methods, this proposed algorithm is able to adapt itself to different daily driving modes. It keeps learning to generate optimal policies to prolong the battery life and improve the HESS efficiency from the data collected during the daily driving. The more a trip is repeated, the better the trip policy will be.

7.2 Future Work

7.2.1 Extended SVM

The extended SVM proposed in chapter 4 is very powerful in feature selection. However, it is sensitive to the size of original data set, because of using the SOCP algorithm to solve it. If the data set is very large, then solving the proposed SVM is very computationally expensive. Fortunately, in HEV applications, very large data sets are not common. But, it still needs some future work to solve this problem. The ideal solution is develop a sequential method to solve this problem, like the Sequential minimal optimization (SMO) algorithm for the standard SVM [206], so that this

optimization problem can be solved step by step. Moreover, in this work, only the signals in the powertrain are considered. Actually, much more data or signals can be also regarded as features, such as the vehicle GPS signal, the date and time of current trip, the weather condition and so on. More experiments can be conducted by using the extended SVM on those data.

7.2.2 IMPC

The IMPC enable to recognize driving pattern with the data un-fully sampled in a much shorter time interval. However, the performance of the IMPC is not as stable as other DPR system in the experiments, especially when the recognition period is less than 100 seconds. In order to stabilize the performance and increase the accuracy for the IMPC, more research could be conducted in optimizing the inertial factor updating function (Equation 5.28) or in designing a more robust dictionary $\boldsymbol{D}$ by employing dictionary-learning algorithms.

7.2.3 IEA

The IEA algorithm works perfectly for a given trip with fixed initial states. However, it still suffers from couple problems for random initial states case and uncertain trip case. For the random initial cases, IEA needs to explore much more states in the state space, which leads to increased difficulties in training the Q-network. Because of the limitation of the computation resources, the presented results in that chapter could not be the optimal results. More works are expected to improve the Q function approximation, including improving the training method or choose a better machine learning model. For the uncertain trip case, a more sophisticated algorithm could be developed to model all repeated trips into a Markov Decision Process. By doing that, only one Q-network is needed for solving this problem. Moreover, this proposed IEA algorithm needs to be further validated on a real EV with a real HESS. Or, it is also very interesting to see the performance of this IEA, if it could be trained by using the real data collected from a real HESS.

7.3 Potential Improvements by Using Nowadays Machine Learning Algorithms

This work was started several years ago. During this work, machine learning was also being developed really fast, especially the Deep Learning algorithms and the Reinforcement Learning. Here, we would like to have some discussion about the potential improvements about this work by using the new proposed machine learning algorithms. First of all, much more sophisticated signal data, including vehicle GPS signals, videos or images from the vehicle on-board camera and etc., can be added to the driving pattern recognition system as features. In order to process these sophisticated features, new machine learning algorithms need to be employed. For example, driving pattern could be also recognized from the real-time videos by using the Convolutional Neural Network (CNN), which is currently widely used in autonomous driving system. Secondly, a deep Q-network model could be used in the IEA to learn the Q-function, so that the estimation of the Q-function can be greatly improved. Moreover, IEA currently needs to repeat the target trip around 200 times, so that it would be able to fully learn the environment and figure out the proper strategy. The Monte-Carlo tree search method implemented in the AlphaGo [192] could be also a solution to reduce the repeated times. More research needs to be conducted to investigate the feasibility of applying the Monte-Carlo tree search into this algorithm. Thirdly, the Trip Mode in IEA is currently recognized by using the Bayesian method in Equation 6.24. According to that equation, the Trip Mode is decided by two items, the trip probability item $Pr(\boldsymbol{P}_i)$ and the current condition item $Pr(\boldsymbol{P}_{driving}|\boldsymbol{P}_i)$. The trip probability item $Pr(\boldsymbol{P}_i)$ is highly related to the driver's habit, life radius and etc. Therefore, we can use the face recognition algorithm based on CNN to recognize current driver and update this item. The second item $Pr(\boldsymbol{P}_{driving}|\boldsymbol{P}_i)$ is not related to the driver, but only decided by the previous driving patterns inside the current trip. Therefore, this item can be updated and calibrated on cloud by using the real-time data collected from other cars inside the relevant area. Last but not least, the Trip Mode data for each driver should be stored on the cloud. These data should also be open to all automakers, so that they can pre-train the IEA for new cars in factory for any driver.

Appendix A

Matlab vs. Python

Autonomie is a very powerful and robust system simulation tool for vehicle energy consumption and performance analysis. Developed in collaboration with General Motors by the Argonne System Modeling and Control Group, Autonomie is a MATLAB based software environment and framework for automotive control-system design, simulation, and analysis. [207] MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include: math and computation, algorithm development, modeling, simulation, and prototyping, data analysis, exploration, and visualization, scientific and engineering graphics, and application development, including Graphical User Interface building. Moreover, Matlab has a graphical simulation platform, Simulink. In Simulink, people can build data-driven models, mathematical models and physical models. Matlab has a solid amount of functions and a large scientific community. However, Matlab is a commercial product and it is very expensive. Furthermore, the algorithms or functions inside Matlab are proprietary, which means you can not see and modify the code of the algorithms you are using. This makes it difficult/impossible for researchers to extend or improve the algorithms in Matlab. Last but not least, Matlab is a high level software package, which eats a lot of additional CPU resources and memory in the backend. It usually costs a lot of time to train a machine learning model or doing some dynamic programming. Therefore, we use Python to develop the HESS model and the IEA algorithm.

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Develop-

ment, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. [208]. Moreover, Python is also very powful in scientific computation, by using its popular packages such as Pandas, Numpy, Scipy and etc. Recently, Python has become a very welcomed tools for developing machine learning algorithms. Besides its useful machine learning Scikit-Learn, Google Inc. also developed a powerful machine learning package, called TensorFlow, which has been widely used in Google's AI projects, such as Goole Brain and AlphaGo. [209]. Python and most of its packages, including TensorFlow, are all open source projects and free to use. Researchers can read and modify all codes in those packages and develop their own packages for reusing and sharing. Moreover, Python is by default already installed and running in most Unix-like operation systems, such as MacOS, Linux-Ubuntu and etc. Therefore, almost no additional computational resources are cost by running a Python scripts compared to using Matlab. Google, Amazon and other high-tech companies even use python to develop their products because of its high efficiency.

Appendix B

Simulator Package and Simulation Environment

This Simulator package is hosted on the Pypi server (`https://pypi.python.org/pypi/simulator/1.0.2`) and it is super easy to install. In a Mac or a Windows PC with python and pip installed, just run following code in the shell or command line to install Simulator.

```
pip install simulator
```

:

Furthermore, Git is used to control the version of Simulator. All source codes of Simulator can be found on the Bitbucket Webpage (`https://bitbucket.org/JasonX_Zhang/simulator`), which can be downloaded to your local machine by using

```
git clone https://bitbucket.org/JasonX_Zhang/simulator
```

:

A simulation environment is also set up on the Amazon Web Service Cloud (AWS)(`http://18.220.213.47:8888/simulator/`)

Bibliography

[1] C. C. Chan and Y. S. Wong, "Electric vehicles charge forward," *IEEE Power and Energy Magazine*, vol. 2, pp. 24–33, Nov 2004.

[2] E. NHTSA, "Joint technical support document: final rule making for 2017 - 2025 light- duty vehicle greenhouse gas emission standards and corporate average fuel economy standards," 2012.

[3] U. E. I. A. (EIA), "Annual energy outlook 2013," p. 70, 2013.

[4] S. J. information report, "Hybrid vehicle (hev) and electric vehicle terminology," 2007.

[5] J. Miller, "Propulsion systems for hybrid vehicles, london: The institution of engineering and technology," *The Institution of Engineering and Technology*, 2004.

[6] X. Wang, "Modeling and experiment of compressed air hybrid engines," *Lost Angeles: University of California*, 2008.

[7] E. Tanaka, "Fuel economy analysis of alternator with kinetic energy storage for a conventional vehicle," *SAE International Journal of Passenger Cars - Mechanical Systems*, vol. 6, no. 2, 2013.

[8] A. Emadi, K. Rajashekara, S. S. Williamson, and S. M. Lukic, "Topological overview of hybrid electric and fuel cell vehicular power system architectures and configurations," *IEEE Transactions on Vehicular Technology*, vol. 54, pp. 763–770, May 2005.

[9] M. Ehsani, Y. Gao, and J. M. Miller, "Hybrid electric vehicles: Architecture and motor drives," *Proceedings of the IEEE*, vol. 95, pp. 719–728, April 2007.

[10] C. C. Chan, A. Bouscayrol, and K. Chen, "Electric, hybrid, and fuel-cell vehicles: Architectures and modeling," *IEEE Transactions on Vehicular Technology*, vol. 59, pp. 589–598, Feb 2010.

[11] A. E. Mehrhad Ehsani, Yimin Gao, "Modern vehicles in modern electric, hybrid electric, and fuel cell vehicles: Fundamentals, theory, and design," *CRC PRESS*, 2005.

[12] J. Larminie and J. Lowry, *Electric Vehicle Technology Explained*. Wiley, 2004.

[13] "The first hybrid vehicle." `http://www.hybrid-vehicle.org`. Accessed 13-March-2012.

[14] A. Kraytsberg and Y. Ein-Eli, "Review on li–air batteries—opportunities, limitations and perspective," *Journal of Power Sources*, vol. 196, no. 3, pp. 886 – 893, 2011.

[15] J.-G. Zhang, D. Wang, W. Xu, J. Xiao, and R. Williford, "Ambient operation of li/air batteries," *Journal of Power Sources*, vol. 195, no. 13, pp. 4332 – 4337, 2010.

[16] L. L. Gaines and R. Cuenca, "Costs of lithium-ion batteries for vehicles," p. 73, 20000501 2000.

[17] A. Burke and H. Zhao, "Present and future applications of supercapacitors in electric and hybrid vehicles," in *Vehicular Technology Conference (VTC Fall), 2015 IEEE 82nd*, pp. 1–5, Sept 2015.

[18] "Electropaedia, battery and energy technologies." `http://www.mpoweruk.com/performance.htm`. Accessed 13-March-2013.

[19] Z. Xing, D. Zuomin, and C. Curran, "Hybrid energy storage system for hybrid and electric vehicles: Review and a new control strategy.," in *ASME International Mechanical Engineering Congress and Exposition*, vol. 4, pp. 91–101, Sept 2011.

[20] X. Zhang, G. Wu, Z. Dong, and C. Crawford, "Embedded feature-selection support vector machine for driving pattern recognition," *Journal of the Franklin Institute*, vol. 352, no. 2, pp. 669 – 685, 2015.

[21] J. Neumann, C. Schnorr, and G. Steidl, "Combined SVM-based feature selection and classification," *Machine Learning*, 2005.

[22] H. Zou, "The Adaptive Lasso and Its Oracle Properties," *Journal of the American Statistical Association*, vol. 101, pp. 1418–1429, Dec. 2006.

[23] J. Yang, X. Huang, Y. Tan, and X. He, "Forecast of driving load of hybrid electric vehicles by using discrete cosine transform and support vector machine," in *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pp. 2227 –2234, june 2008.

[24] J.-M. Tarascon and M. Armand, "Issues and challenges facing rechargeable lithium batteries," *Nature*, vol. 414, pp. 359–367, nov 2001.

[25] S. Pacala and R. Socolow, "Stabilization wedges: Solving the climate problem for the next 50 years with current technologies," *Science*, vol. 305, no. 5686, pp. 968–972, 2004.

[26] L. Lu, X. Han, J. Li, J. Hua, and M. Ouyang, "A review on the key issues for lithium-ion battery management in electric vehicles," *Journal of Power Sources*, vol. 226, pp. 272–288, 2013.

[27] N. Nitta, F. Wu, J. T. Lee, and G. Yushin, "Li-ion battery materials: Present and future," *Materials Today*, vol. 18, no. 5, pp. 252–264, 2015.

[28] H. Vikström, S. Davidsson, and M. Höök, "Lithium availability and future production outlooks," *Applied Energy*, vol. 110, pp. 252 – 266, 2013.

[29] T. M. Bandhauer, S. Garimella, and T. F. Fuller, "A critical review of thermal issues in lithium-ion batteries," *Journal of The Electrochemical Society*, vol. 158, no. 3, pp. R1–R25, 2011.

[30] M. N. Richard and J. R. Dahn, "Accelerating rate calorimetry study on the thermal stability of lithium intercalated graphite in electrolyte. i. experimental," *Journal of The Electrochemical Society*, vol. 146, no. 6, pp. 2068–2077, 1999.

[31] R. Spotnitz and J. Franklin, "Abuse behavior of high-power, lithium-ion cells," *Journal of Power Sources*, vol. 113, no. 1, pp. 81 – 100, 2003.

[32] G. Venugopal, "Characterization of thermal cut-off mechanisms in prismatic lithium-ion batteries," *Journal of Power Sources*, vol. 101, no. 2, pp. 231 – 237, 2001.

[33] L. Moshuchak, M. Bulinski, W. Lamanna, R. Wang, and J. Dahn, "Direct comparison of 2,5-di-tert-butyl-1,4-dimethoybenzene and 4-tert-butyl-1,2-dimethoxybenzene as redox shuttles in lifepo4-based li-ion cells," *Electrochemistry Communications*, vol. 9, no. 7, pp. 1497 – 1501, 2007.

[34] J. Fan and S. Tan, "Studies on charging lithium-ion cells at low temperatures," *Journal of The Electrochemical Society*, vol. 153, no. 6, pp. A1081–A1092, 2006.

[35] "Lithium battery failures." `http://www.mpoweruk.com/lithium\_failures.htm`. Accessed 12-Oct-2015.

[36] P. Arora, R. E. White, and M. Doyle, "Capacity fade mechanisms and side reactions in lithium-ion batteries," *Journal of The Electrochemical Society*, vol. 145, no. 10, pp. 3647–3667, 1998.

[37] S. Matsuta, Y. Kato, T. Ota, H. Kurokawa, S. Yoshimura, and S. Fujitani, "Electron-spin-resonance study of the reaction of electrolytic solutions on the positive electrode for lithium-ion secondary batteries," *Journal of The Electrochemical Society*, vol. 148, no. 1, pp. A7–A10, 2001.

[38] U. Manuals, "Electric vehicle battery test procedures manual," vol. Reversion 2.

[39] "Wikipedia." `https://en.wikipedia.org/wiki/State_of_health`. Accessed 21-Jan-2014.

[40] D. Andrea, *Battery Management Systems for Large Lithium-Ion Battery Packs, first ed.* Artech House, September 30, 2010.

[41] M. Zhu, W. Hu, and N. C. Kar, "The soh estimation of lifepo4 battery based on internal resistance with grey markov chain," in *2016 IEEE Transportation Electrification Conference and Expo (ITEC)*, pp. 1–6, June 2016.

[42] L. W. Juang, "Online battery monitoring for state-of-charge and power capability prediction," *Master dissertation*, vol. University of Wisconsin Madison, 2010.

[43] "Battery life and how to improve it." `http://www.mpoweruk.com/life.htm\#changes`. Accessed 29-May-2015.

[44] J. Zhou, Z. He, M. Gao, and Y. Liu, "Battery State of Health Estimation Using The Generalized Regression Neural Network," *2015 8th International Congress on Image and Signal Processing (CISP 2015)*, no. Cisp, pp. 1396–1400, 2015.

[45] N. Omar, M. A. Monem, Y. Firouz, J. Salminen, J. Smekens, O. Hegazy, H. Gaulous, G. Mulder, P. V. den Bossche, T. Coosemans, and J. V. Mierlo, "Lithium iron phosphate based battery – assessment of the aging parameters and development of cycle life model," *Applied Energy*, vol. 113, pp. 1575 – 1585, 2014.

[46] B. Saha, K. Goebel, and J. Christophersen, "Comparison of prognostic algorithms for estimating remaining useful life of batteries," *Transactions of the Institute of Measurement and Control*, 2009.

[47] M. Dalal, J. Ma, and D. He, "Lithium-ion battery life prognostic health management system using particle filtering framework," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 225, no. 1, pp. 81–90, 2011.

[48] S. Wang, L. Zhao, X. Su, and P. Ma, "Prognostics of lithium-ion batteries based on battery performance analysis and flexible support vector regression," *Energies*, vol. 7, no. 10, p. 6492, 2014.

[49] J. Li, C. Lyu, L. Wang, L. Zhang, and C. Li, "Remaining capacity estimation of li-ion batteries based on temperature sample entropy and particle filter," *Journal of Power Sources*, vol. 268, pp. 895 – 903, 2014.

[50] J. Li, L. Wang, C. Lyu, L. Zhang, and H. Wang, "Discharge capacity estimation for li-ion batteries based on particle filter under multi-operating conditions," *Energy*, vol. 86, pp. 638 – 648, 2015.

[51] A. Widodo, M.-C. Shim, W. Caesarendra, and B.-S. Yang, "Intelligent prognostics for battery health monitoring based on sample entropy," *Expert Systems with Applications*, vol. 38, no. 9, pp. 11763 – 11769, 2011.

[52] G. K. Prasad and C. D. Rahn, "Model based identification of aging parameters in lithium ion batteries," *Journal of Power Sources*, vol. 232, pp. 79 – 85, 2013.

[53] A. P. Schmidt, M. Bitzer, Árpád W. Imre, and L. Guzzella, "Model-based distinction and quantification of capacity loss and rate capability fade in li-ion batteries," *Journal of Power Sources*, vol. 195, no. 22, pp. 7634 – 7638, 2010.

[54] S. K. Rahimian, S. Rayman, and R. E. White, "Comparison of single particle and equivalent circuit analog models for a lithium-ion cell," *Journal of Power Sources*, vol. 196, no. 20, pp. 8450 – 8462, 2011.

[55] J. Li, C. Lyu, L. Wang, and T. Ge, "Model-based method for estimating licoo2 battery state of health and behaviors," in *2016 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 1–4, June 2016.

[56] H. Becker, "Low voltage electrolytic capacitor," July 23 1957. US Patent 2,800,616.

[57] D. Boos, "Electrolytic capacitor having carbon paste electrodes," Oct. 27 1970. US Patent 3,536,963.

[58] A. Burke, "Ultracapacitor technologies and application in hybrid and electric vehicles," *International Journal of Energy Research*, vol. 34, no. 2, pp. 133–151, 2010.

[59] A. W. Stienecker, T. Stuart, and C. Ashtiani, "An ultracapacitor circuit for reducing sulfation in lead acid batteries for mild hybrid electric vehicles," *Journal of Power Sources*, vol. 156, no. 2, pp. 755 – 762, 2006.

[60] D. Eichenberg, *Baseline Testing of the Ultracapacitor Enhanced Photovoltaic Power Station*. 2001.

[61] A. Schneuwly, "Ultracapacitors improve reliability for wind turbine pitch systems," *White paper-Maxwell Technologies.*

[62] M. T. Inc., "Top 10 reasons for using ultracapacitors in your system designs," *MAXWELL TECHNOLOGIES WHITE PAPER*, 2013.

[63] B. E. Conway, *Electrochemical supercapacitors : scientific fundamentals and technological applications*. New York, N.Y. Kluwer Academic / Plenum Publishers, 1999.

[64] R. Koetz and M. Carlen, "Principles and applications of electrochemical capacitors," *Electrochimica Acta*, vol. 45, no. 15–16, pp. 2483 – 2498, 2000.

[65] "General ultracapacitor overview." `http://www.solrayo.com/SolRayo/General_Overview.html`. Accessed 31-June-2016.

[66] A. Burke, Z. Liu, and H. Zhao, "Present and future applications of supercapacitors in electric and hybrid vehicles," in *2014 IEEE International Electric Vehicle Conference (IEVC)*, pp. 1–8, Dec 2014.

[67] R. King, X. Huang, and G. Kilinski, "Vehicle propulsion system," July 19 2007. US Patent App. 11/614,412.

[68] J. Dixon and M. Ortuzar, "Ultracapacitors dc-dc converters in regenerative braking system," *IEEE Aerospace and Electronic Systems Magazine*, vol. 17, no. 8, p. 16–21, 2002.

[69] R. Carter and A. Cruden, "Strategies for control of a battery/supercapacitor system in an electric vehicle," *2008 International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, 2008.

[70] L. Rosario and P. C. K. Luk, "Applying management methodology to electric vehicles with multiple energy storage systems," *2007 International Conference on Machine Learning and Cybernetics*, 2007.

[71] B. A. Niemoeller and P. T. Krein, "Battery-ultracapacitor active parallel interface with indirect control of battery current," *2010 Power and Energy Conference At Illinois (PECI)*, 2010.

[72] J. Schindall, "The charge of the ultracapacitors," *IEEE Spectrum*, vol. 44, pp. 42–46, Nov 2007.

[73] J. Larminie and J. Lowry, *Electric vehicle technology explained*. J. Wiley, 2003.

[74] A. Emadi, *Handbook of automotive power electronics and motor drives*. Taylor, Francis, 2005.

[75] A. Jossen, "Fundamentals of battery dynamics," *Journal of Power Sources*, vol. 154, no. 2, p. 530–538, 2006.

[76] I. Buchmann, "How to prolong lithium based batteries." http://www.batteryuniversity.com/parttwo-34.htm. Accessed: June 16, 2014.

[77] I. Buchmann, "Discharge methods." http://www.batteryuniversity.com/partone-16.htm. Accessed: June 16, 2014.

[78] S. S. Choi and H. S. Lim, "Factors that affect cycle-life and possible degradation mechanisms of a li-ion cell based on licoo2," *Journal of Power Sources*, vol. 111, no. 1, p. 130–136, 2002.

[79] J. M. Miller, "Will market accept battery-ultracapacitor combinations?." http://planetee.com/events/. Accessed: June 18, 2014.

[80] A. Kuperman and I. Aharon, "Battery-ultracapacitor hybrids for pulsed current loads: A review," *Renewable and Sustainable Energy Reviews*, vol. 15, no. 2, pp. 981–992, 2011.

[81] M. Pagano and L. Piegari, "Hybrid electrochemical power sources for onboard applications," *IEEE Transactions on Energy Conversion*, vol. 22, no. 2, pp. 450–456, 2007.

[82] D. Cericola, P. Ruch, R. Katz, P. Novak, and A. Wokaun, "Simulation of a supercapacitor/li-ion battery hybrid for pulsed applications," *Journal of Power Sources*, vol. 195, no. 9, pp. 2731–2736, 2010.

[83] M. Penella and M. Gasulla, "Runtime extension of low-power wireless sensor nodes using hybrid-storage units," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 4, p. 857–865, 2010.

[84] H. Liu, Z. Wang, J. Cheng, and D. Maly, "Improvement on the cold cranking capacity of commercial vehicle by using supercapacitor and lead-acid battery hybrid," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 3, pp. 1097–1105, 2009.

[85] C. Menachem and H. Yamin, "High-energy, high-power pulses battery for long-term applications," *Journal of Power Sources*, vol. 136, no. 2, pp. 268–275, 2004.

[86] S. H. Choi, J. Kim, and Y. S. Yoon, "Fabrication and characterization of a licoo2 battery-supercapacitor combination for a high-pulse power system," *Journal of Power Sources*, vol. 138, no. 1-2, pp. 360– 363, 2004.

[87] X. Hu, Z. Deng, J. Suo, and Z. Pan, "A high rate, high capacity and long life (limn2o4 ac)/li4ti5o12 hybrid battery- supercapacitor," *Journal of Power Sources*, vol. 187, no. 2, pp. 635–639, 2009.

[88] A. Kuperman and I. Aharon, "Battery-ultracapacitor hybrids for pulsed current loads: A review," *Renewable and Sustainable Energy Reviews*, vol. 15, no. 2, pp. 981–992, 2011.

[89] Y. Gao and M. Ehsani, "Parametric design of the traction motor and energy storage for series hybrid off-road and military vehicles," *IEEE Transactions on Power Electronics*, vol. 21, no. 3, pp. 749–755, 2006.

[90] J. Cao and A. Emadi, "A new battery/ultra-capacitor hybrid energy storage system for electric, hybrid and plug-in hybrid electric vehicles," *2009 IEEE Vehicle Power and Propulsion Conference*, 2009.

[91] A. Allegre, A. Bouscayrol, and R. Trigui, "Influence of control strategies on battery/supercapacitor hybrid energy storage systems for traction applications," *2009 IEEE Vehicle Power and Propulsion Conference*, 2009.

[92] X. Liu, Q. Zhang, and C. Zhu, "Design of battery and ultracapacitor multiple energy storage in hybrid electric vehicle," *2009 IEEE Vehicle Power and Propulsion Conference*, 2009.

[93] J. M. Miller, "Trends in vehicle energy storage systems: Batteries and ultracapacitors to unite," *2008 IEEE Vehicle Power and Propulsion Conference*, 2008.

[94] A. M. V. Voorden, L. M. R. Elizondo, G. C. Paap, J. Verboomen, and L. V. D. Sluis, "The application of super capacitors to relieve battery-storage systems in autonomous renewable energy systems," *2007 IEEE Lausanne Power Tech*, 2007.

[95] A. Govindaraj, S. Lukic, and A. Emadi, "A novel scheme for optimal paralleling of batteries and ultracapacitors," *2009 IEEE Energy Conversion Congress and Exposition*, 2009.

[96] A. Khaligh and Z. Li, "Battery, ultracapacitor, fuel cell, and hybrid energy storage systems for electric, hybrid electric, fuel cell, and plug-in hybrid electric vehicles: State of the art," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 6, pp. 2806–2814, 2010.

[97] D. L. Cheng and M. G. Wismer, "Active control of power sharing in a battery/ultracapacitor hybrid source," *2007 2nd IEEE Conference on Industrial Electronics and Applications*, 2007.

[98] M. B. Camara, B. Dakyo, H. Gualous, and C. Nichita, "Full bridge converter for embedded energy share between battery and supercapacitors," *2009 35th Annual Conference of IEEE Industrial Electronics*, 2009.

[99] V. A. Shah, S. G. Karndhar, R. Maheshwari, P. Kundu, and H. Desai, "An energy management system for a battery ultracapacitor hybrid electric vehicle," *2009 International Conference on Industrial and Information Systems (ICIIS)*, 2009.

[100] X. Rui, H. Hongwen, Z. Xiaowei, and W. Yi, "Simulation study on hybrid ultracapactior-battery power system for phev," *2010 2nd International Conference on Future Computer and Communication*, 2010.

[101] J. M. Miller, U. Deshpande, T. J. Dougherty, and T. Bohn, "Power electronic enabled active hybrid energy storage system and its economic viability," *2009 Twenty-Fourth Annual IEEE Applied Power Electronics Conference and Exposition*, 2009.

[102] Y. Yuanbin, W. Qingnian, H. Changjian, and W. Boshi, "The feasibility and superiority of super-capacitors on mild hybrid electric vehicle," *2009 International Conference on Mechatronics and Automation*, 2009.

[103] Z. Amjadi and S. Williamson, "Power-electronics-based solutions for plug-in hybrid electric vehicle energy storage and management systems," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 2, pp. 608–616, 2010.

[104] M. Glavin, P. K. Chan, S. Armstrong, and W. Hurley, "A stand-alone photovoltaic supercapacitor battery hybrid energy storage system," *2008 13th International Power Electronics and Motion Control Conference*, 2008.

[105] O. Onar and A. Khaligh, "Dynamic modeling and control of a cascaded active battery/ultra-capacitor based vehicular power system," *2008 IEEE Vehicle Power and Propulsion Conference*, 2008.

[106] N. Kim, A. Rousseau, and E. Rask, "Autonomie model validation with test data for 2010 toyota prius," in *SAE Technical Paper*, SAE International, 04 2012.

[107] J. Voelcker, "Toyota hybrid battery replacement cost guide (2016 update)." `http://www.greencarreports.com/news/1078138_toyota-hybrid-battery-replacement-cost-guide`. Accessed: December 1, 2016.

[108] S. Buller, E. Karden, D. Kok, and R. W. D. Doncker, "Modeling the dynamic behavior of supercapacitors using impedance spectroscopy," *IEEE Transactions on Industry Applications*, vol. 38, pp. 1622–1626, Nov 2002.

[109] E. Ericsson, "Independent driving pattern factors and their influence on fuel-use and exhaust emission factors," *Transportation Research Part D: Transport and …*, vol. 6, 2001.

[110] X. Huang, Y. Tan, and X. He, "An intelligent multifeature statistical approach for the discrimination of driving conditions of a hybrid electric vehicle," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, pp. 453 –465, june 2011.

[111] H. Zhang, J. Wang, and Y. Shi, "Robust sliding-mode control for markovian jump systems subject to intermittent observations and partially known transition probabilities," *Systems Control Letters*, vol. 62, no. 12, pp. 1114 – 1124, 2013.

[112] D. Di Domenico, G. Fiengo, and A. Stefanopoulou, "Lithium-ion battery state of charge estimation with a kalman filter based on a electrochemical model," in *Control Applications, 2008. CCA 2008. IEEE International Conference on*, pp. 702–707, 2008.

[113] H. He, R. Xiong, X. Zhang, F. Sun, and J. Fan, "State-of-charge estimation of the lithium-ion battery using an adaptive extended kalman filter based on an improved thevenin model," *Vehicular Technology, IEEE Transactions on*, vol. 60, no. 4, pp. 1461–1469, 2011.

[114] H. Zhang, Y. Shi, and J. Wang, "Observer-based tracking controller design for networked predictive control systems with uncertain markov delays," *International Journal of Control*, vol. 86, no. 10, pp. 1824–1836, 2013.

[115] R. Wang and S. Lukic, "Review of driving conditions prediction and driving style recognition based control algorithms for hybrid electric vehicles," in *Vehicle Power and Propulsion Conference (VPPC), 2011 IEEE*, pp. 1–7, 2011.

[116] L. Hermes and J. Buhmann, "Feature selection for support vector machines," in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 2, pp. 712–715 vol.2, 2000.

[117] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for svms," in *Advances in Neural Information Processing Systems 13*, pp. 668–674, MIT Press, 2000.

[118] P. Bradley and O. Mangasarian, "Feature selection via concave minimization and support vector machines.," *ICML*, vol. 6, 1998.

[119] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, "1-norm support vector machines," in *The Annual Conference on Neural Information Processing Systems 16*, pp. 168–174, MIT Press, 2004.

[120] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[121] C. Bhattacharyya, "Second order cone programming formulations for feature selection," *The Journal of Machine Learning Research*, vol. 5, pp. 1417–1433, 2004.

[122] H. Zou, "An improved 1-norm svm for simultaneous classification and variable selection," in *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, pp. 675–681, 2007.

[123] Y. Xu, P. Zhong, and L. Wang, "Support vector machine-based embedded approach feature selection algorithm," *Journal of Informa-tion and Computational …*, vol. 5, no. 10771213, pp. 1155–1163, 2010.

[124] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, pp. 129–159, Jan. 2001.

[125] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[126] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, "Applications of second-order cone programming," *Linear Algebra and its Applications*, vol. 284, pp. 193–228, Nov. 1998.

[127] A. Antoniou and W.-S. Lu, *Practical optimization - algorithms and engineering applications*. Springer, 2007.

[128] K. Bache and M. Lichman, "UCI machine learning repository," 2013.

[129] A. Barth and U. Franke, "Estimating the driving state of oncoming vehicles from a moving platform using stereo vision," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, pp. 560 –571, dec. 2009.

[130] B. Morris and M. Trivedi, "Learning, modeling, and classification of vehicle track patterns from live video," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 9, pp. 425 –437, sept. 2008.

[131] Y. Murphey, J. Park, Z. Chen, M. Kuang, M. Masrur, and A. Phillips, "Intelligent hybrid vehicle power control-part i: Machine learning of optimal vehicle power," *Vehicular Technology, IEEE Transactions on*, vol. 61, pp. 3519–3530, Oct 2012.

[132] S.-I. Jeon, S.-T. Jo, Y.-I. Park, and J.-M. Lee, "Multi-mode driving control of a parallel hybrid electric vehicle using driving pattern recognition," *Journal of dynamic systems, measurement, and control*, vol. 124, pp. 141–149, Oct 2002.

[133] J. Park, Z. Chen, L. Kiliaris, M. Kuang, M. Masrur, A. Phillips, and Y. Murphey, "Intelligent vehicle power control based on machine learning of optimal control parameters and prediction of road type and traffic congestion," *Vehicular Technology, IEEE Transactions on*, vol. 58, pp. 4741–4756, Nov 2009.

[134] J. C. Herrera and A. M. Bayen, "Incorporation of Lagrangian measurements in freeway traffic state estimation," *Transportation Research Part B: Methodological*, vol. 44, no. 4, pp. 460–481, 2010.

[135] H. B. Celikoglu, "An Approach to Dynamic Classification of Traffic Flow Patterns," *Computer-Aided Civil and Infrastructure Engineering*, vol. 28, pp. 273–288, Apr. 2013.

[136] L. Mihaylova, R. Boel, and A. Hegyi, "Freeway traffic estimation within particle filtering framework," *Automatica*, 2007.

[137] M. Montazeri-Gh and M. Naghizadeh, "Development of the tehran car driving cycle," *International Journal of Environment and Pollution*, vol. 30, pp. 106 – 118, 2007.

[138] Y. Ma, M. Chowdhury, A. Sadek, and M. Jeihani, "Real-time highway traffic condition assessment framework using vehicle - infrastructure integration (vii) with artificial intelligence (ai)," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, pp. 615–627, Dec 2009.

[139] M. Borner, L. Andreani, P. Albertos, and R. Isermann, "Detection of lateral vehicle driving conditions based on the characteristic velocity," *Proc. Triennial World Congr. Int. Fed. Autom. Control*, Jul 2002.

[140] J. Krumm, "How people use their vehicles: Statistics from the 2009 national household travel survey," *SAE 2012 World Congress and Exhibition*, April 2012.

[141] E. Candes and M. Wakin, "An introduction to compressive sampling," *Signal Processing Magazine, IEEE*, vol. 25, pp. 21 –30, march 2008.

[142] M. Montazeri-Gh, A. Ahmadi, and M. Asadi, "Driving condition recognition for genetic-fuzzy hev control," in *Genetic and Evolving Systems, 2008. GEFS 2008. 3rd International Workshop on*, pp. 65 –70, march 2008.

[143] R. Langari and J.-S. Won, "Intelligent energy management agent for a parallel hybrid vehicle-part i: system architecture and design of the driving situation identification process," *Vehicular Technology, IEEE Transactions on*, vol. 54, pp. 925 – 934, may 2005.

[144] T. R. Carlson and R. C. Austin, "Development of speed correction cycles," in *Sierra Research, Inc.*, (Sacramento, CA), pp. Report SR97–04–01, 1997.

[145] Transportation Research Board, Washington, DC, *Highway Capacity Manual 2000*, 2000.

[146] E. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *Information Theory, IEEE Transactions on*, vol. 52, pp. 5406 –5425, dec. 2006.

[147] D. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, pp. 1289 –1306, april 2006.

[148] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *Information Theory, IEEE Transactions on*, vol. 52, pp. 489 – 509, feb. 2006.

[149] W. Lu, "Selected topics in digital signal processing: Sparse signal processing and compressed sensing." Notes of Course ELEC 639A, Department of Electrical Energineering, University of Victoria, Spring 2011.

[150] I. Tosic and P. Frossard, "Dictionary learning," *Signal Processing Magazine, IEEE*, vol. 28, pp. 27 –38, march 2011.

[151] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *Signal Processing, IEEE Transactions on*, vol. 41, pp. 3397 –3415, dec 1993.

[152] J. Tropp, "Greed is good: algorithmic results for sparse approximation," *Information Theory, IEEE Transactions on*, vol. 50, pp. 2231 – 2242, oct. 2004.

[153] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *Information Theory, IEEE Transactions on*, vol. 55, pp. 2230–2249, May 2009.

[154] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[155] D. Needell and R. Vershynin, "Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, pp. 310–316, April 2010.

[156] D. Needell, J. Tropp, and R. Vershynin, "Greedy signal recovery review," in *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*, pp. 1048–1050, Oct 2008.

[157] Y. Chen, N. Nasrabadi, and T. Tran, "Hyperspectral image classification using dictionary-based sparse representation," *Geoscience and Remote …*, vol. 49, no. 10, pp. 3973–3985, 2011.

[158] S. Mallat, *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, 3rd ed., 2008.

[159] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proceedings of the IEEE*, vol. 98, pp. 1031–1044, June 2010.

[160] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, pp. 210–227, Feb 2009.

[161] J. Pillai, V. Patel, and R. Chellappa, "Sparsity inspired selection and recognition of iris images," in *Biometrics: Theory, Applications, and Systems, 2009. BTAS '09. IEEE 3rd International Conference on*, pp. 1–6, Sept 2009.

[162] C.-H. Zheng, D. Zhang, T.-Y. Ng, S. C. K. Shiu, and D.-S. Huang, "Metasample-based sparse representation for tumor classification," *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, vol. 8, pp. 1273–1282, Sept 2011.

[163] O. S. Guo Zhaohui, Wittman Todd, "L1 unmixing and its application to hyperspectral image enhancement," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 7334 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, May 2009.

[164] Y. Chen, N. Nasrabadi, and T. Tran, "Sparsity-based classification of hyperspectral imagery," in *Geoscience and Remote Sensing Symposium (IGARSS), 2010 IEEE International*, pp. 2796–2799, July 2010.

[165] R. Nock and F. Nielsen, "On weighting clustering," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, pp. 1223–1235, Aug 2006.

[166] M. Aharon, M. Elad, and A. Bruckstein, "k -svd: An algorithm for designing overcomplete dictionaries for sparse representation," *Signal Processing, IEEE Transactions on*, vol. 54, pp. 4311–4322, Nov 2006.

[167] R. Rubinstein, A. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, pp. 1045–1057, June 2010.

[168] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning.," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 4, pp. 791–804, 2012.

[169] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Img. Sci.*, vol. 2, pp. 183–202, Mar. 2009.

[170] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *Image Processing, IEEE Transactions on*, vol. 18, pp. 2419 –2434, nov. 2009.

[171] F. S. Garcia, A. A. Ferreira, and J. A. Pomilio, "Control strategy for battery-ultracapacitor hybrid energy storage system," *2009 Twenty-Fourth Annual IEEE Applied Power Electronics Conference and Exposition*, 2009.

[172] A. A. A. Al-Karakchi, G. Lacey, and G. Putrus, "A method of electric vehicle charging to improve battery life," *2015 50th International Universities Power Engineering Conference (UPEC)*, 2015.

[173] S. Bradley, A. Hax, and T. Magnanti, *Applied Mathematical Programming*. Addison-Wesley Publishing Company, 1977.

[174] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits and Systems Magazine*, vol. 9, pp. 32–50, Third 2009.

[175] R. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press, 1 ed., 1957.

[176] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*. The MIT Press, 2012.

[177] A. Gosavi, "Reinforcement learning: A tutorial survey and recent advances," *INFORMS J. on Computing*, vol. 21, pp. 178–192, Apr. 2009.

[178] P. J. Werbos, "Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 17, 1987.

[179] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Artificial neural networks," ch. Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems, pp. 81–93, Piscataway, NJ, USA: IEEE Press, 1990.

[180] J. M. Mendel and R. W. McLaren, "A prelude to neural networks," ch. Reinforcement-learning Control and Pattern Recognition Systems, pp. 287–318, Upper Saddle River, NJ, USA: Prentice Hall Press, 1994.

[181] R. Crites and A. Barto, "Improving elevator performance using reinforcement learning," in *Advances in Neural Information Processing Systems 8*, pp. 1017–1023, MIT Press, 1996.

[182] D. Pepyne, *Application of Q-learning to Elevator Dispatching*. University of Massachusetts at Amherst, 1995.

[183] W. Zhang and T. G. Dietterich, "A reinforcement learning approach to job-shop scheduling," in *In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1114–1120, Morgan Kaufmann, 1995.

[184] T. K. Das, A. Gosavi, S. Mahadevan, and N. Marchalleck, "Solving semi-markov decision problems using average reward reinforcement learning," *Management Science*, vol. 45, no. 4, pp. 560–574, 1999.

[185] A. Gosavi, "Reinforcement learning for long-run average cost," *European Journal of Operational Research*, vol. 155, no. 3, pp. 654 – 674, 2004. Traffic and Transportation Systems Analysis.

[186] N. Akar and C. Sahin, *Reinforcement Learning as a Means of Dynamic Aggregate QoS Provisioning*, pp. 100–114. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[187] P. Pontrandolfo, A. Gosavi, O. G. Okogbaa, and T. K. Das, "Global supply chain management: A reinforcement learning approach," *International Journal of Production Research*, vol. 40, no. 6, pp. 1299–1317, 2002.

[188] A. Gosavi, "A reinforcement learning algorithm based on policy iteration for average reward: Empirical results with yield management and convergence analysis," *Machine Learning*, vol. 55, no. 1, pp. 5–29, 2004.

[189] A. Gosavi, N. Bandla, and T. K. Das, "A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking," *IIE Transactions*, vol. 34, no. 9, pp. 729–742, 2002.

[190] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, *Hyper-Heuristics: An Emerging Direction in Modern Search Technology*, pp. 457–474. Boston, MA: Springer US, 2003.

[191] M. Ghavamzadeh, S. Mahadevan, and R. Makar, "Hierarchical multi-agent reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 13, no. 2, pp. 197–229, 2006.

[192] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, jan 2016.

[193] S. J. Moura, J. L. Stein, and H. K. Fathy, "Battery-health conscious power management in plug-in hybrid electric vehicles via electrochemical modeling and stochastic control," *IEEE Transactions on Control Systems Technology*, vol. 21, pp. 679–694, May 2013.

[194] P. Ramadass, B. Haran, P. M. Gomadam, R. White, and B. N. Popov, "Development of first principles capacity fade model for li-ion cells," *Journal of The Electrochemical Society*, vol. 151, no. 2, pp. A196–A203, 2004.

[195] Y. Zhao, R. Fu, and S. y. Choe, "Modeling of sei formation based on a electrochemical reduced order model for li(mnnico)o2/carbon polymer battery," in *2015 IEEE Vehicle Power and Propulsion Conference (VPPC)*, pp. 1–4, Oct 2015.

[196] A. a. Mamun, I. Narayanan, D. Wang, A. Sivasubramaniam, and H. K. Fathy, "Battery health-conscious online power management for stochastic datacenter demand response," in *2016 American Control Conference (ACC)*, pp. 3206–3211, July 2016.

[197] Q. Gong, Y. Li, and Z. R. Peng, "Trip-based optimal power management of plug-in hybrid electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 57, pp. 3393–3401, Nov 2008.

[198] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods," *SIAM Review*, vol. 45, pp. 385–482, 2003.

[199] R. Hooke and T. A. Jeeves, "" direct search" solution of numerical and statistical problems," *J. ACM*, vol. 8, pp. 212–229, Apr. 1961.

[200] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, pp. 246–257, June 1991.

[201] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, oct 1986.

[202] T. Kurban and E. Beşdok, "A comparison of rbf neural network training algorithms for inertial sensor based terrain classification," *Sensors*, vol. 9, no. 8, pp. 6312–6329, 2009.

[203] S. Adam, L. Busoniu, and R. Babuska, "Experience replay for real-time reinforcement learning control," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, pp. 201–212, March 2012.

[204] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *CoRR*, vol. abs/1511.05952, 2015.

[205] J. N. Foerster, N. Nardelli, G. Farquhar, P. H. S. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," *CoRR*, vol. abs/1702.08887, 2017.

[206] J. C. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," tech. rep., Microsoft Research, April 21, 1998.

[207] Autonomie, "Autonomie homepage." `http://www.autonomie.net/expertise/Autonomie.html`. Accessed: April 16, 2017.

[208] P. Org., "Python summary." `https://www.python.org/doc/essays/blurb/`. Accessed: April 16, 2017.

[209] T. Team, "Tensor flow." https://www.tensorflow.org. Accessed: June 16, 2014.