# Driving Data Pattern Recognition for Intelligent Energy Management of Plug-in Hybrid Electric Vehicles

by

Sreejith Munthikodu
B. Tech., National Institute of Technology, Calicut, 2009

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

MASTER OF APPLIED SCIENCE
in the Department of Mechanical Engineering

# Supervisory Committee

Driving Data Pattern Recognition for Intelligent Energy Management of Plug-in Hybrid Electric Vehicles

by

Sreejith Munthikodu
B. Tech., National Institute of Technology, Calicut, 2009

**Supervisory Committee**

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

Dr. Curran Crawford, (Department of Mechanical Engineering)
**Departmental Member**

Dr. Caterina Valeo (Department of Mechanical Engineering)
**Departmental Member**

# Abstract

**Supervisory Committee**
Dr. Zuomin Dong, (Department of Mechanical Engineering)
**Supervisor**
Dr. Curran Crawford, (Department of Mechanical Engineering)
**Departmental Member**
Dr. Caterina Valeo (Department of Mechanical Engineering)
**Departmental Member**

This work focuses on the development and testing of new driving data pattern recognition intelligent system techniques to support driver adaptive, real-time optimal power control and energy management of hybrid electric vehicles (HEVs) and plug-in hybrid electric vehicles (PHEVs). A novel, intelligent energy management approach that combines vehicle operation data acquisition, driving data clustering and pattern recognition, cluster prototype based power control and energy optimization, and real-time driving pattern recognition and optimal energy management has been introduced. The method integrates advanced machine learning techniques and global optimization methods form the driver adaptive optimal power control and energy management. Fuzzy C-Means clustering algorithm is used to identify the representative vehicle operation patterns from collected driving data. Dynamic Programming (DA) based off-line optimization is conducted to obtain the optimal control parameters for each of the identified driving patterns. Artificial Neural Networks (ANN) are trained to associate each of the identified operation patterns with the optimal energy management plan to support real-time optimal control. Implementation and advantages of the new method are demonstrated using the 2012 California household travel survey data, and driver-specific data collected from the city of Victoria, BC Canada.

# Table of Contents

# List of Tables

# List of Figures

# Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Zuomin Dong, for providing me the opportunity to study at the University of Victoria and take up this extremely rewarding research work. This research would not have been possible without his constant guidance, motivation and support. I am grateful to him for introducing me to the amazing world of Artificial Intelligence (AI) and Machine Learning (ML). Through the research work, I developed a passion for AI and ML, transforming my career goals and ambitions. I am now committed to helping the traditional engineering branches be a part of the ongoing Artificial Intelligence revolution.

I would like to thank Dr. Yanbiao Feng, post-doctoral fellow at the University of Victoria, for his valuable assistance in testing my new methods through driver adaptive optimal energy management research, and on advanced optimization and simulation tasks. I would also like to thank my friends in the UVic Clean Transportation Research Team for their encouragements and supports.

Financial support from the Natural Science and Engineering Research Council of Canada, and Dennis & Phyllis Washington Foundation are gratefully acknowledged.

I thank my daughter, Aria Sreejith, for bringing all the joy to our life. Lastly, I would like to express my heartfelt appreciation to my beloved wife, Vasavi Kakkat, for taking care of the family in the best possible way while I was a full-time student. This milestone would not have been possible without her support.

# 1 Introduction

## 1.1 Background

### 1.1.1 Global Warming and $CO_2$ emissions

Global warming is the most significant environmental challenge our planet is facing today. The increased quantity of greenhouse gases causes the earth to warm up, triggering climate change and rising sea level. Glacial ice and air bubbles trapped in it helped scientists build a record of earth's past 800,000 year climates [1]. This data has shown that earth has cycled between ice ages and warm interglacial periods. Some of these interglacial periods were even hotter than today. However, the rate at which our atmosphere is warming up currently is at least 20 times faster than normal. While minor variations in the earth orbit can be attributed to climate change in the past, the current warming trend is triggered by human activities. This is evident from the atmospheric $CO_2$ study on ice cores that revealed the unusually high concentration of $CO_2$ in the earth's atmosphere today [2]. As shown in Figure 1.1, the concentration of $CO_2$ post-industrial revolution in the earth's atmosphere increased exponentially, breaking all-time high levels in 1950 at about 300 ppm. The $CO_2$ levels are estimated to reach 800 ppm in the year 2100 at the current annual emission trends, causing a rise in temperature by $4^0C$. Considering this potentially catastrophic scenario, countries in the world pledged to reduce the greenhouse gas emissions in the 2016 Paris agreement.

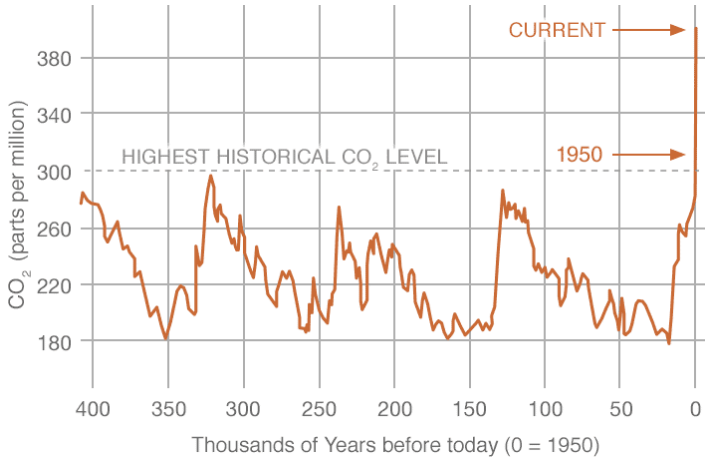



Figure 1. 1 Historical CO2 levels in the earth's atmosphere. Credit: NOAA.

### 1.1.2 Role of the Automotive Industry

In accordance with the Paris agreement, the world's automakers are committed to reducing $CO_2$ emissions. Globally, road transportation accounts for about 16% of synthetic $CO_2$ emissions as shown in Figure 1.2 [3].

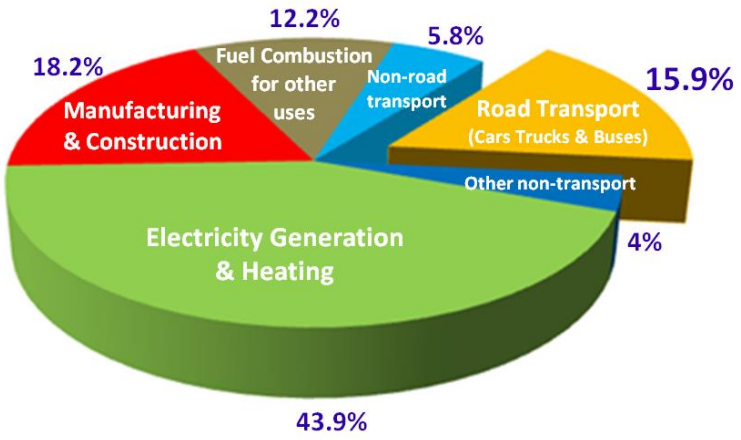



Figure 1. 2 Sources of synthetic CO2 emissions. Credit: OCIA

2018 production statistics from the International Organization of Motor Vehicle Manufacturers shows about 95.6 million cars and commercial vehicles were added in 2018 alone [4]. Automotive industry worldwide is under pressure to cut not only $CO_2$ emissions but also other pollutants such as NO2, particulate matter and fluorinated greenhouse gases from mobile air-conditioning systems. An integrated approach is followed by the automotive industry and legislation to reduce emissions. The approach focuses primarily on improving the performance of new vehicles to reduce emissions. Also, manufacturers invest efforts towards using sustainable fuels such as Hydrogen and Natural Gas instead of fossil fuels. Driver behavior improvement and modernization of road infrastructure are also being implemented to cut down the emissions.

With the number of vehicles on the road exploding each year, reducing the emissions from road transport is vital in mitigating global warming. Original Equipment Manufacturers (OEMs) primarily employ two methods to achieve reduced emissions and higher fuel economy [5]. The first method focuses on the design and structure of the vehicles to reduce losses such as aerodynamic losses, braking losses, and rolling resistance losses. The second

method focuses on powertrain to improve the efficiency of energy conversion. Hybridization of the powertrain is a widely used technology used by the automotive industry to improve efficiency and reduce emissions. It not only improves the efficiency of energy conversion but also reduces energy lost during braking.

### 1.1.3 Overview of Hybrid Electric Vehicles

Hybrid vehicle (HV) is equipped with two or more energy storage devices with associated power converters. In a Hybrid Electric Vehicle (HEV), an Internal Combustion Engine (ICE) and an electric motor are used to propel the vehicle. Each of the energy sources is capable of meeting the energy demand independently or together depending upon the control strategy adopted. The concepts of electric vehicle (EV) and hybrid electric vehicle (HEV) are not new. The first hybrid electric vehicle was around as early as in 1889. But, after the first world war, the advancements in internal combustion engine technology lead to reduced interest in electric and hybrid electric vehicles. However, with increased awareness on emissions and global warming, and also fuelled by the advancements in power electronics to support optimization of hybrid vehicles, the 1990s witnessed a renewed interest in hybrid electric vehicles. Toyota unveiled its first hybrid electric vehicle, Prius in 1998, which opened a new segment of HEV in the commercial automotive industry.

In a conventional vehicle, the energy demand is met with the ICE alone. Hence, the ICE has to be sized to accommodate the occasional high energy demand. This oversizing forces the ICE to operate far from its best efficiency point during the majority of its operations. In an HEV, the energy storage system (ESS) can provide extra power in the event of high power demand. Also, if ICE delivers power more than the demand, the excess energy can be stored in the ESS. Hence, the ICE in an HEV can be downsized and it can be operated always at or near its best efficiency point. This increases the overall powertrain efficiency significantly.

Also, in a conventional vehicle, a large amount of energy is wasted during braking. Hybridization can recover a major portion of this energy by means of the regenerative braking system, which can convert the kinetic energy of the vehicle into electric energy and can store in ESS. This feature enables a hybrid electric vehicle to have much superior

mileage than an ICE vehicle, especially on frequent start-stop driving cycles such as urban driving cycle, transit vehicle driving cycle and delivery vehicle driving cycle.

HEV can be classified into a series hybrid, parallel hybrid and series-parallel hybrid based on the powertrain architecture. In a series hybrid architecture, the power to the transmission is provided only by the electric motor. An internal combustion engine is used to generate electricity from fossil fuel by means of a generator. The generated electric energy is either used to propel the vehicle or stored in the battery. The electric motor can derive power from the generator, the battery or both. In a parallel hybrid configuration, both the ICE and the motor can deliver power to the drivetrain in tandem. The series-parallel hybrid configuration is a combination of properties of both the series and parallel architectures.

A plug-in hybrid electric vehicle (PHEV) is a type of HEV with a much larger battery ESS on board, and the vehicle's ESS can be charged from the external power grid when the vehicle is not in use. Ideally, a PHEV is designed to deplete the energy obtained from the external charge and stored in the ESS between two charges, and a smaller amount of charge is maintained to support the HEV operation of the vehicle. At present, a PHEV normally operates on a sequenced charge depletion (CD) and charge sustaining (CS) modes. During the initial CD mode, the vehicle operates as a pure electric vehicle (PEV) or battery electric vehicle (BEV), using the grid charge filled electric energy in the ESS to propel the vehicle. When the storage energy dropped to a certain threshold, the vehicle switches to its CS mode and operates as a regular HEV, maintaining a certain level of charge in the battery ESS using the surplus power from the engine and through regenerative braking. A PHEV also has a much more powerful electric drive compared to a full HEV. The battery ESS and electric drive of many of these PHEVs are large enough to propel the vehicle in its pure electric mode under all major drive cycles, forming the so-called Extended Range Electric Vehicle (EREV).

### 1.1.4 Power Control and Energy Management Challenges for HEV/PHEV

The powertrain control strategies in HEV are used to control the flow of power from two or more power converters to meet objectives such as vehicle power, speed and acceleration demands, low fuel consumption, reduced emissions, maintaining battery state of charge (SOC), and enhancing driveability. There exist a global minimum for the constrained

optimization problem of minimizing the cost function in an HEV power control problem. The variables to be controlled are normally the operating speed and torque of the ICE and electric motors/generators (M/Gs). In addition, the energy management issue of the ESS needs to be considered. For the PHEV, when and how quickly the grid charge-obtained electric energy in the ESS should be used during the trip between two subsequent charges make the optimal energy management of the vehicle more challenging.

The optimal energy management requires prior knowledge of the driving conditions to obtain the globally optimal performance and operation cost of the HEV/PHEV. While the control and energy management of HEV/PHEV is only optimized using a number of classic statistical driving patterns for the city and highway operations. These driving cycles serve as useful benchmarks to measure the performance and fuel economy of the vehicle globally. However, their use as the foundations for optimal power control and energy management of HEV/PHEV is totally inadequate since no driver follows these driving cycles precisely in their daily vehicle use.

Researches over the past couple of decades resulted in different advanced energy management strategies for HEV. Although optimization techniques such as Dynamic Programming helped to find the global optimum solution, the real-time implementation of energy management in HEV is still a challenging issue and many research have been carried out in this area. The energy management strategies can be classified into rule-based strategy, equivalent consumption minimization strategy, dynamic programming, and optimal control methods, adaptive methods based on driving cycle prediction and adaptive methods based on driving cycle pattern recognition [6]. Lately, the focus is more towards using intelligent machine learning algorithms to get information about future driving conditions. The ability to predict a driver's driving conditions can be used to identify the "customized driving cycles" for each driver's specific trip, to improve the power control and energy management optimization so that the quasi-optimal solution can reach closer to the global optimum. This research is devoted to producing accurate driving pattern prediction for a driver using collected vehicle operation data of a driver, to support the optimal energy management of a PHEV using the predicted driving pattern.

## 1.2 Research Contributions

A novel, intelligent, driver-specific, adaptive energy management approach based on pattern recognition from driving data, driving prototype-based power control and energy optimization, and real-time driving pattern recognition and optimal control has been proposed in this work.

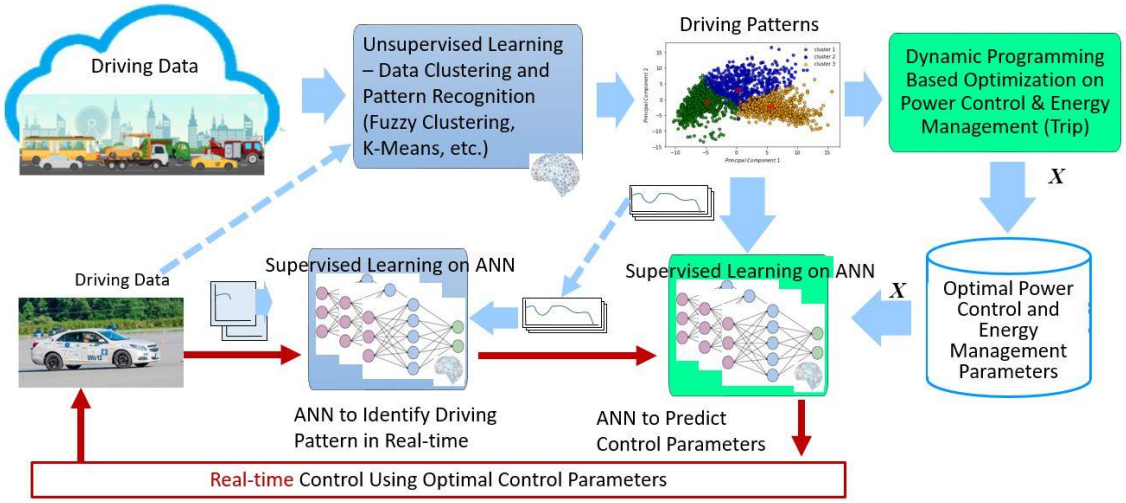



Figure 1. 3 Intelligent powertrain control and energy management strategy

In the first section of the research, a simplified representation technique to represent any given driving cycle with only a set of predefined cycle blocks is developed. A cycle block is a pattern representing a driving event in a 5-second window on a standard driving cycle. 200 such cycle blocks are created from 36 standard driving cycles to form a cycle block library that can be used to represent any driving cycle. In the block representation, the driving cycle is expressed as a sequence of cycle blocks selected from the cycle block library. This simplified representation is later used for efficient pattern recognition and quick real-time driving cycle pattern prediction.

In the second section of the research, the 2010-2012 California Household Travel Survey (CHTS) data was converted into the block representation. Then the unsupervised machine learning algorithm, Fuzzy C-Means clustering (FCM), was performed to identify unique driving patterns in the California state [7]. The CHTS data was then labeled based on the

patterns obtained from the FCM clustering and the labeled data was used to train a sequence of Artificial Neural Networks (ANN) to predict the label of any driving cycle in real-time. Similarly, the FCM clustering algorithm was applied to the driver-specific driving data collected from the city of Victoria, BC. The data was then labeled based on the unique driving patterns and the labeled data was used to train a sequence of ANNs to classify any driving cycle into one of the identified unique driving cycles.

Dynamic Programming (DP) based off-line optimization is performed on the identified representative driving cycles in the Driver-specific driving data to get global optimal control parameters. The library of representative driving cycles and their respective optimal control parameters are used for real-time energy management. Ideally, another ANN is trained with representative driving cycles as the input and respective optimal control parameters as the output to support real-time implementation. However, due to a small number of identified representative patterns, in this research work, a look-up table is used instead of training an ANN for getting the optimal control parameters. During real-time energy management, the trained sequence of ANNs classify the current driving data into one of the identified representative driving cycles and the respective optimal control parameters are selected using a look-up table for energy management. In this work, the introduced data clustering and pattern recognition methods have been tested using the acquired vehicle operation data from California and from Victoria. To put the new method into real tests on driver adaptive intelligent energy management of PHEVs is beyond the scope of this research. To better demonstrate the benefits of the newly introduced methods, simulation results from driver adaptive intelligent energy management are included. This following work has been conducted by Dr. Yanbiao Feng, a post-doctoral fellow at the University of Victoria. The tests were made using the MATLAB/Simulink models of the UVic EcoCAR2 that is a 4WD series-parallel multiple-regime PHEV.

## 1.3 Thesis Outline

Chapter 1 briefly discusses the problems of global warming and the role of the automotive industry in reducing $CO_2$ and other emissions. Then an overview of different strategies employed by the automotive industry to reduce emissions is discussed with an emphasis on hybrid electric vehicles. Then challenges in HEV energy management are briefly

introduced. The chapter ends with a summary of the research contributions and thesis outline.

A review of recent relevant literature is done in Chapter 2. The evolution of energy management strategies in HEV over the past couple of decades is discussed. Chapter 3 describes the vehicle operation data that is used in this research. The first set of data is a set of standard driving cycles that are used to generate the cycle blocks for driving cycle block representation. The second data is the publically available CHTS data. The third data is the driver-specific driving data collected from the city of Victoria, BC. Pattern recognition and real-time pattern prediction methodologies are illustrated on the CHTS data and the Driver-specific driving data.

Chapter 4 describes the methodology used to create the library of cycle blocks from standard driving cycles. In Chapter 5, pattern recognition is performed on the CHTS data and the physical interpretation of the resulted representative driving cycles are discussed. In Chapter 6, a sequence of supervised machine learning models based on ANN is trained to predict the pattern in driving cycles in real-time.

Chapter 7 discusses the pattern recognition and real-time pattern prediction algorithms applied to the Driver-specific driving data collected from Victoria. Software implementation of the research work is discussed in Chapter 8. Feasibility of the intelligent energy management strategy is tested on a series-parallel hybrid electric vehicle in Chapter 9. Theoretical maximum possible improvement in fuel consumption over a rule-based controller is evaluated by using a dynamic programming based off-line controller. The last chapter discusses the conclusion and future works.

# 2 Related Topic Review

## 2.1 Energy Management in PHEV

The design and control optimization of HEV/PHEVs' powertrain systems deals with identifying the best powertrain architecture, determining the optimal powertrain component sizes and introducing an optimal control strategy that finds the best combination of powertrain control parameters during vehicle operation, particularly the output speeds and torques of the ICE and motors/generators. The optimal energy management of a PHEV focuses on the effective utilization of the energy stored in the battery ESS and produced by the ICE to minimize loss and prolong battery life. The controlled battery charge/discharge rates, and depth of charge (DOC)/state of charge (SOC) during a trip determine when and how quickly the grid charge-obtained electric energy in the ESS is used during the trip.

This research work considers the optimization of energy management for a vehicle with fixed powertrain architecture and fixed powertrain components. The research focuses on how the power demand should be distributed between the ICE and the electric motor, depending on the driving conditions, and how the energy stored in the battery ESS is used, so that fuel consumption and emissions are minimized.

Many types of research have been carried out on various control strategies over the past couple of decades. A review of optimal energy management strategies for HEV and PHEV can be found in [6] [8]. The optimal energy management strategies can be classified into rule-based and optimization-based.

### 2.1.1 Rule-Based Energy Management Strategies

Rule-based energy management strategy employs rule tables or flow charts to choose the operating points of the different energy converters. The rules are created based on heuristics, human intelligence or mathematical model. The strategy makes decisions based on instantaneous inputs and prior knowledge of the driving cycle is not necessary. Hence, rule-based energy management can be easily implemented in real-time.

The first documented intelligent controller for HEV was contributed by Baumann et al. (2000) [5]. The authors covered the two major challenges in HEV design; determining the

optimum size of electric motor and ICE, and determining the operation strategy. In this article, the authors introduced the concept of degree of hybridization (DOH), which helps to decide what control strategy to use on which energy converter. If the hybrid architecture is ICE dominated, the control strategy is focused more on improving efficiency or fuel consumption on the ICE rather than on the electric motor since the latter will usually be downsized being the non-dominant energy converter. Authors considered a case study with a parallel HEV with a DOH of 0.48, ICE dominant. The vehicle operation strategy directed at forcing the ICE to act at or near its best efficiency point or least fuel consumption point. The non-dominant energy converter, electric motor, either absorbs the excess power generated by the ICE or provides the extra power at peak load. The control strategy was implemented using a fuzzy logic controller (FLC). The input to the controller is acceleration demand from the driver, desired torque from the non-dominant energy converter and state of charge (SOC) of the battery. Depending on the input and the two modes of operation; high efficiency and low fuel consumption, the output from the controller give how much power from each energy converter is required. If the energy demand from the electric motor is negative, it functions as a generator and stores the excess energy. Based on the different possible scenarios and expert's knowledge on the system, a set of 847 rules were defined. For each input to the FLC, membership values are assigned which defines the probability of each rule in the rule base that applies to the current input state. A center of gravity method is used to determine the final output from these membership values. Although this approach improved efficiency as well as fuel economy, it required expert knowledge to create the rule base and no optimality was considered. The rule-based control strategies can be further classified into deterministic rule-based and fuzzy rule-based energy management strategies [8].

### 2.1.2 Global Optimization-Based Energy Management Strategies

Constrained optimization techniques are considered a better alternative to rule-based control strategies in HEV design. In 2000, a research team from Ohio State University proposed a constrained optimization approach for energy management in HEV using Dynamic Programming (DP) technique [9]. In [10] R. Wang et al. (2012) discussed the procedures for implementing Dynamic Programming (DP) technique to find the globally

optimal use of ICE and EM over a pre-defined driving cycle. In the DP approach, a cost function is defined which is usually the sum of fuel consumption and emission. It is a function of state variables and control variables of the system. The objective of the optimization problem is to minimize the cost function with respect to control variables such as torque output from the electric motor, torque output from the ICE and gear ratio, subjected to constraints such as meeting power demand from the driver, maximum SOC, minimum SOC, speed of motor and speed of engine. As per Bellman's principle of optimality, a recursive approach can be used to solve such optimization problems [10] [11] [12]. Being a global optimization algorithm, DP ensures the convergence to the global optimum. However, real-time implementation of the DP technique on HEV design is not feasible directly as prior knowledge on the driving cycle is required. But it can be used as a benchmark to test the performance of real-time control strategies. Also, the results from DP optimization can be used to improve the rule base in a rule-based control strategy so that the performance of rule-based control strategies achieve performance near global optimum. In [11] techniques such as Stochastic Dynamic Programming (SDP), Power Split Ratio (PSR) based control and Supervised Machine Learning methods are discussed. All these algorithms exploit the results obtained from DP optimization for better real-time energy management.

### 2.1.3 Equivalent Consumption Minimization Strategy

In 1990, Paganelli G introduced Equivalent Consumption Minimization Strategy (ECMS) [13] [14]. He proposed an instantaneous optimization problem for energy management in HEV instead of the global optimization problem in the DP approach. ECMS introduced a cost function that is dependent only on the current system variables. For a charge sustaining HEV, the cost function is the sum of instantaneous fuel consumption and equivalent fuel consumption related to SOC variation. Equivalent fuel consumption is computed using equivalence factors, which are dependent on the driving conditions. With the slightly sub-optimal solution and optimization based only on the current system variables, this method has the potential for real-time EMS controls. However, the challenge is, finding the optimum equivalence factors at the current driving cycle. ECMS is a rigid system and a slight deviation from the optimal values of the equivalence factors will result in an

unacceptable solution. Hence, without prior knowledge of the driving conditions, real-time implementation of ECMS is not practical.

### 2.1.4 Adaptive ECMS Using Driving Cycle Prediction

Two approaches to adapt ECMS for real-time energy management are discussed in [15]. The first approach makes use of driving cycle prediction techniques to find the optimal value of equivalence factors. In the second approach, pattern recognition techniques are employed to detect the current driving conditions and this information is used to feed the ECMS with better equivalent energy consumption. In [14], an adaptive ECMS technique, which is based on driving cycle prediction, is discussed. It uses an algorithm that combines past driving data and predicted driving data to estimate the current mission. This information is then used to evaluate and feed the optimum equivalence factors to the ECMS. The algorithm is run periodically at a fixed interval and the equivalence factors are updated accordingly.

### 2.1.5 Adaptive ECMS Using Pattern Recognition

Gu and Rizzoni (2006) [16] proposed an adaptive ECMS that uses pattern recognition algorithms to classify the current driving condition into one of the previously identified driving patterns. Authors used three typical driving patterns as a reference and evaluated the optimum equivalence factors for each offline. For real-time implementation, they analyzed a sliding window of past driving records and used the features to classify the past driving data into one of the reference driving patterns. Respective equivalence factors are used in the ECMS for real-time energy management.

A similar approach was extensively used in recent researches for HEV energy management. The basic idea behind this approach is to find the closest representative driving cycle to current driving conditions and use the pre-computed equivalence factors of the representative driving cycle for ECMS of current driving conditions. In [17], authors selected four reference driving cycles that represent city, highway, high acceleration, aggressive and low-speed stop and go driving conditions. Different statistical parameters such as mean speed and acceleration, standard deviations of speed and acceleration and percentage of idle events are used to represent the features of reference driving cycles. The

equivalent factors are computed offline for these reference driving cycles. While real-time implementation, a past driving window of 600 seconds is used for pattern recognition. Features are extracted for the past driving window and in the feature space, Euclidean distance to each of the reference driving patterns are computed. For ECMS, the equivalence factor of the reference driving cycle that is closest to the current driving conditions are used.

### 2.1.6 Model Predictive Control (MPC)

A Model Predictive Control strategy for energy management in HEV is discussed in [18]. It uses historical driving data to predict future driving conditions over a prediction horizon and DP is used to solve the optimization problem within the prediction horizon. Authors used a multi-step Markov-based predictive method to estimate the velocity and acceleration in the prediction horizon. From current velocity and current acceleration in each of the sample driving cycles, acceleration of the next fixed number of steps are recorded. The probability distribution of the acceleration is estimated at each step. Using the current velocity and current acceleration, and the estimated probability distribution, the velocity, and acceleration in the prediction horizon are predicted. Then a constrained optimization problem is formulated in the prediction horizon for the energy management in HEV. DP is used to solve the optimization problem and the resulting control parameters are used for energy management.

## 2.2 Pattern Recognition

Different clustering algorithms for grouping data based on similarity are described in [19]. Such algorithms can be classified into hierarchical clustering and partitional clustering. Hierarchical clustering organizes the data into a nested sequence of groups. Partitional clustering creates a single partition of the data. Hierarchical clustering can be considered as a sequence of partitional clustering. K-Means clustering algorithm and Fuzzy-C Means (FCM) clustering algorithms are the two most popular partitional clustering algorithms. In the K-Means clustering algorithm, as explained in [19], the given data is clustered into K clusters such that the squared error of the samples within the same clusters is minimized. In [20], the authors used the K-Means clustering algorithm to generate standard driving

cycles for the city of Tehran. They grouped driving cycles into 4 clusters and extracted representative driving cycles from each cluster. These representative driving cycles were then used to generate a standard driving cycle.

A modified version of the K-Means clustering algorithm, called the Fuzzy C-Means algorithm, was first proposed by Dunn [21] in 1973. It is based on fuzzy logic, where the members in a cluster can belong to more than one cluster with an associated degree of membership. In [22], authors compared FCM and K-Means algorithm and concluded that when well-separated cluster structures exist in the data, the K-Means algorithm is preferred for its efficient computations. However, the FCM algorithm is found suitable for handling overlapping clusters.

# 3 Collection of Vehicle Operation Data

Three different datasets were used in this research work to build and test the machine learning models to support the intelligent powertrain control and energy management strategy. The first dataset contains 24 standard driving cycles to support data abstraction. These driving cycles are a statistical representation of all possible driving behavior from which a library of driving patterns can be extracted. Driving cycles are represented with these driving patterns to get a simplified block representation which is easier for the machine learning algorithms to process. The second dataset, openly available California Household Travel Survey dataset, is a large, real, raw driving data. It is used to implement and test the machine learning algorithms on large scale dataset collected from different drivers. The third dataset is a collection of 1500 personal driving cycles that are collected from a single driver in the city of Victoria, Canada. This driver-specific data is needed to demonstrate the benefits of the driver adaptive powertrain control and energy management which is based on individual driving behavior. Although the datasets used in this research work contains driving cycles from domestic vehicles including passenger cars, the approach can be used for the real-time powertrain control and energy management of any class of HEV/ PHEVs.

## 3.1 Standard Driving Cycles

Performance of a vehicle such as fuel consumption and emission is estimated from dynamometer while vehicle follows a driving cycle, which is a pre-defined set of points representing speed versus time. Duration of the driving cycle used for testing is often limited to reduce the operation cost of the test. Hence, it is important to represent the actual driving conditions in a small test driving cycle to get a better estimate of the vehicle performance. Driving cycle construction includes steps such as collecting real-world driving data, segmenting the driving data, constructing cycles, and evaluating and selecting the final cycle. Different cycle construction methodologies for light-duty vehicles are summarized in [23]. These are micro-trip based cycle construction, trip segment based cycle construction, cycle construction based on pattern recognition and modal cycle construction.

The first part of this research work develops a simplified block representation of driving cycles. It uses patterns in 5 seconds long segments in the driving cycles, called cycle blocks, to represent the cycle in block form. Each cycle block is a pattern representing a driving event in the 5-second window. Driving cycles are considered as a sequence of cycle blocks. An infinite number of cycle blocks are required to represent all possible driving cycles. However, in the simplified block representation proposed here, a set of 200 cycle blocks are selected such that any driving cycle can be represented as a sequence of these 200 cycle blocks with acceptable accuracy. The details of block representation are discussed in Chapter 4.

To build a library of 200 cycle blocks, 24 standard driving cycles from ADVISOR, an advanced vehicle simulator developed by the National Renewable Energy Laboratories (NREL) in 1994 [24], and 12 driving cycles from the CHTS data are used. 24 standard driving cycles, listed in Table 3.1, are handpicked from the standard driving cycles available in ADVISOR. These driving cycles are a statistical representation of all possible driving behavior. Driving cycles that do not represent real-life driving conditions are not included. Randomly selected 4 driving cycles from the 24 standard driving cycles are shown in Figure 3.1. The 12 driving cycles from the CHTS data are chosen by performing an approximate clustering on the whole dataset and choosing the driving cycles closest to the resulted cluster centers. The library of 36 driving cycles are used to generate 7652 cycle blocks and a subset of 200 cycle blocks are selected for the library of cycle blocks.

Table 3. 1 Standard driving cycles selected from ADVISOR

| Sl. No. | Driving Cycle Name | Sl. No. | Driving Cycle Name |
|---|---|---|---|
| 1 | CYC_ARB02 | 13 | CYC_UDDS |
| 2 | CYC_SC03 | 14 | CYC_Viking |
| 3 | CYC_REP05 | 15 | CYC_OCRef |
| 4 | CYC_NYCCOMP | 16 | CYC_Highway |
| 5 | CYC_LA92 | 17 | CYC_HHDDT65 |
| 6 | CYC_INRETS | 18 | CYC_Cruise3 |
| 7 | CYC_INDIA_HWY_SAMPLE | 19 | CYC_WVUSUB |
| 8 | CYC_IM240 | 20 | CYC_WVUINTER |
| 9 | CYC_HWFET | 21 | CYC_WVUCITY |
| 10 | CYC_HL07 | 22 | CYC_US06_HWY |
| 11 | CYC_CSHVR_Driver | 23 | CYC_US06 |
| 12 | CYC_COMMUTER | 24 | CYC_UDDSHDV |

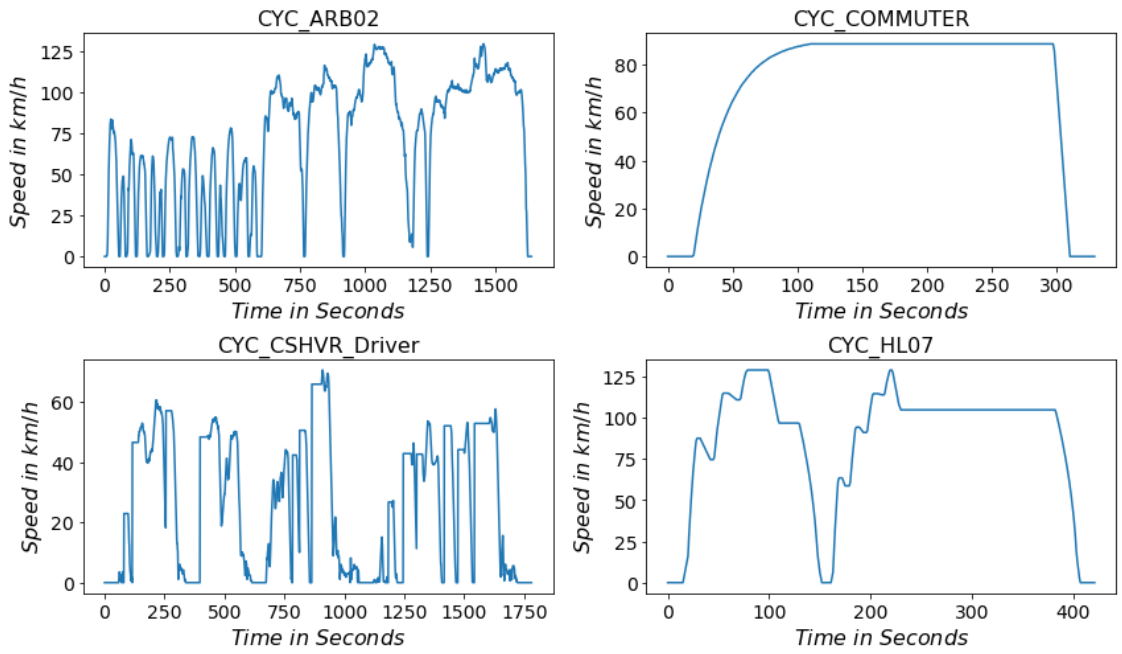



Figure 3. 1 Randomly selected 4 standard driving cycles from ADVISOR

## 3.2 Driving Data from California

### 3.2.1 Data Source

The block representation technique, unsupervised machine learning for identifying unique driving patterns and supervised machine learning for predicting the patterns are demonstrated on driving cycle data available from the 2010-2012 California Household Travel Survey (CHTS) [7]. The survey was conducted by the California Department of Transpiration (Caltrans). Travel information was collected from all of California's 58 counties and portions of three adjacent counties from Nevada. Beginning in January 2012, Computer Assisted Telephone Interviewing (CATI), online, global positioning systems (GPS), and on-board diagnostic board (OBD) were used for daily driving data collection that lasted for a whole year.

Researchers at the National Renewable Energy Laboratory (NREL) processed a sample of driving data from CHTS and created second-by-second vehicle speed profiles. Data was collected from light, medium and heavy-duty vehicles. Erroneous data in the raw GPS data was filtered by NREL using a GPS data filtration routine. Daily travel tables in CHTS,

which contains second-by-second speed and acceleration profiles followed by a vehicle in a whole day, is used in this research.

### 3.2.2 Data Pre-processing

The raw data provided in CHTS made no distinction between key-on idling events and key-off parked periods. In this study, an idle event is assumed as a key-off parking period if it lasts for more than 300 seconds. Based on this assumption, daily driving data was split between parking periods to create multiple micro-trips. Also, zero speed points are largely excluded in the original data to save space. This resulted in idling events represented as a single point in the driving cycle. In the pre-processing stage, these points were added to create a complete driving cycle.

On dividing the daily driving data based on parking periods, a total of 65653 driving cycles were generated. A sample of 20,000 driving cycles was drawn and explored for total trip length. A box plot summarizing the trip length is shown in Figure 3.2.

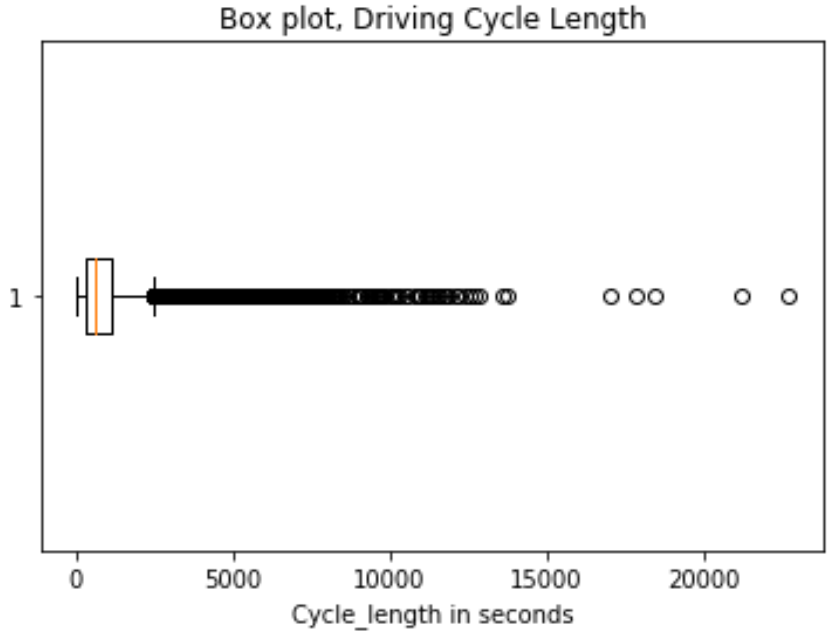



Figure 3. 2 Boxplot before removing the outliers

As seen in the box plot, the trip length is widely spread from a minimum of 6 seconds to a maximum of 22,000 seconds. The circles in the box plot represent outliers in the data. In the pre-processing stage, these outliers are removed from the data. All data points that are

less than the lower whisker in the box plot, $(Q1 - 1.5 * IQR)$ or that is higher than the upper whisker, $(Q3 + 1.5 * IQR)$, are considered as outliers where,

$Q1$ is the first quartile of the distribution,

$Q3$ is the third quartile of the distribution,

and $IQR$ is the Inter Quartile Range, $Q3 - Q1$.

After removing the outliers from the data, the resulting driving cycles have a trip length ranging from 6 seconds to 1600 seconds. A box plot showing the distribution of the trip length after removing the outliers is shown in Figure 3.3.

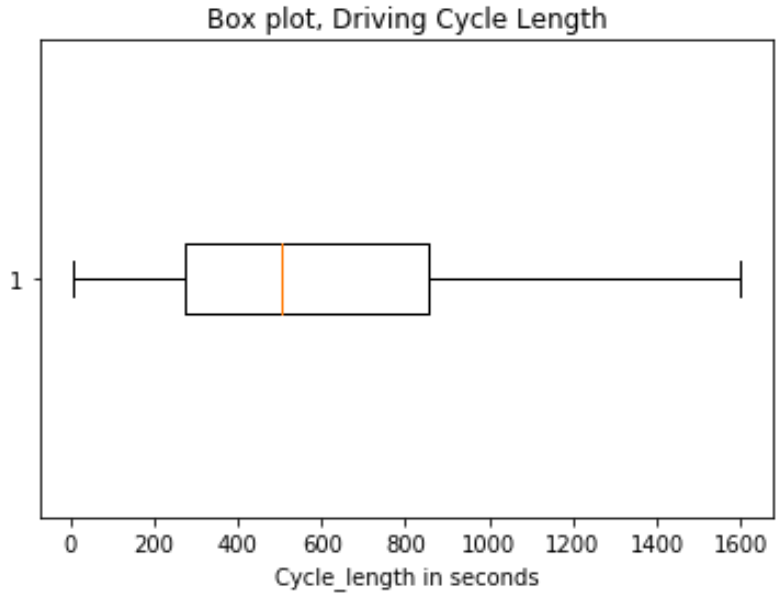



Figure 3. 3 Boxplot after removing outliers

A combination of hierarchical and partitional clustering was performed on the CHTS data. The dataset was firstly divided into 3 based on the trip length. FCM clustering algorithm was performed on each dataset independently. First data set, short trip dataset, contained all driving cycles with the trip length between 300 seconds and 500 seconds. Samples with trip length less than 300 seconds were omitted from the study due to lack of significant patterns. The second data set, medium trip dataset, included all driving cycles with trip length greater than or equal to 500 seconds, but less than 1000 seconds. The last dataset, long trip dataset, contained all driving cycles with trip length greater than 1000 seconds.

This partition was done in order to enable the clustering algorithm to distinguish between driving cycles with similar patterns but with significantly different trip length. Energy management strategies in HEV are dependent on trip lengths and similar driving cycles with different total trip length may have different optimal control strategies. Clustering algorithms were performed independently on these data sets and the driving cycles corresponding to the cluster centers were selected as the representative driving cycles, representing the driving conditions and driver behaviors in the state of California.

### 3.3 Collected Driver-specific Driving Data

The advantages of the proposed optimal energy management strategy are demonstrated on a driver-specific driving data. Data were collected from a single passenger car for trips in and around Victoria, British Columbia. GPS based mobile application was used for the data collection. The initial data consisted of speed data in km/hr for every 3 seconds from 30 trips. Missing values were filled with an average of the adjacent available speed data. The resulting driving cycles were smoothed to limit the maximum acceleration/ deceleration to 5 km/hr/second. Since more data is required to train machine learning models, data augmentation was performed to artificially synthesis more data from these 30 driving cycles. 50 driving cycles were generated from each of the 30 driving cycles by introducing random noises. Each speed data point is iteratively selected and modified depending on its value. If the original speed is less than 15 km/hr, a random value is selected from the range of -30 to 30 and added to the original speed. If the resulting value is negative, it is capped at 0 km/hr. If the speed is greater than 15 km/hr, a random value is selected from the range of -10 to 10 and added. This resulted in a dataset with 1500 driving cycles that represents a driver-specific driving data from the city of Victoria. Randomly selected 2 driving cycles from the Driver-specific driving data are shown in Figure 3.4.

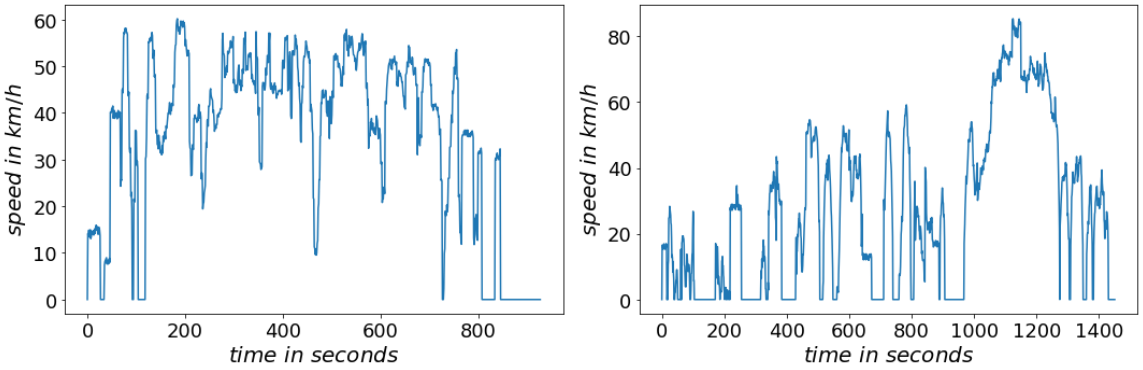



Figure 3. 4 Randomly selected 2 driving cycles from Driver-specific driving data

# 4 Data Abstraction

This chapter introduces a method to better represent the collected driving data to support driving pattern recognition. The abstraction of the driving data should truly represent the driving speed variations in a more compact form that is easy for computer processing and analyses using an intelligent system. The abstraction describes the definition of a cycle block, how cycle blocks are extracted from standard driving cycles, how a subset of such cycle blocks are selected to form the library of cycle blocks that can be used to represent any driving cycle in the block representation. Evaluation of how well the block representation represents the original driving cycle is also discussed.

## 4.1 Cycle Block Extraction

A cycle block is defined as a pattern in the driving cycle that lasts for 5 seconds. It represents a driving event on a 5-second window. Cycle blocks with different time span were tried and evaluated iteratively and found that the quality of block representation in preserving the details in the original driving cycle is at its maximum at a time window of 5 seconds. A cycle block library is a collection of 200 cycle blocks selected from 36 standard driving cycles. To illustrate the cycle block and the library of cycle blocks, a randomly selected driving cycle from the 36 standard driving cycles is shown in Figure 4.1.

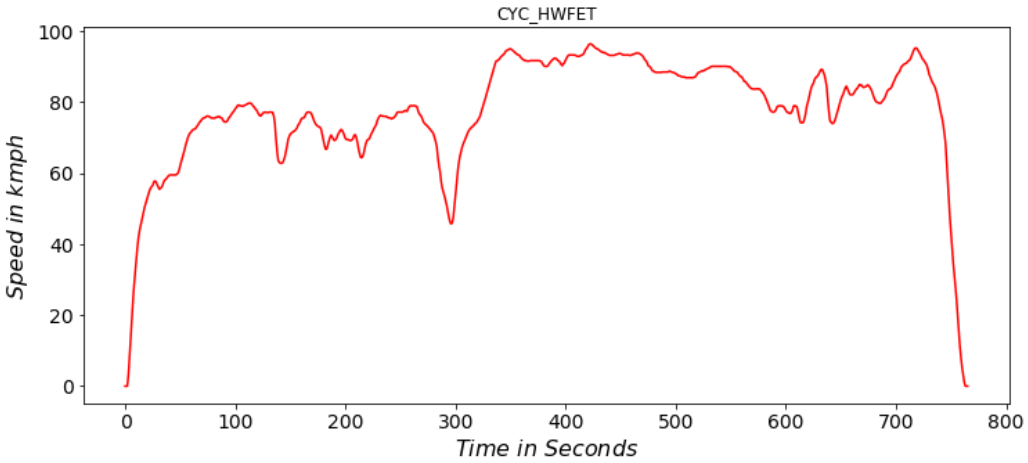



Figure 4. 1 Random driving cycle from ADVISOR

If the driving cycle is divided into segments of 5 seconds, each segment represents a cycle block. So any driving cycle can be considered as a sequence of cycle blocks. 20 such cycle blocks that make the first 100 seconds of the above driving cycle are shown in Figure 4.2.

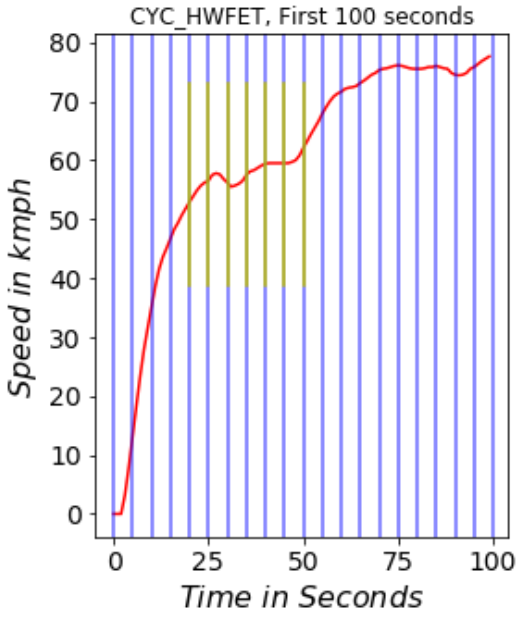



Figure 4. 2 Cycle blocks constituting the first 100 seconds of the driving cycle shown in Figure 4.1

If cycle blocks are named as *"Block1", "Block2"* etc., driving cycles can be represented as a series of cycle block names. For example, a portion of the above driving cycle from 20$^{th}$ second to the 49$^{th}$ second is highlighted in Figure 4.2. This portion can be represented as 5 cycle blocks joined together along the time axis, as shown in Figure 4.3. If those cycle blocks are named as *"Block1"* through *"Block5",* the highlighted portion of the above driving cycle can be represented in the block representation as,

Block1 – Block2 – Block3 – Block4 – Block5

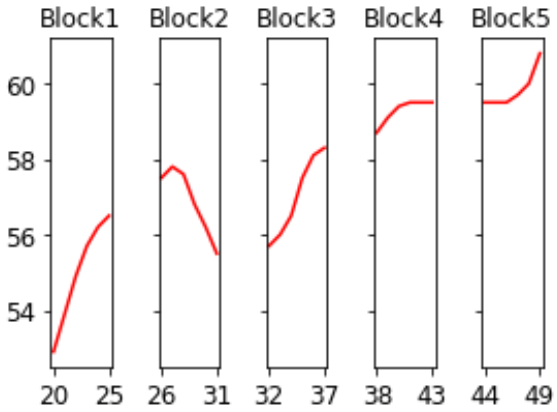



Figure 4. 3 Driving cycle represented as a sequence of cycle blocks

Since an infinite number of cycle blocks are possible, it is not practical to name every cycle block for the block representation. To overcome this, a subset of cycle blocks are selected. Similar cycle blocks are grouped together and a representative cycle block is selected from each of the group. One cycle block represents all the cycle blocks having similar patterns in the 5-second window. Any given cycle block can be associated with one of the cycle blocks in the cycle block library. In block representation, any given driving cycle is firstly divided into segments of 5 seconds duration. Then each segment is replaced with the respective cycle block from the cycle block library to get the block representation. In the block representation, only the string representing the block name is used. Hence, any driving cycle can be represented as a sequence of strings.

As mentioned in Chapter 3, a set of 36 standard driving cycles were used to create cycle block library. These 36 driving cycles cover many of the driving events possible. Firstly, 36 standard driving cycles were divided into segments of 5 seconds to get 7652 cycle blocks. All of these 7652 cycle blocks could be included in the cycle block library. However, a higher number of cycle blocks would increase the computations involved in converting a driving cycle to its block representation. Also, many cycle blocks in the 7652 cycle blocks are similar or differ only slightly. So these 7652 cycle blocks were grouped in different clusters based on the similarity in their patterns and one representative cycle block was selected from each cluster and included in the cycle block library.

Unsupervised machine learning algorithm was used to group the 7652 cycle blocks based on similarity in their patterns. Prior to performing machine learning algorithms, features must be extracted from each of the cycle blocks, in a process called feature extraction. A feature quantifies the pattern in the cycle block it is extracted from. Machine learning algorithm uses these features to compare different cycle blocks and group similar cycle blocks in the same cluster.

For the feature extraction, a quadratic polynomial is fitted on each pattern in the 5-second window in a cycle block. This gives three polynomial coefficients, $A0, A1$ and $A2$ that forms the features. Here, the first coefficient, $A0$, represents the initial velocity in the 5 seconds window of the cycle block. $A1$ represents the initial acceleration and $A2$ represents half of the initial rate of change of acceleration. For the *"Block2"* in Figure 4.3, the coefficients obtained by fitting a quadratic polynomial are given below.

$$A0 = 57.55714286$$

$$A1 = 0.32571429$$

$$A2 = -0.1142857$$

*"Block2"* can now be mathematically expressed as:

$$time = [0, 1, 2, 3, 4] \quad (4.1)$$

$$speed = A0 + A1x + A2x^2 \quad (4.2)$$

A plot of *"Block2"* and *"Block2"* generated using the polynomial expression are shown in Figure 4.4.

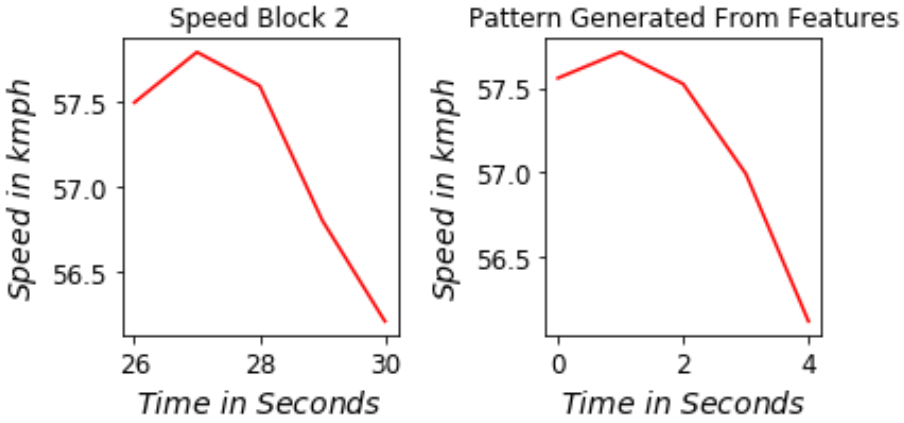



Figure 4. 4 Cycle block and pattern generated from polynomial coefficients

Features were extracted from each of the 7652 cycle blocks to get a data matrix as shown in table 4.1. Each row represents a cycle block and the columns represent the polynomial coefficients. *"Block2"* from the previous example is entered in the first row for illustration. Clustering algorithms can now be applied to this data matrix to group the cycle blocks based on similarity in the polynomial coefficients.

Table 4. 1 Pattern represented with quadratic polynomial coefficients.

| **Sl. No** | **A0** | **A1** | **A2** |
|---|---|---|---|
| 1 | 57.557 | 0.3257 | −0.1142 |
| 2 | | | |
| 3 | | | |
| . . . | . . . | . . . | . . . |
| 7652 | | | |

## 4.2 Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that helps to train computers to learn from data. With the recent advancements in deep learning, computers can be trained to perform advanced tasks such as image classification and segmentation, and Natural Language Processing (NLP). This enabled the implementation of ML in a variety of applications such as autonomous driving and language translation.

The process of making computers learn from data using ML algorithms is known as training. Based on the training method, ML algorithms can be classified as supervised and unsupervised machine learning. In a supervised machine learning, data is given to the model with the respective output label. The purpose of training is to find a mathematical function that can map given input to respective output so that it can be used to predict output with high accuracy for data that was not used to train the ML model. Training a supervised ML model is an optimization process in which the model initially assumes a random mathematical function that represents the mapping between inputs and outputs. A prediction is made using the current mathematical function on all or some of the training examples. An objective function is then defined that quantifies the error made by the model. In the optimization process, the ML model is allowed to predict using the training

data iteratively. In each iteration, the parameters of the mapping function are optimized to minimize the objective function, training loss. When the solution converges, the initially assumed mathematical function would represent very close to the ground truth that maps the input to the output. The performance of a trained ML model is evaluated using its prediction on test data, which is a data that is not used for training and the output is known. A schematic diagram representing supervised ML is shown in Figure 4.5. There are many supervised machine learning algorithms available such as Linear Regression, Logistic Regression, Support Vector Machines, and Artificial Neural Networks (ANN).

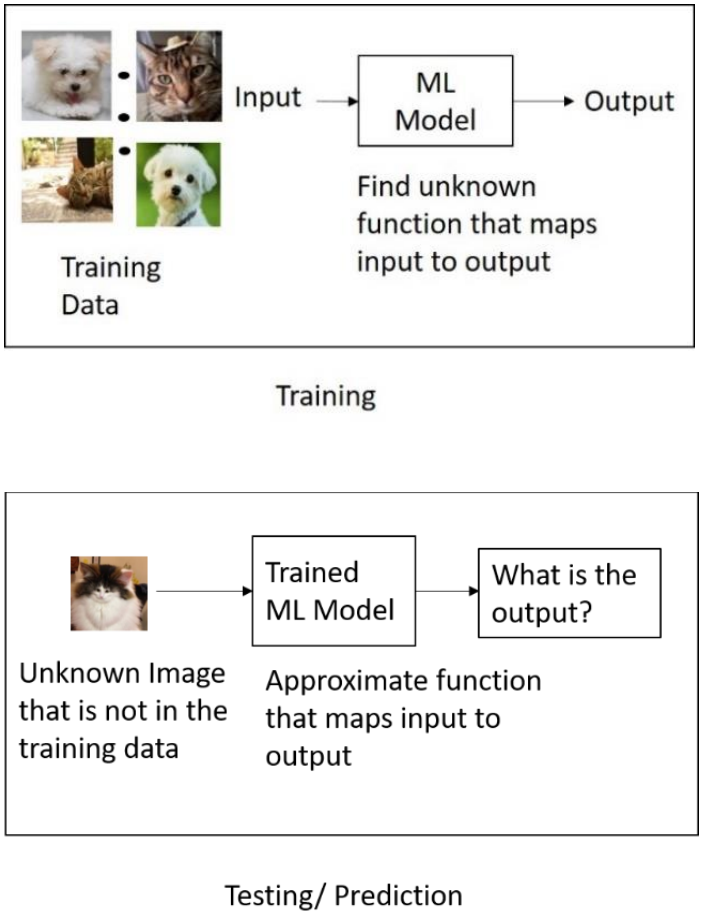



Figure 4. 5 Supervised Machine Learning

In an unsupervised machine learning, also called clustering, only training data is given to the ML model. The output of the data is not known. Clustering algorithms learn the data, find similarities and group data into different clusters such that samples within a single cluster are as similar as possible while the samples in different clusters as different as

possible. The objective function in unsupervised learning is often the sum of squared distance between all samples within the same cluster. The optimization aims at minimizing the sum of the squared error across all clusters. Examples of Unsupervised machine learning algorithms are K-Means algorithm and Fuzzy C-Means (FCM) algorithm.

### 4.3 Fuzzy C-Means Clustering (FCM) Algorithm

Fuzzy logic was developed by Prof. Lofti A. Zadeh at the University of California at Berkeley, in the 1960s, to handle the concept of partial truth [25]. As per binary logic in traditional computing models, a statement is either completely true or completely false. But a fuzzy logic can model statements that are partially true. It is a part of the field of soft computing, which uses inexact but useful solutions to computationally hard tasks. Similar to the human mind, soft computing can handle partial truths, uncertainty, imprecision, and approximation.

FCM is a clustering algorithm based on fuzzy logic, first proposed by Dunn [21] in 1973. It is a modified version of the K-Means clustering algorithm. In K-Means, samples in the data are forced to belong to exactly one of the many clusters. Hence it is a hard clustering technique. However, in FCM clustering, samples in the data can belong to more than one cluster with an associated degree of membership. Data points at the boundary are not forced to belong to one cluster. Hence FCM clustering is considered as a soft clustering technique. The fuzzy logic used in FCM clustering empowers it to handle uncertainties associated with grouping data. If a data point lies at the boundary of the clusters, it may have equal membership values to belong to some or all of the clusters. Such data can give more insights into the underlying structure of the dataset and they may be grouped together as separate clusters.

Most of the real-world data are not well separated. Hence it is important to handle overlapping clusters when partitioning such data. Even though the K-Means algorithm is slightly more computationally efficient than FCM clustering, the ability of the latter to handle overlapping clusters made it a suitable algorithm for clustering [22]. Also, the flexibility to set a threshold on the membership value enables to extract a crisp set from the fuzzy partition with samples within the clusters having very high similarity. Moreover, the fuzzy partition coefficient serves as a measure to quantify the quality of resulting clusters.

This can be used to find the optimal number of clusters for a dataset. So the FCM algorithm was used in all the clustering tasks in this research work. Fuzzy C-Means algorithm proposed in [26] is described in detail below.

In set theory, an ordinary set or a crisp set is a set where an element is either a member of it or not. More formally, a set A is said to be a crisp set when,

$For\ all\ x,$

$I_A(X) = 0\ if\ x\ does\ not\ belongs\ to\ A$ (4.3)

$I_A(X) = 1\ if\ x\ belongs\ to\ A$ (4.4)

$where, I_A(X)\ is\ the\ indicator\ function\ of\ A$

Here, the indicator function of x with respect to A indicates if x belongs to A. Ordinary set is based on binary logic and hence the indicator function can only take binary values. In other words, any point x either belongs to A or does not belongs to A. Unlike a crisp set, a fuzzy set allows an element to belongs to more than one set simultaneously with an associated degree of membership. A set is said to be a fuzzy set if,

$There\ exists\ at\ least\ one\ x\ such\ that,$

$0 \leq \mu_A(x) \leq 1$ (4.5)

$where, \mu_A(x)\ is\ the\ membership\ function\ of\ A$

Fuzzy C – partition can be defined based on fuzzy set theory. Consider a set,

$S = \{ x_1, x_2, x_3 \dots \dots x_N\}$ (4.6)

Fuzzy C – partition of $S$ into $C$ clusters, represented by $(U, S)$ where,

$U = cluster\ membership\ matrix = ((u_{i,j}))_{NxC}$ (4.7)

where,

$u_{i,j}$ = membership value of $i^{th}$ sample to the $j^{th}$ fuzzy set,

$1 \leq i \leq N$

$1 \leq j \leq C$

Satisfying the following properties:

i. $0 \leq u_{i,j} \leq 1 \qquad \forall\ i, j$ (4.8)

ii. $\sum_{j=1}^{C} u_{i,j} = 1 \qquad \forall\ i = 1, 2, 3, \dots \dots N$ (4.9)

iii. $0 \leq \sum_{i=1}^{n} u_{i,j} \leq n \qquad \forall\ j = 1, 2, 3, \dots \dots C$ (4.10)

The fuzzy partition can be formulated as an optimization problem with an objective to minimize generalized least square error, as given below.

$$minimize\ J_m(U,S,A)\ with\ respect\ to\ U\ for\ a\ given\ S,A\ and\ m\ where,$$

$$J_m(U,S,A) = \sum_{i=1}^{N}\sum_{j=1}^{C}(u_{i,j})^m\,(d_{i,j})^2 \tag{4.11}$$

$$(d_{i,j})^2 = squared\ distance\ of\ x_i\ from\ v_j = (x_i - v_j)^T A((x_i - v_j)) \tag{4.12}$$

$$m = weighting\ exponent;\quad 1 < m < \infty$$

$$v_j = weighted\ mean\ of\ j^{th}\ cluster = \frac{\sum_{i=1}^{N} x_i(u_{i,j})^m}{\sum_{i=1}^{N}(u_{i,j})^m}, \tag{4.13}$$

$$j = 1,2,3\ldots.C$$

$$S = given\ data\ set$$

$$A = positive\ definite\ (nxn)\ weight\ matrix$$

The weighting exponent, $m$, controls the relative weight on each of the squared errors, $(d_{i,j})^2$. The value of $m$ is found experimentally. For most of the data, $1.5 \le m \le 3$ gives good results [26]. If the positive definite weight matrix is chosen as the identity matrix, the squared distance in the feature dimensional space is Euclidean norm.

FCM Algorithm

i. *Given dataset $S$, positive definite matrix$A$, weighting exponent $m$, and the number of clusters, $C$.*
ii. *Choose an initial cluster membership matrix $((u_{i,j}))_{NxC}$.*
iii. *Compute $v_j$ for $j = 1,2,3\ldots.C$.*
iv. *Calculate new $u_{i,j}$ as per below equation.*

$$(u_{i,j}) = \frac{1}{\left[\sum_{k=1}^{C}\left[\frac{(d_{i,j})^2}{(d_{i,k})^2}\right]^{2/(m-1)}\right]} \qquad 1 \le i \le N,\ 1 \le j \le C \tag{4.14}$$

v. *Check the convergence criteria,*

$$\left\| u_{i,j} - u_{i-1,j-1} \right\|_{norm\,1} < \varepsilon \tag{4.15}$$

*Go to (iii) if convergence criteria are not met.*

## 4.4 Cycle Block Selection Using FCM Clustering

FCM clustering algorithm is used in this study to select a subset of cycle blocks from a superset of 7652 cycle blocks extracted from 36 standard driving cycles. Cycle blocks are selected such that the number of cycle blocks in the subset is minimized while the metric used to evaluate how well the block representation represents the original driving cycle, is maximized.

The hypothesis behind cycle block selection using a clustering algorithm is that, of the 7652 cycle blocks, many are similar. If similar cycle blocks are grouped together to form clusters, the center of the cluster can be used to replace any of the cycle blocks in the same cluster. Hence, reasonably good representation of any driving cycle can be achieved with a much smaller subset of the 7652 cycle blocks. The optimum number of cycle blocks to be selected is not known. If all 7652 cycle blocks are selected, that would result in redundant cycle blocks included in the cycle block library and if fewer cycle blocks are selected, the block representation would not accurately represent the original driving cycle. How the FCM clustering algorithm is used to create the cycle block library is demonstrated in the following section. For the purpose of demonstration, only 9 cycle blocks are selected and included in the cycle block library from 7652 cycle blocks. Expecting only 9 cycle block to represent any given driving cycle is highly optimistic. In the final cycle block library, 200 cycle blocks are included. An iterative approach used to tune the number of cycle blocks in the cycle block library to 200 is discussed in the final section of this chapter.

### 4.4.1 Feature Extraction

In feature extraction, each of the 36 standard driving cycles is divided into 7652 segments of 5 seconds duration or cycle blocks. Data analysis tool for the Python programming language, Pandas [27] is used for data pre-processing. As discussed in section 4.1 in this chapter, each segment is fitted with a quadratic polynomial and polynomial coefficients are used as the features. Numpy [28] [29], the fundamental package for scientific

computing with Python, is used extensively in this research work, including for fitting the quadratic polynomial on the cycle block. Data matrix with extracted features from the 7652 cycle blocks shown in table 4.2.

Table 4. 2 Extracted features from the 7652 cycle blocks

| | A0 | A1 | A2 |
|---|---|---|---|
| **0** | 0.000000 | 0.000000 | 0.000000 |
| **1** | 0.048571 | 0.042857 | 0.014286 |
| **2** | 1.368571 | 1.622857 | 1.614286 |
| **...** | ... | ... | ... |
| **7649** | 4.631429 | -0.582857 | 0.085714 |
| **7650** | 3.985714 | 0.358571 | -0.007143 |
| **7651** | 5.374286 | -0.158571 | -0.292857 |

7652 rows × 3 columns

### 4.4.2 Feature Scaling

Feature scaling is an important pre-processing step in any machine learning pipeline. When different features in the data set have different dimensional range, many of the machine learning algorithms, including FCM clustering, will not work correctly. This is because of the fact that ML algorithms compute the distance between different samples in the feature space. If the distribution of one feature has a very high range compared to other, the distance calculation will be highly influenced by the feature with a high range. For example, a summary of the three features in the cycle block data matrix is given in table 4.3.

Table 4. 3 Summary of the dataset before feature scaling

| | A0 | A1 | A2 |
|---|---|---|---|
| count | 7652.000000 | 7652.000000 | 7652.000000 |
| mean | 47.927433 | 0.016824 | -0.002905 |
| std | 38.214073 | 2.367633 | 0.373062 |
| min | -7.542857 | -24.440001 | -7.428571 |
| 25% | 9.173572 | -0.454643 | -0.085714 |
| 50% | 45.374287 | 0.000000 | 0.000000 |
| 75% | 84.710714 | 0.630000 | 0.057143 |
| max | 132.894287 | 40.114285 | 9.400000 |

Table 4. 4 Summary of data after feature scaling

| | A0 | A1 | A2 |
|---|---|---|---|
| count | 7.652000e+03 | 7.652000e+03 | 7.652000e+03 |
| mean | 6.357811e-17 | -3.086049e-17 | 1.585826e-17 |
| std | 1.000065e+00 | 1.000065e+00 | 1.000065e+00 |
| min | -1.451657e+00 | -1.033028e+01 | -1.990587e+01 |
| 25% | -1.014187e+00 | -1.991420e-01 | -2.219857e-01 |
| 50% | -6.681275e-02 | -7.106102e-03 | 7.787216e-03 |
| 75% | 9.626232e-01 | 2.589987e-01 | 1.609692e-01 |
| max | 2.223589e+00 | 1.693671e+01 | 2.520622e+01 |

The coefficient *A0*, has a mean of 47.927 and a standard deviation of 38.214, whereas the other two features have much smaller means and standard deviations. Hence, while computing the distance between two cycle blocks in the 3-dimensional feature space, the differences in coefficients *A1* and *A2* are negligibly small compared to the difference in *A0*. So the features have to be normalized before applying ML algorithms.

There are many normalization techniques available to pre-process the data. Standard scaling is a widely used method to normalize the data where features are standardized by removing the mean and scaling to unit variance.

$$\hat{X} = \frac{X - mean(X)}{variance(X)} \tag{4.16}$$

where,

$\hat{X}$ is the standardized feature,

$X$ is the original feature,

$mean(X)$ is the mean of the original feature,

$variance(X)$ is the variance of the original feature.

StandardScaler module in Scikit-learn, a machine learning framework for Python programming language [30] is used to normalize the data in this study. Summary of the three features in the cycle block data matrix after standard scaling is shown in table 4.4. The pre-processed features now have zero mean and unit variance.

### 4.4.3 Principal Component Analysis

Principal component analysis (PCA) is a dimensionality reduction technique by which the data is projected onto a lower-dimensional uncorrelated orthogonal feature space having maximum variance.

The dataset is first centered around the origin by normalizing with a standard scaler. This is done because PCA rotates and scales data in order to find uncorrelated axes. This is effectively done only when any linear translation of the data has already been removed. The covariance matrix of the normalized data is then evaluated. The principal components are the eigenvectors of the covariance matrix and the respective eigenvalues represent how much variance contained in each principal component. So ordering eigenvectors in the decreasing order of their eigenvalues would give the required principal components in their decreasing order of significance. The required number of principal components are selected and the original data is projected onto the new lower dimensional feature space spanned by the selected principal components.

PCA is primarily used to reduce the number of features to 2 or 3 so that the data can be visualized. If a data has N dimensions where N is greater than 3, the data cannot be

visualized. Plotting random 3 features would result in significant loss of data and hence would not yield an acceptable visualization of the data. PCA can be used to transform such high dimensional data into low dimensional feature space and the data can be plotted with first two or three principal component directions.

Apart from the advantage of being able to visualize the data, lower number of features significantly decreases computational requirements. Transforming data using PCA often improves the performance of the machine learning algorithms. Since the cycle block data has only 3 features, PCA is used only as a mean to transform the data and no dimensionality reduction is required. Hence all the three principal components are used for the clustering task. PCA module in Scikit-learn [30] is used for applying principal component analysis in this work.

### 4.4.4 FCM Clustering

Scikit-Fuzzy (sk-fuzzy) [31], which is a collection of fuzzy logic algorithms intended for use in the SciPy Stack, written in the Python computing language, is used for clustering the cycle blocks into different groups.

The inputs to the sk-fuzzy algorithm are the cycle block dataset after applying principal component analysis, the desired number of clusters, array exponentiation applied to the membership function at each iteration, stopping criteria and a maximum number of iterations allowed. Algorithm returns cluster centers, final fuzzy c-partitioned matrix, final Euclidean distance matrix, objective function history, number of iterations run and final fuzzy c-partition coefficient [31]. In this research work, the FCM algorithm is applied with a weighting exponent of 2 and initial fuzzy c – partitioned matrix is initialized randomly. The effects of weighting exponent or initial fuzzy c – partitioned matrix on the resulting clusters are not studied.

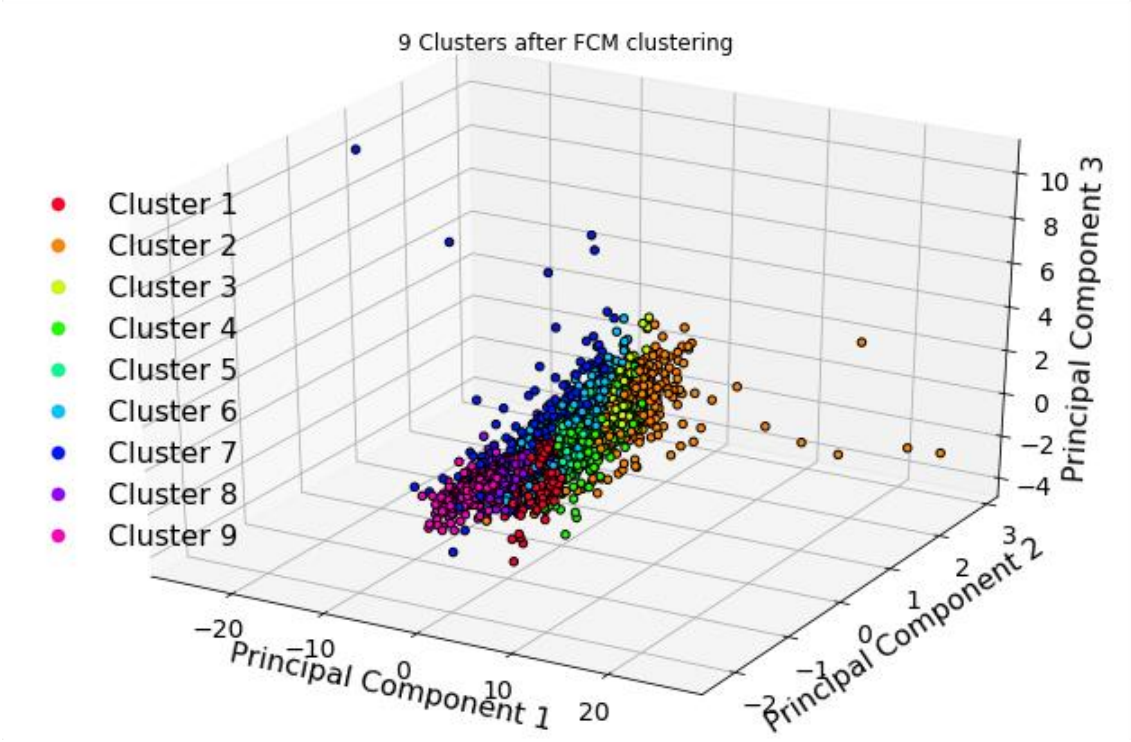
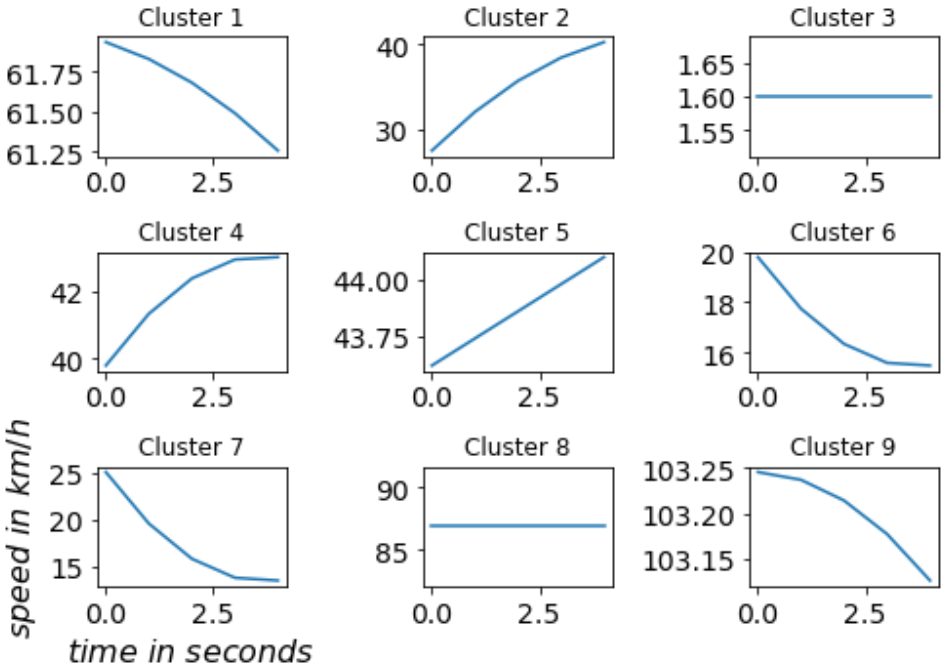
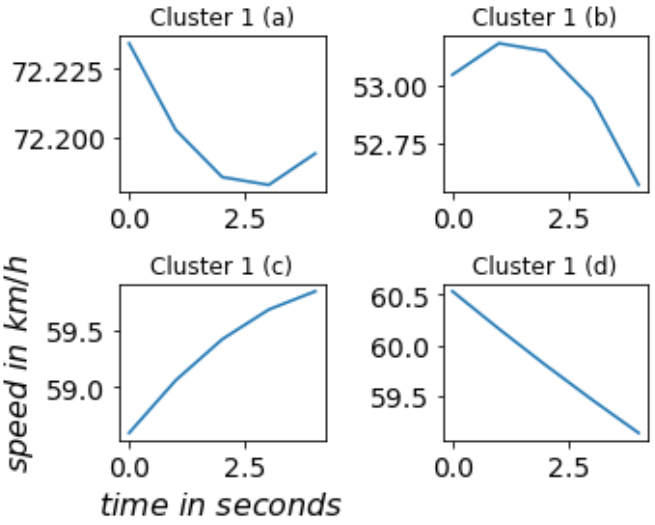



Figure 4. 6 Clusters formed by the FCM algorithm

Fuzzy c-partitioned matrix is an array that contains the membership value of each data point to belong to each of the 9 clusters. Each data point is assigned to the cluster with the maximum membership value. The resulting clusters are shown in Figure 4.6. Figure 4.7 (a) shows the 9 cycle blocks extracted from each of the 9 clusters. Cycle block one represents a pattern corresponding to driving events at about 61 km/h . 4 randomly selected cycle blocks from cluster 1 are shown in Figure 4.7 (b). It can be seen that all of these cycle blocks have a pattern that represents driving events between 50 km/h and 60 km/h. Similarly, other cycle blocks also represent a specific driving pattern in the 5-second window. The resulted cycle block library, along with respective polynomial coefficients, are shown in Table 4.5. If these 9 cycle blocks are used to represent a driving cycle in the block form, the cycle blocks in the original driving cycle are replaced by these 9 cycle blocks that are most similar to the cycle block being replaced.




(a)




(b)

Figure 4. 7 (a): Cluster centers from all 9 clusters. (b): 4 randomly selected cycle blocks from cluster 1

Table 4. 5 Selected 9 cycle blocks with their respective polynomial coefficients.

| | A0 | A1 | A2 | Block |
|---|---|---|---|---|
| 0 | 14.931429 | -5.732857e+00 | 9.357143e-01 | B1 |
| 1 | 19.782858 | -2.355714e+00 | 3.214286e-01 | B2 |
| 2 | 1.600000 | -2.515102e-17 | 4.148413e-17 | B3 |
| 3 | 58.599998 | -4.624683e-15 | 1.541561e-15 | B4 |
| 4 | 102.559998 | 4.000000e-02 | -1.588056e-15 | B5 |
| 5 | 22.462856 | 5.674286e+00 | -6.285715e-01 | B6 |
| 6 | 40.125713 | 2.385714e-01 | -7.142857e-03 | B7 |
| 7 | 86.900002 | -1.436228e-14 | 1.609783e-15 | B8 |
| 8 | 31.631428 | 1.997143e+00 | -2.142857e-01 | B9 |

If an unknown cycle block with coefficients $A_0$, $A_1$ and $A_2$ is given to the trained FCM clustering algorithm, it can predict the cluster the unknown cycle block belongs to. The trained algorithm computes the distance of the unknown data from each of the cluster centers and assigns membership values. This information can be used to choose which of the 9 feature block should be used to replace the unknown cycle block. This procedure is used to convert any given driving cycle into block representation using the cycle blocks in the cycle block library.

In order to quantify the error in approximating the original driving cycle using the block representation, coefficient of determination (r-squared value) is used. It is a number that indicates how well a regression line predicts the value of a dependent variable than the mean of that variable. It also shows how much of the variance in the dependent variable is explained by the independent variable. Coefficient of determination value can be expressed as,

$$r^2 = \left[\frac{\sum Z_X Z_Y}{n-1}\right]^2 \tag{4.17}$$

where,

$Z - score\ of\ the\ independent\ variable,$

$$Z_X = \frac{X-\bar{X}}{\sigma_X} \quad (4.18)$$

$Z - score\ of\ the\ independent\ variable,$

$$Z_Y = \frac{Y-\bar{Y}}{\sigma_Y} \quad (4.19)$$

$r$ is the Pearson's r correlation coefficient

$n$ is the number of samples in the distribution

$\sigma_X$ is the standard deviation of the independent variable, $X$

$\sigma_Y$ is the standard deviation of the dependent variable, $Y$

$\bar{X}$ is the mean of the independent variable, $X$

$\bar{Y}$ is the mean of the dependent variable, $Y$

the coefficient of determination ranges from zero to one. It has a value of one when there is a perfect linear correlation between the two variables and a value of zero when there is no linear correlation between the two variables.

In the cycle block selection task, speed values at each second in a given driving cycle can be considered as the independent variable whereas speed generated by using the polynomial coefficients corresponding to the cycle blocks that represent the same driving cycle can be considered as the dependent variable. In a perfect block representation of the driving cycle, the speed of both the generated driving cycle and actual driving cycle should perfectly match. This means there should be a perfect linear correlation between the two variables. A plot of dependent variable vs independent variable, in this case, should give a straight line corresponding to the equation $X = Y$ and the coefficient of determination will be 1. However, this is not possible since we are using only a very small number of cycle blocks to represent a driving cycle. If we include all the 7652 driving segments from the 36 standard driving cycles, the block representation can perfectly represent any driving cycle from those 36 standard driving cycle. But it cannot represent a driving cycle that is different from those 36 cycles with the same accuracy. The task in the cycle block selection is to select the minimum number of cycle blocks to represent any driving cycle with acceptable accuracy. Hence, a quantification of the error in block representation should be used to tune the number of cycle blocks to be included in the cycle block library.

9 random driving cycles from the CHTS data, three each from each dataset that was divided based on trip length, were used to evaluate the performance of the block representation. One of the 9 driving cycles used for evaluation is shown in Figure 4.8.

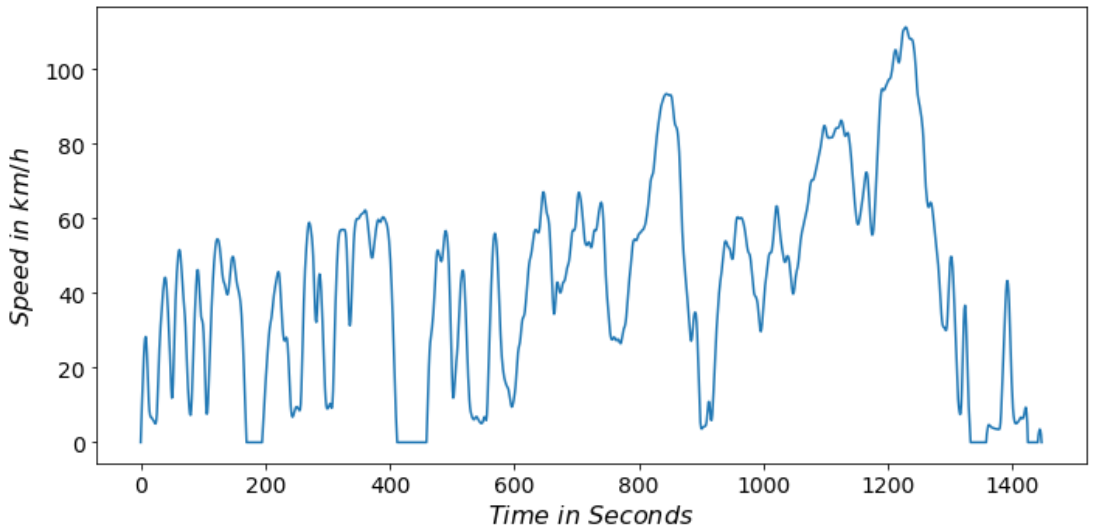



Figure 4. 8 Randomly selected one driving cycle from the 9 evaluation driving cycles

In order to convert the driving cycle in the block representation, firstly, one cycle was divided into segments of 5 seconds each and features, the quadratic polynomial coefficients $A_0$, $A_1$ and $A_2$, were extracted. Then these features were transformed by the same standard scaler and principal component analysis functions that were used during the training of the FCM clustering algorithm. The transformed features were then given to the trained FCM clustering model which predicts the clusters that each input cycle block belongs to. This gave a sequence of cycle block representation of the given driving cycle. Similarly, other driving cycles were also converted into block representation. The length of the block representation depends on how many segments of 5 seconds can the original driving cycle be divided. The block representation of the driving cycle shown in Figure 4.8 using 9 cycle blocks in the cycle block library is shown in table 4.6.

Table 4. 6 Driving cycle in Figure 4.8 in block representation with 9 cycle blocks.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 279 | 280 | 281 | 282 | 283 | 284 | 285 | 286 | 287 | 288 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | B8 | B8 | B1 | B6 | B6 | B6 | B3 | B3 | B5 | B1 | ... | B1 | B1 | B6 | B6 | B6 | B3 | B6 | B6 | B6 | B6 |

1 rows × 289 columns

In order to evaluate how well the block representation captures patterns in the original driving cycle, these block representations can be used to generate the driving cycle that it represents. This is done by converting each feature block in the block representation back to speed vs time data. Respective polynomial coefficients $A_0$, $A_1$ and $A_2$ are used to generate the speed data for each block and they are plotted against respective time to give an approximation of the original driving cycle that the block representation represents. The generated driving cycle is shown along with the original driving cycle and coefficient of determination in Figure 4.9.

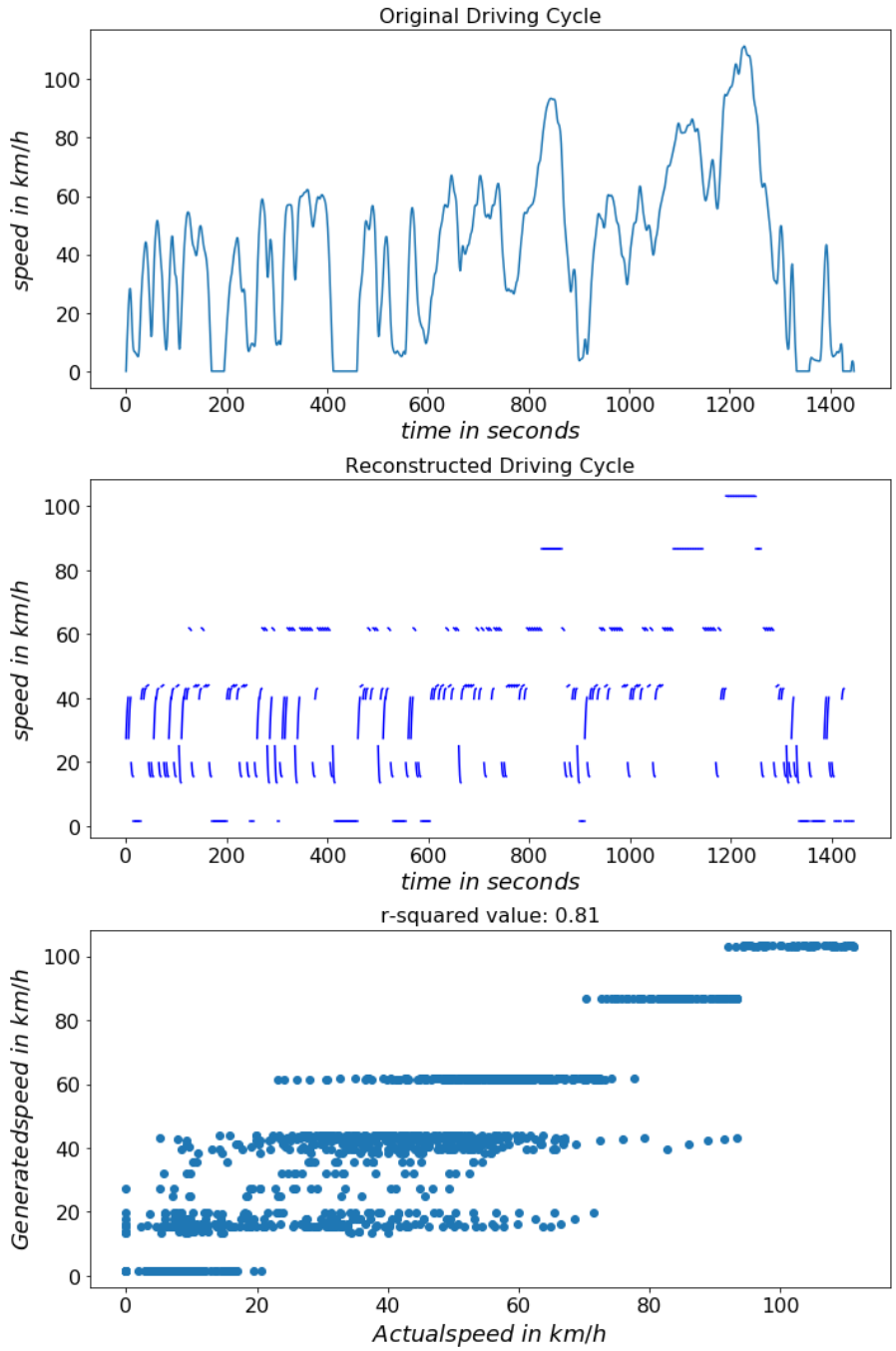



Figure 4. 9 Performance of the block representation on the driving cycle in Figure 4.8

The error in the above block representation is quantified by calculating the coefficient of determination using the actual speed data and the generated speed data. A plot of the actual speed vs generated speed is shown in Figure 4.9. In the ideal case, this should be a straight line with $45^{o}$ slope. It can be seen that the 9 cycle blocks are not sufficient to represent the

given driving cycle. However, with only 9 cycle blocks, a lot of information about the original driving cycle is captured. The mean coefficient of determination over the nine driving cycles was found to be 0.793.

### 4.4.5 Tuning Number of Cycle Blocks

As discussed in the previous section, with only 9 cycle blocks, the mean coefficient of determination over the nine driving cycles was only 0.793, which is far from the ideal value of 1.0. Using a higher number of cycle blocks will improve the accuracy of block representation at the cost of higher computation. Also, too much information on the driving cycle data may make the pattern recognition algorithm more sensitive to noise like patterns. Hence, the optimum number of cycle blocks was estimated iteratively by evaluating how well the block representation represents the information in the original driving cycle with varying number of cycle blocks. FCM clustering algorithm was performed over a range of a number of clusters. The nine randomly selected driving cycles from each of the three California datasets were used to evaluate the performance of block representation. During each iteration, these nine driving cycles were reconstructed from the block representation and the respective coefficient of determination were calculated. The overall performance of the block representation with the current number of cycle blocks in the cycle block library was represented by the mean coefficient of determination. The number of cluster centers at which the lowest coefficient of determination is greater than 0.8 and mean coefficient of determination is greater than 0.9 was selected as the optimum number of clusters.

The plot of coefficient of determination vs number of cluster centers/ number of cycle blocks in the cycle block library is shown in Figure 4.10. Initially, with a lower number of cycle blocks, the error is significantly higher. After about 150 number of clusters, the improvement in the representation accuracy reduced. At about 200 cluster centers, the coefficient of determination plateaued and hence for the final cycle block library, the 7652 cycle blocks were clustered into 200 groups and one representative cycle block was selected from each cluster.

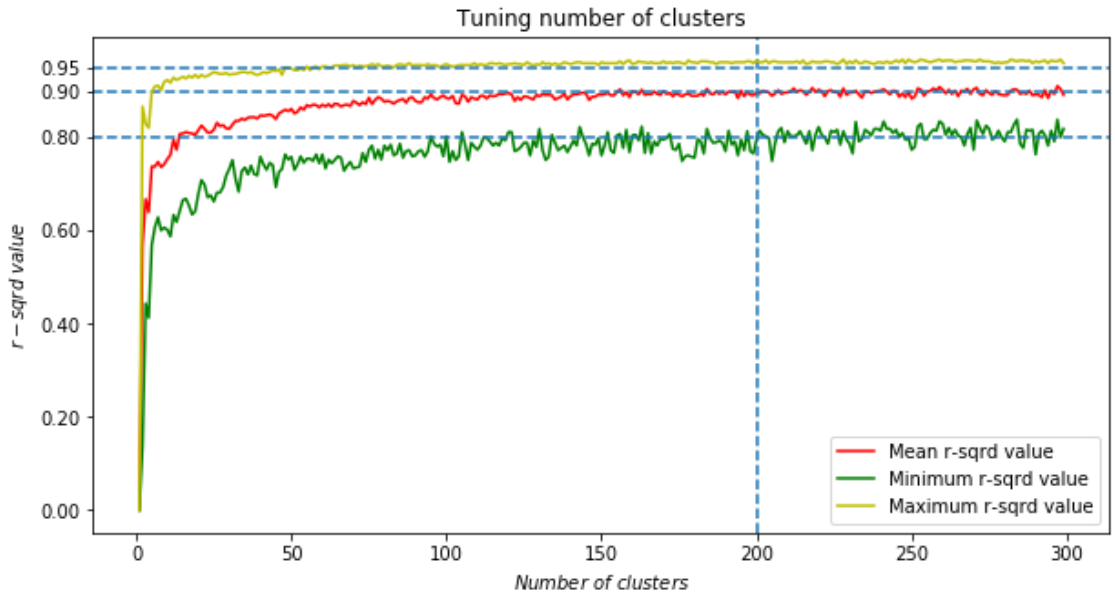



Figure 4. 10 Tuning the number of cycle blocks in the feature block library

Each cluster center represents a cycle block that is included in the final cycle block library. These 200 cycle blocks are used to represent any given driving cycle. California driving cycle is shown in Figure 4.8, its reconstructed driving cycle using the 200 cycle blocks and the respective coefficient of determination are shown in Figure 4.11. The mean coefficient of determination over the nine driving cycles was found to be 0.916, which is significantly higher than that obtained using only 9 cycle blocks. Also, the reconstructed driving cycle with 200 cycle blocks is successful in preserving most of the patterns in the original driving cycles. Since we are interested in finding overall patterns in the driving data in California, this block representation serves as features to compare different driving cycles. Similar driving cycles will have a sequence of similar cycle blocks. So, comparing the sequence of driving blocks will effectively find similarities and differences between different driving cycles. Hence the simplified representation of the driving cycle with 200 cycle blocks can be used for analyzing patterns.

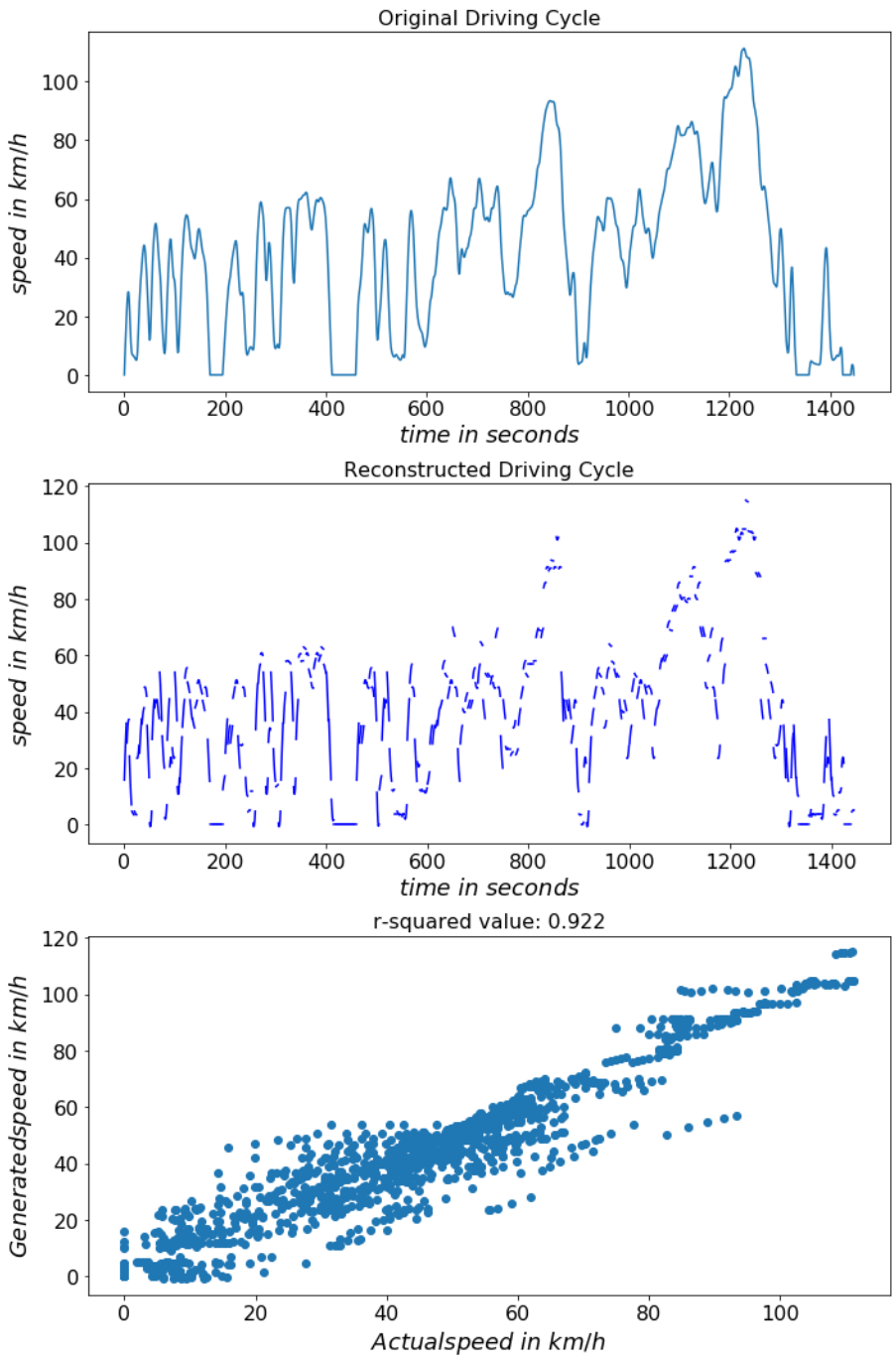



Figure 4. 11 Performance of the block representation using 200 standard cycle blocks on the selected random driving cycle

# 5 Driving Cycle Pattern Recognition

As discussed in Chapter 3, CHTS data is pre-processed and divided into three datasets based on the trip length. The FCM pattern recognition algorithm is applied to each of this dataset to group driving cycles with similar patterns together. In order to reduce the computation involved in pattern recognition and later in training an artificial neural network for real-time prediction, a sample of 2000 driving cycles each were drawn from each dataset. FCM algorithm is applied to each of this dataset to group similar driving cycles together and to get representative driving cycles from each group.

## 5.1 Block Representation

All the dataset are converted to the block representation for machine learning. The FCM clustering model that was used to select 200 standard cycle blocks is used to transform raw time vs speed driving data into block representation. Since another FCM clustering model is used in this study for clustering the CHTS data, the first model used to generate feature block library is referred to as FCM model 1 and the second FCM model that is used for pattern recognition in the California driving cycles are referred to as FCM model 2, henceforth.

Each driving cycle was firstly divided into segments of 5 seconds. A quadratic polynomial was fitted onto each segment to get the coefficients *A1*, *A2*, and *A3*. FCM model 1 was already trained to assign any of these cycle blocks into one of the 200 cycle blocks in the cycle block library. The coefficients were then transformed with standard scaler and PCA that were used in training FCM model 1. Coefficients were then inputted to the FCM model 1, which predicts cycle block clusters that each segment belongs to. Each of the segment was assigned with a label corresponding to the cycle block. Hence, each driving cycle is represented by a sequence of labels.

A random driving cycle in its block representation is shown below.

B89 B91 B49 B151 B100 B32 B49 B0 B68 B64 B64 B64 B64 B64 B64 B64 B64 B102 B54 B134 B186 B177 B196 B29 B34 B93 B94 B91 B89 B144 B165 B163 B80 B29 B85 B34 B101 B3 B134 B177 B123 B28 B114 B133 B57 B51 B43 B159 B159 B2 B96 B95

Standard cycle blocks are labeled from B0 to B199. For the above driving cycle, the first 5-second segment is more similar to B89. So in the block representation, all the speed – time data for the first 5 seconds are represented by a string B89. The same process is repeated for all the 5-second segments and the sequence of cycle block labels are obtained. For the above driving cycle, the total trip length is 260 seconds. Hence there are 52 blocks in the sequence. FCM model 2, that identify unique driving patterns in the driving cycles, pays no attention to the meaning of each cycle block. It focuses on the relative distribution of different standard cycle blocks over the entire trip and finds similarities in the distribution.

## 5.2 Feature Extraction

The California driving dataset in block representation is a sequence of strings, which represent the respective standard cycle block label. Since the labels are strings, there is no order between the labels and hence the data has a nominal level of measurement. Most machine learning algorithms, including the FCM clustering algorithm, can handle only quantitative variables. Hence, the nominal variable has to be encoded to a quantitative variable.

There are various techniques, such as one-hot encoding and label encoding, available for machine learning applications to convert categorical variables into quantitative variables. However, the cycle block label is a high cardinality categorical variable. It can take one of the 200 possible values. One-hot encoding of such a variable would explode the feature dimension to 40,000. In addition to very high computation and memory requirements, the curse of dimensionality also comes into play when dataset with such a huge number of features are used for machine learning applications. If label encoding is used, each block will be given a numerical value. For example, B1 can be labeled as 1, B199 as 199, etc. However, the machine learning algorithm will try to learn from the intervals between the different labels. This may lead the algorithm to give more importance to B199 than B1. But both B1 and B199 represent cycle blocks and there exists no order in their labels. Each standard cycle blocks are equally important in training the algorithm to cluster. Hence both label encoding and one-hot encoding are not suitable for high cardinality categorical variable.

Since our dataset consists of a sequence of strings, inspired by Natural Language Processing (NLP) field in machine learning, term frequency-inverse document frequency (Tf-idf) is used for feature extraction. Term frequency refers to how frequently a term appears in a document. For our data, this refers to the frequency with which each cycle block appears in a particular driving cycle.

Term frequency gives equal importance to all the cycle blocks in the driving cycle. However, there may be some cycle blocks more common than others. Common cycle blocks provide less information in distinguishing between different cycles and also their count shadows rare but more interesting cycle blocks. It is the rare driving events that distinguish one driving cycle from the other. Inverse document frequency finds cycle blocks that are common among all driving cycles and reduce their significance in finding patterns in the data.

Mathematically Tf-idf can be described as given below.

$$tf(t,d) = Term\ Frequency$$
$$= Frequency\ of\ a\ particular\ cycle\ block\ in\ a\ driving\ cycle$$

$$tfidf(t,d) = tf(t,d) * idf(t,d) \tag{5.1}$$

$$InverseDocument\ Frequency,$$

$$idf(t,d) = log_{10}\left[\frac{1+n_d}{1+df(d,t)}\right] + 1 \tag{5.2}$$

$$n_d = Total\ number\ of\ driving\ cycles$$

$$df(d,t) = Number\ of\ driving\ cycles\ that\ contains\ the\ cycle\ block\ t$$

The resulting $tfidf(t,d)$ is then normalized by the Euclidean norm.

Tf-idfVectorizer function in scikit-learn for Python is used for feature extraction [30]. For the long and medium trips, each driving cycle was divided into 4 segments and Tf-idfVectorizer was applied to each segment. This counts the number of times each standard feature block appears in each of the 4 segments, apply idf transformation and join resulting features along the row to get the final feature of 800 dimensions. This process of dividing a single driving cycle into 4 segments before feature extraction helps to distinguish between cycles having similar cycle blocks but different patterns due to the difference in the location of the distribution of these cycle blocks. For example, the dummy driving cycles B1 B1 B2 B2 and B2 B2 B1 B1 would give the same feature vector since both cycles have the same

cycle block frequency. However, one cycle could be the mirror image of the other. Splitting the driving cycle into multiple segments and performing feature extraction and then concatenating each segment would help better distinguish even if two cycles have similar feature block count but different distribution in time. For the short trips, only two segments are used for feature extraction.

### 5.3 FCM Clustering

Standard scaler was used to transform the data matrix after feature extraction. PCA was used to transform the extracted features and data visualization. The 800-dimensional long and medium trip data and 400-dimensional short trip data were transformed using PCA and the first three principal components were used to visualize the data. FCM clustering algorithm was applied to the transformed dataset. The number of clusters is a parameter that is to be given to the algorithm. Estimating the optimum number of clusters that best fits the underlying data is a challenging task.

Fuzzy Partition Coefficient (FPC) is a parameter that expresses the fuzziness in a given fuzzy partition [26] [32] [33]. Consider the fuzzy partition of $N$ objects into $C$ clusters,

$$FPC = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}(u_{i,j})^2 \tag{5.3}$$

where, $u_{i,j}$ = membership value of $i^{th}$ sample to the $j^{th}$ fuzzy set; FPC varies between 1, for hard partition, and $\frac{1}{C}$, for completely fuzzy sets. If the FPC takes the lowest value, close to $\frac{1}{C}$, that means there is no clustering tendency in the dataset or the algorithm failed to reveal the clusters. FPC takes its highest value when the given dataset is partitioned into the optimal number of clusters that is inherent in the dataset. Hence, to find the optimal number of clusters, the given dataset can be partitioned into a range of possible optimal numbers of clusters iteratively and FPC can be compared. A plot of the number of clusters vs FPC would have a global maximum corresponding to the optimal number of clusters.

In this work, a number of clusters between 2 and 10 were tried and the plot of FPC vs the number of clusters obtained is shown in figure 5.1. Also, a plot of the first two principal components of the long trip data before assigning any cluster is shown in figure 5.2.

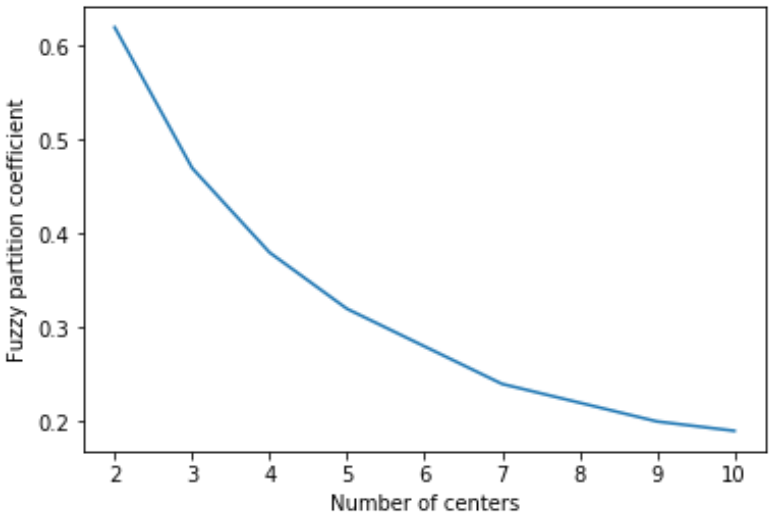



Figure 5. 1 FPC vs Number of clusters for the long trip dataset

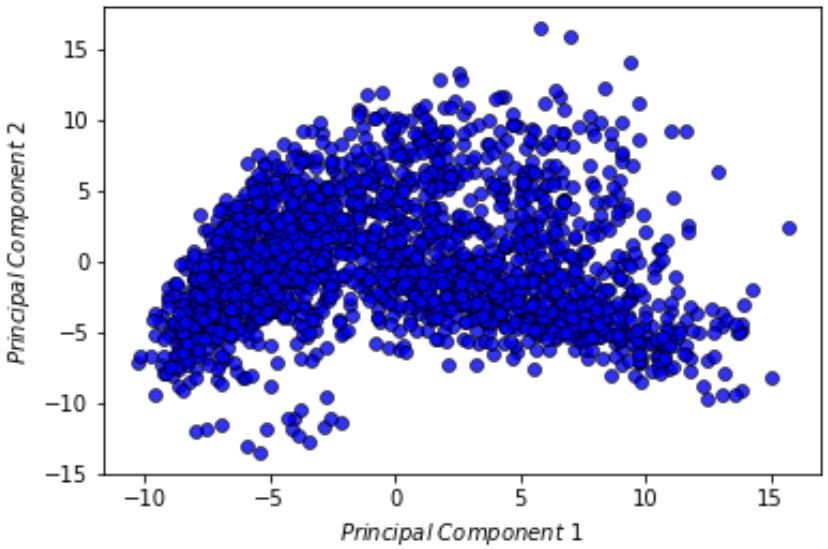



Figure 5. 2 Plot of first two principal components of the long trip dataset

It can be seen from Figure 5.2 that the data is highly overlapping and there are no distinct clusters in the data. Hence, the optimum number of clusters is not obvious. Any attempt to partition the data would result in a decrease in FPC. Same is reflected in Figure 5.1, where the FPC is found to be decreasing with an increasing number of clusters. Ideally, if distinct

clusters existed without much overlap, the FPC vs the number of cluster centers graph would have a global maximum at the optimum number of cluster centers.

The CHTS dataset was collected from all of California's 58 counties and portions of three adjacent counties from Nevada. The geographical area of data collection was spread over different cities, terrains, and demographics. Hence, a very high number of driving patterns are possible that reflect the driving conditions and the driver's behavior. Hence distinct clusters are highly unlikely. One solution to this problem could be to use data from a single driver driving in the same city to find driver-specific patterns and perform HEV energy management specific to that driver. The proposed energy management approach is driver-specific. The pattern recognition is done on data from a single vehicle. This data will more likely have distinct patterns without much overlapping.

Due to the lack of obvious clusters in the data, an iterative approach that relies on the physical interpretation of the clusters was used to find the optimum number of clusters in each of the three datasets in the CHTS data. FCM clustering was performed with a number of clusters and for each resulting clusters, random members were visually inspected for physical interpretation. Driving cycles within the same cluster should have similar patterns and driving cycles in different clusters should have different patterns. Based on the iterative approach, the optimum number of clusters was found to be 3 for all the three datasets. Having more clusters resulted in similar driving cycles falling in different clusters. The resulting clusters from the three datasets are shown in Figure 5.3, 5.4 and 5.5. The cluster centers are shown in red square markers.

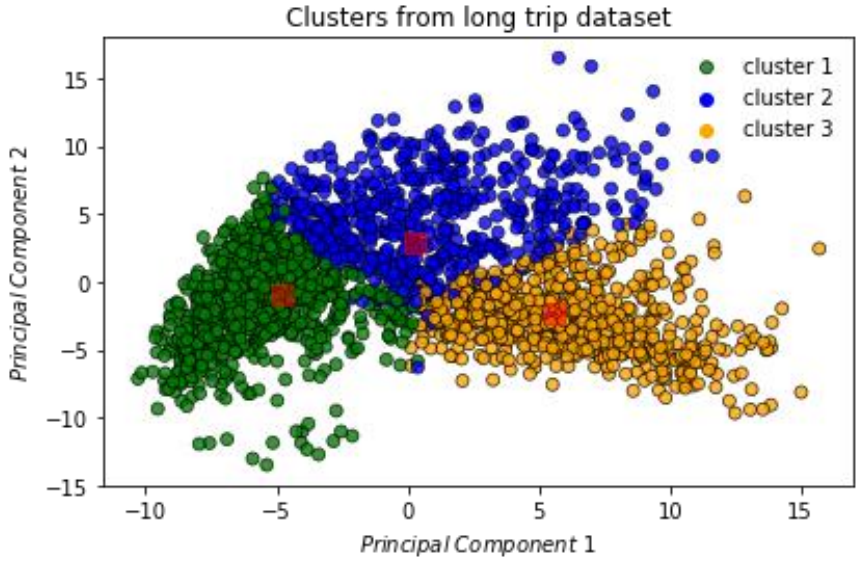



Figure 5. 3 Cluster from the long trip dataset

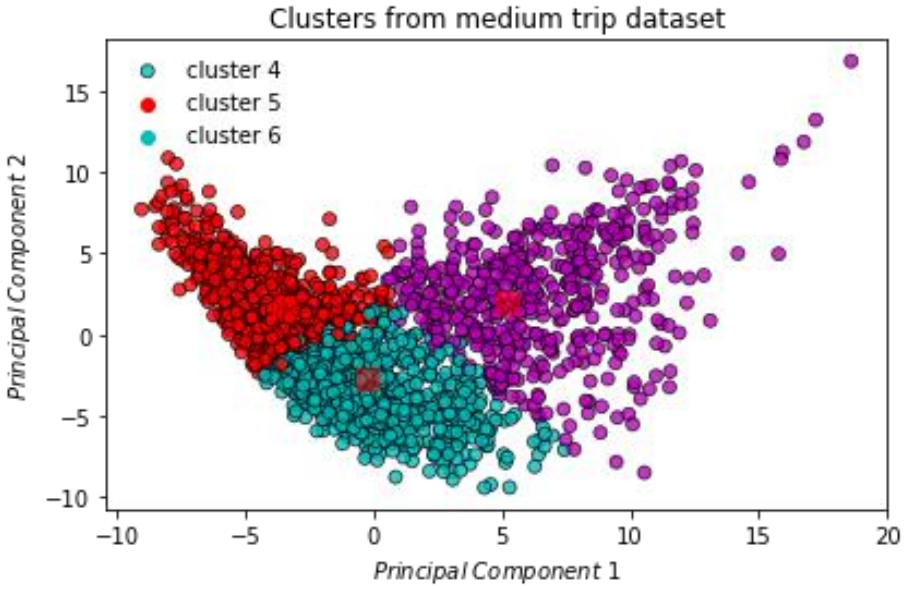



Figure 5. 4 Clusters from the medium trip dataset

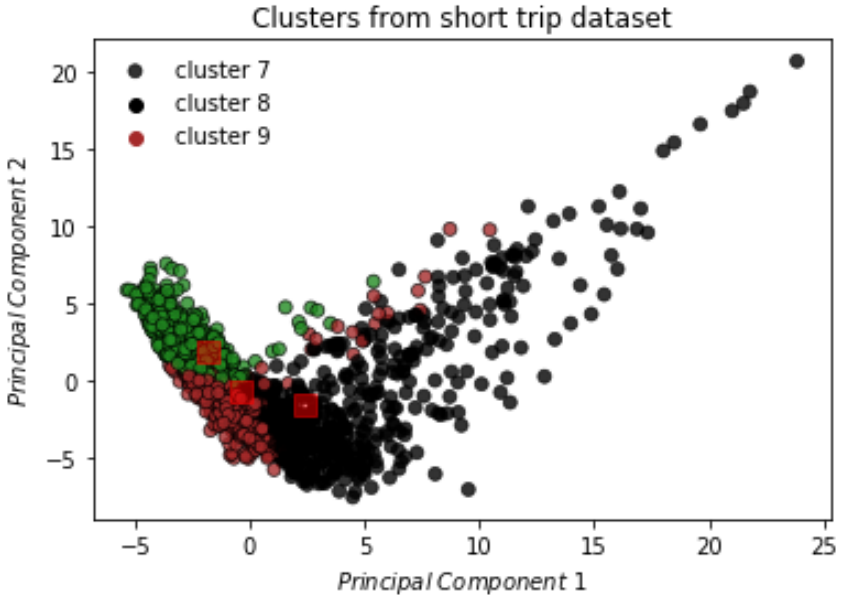



Figure 5. 5 Clusters from the short trip dataset

## 5.4 Physical Interpretation of the Clusters

### 5.4.1 Long Trips

2000 samples in the long trip driving cycle data are grouped into three clusters on applying FCM clustering. The cluster centers are selected from each of the clusters to form representative driving cycles that represent the unique patterns in the long trip dataset. The representative driving cycles are shown in Figure 5.6.

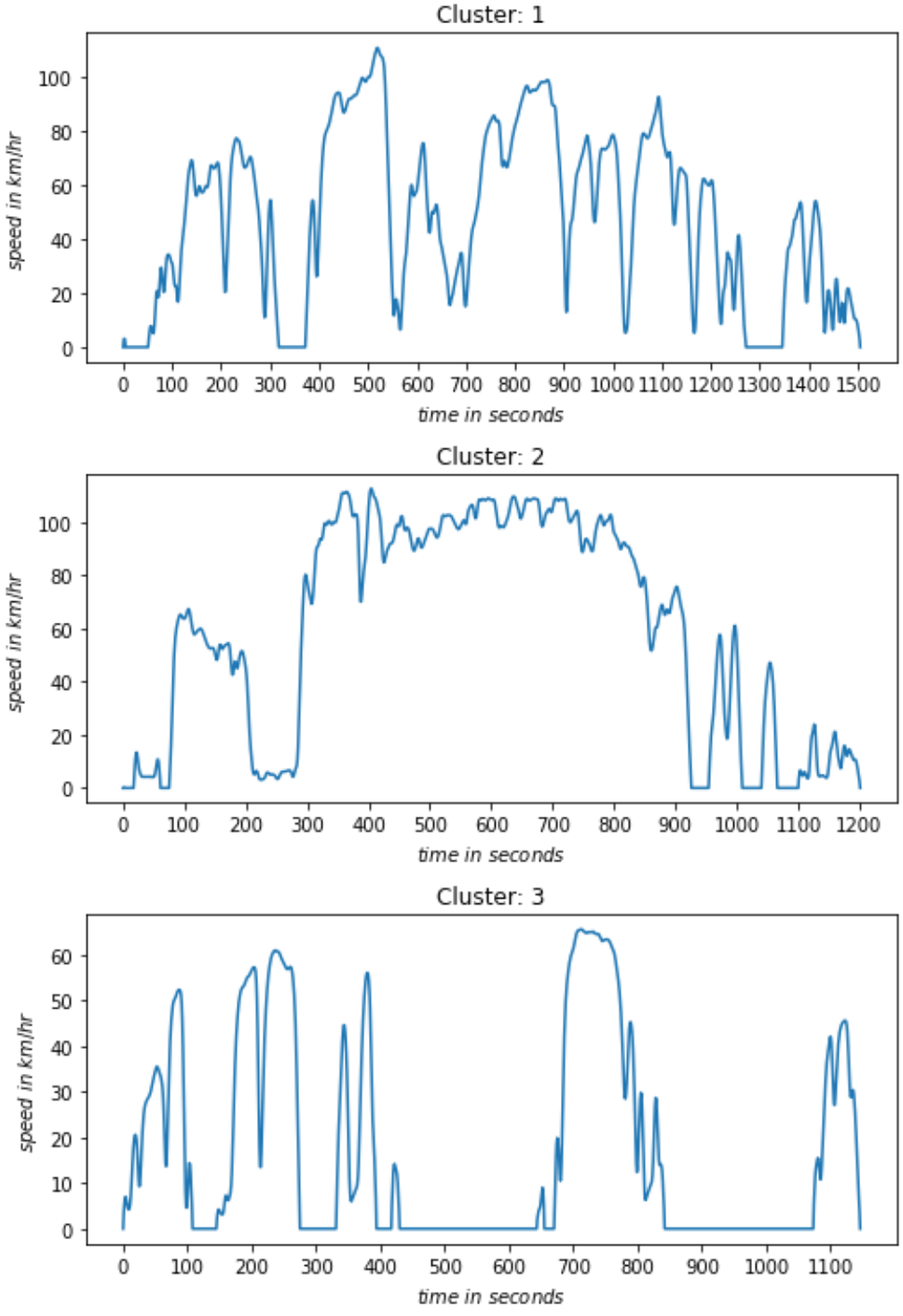



Figure 5. 6 Representative driving cycles from the long trip dataset

Cluster 0 represents a 2-lane freeway driving conditions. Vehicle speed reaches freeway speed range but fails to cruise at that speed for a long time possibly due to traffic signals

and high traffic. Cluster 1 represents a vehicle entering a multi-lane freeway from an urban driving condition, cruising at highway speeds and finally exiting back onto an urban road. Cluster 2 represents urban driving condition with frequent stops and idling. A bar chart showing the number of driving cycles in each of the three clusters is shown in Figure 5.8.

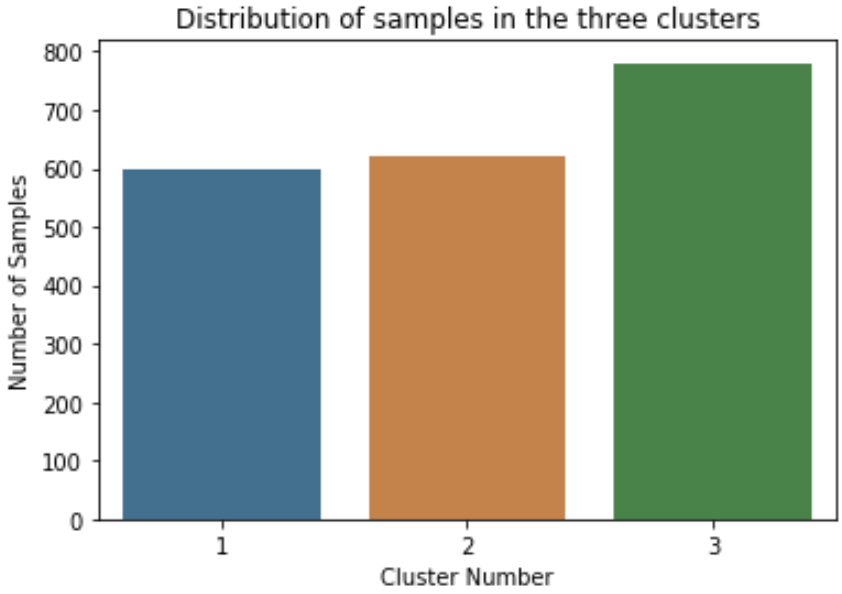



Figure 5. 7 Number of driving cycles in each cluster in the long trip dataset

Appendix B1 shows 2 randomly selected driving cycles from each of the three clusters. As expected, the samples within the same cluster show similar patterns and samples between two clusters differs in their patterns.

### 5.4.2 Medium Trips

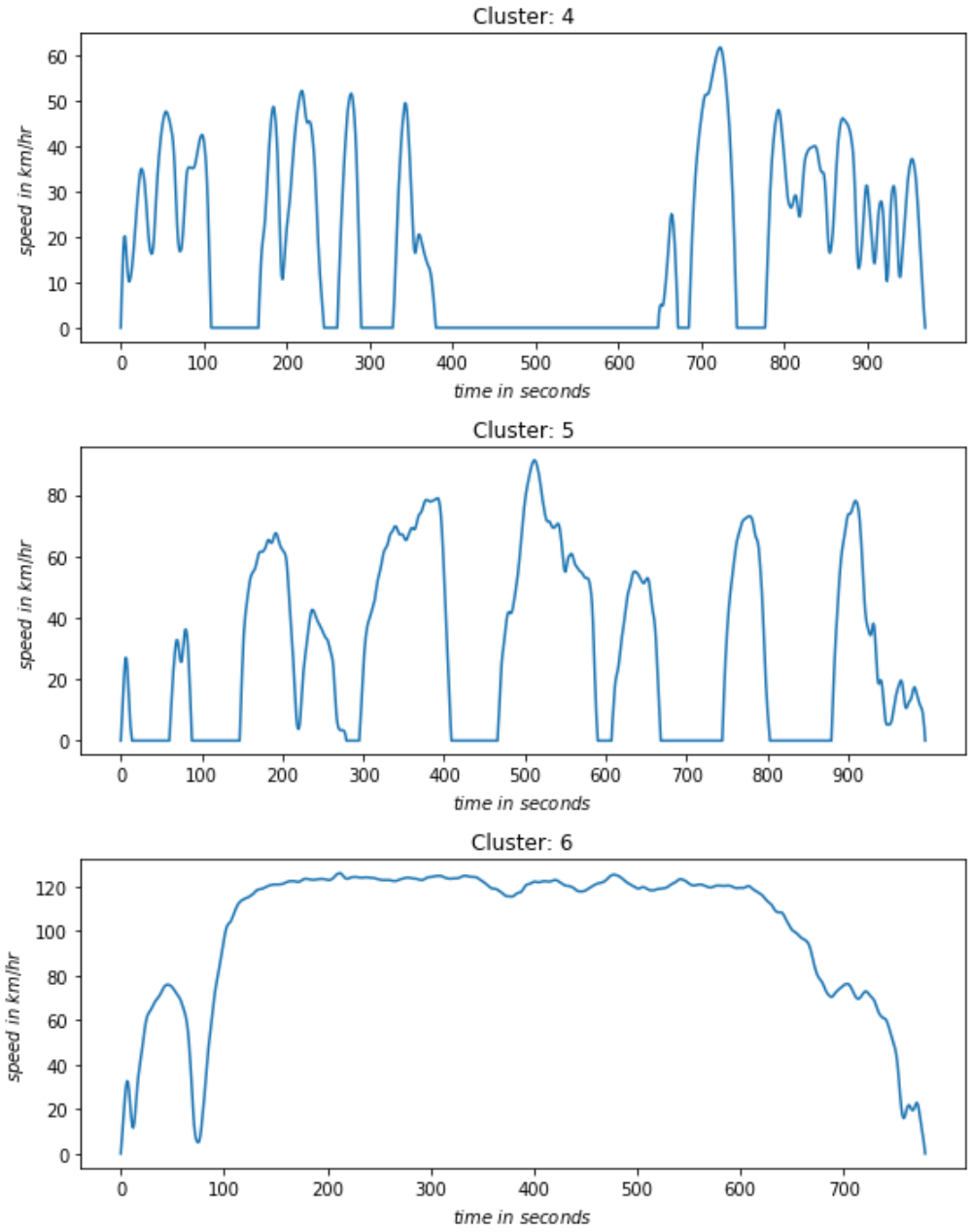



Figure 5. 8 Representative driving cycles from the medium trip dataset

The three clusters resulted from the FCM clustering of the medium trip driving data are shown in Figure 5.8. Cluster 4 shows the driving patterns corresponding to urban driving conditions. Cluster 5 represents a 2 lane freeway driving condition with signals and high traffic. Cluster 6 represents a multilane freeway driving conditions. The overall patterns

observed in the medium trip dataset are similar to that observed in the long trip dataset except that the trip length is more in the long trip dataset.

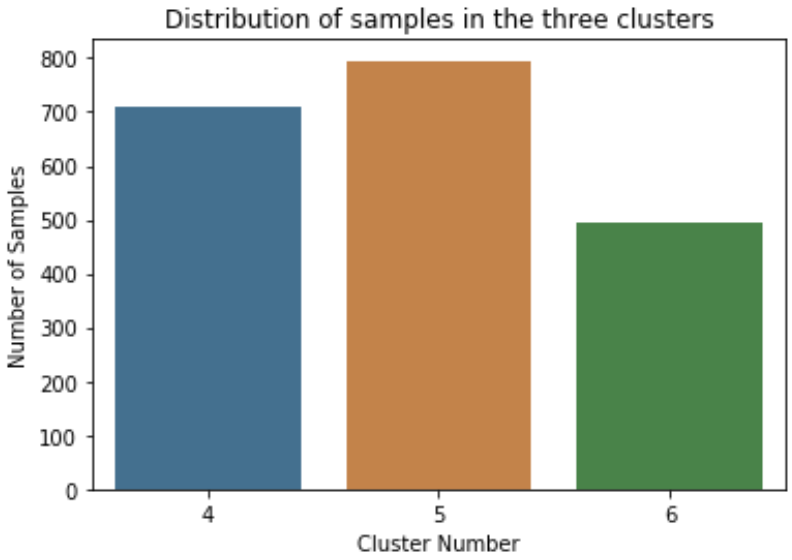



Figure 5. 9 Number of driving cycles in each cluster in the medium trip dataset

Distribution of the number of driving cycles in each of the three clusters is shown in the bar graph in Figure 5.9. Randomly selected 2 driving cycles from each of the clusters are shown in Annexure B2.

### 5.4.3 Short Trips

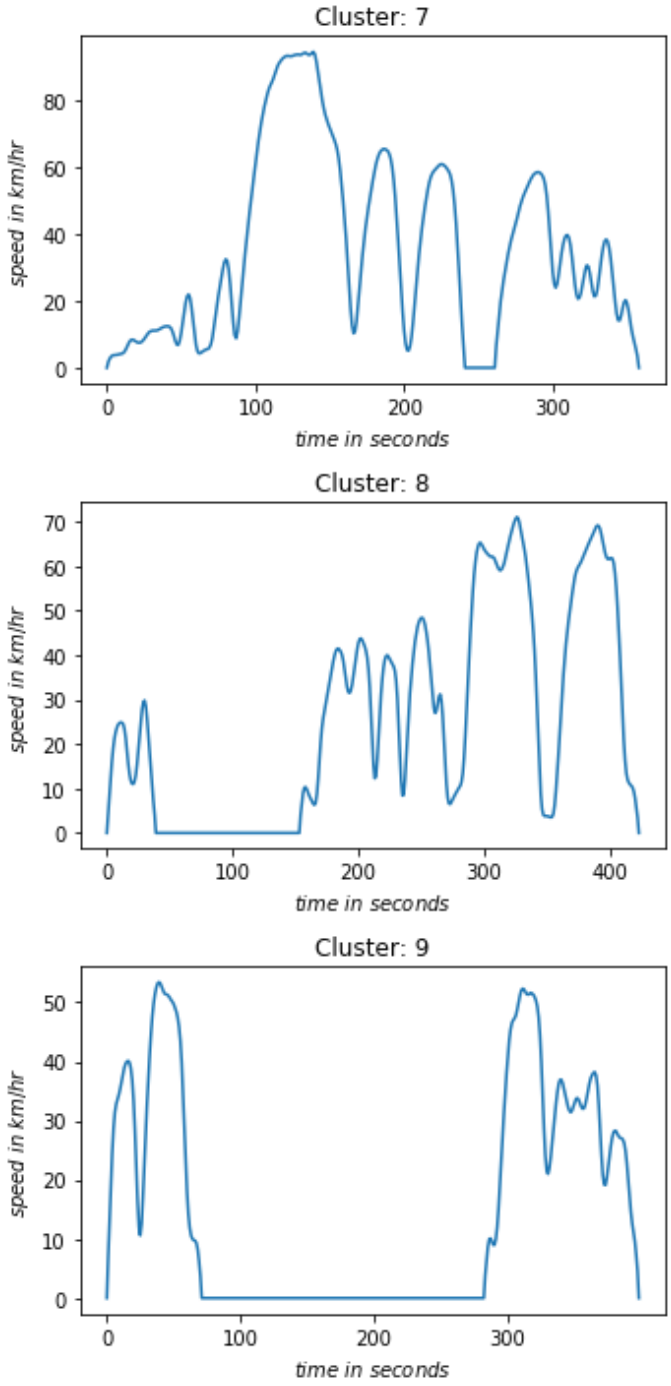



Figure 5. 10 Representative driving cycles from the short trip dataset

Cluster centers of the three clusters resulted from the short trip data are shown in Figure 5.10. Cluster 7 represents a short trip on the highway. Cluster 8 represents possibly rural

driving conditions. Cluster 9 represents urban driving conditions where the vehicle speed rarely exceeding 60 km/hr. Also, there is a lot of idling events suggesting frequent stop signals corresponding to urban driving conditions. Figure 5.11 shows the distribution of samples in the three clusters in the short trip dataset. Annexure B3 shows randomly selected 2 driving cycles from each of the 3 clusters.

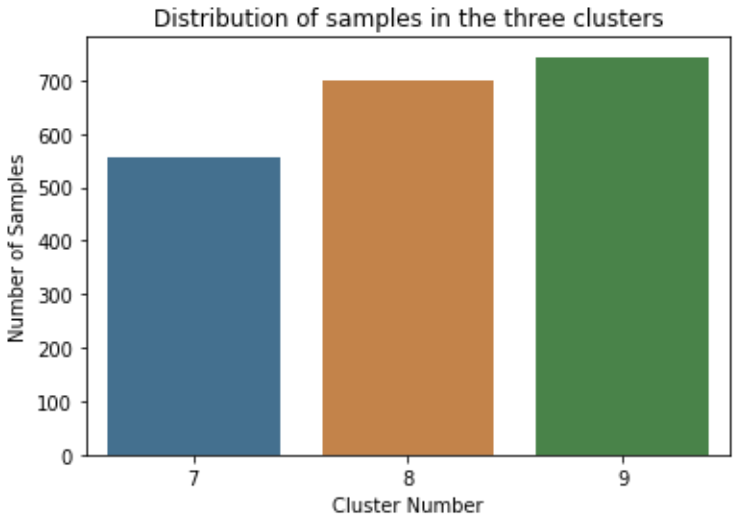



Figure 5. 11 Number of driving cycles in each cluster in the medium trip dataset

# 6 Real-time Prediction with Artificial Neural Networks

In Chapter 5, nine representative driving cycles that represent the driving conditions in the state of California are obtained by performing FCM clustering. Optimal control parameters for energy management in HEV can be evaluated for each of these 9 representative driving cycles by Dynamic Programming based optimization algorithm. Real-time optimal energy management identifies the current trip as one of the 9 representative driving cycles by predicting pattern in the driving cycle and uses respective optimal control parameters.
After performing FCM clustering, the unlabelled 6000 driving cycles in the CHTS data can be labeled as per their respective cluster numbers. This labeled data can be used to train a supervised machine learning model that can predict the label of any driving cycle. One hidden layer Artificial Neural Networks (ANN) is used for real-time pattern prediction.

## 6.1 Artificial Neural Network

Artificial Neural Networks are a category of supervised machine learning algorithms that are inspired by the neurons in the human brain. A neural network is a collection of layers of neurons which are linked together. In a feed-forward neural network, the neurons from one layer are connected to the neurons in the next layer. The first layer is called the input layer which has the features of the respective sample as the values. The last layer is called the output layer which will have the value that the algorithm is trying to predict. All the layers between the input and output layer are called hidden layers. The number of hidden layers can vary depending upon the problem. For a complex problem with much complex non-linear decision boundaries, more than one hidden layer may be required and such networks are called deep neural networks. In this project, we are using a neural network with only one hidden layer. A schematic diagram of a neural network with 1 hidden layer is shown in Figure 6.1.

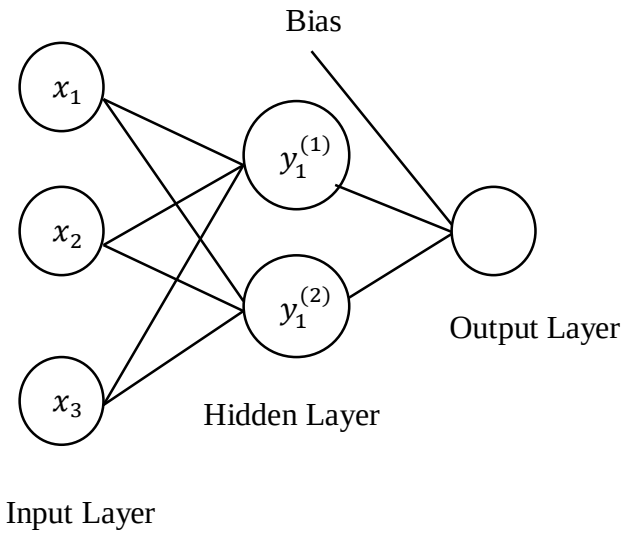



Figure 6. 1 A simple neural network architecture with one hidden layer.

The neurons from the input layer are given appropriate weights to calculate each of the neurons in the first hidden layer. The weighted sum of neurons from the previous layer is then transformed with an activation function that results in the value of neurons in the first hidden layer. The process of finding the values of the neurons in the next layer using the values in the current layer is termed as forward propagation. The value of the neuron in the output layer is the output of the neural network. For a neural network with L layers, there will be an L-1 set of weights each corresponding to layers from 1 to L-1. The initial weights are selected randomly and used to make the initial prediction. The error in prediction is then evaluated by comparing the predicted value with the actual value. Then a loss function is defined that quantifies the error in prediction. The objective now is to minimize this loss function with respect to the weights. Once the weights are optimized, the trained ANN is used to make a prediction for samples for which output is not known.

## 6.2 Data for Training ANN

The California driving data was initially divided into three groups based on the trip length. Each of the three groups, long, medium and short trips, have 2000 driving cycles. After performing FCM clustering on these groups, a total of 9 clusters was obtained. Clusters are labeled from 1 through 9. Then all the 6000 driving cycles are combined to form a data

matrix with each driving cycle labeled as per the cluster it belongs to. This data matrix is used to train the ANN for predicting the label of any driving cycle.

## 6.3 Feature Extraction

A similar approach used for the feature extraction for FCM clustering is used here as well. Firstly, the 6000 raw driving cycles were represented in block representation using 200 cycle blocks. Then the data was split into training and test datasets. 4500 driving cycles were used as the training data and they were used for training the ANN. Remaining 1500 driving cycles were used to test the performance of the trained ANNs in real-time pattern prediction.

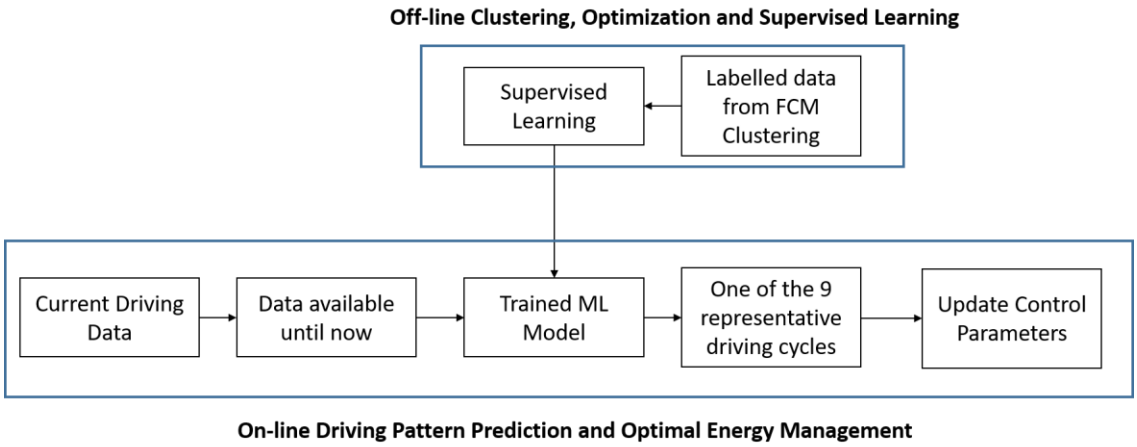



Figure 6. 2 Real-time optimal energy management

A schematic diagram showing real-time optimal energy management is shown in Figure 6.2. The ANN should be able to predict the pattern of an ongoing trip at the beginning of the trip itself for the energy management strategy to be effective. It is of no use if the ANN predicts the correct pattern using all the data after completion of the current trip. Hence the ANN needs to be trained to predict patterns in an ongoing trip using whatever data available in real-time. The driving cycles are represented in block form for supervised learning. Here, when a trip begins, the speed from the first 5 seconds are mapped to respective cycle block from the cycle block library. Only this cycle block is available for the ANN to predict the pattern. Hence the ANN should be trained only on the first cycle blocks from the training data for this task. When another 5 seconds elapse, ANN gets another cycle block. It can

now use the first two cycle blocks to make a more informed prediction. Hence a second ANN that is trained on the first two cycle blocks from all the driving cycles in the training data is required for this task. Similarly, a different ANN trained on additional cycle blocks are required after every 5 seconds to make a real-time prediction. So a sequence of 320 ANNs were trained and this library of ANNs was used to make real-time predictions on the CHTS data.

For the training of $i^{th}$ ANN that is trained to predict the pattern after the first $5i$ seconds of the trip, only the first *i* cycle blocks from each of the driving cycles in the training dataset were used. TfidfVectorizer was applied to these block representation of the incomplete driving cycles. Since TfidfVectorizer counts and scales the number of occurrences of each of the 200 cycle blocks in each driving cycles, irrespective of the value of *i*, the number of features in the driving cycle will be 200. This enables to use the same ANN architecture for training all the 320 ANNs. This features, along with respective labels that represent the cluster, is used to train the ANNs. For real-time prediction, respective ANN from the trained ANN library based on the time elapsed was used.

For the real-time prediction, respective ANN is selected from the sequence of trained ANNs. This depends on the time elapsed after the trip begins. After the first 5 seconds into the trip, the first ANN from the sequence will make a prediction. After 10 seconds, the second ANN will use 10 seconds of the driving data to associate the current trip to its respective representative driving cycle. So a growing length of 5-second interval of data is used to update the initial pattern identified. Although, the real-time pattern identification is performed only after every 5 seconds, at any instant the current trip will have one of the representative driving cycle associated with it. Those representative driving cycles each have an optimum powertrain control and energy management strategy that controls the control parameters every second of the trip. Thus, even though the real-time pattern prediction is not done every second, at all times the ongoing trip will have a controller that performs every second.

### 6.4 ANN Architecture

The architecture of all the 320 ANNs is maintained the same. They are a single layer feed-forward artificial neural network with 500 neurons in the hidden layer. Since our data

matrix has 200 features, the number of layers in the input layer is fixed at 200. Similarly, since the target variable has 9 categories, cluster 1 through cluster 9, the number of neurons in the output layer is fixed at 9. The ANN architecture used is shown in Figure 6.2.

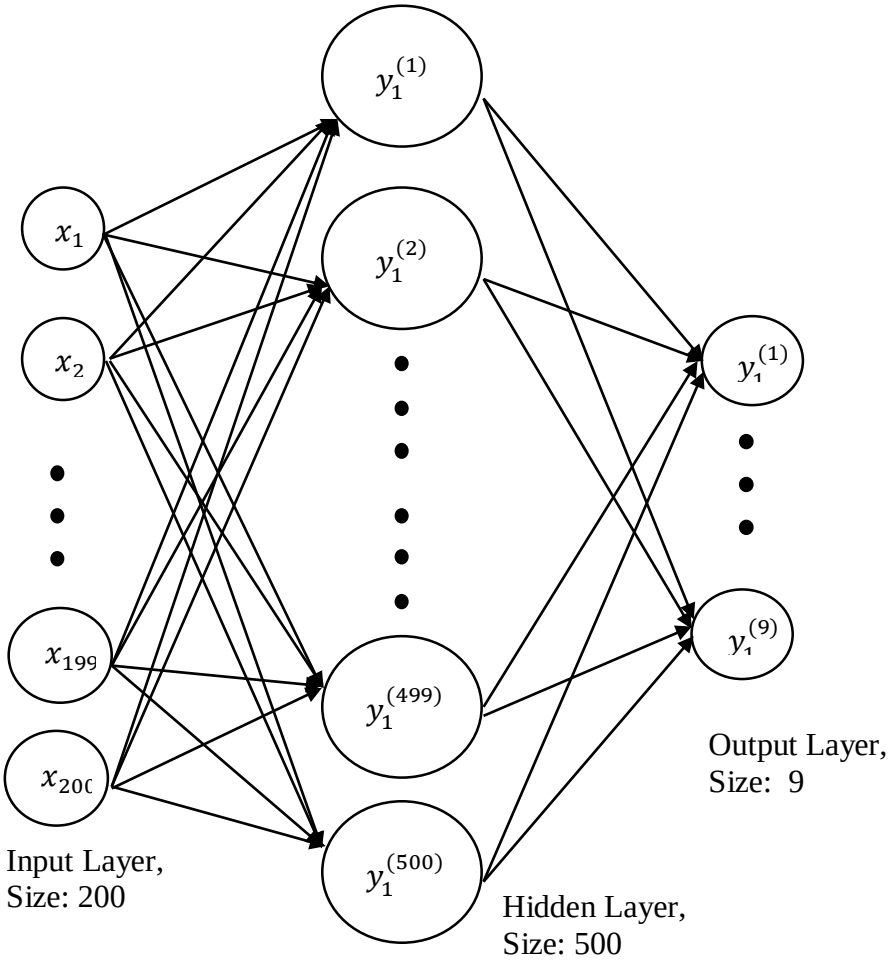



Figure 6. 3 Neural network architecture for real-time pattern prediction.

## 6.5 Training ANN

The dataset was divided randomly into a training and a test dataset in 75:25 proportion. The larger dataset was used to train the 320 ANNs and the smaller dataset was used to assess the performance of the trained model. If the performance of the trained model on the training dataset is significantly higher than that on the test dataset, then the model is said to be overfitting on the training dataset. Although such models give high accuracy on training data, it fails to give acceptable prediction performance on data that was not used

to train the model. Hence it is important to express the performance of machine learning models on test datasets. The training dataset for the ANN in this work has 4500 driving cycles and the test dataset has 1500 driving cycles.

The training was performed iteratively 320 times, which is the maximum number of cycle blocks in the longest trip in the CHTS dataset. Each of the 320 ANNs was trained in sequence starting from ANN1 that uses only the first cycle block from all the training dataset. The last ANN in the sequence uses all available cycle blocks from the training data. During each iteration, the prediction accuracies on the training data as well as on the test data are recorded.

MLPClassifier in the neural network module in scikit-learn for Python was used for building and training the ANNs [34]. Stochastic Gradient Descent optimization was used to update the weights in the network and "ReLu" activation function was used to introduce nonlinearity. L2 regularization parameter of 7 was used to prevent overfitting. A learning rate of 0.001 was used that controls the step size in updating the weights. 4500 samples in the training dataset were used to train the model and 1500 samples from the test dataset were used to evaluate the performance of the model.

## 6.6 Testing Performance of ANN

The ANNs were trained in sequence using one additional cycle block in each iteration. Since more information about the trip is available after each iteration, the prediction performance is expected to improve as the trip progresses. The prediction accuracies on both training as well as test dataset are recorded during each iteration to check if the ANNs are overfitting on the training data. A plot of the training and test accuracies are shown in Figure 6.4.


Figure 6. 4 Real-time prediction performance of trained ANNs

The highest training accuracy was found to be 93.87% and peak test accuracy was found to be 89.5%. Since the difference between training and test accuracies is not high, the models are not overfitting on the training data. However, it can be observed that the prediction performance on the test data is only about 30% after the end of 250 seconds and 60% after the end of 500 seconds. By the time the model attains 60% prediction accuracy, the trips with trip length in the range of short trip dataset would be completed and the real-time optimal energy management is highly ineffective.

The real-time pattern prediction performance of the trained ANNs on two randomly selected driving cycles from the CHTS data are illustrated in Figure 6.5 and Figure 6.6.

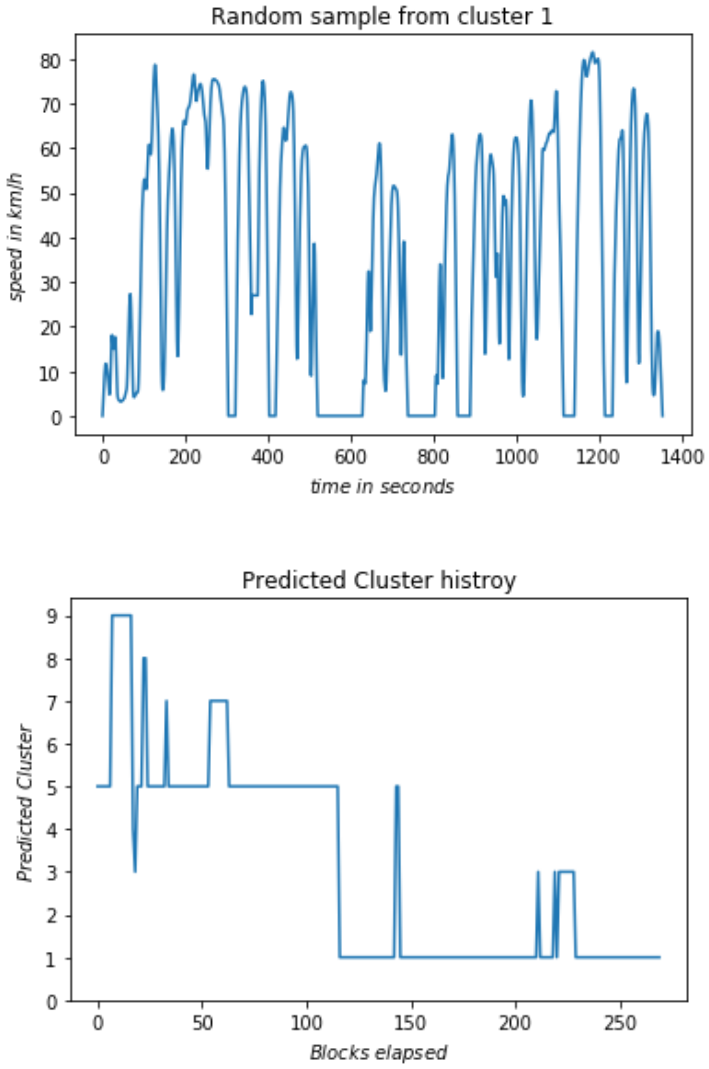



Figure 6. 5 Real-time prediction performance on a driving cycle from CHTS dataset

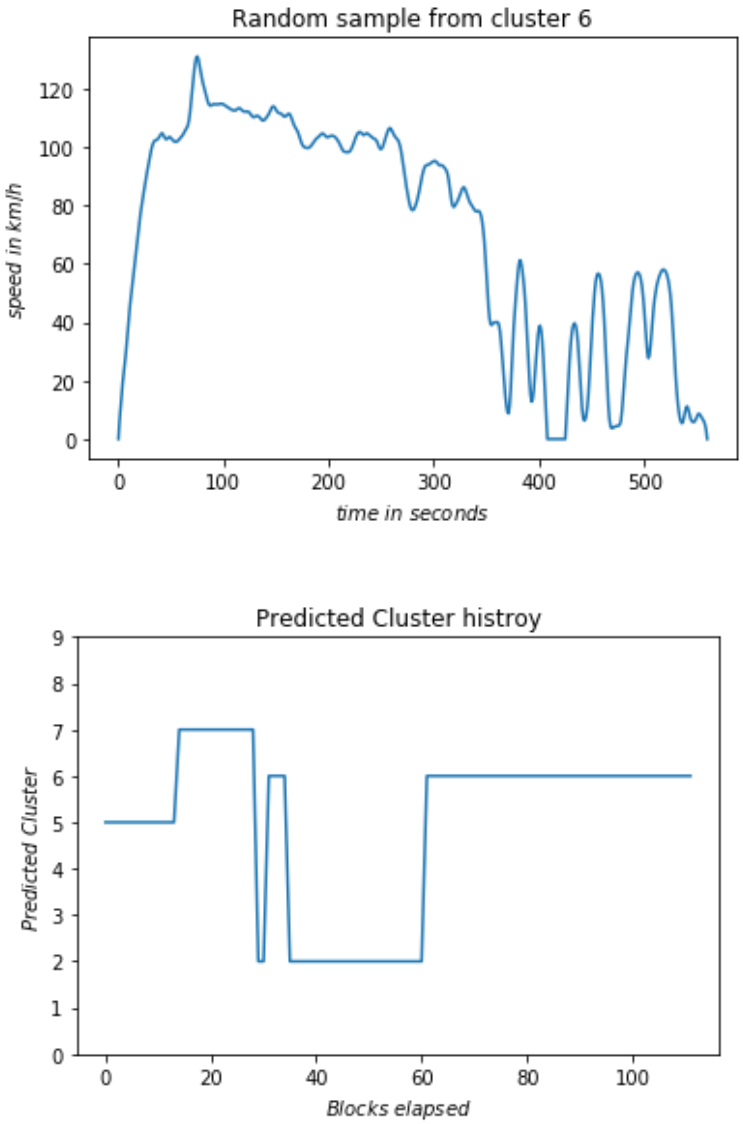



Figure 6. 6 Real-time prediction performance on a driving cycle from CHTS dataset

In Figure 6.5, the random driving cycle selected is from cluster 1. ANNs struggle to predict the correct label in the beginning. Only after about 600 seconds, the correct label was predicted for the first time. Here, for the first half of the trip, the proposed energy management would be highly ineffective. Similarly, for driving cycle in Figure 6.6, the ANNs predicts the correct label consistently only after half the way into the trip.

This poor prediction performance of the ANNs in predicting the pattern early in the trip can be attributed to the data. The CHTS data was collected from a survey area spanning across all the counties in the state of California. Also, each driving cycle is mostly from

different drivers and different vehicles. Hence, a huge number of overlapping patterns are possible in the data as discussed in Chapter 5. Here, the pattern recognition algorithm performs at a lower resolution in terms of the patterns. It tends more towards clustering the driving cycles based on the overall speed. Also, there is significant overlapping in some of the 9 representative driving cycles. For example, cluster 2 and cluster 6 are both representing a multilane freeway driving conditions. The only difference is that the two cycles have different cycle lengths. For the real-time pattern prediction to work better, there have to be unique driving events that are common among all driving cycles in the same cluster. This could be possible only if the data is collected from the same vehicle driven ideally in and around the same city. In order to demonstrate this, the same pattern recognition and pattern prediction machine learning pipelines are applied to a driver-specific driving data collected from the city of Victoria, British Columbia in Chapter 7.

# 7 Pattern Recognition and Prediction on Driver-specific Driving Data

In Chapters 5 and 6, pattern recognition on CHTS data resulted in nine representative driving cycles. However, due to the wide source of the data, the patterns in the driving cycles were overlapping. Hence the was no optimal number of clusters in the CHTS data. This resulted in the algorithm to classify the driving cycles more based on the speed rather than the patterns. Hence, driving cycles in the same cluster were not having the exact same patterns. So Dynamic Programming based optimal control parameters on the cluster centers may not be optimum for all the driving cycles in the clusters that they represent. Hence the proposed energy management strategy has to be implemented driver-specific to be most effective.

In this chapter, driving cycle data was collected from a single passenger car over a one month period from the city of Victoria, British Columbia. The same machine learning pipeline that was applied to the CHTS data was applied to the driver-specific driving cycles as well. Driving cycles were first converted into the block representation and FCM clustering algorithm was performed to get unique representative driving patterns. Then a sequence of ANNs was trained to classify any driving cycle into one of the identified representative driving cycles.

## 7.1 Pattern Recognition

1500 driving cycles from the driver-specific driving data were converted into block representation by using the 200 cycle blocks in the cycle block library. As discussed in Chapter 5, TfidfVectorizer was used to convert the string representation of the driving cycles into numerical values so that machine-learning algorithms can be applied to it. PCA was applied for dimensionality reduction to visualize the data. FCM clustering algorithm was applied to the resulted data to group it into different clusters based on the similarity in the patterns.

In order to find the optimum number of clusters in the driver-specific driving data, FCM clustering was applied iteratively with a number of clusters varying from 2 to 13. A plot of fuzzy partition coefficient vs the number of clusters is shown in Figure 7.1. The curve has a global maximum at a number of clusters equal to 6. This indicates that the quality of

clustering is at its best when the data were grouped into 6 clusters. This implies there exist 6 unique representative driving cycles in the Driver-specific driving data.

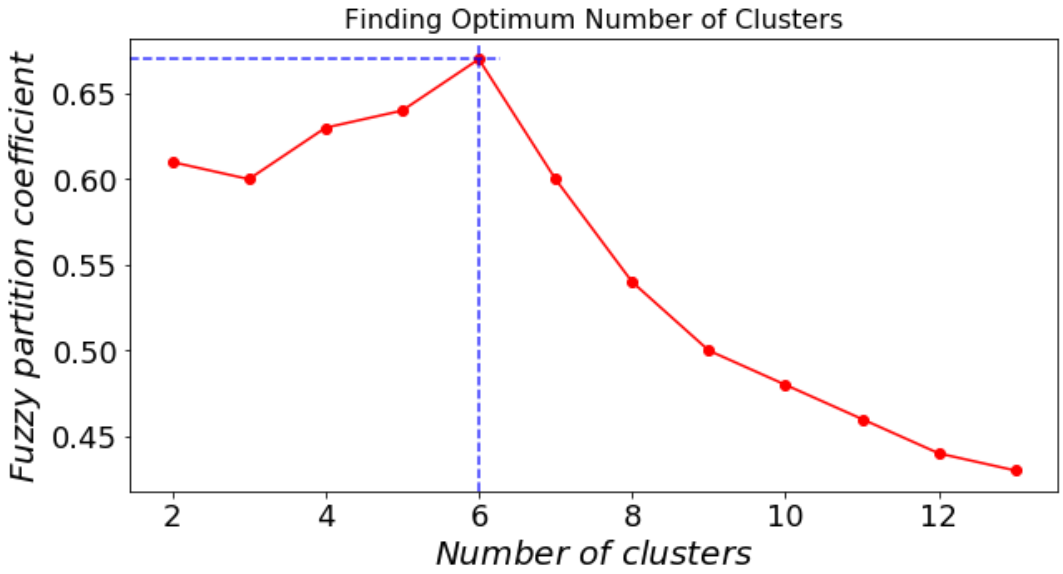



Figure 7. 1 Choosing the optimum number of clusters

Once the optimum number of clusters was found to be 6, the FCM clustering algorithm was performed on the Driver-specific driving data with a number of clusters of 6. A plot of the first two principal components of the data after clustering is shown in Figure 7.2. Also, a plot of the first three principal components is shown in Figure 7.3.

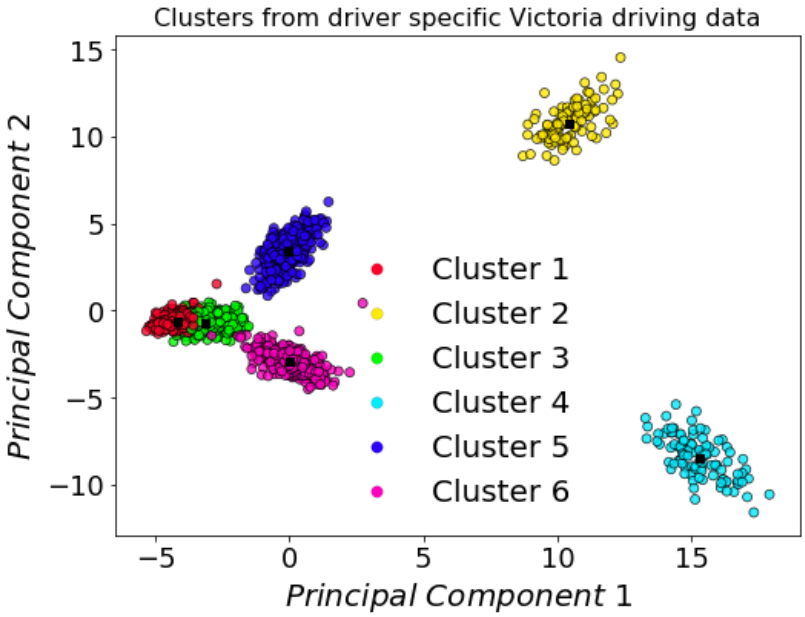



Figure 7. 2 Plot of the first 2 principal components showing the 6 clusters

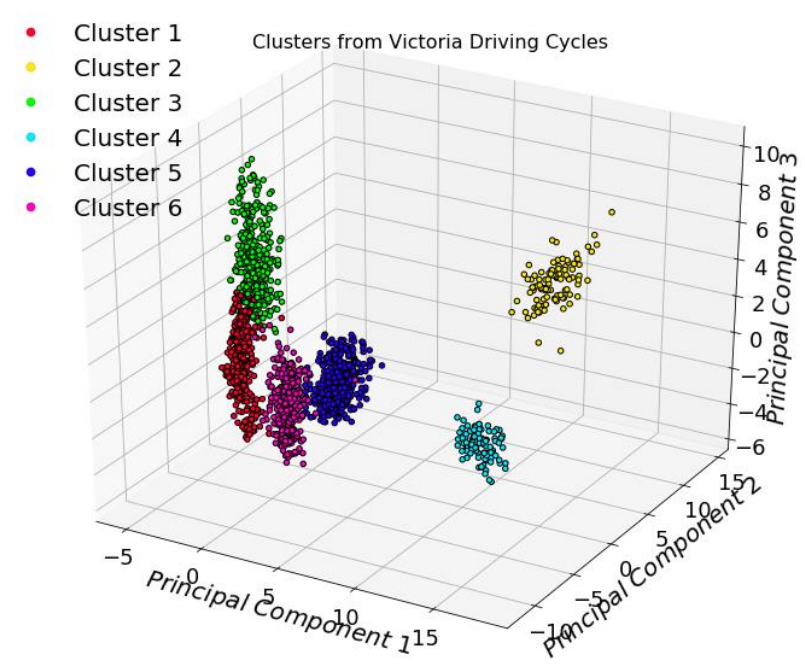
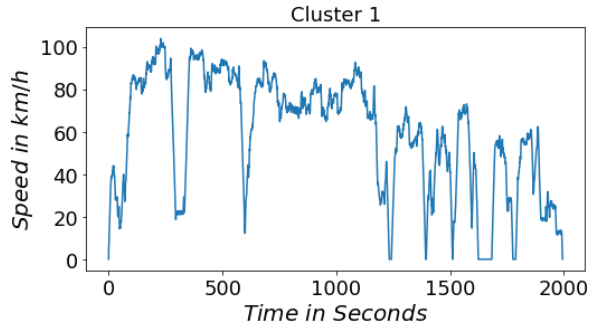


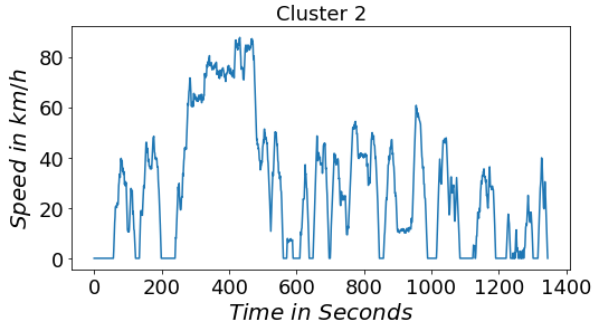

Figure 7. 3 Plot of the first 2 principal components showing the 6 clusters

As expected, there exist 6 distinct clusters in the Driver-specific driving data. It seems like there is some overlapping in the clusters in Figure 7.2. However, this is due to the

inaccuracies in representing the data by using only the first two principal components. The original data has 200 features and Figure 7.2 uses only 2 PCA transformed features to plot the data. Hence, there are some data lost in visualization. In Figure 7.3, the first three principal components are used to plot the data. The additional one dimension compared to Figure 7.3 added more information to the plot and hence the clusters are less overlapping. If all principal components are used, the clusters will be more distinct in the 200-dimensional feature space.

## 7.2 Representative Driving Cycles



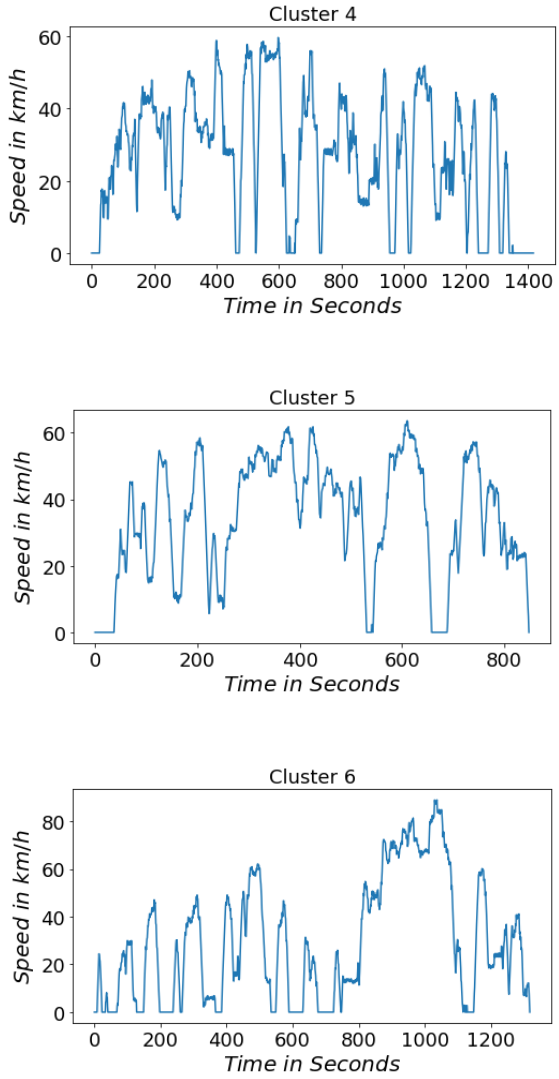




Figure 7. 4 Representative driving cycles from driver-specific driving data from Victoria

The 6 driver-specific representative driving cycles from the Driver-specific driving data is shown in Figure 7.4. Cluster 1 represents a driving cycle where a large portion of the trip is on a 2 lane highway and the last portion of the trip is on urban driving conditions. Cluster 3 shows a similar pattern in reverse order. Hence cluster 1 and cluster 3 could be from the same route but in different directions. Cluster 2 represents a driving pattern wherein the first half of the trip, the vehicle enters the highway from urban driving conditions, cruises at highway speeds and exit onto urban driving conditions. The second half of the trip is entirely on urban driving conditions. Cluster 6 is roughly the mirror image of cluster 2,

indicating both are possibly the same trip but in different directions. Cluster 4 and cluster 5 shows urban driving conditions, with more traffic and stop signals in cluster 4.

Randomly selected 2 driving cycles from each of the 6 clusters are shown in Annexure D. Driving cycles from the same clusters are very similar to their respective cluster centers shown in Figure 7.4. Hence, optimal control parameters for the centers of the clusters can be used on any of the driving cycles from the same cluster for optimal energy management.

## 7.3 Real-time Prediction

As discussed in Chapter 6, a sequence of ANNs was trained to classify any given driving cycle into one of the identified six representative driving cycles. 1500 driving cycles from the Driver-specific driving data were labeled as per the respective cluster number. The data was then divided into training and test dataset. The training dataset had 1500 driving cycles and the test dataset had 500 driving cycles. The same ANN architecture shown in Chapter 6 is used here as well. Also, the same parameters of the MLP Classifier are used. The maximum cycle blocks in the longest driving cycle in the Driver-specific driving data is 380. Hence 380 ANNs were trained in sequence, with each ANN getting one additional cycle block to make the prediction. During each iteration, the performance of the trained ANN was evaluated on the training dataset as well as the test dataset. A plot of the training and test accuracy in each iteration is plotted Figure 7.5.

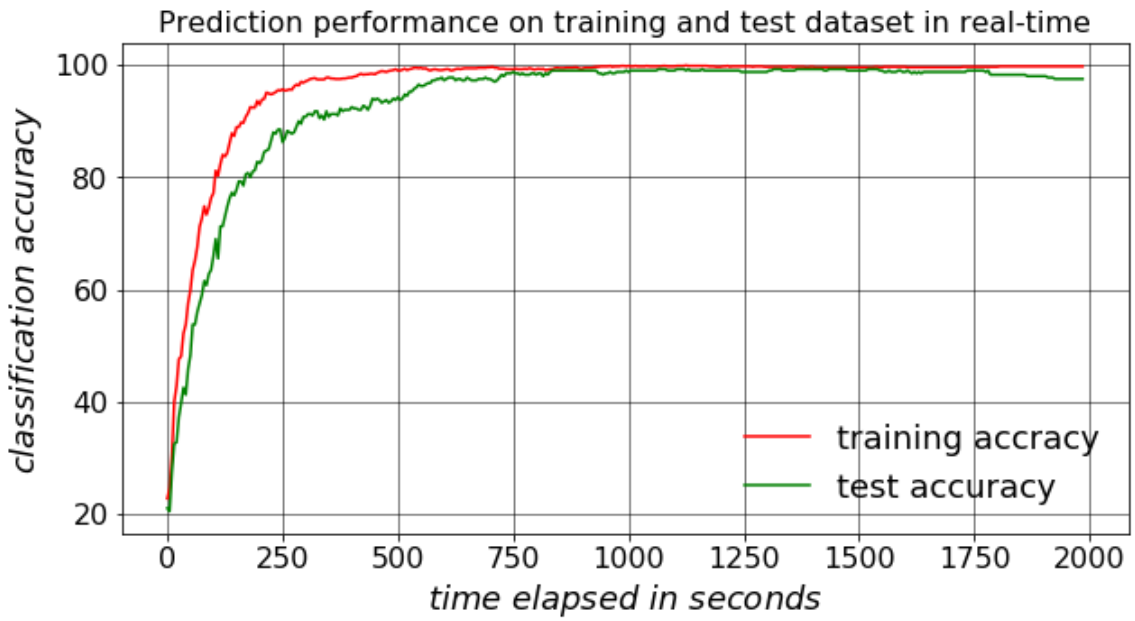



Figure 7. 5 Real-time pattern prediction accuracy

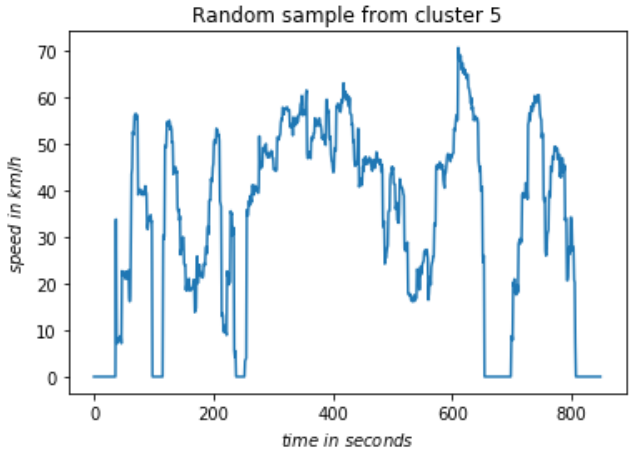
The training accuracy and test accuracy are close to each other. Hence, there is no overfitting in the machine learning model. After the first 5 seconds, the first ANN in the sequence is about 20% accurate in classifying the driving cycles. This is better than a random guess. The classification performance quickly reached about 80% after 160 seconds and 87% after 250 seconds. After 750 seconds of the trip, ANN is about 98% accurate in classifying the driving cycles in the test dataset. The peak classification accuracy of 99% is reached after 1000 seconds into the trip.

A comparison of the classification performance using ANN on the Victoria and California dataset are shown in table 7.1. Since the Driver-specific driving data has distinct clusters, the classification accuracy is much better compared to the CHTS dataset. After 250 seconds, ANN can classify any driving cycles in the driver-specific driving data with 87% accuracy. However, it is only 30% for the California driving data. Thus, the ANN is able to classify driving cycles in the Driver-specific driving data accurately much earlier in the trip.

Table 7. 1 Comparison of classification accuracy on California and Driver-specific driving data

| Time elapsed in seconds | Prediction accuracy (Victoria) | Prediction accuracy (California) |
|---|---|---|
| 250 | 87% | 30% |
| 500 | 93% | 60% |
| 750 | 98% | 72% |
| 1000 | 99% | 85% |
| 1500 | 99% | 89% |

The prediction performance of the trained ANNs on two randomly selected driving cycles from the Driver-specific driving data are shown in Figure 7.6. The predicted cluster history plot shows the history of labels predicted by the respective ANN after every 5 seconds in the trip. The predictions are very quick, accurate and stable. Thus the driver-specific Driver-specific driving data is a very good candidate for the proposed energy management strategy.



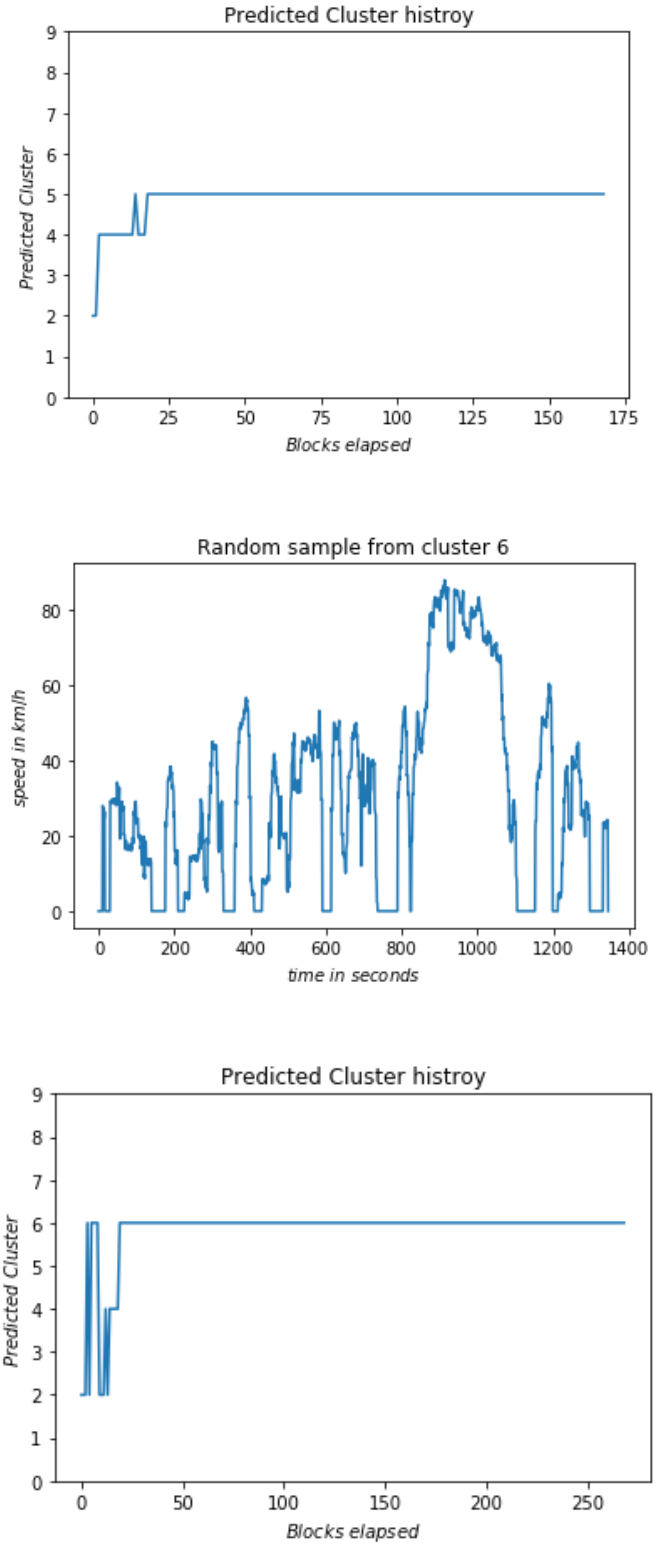



Figure 7. 6 Real-time prediction performance on two randomly selected driving cycles

# 8 Software Implementation

Python 3.6 was used primarily in this research work for implementing the pattern recognition and pattern prediction tasks for supporting optimal power control and energy management for PHEV's. Jupyter Notebook was used as the Interactive Development Environment (IDE).

## 8.1 Data Preprocessing

The various libraries used in the data preprocessing stage are shown in the screenshot in Figure 8.1.

```
import pandas as pd
import numpy as np
import glob
import os
import re
import matplotlib.pyplot as plt
import datetime
import warnings
```

Figure 8. 1 Various libraries used for preprocessing the data

The raw data was imported using pandas version 0.24 for preprocessing as a pandas data frame. Pandas is an open-source library for the Python programming language that proves high-performance, easy-to-use data structures, and data analysis tools. First five driving cycles from CHTS data in pandas data frame is shown in Figure 8.2.

| | time | speed | length |
|---|---|---|---|
| 0 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,... | [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.38697917336, ... | 629 |
| 1 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,... | [0.0, 1.26577356771, 2.02961276525, 2.95781213... | 612 |
| 2 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,... | [0.0, 2.1955357433, 3.80379462946, 5.563541552... | 1051 |
| 3 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,... | [0.0, 3.0858628392900003, 5.39709786086, 7.546... | 2822 |
| 4 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,... | [0.0, 1.23385379092, 2.18284910193, 3.45937426... | 952 |

Figure 8. 2 First five CHTS driving cycles in pandas data frame

Modules os, re and glob is used to find the required pathnames to the raw data. Matplotlib, a python 2-D plotting library is used for plotting all the figures in this research work.

For the CHTS data, the 24 hours long driving cycles were split into micro-trips at idle events greater than 300 seconds. Also, zero speed and respective timestamps were inserted at all missing locations as the original data represented an idling event with only two timestamps; first at the beginning of the Idling event and second at the end of the Idling event. A random raw driving cycle before splitting into micro-trips is shown in Figure 8.3. This driving cycle was split into 5 micro-trips at idling events greater than 300 seconds. The 5$^{th}$ micro-trip resulted from the driving cycle in Figure 8.3 is shown in Figure 8.4.

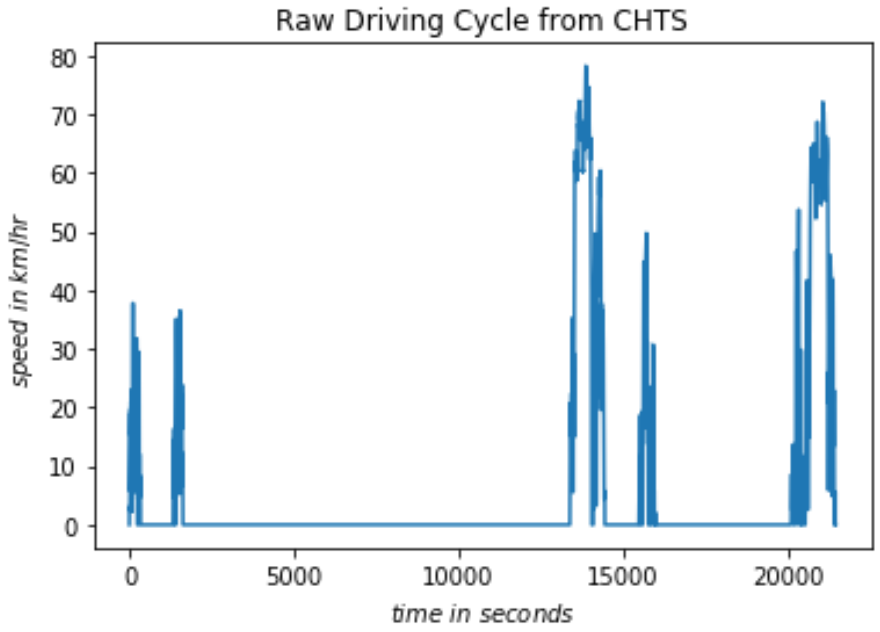



Figure 8. 3 Random raw driving cycle from CHTS data

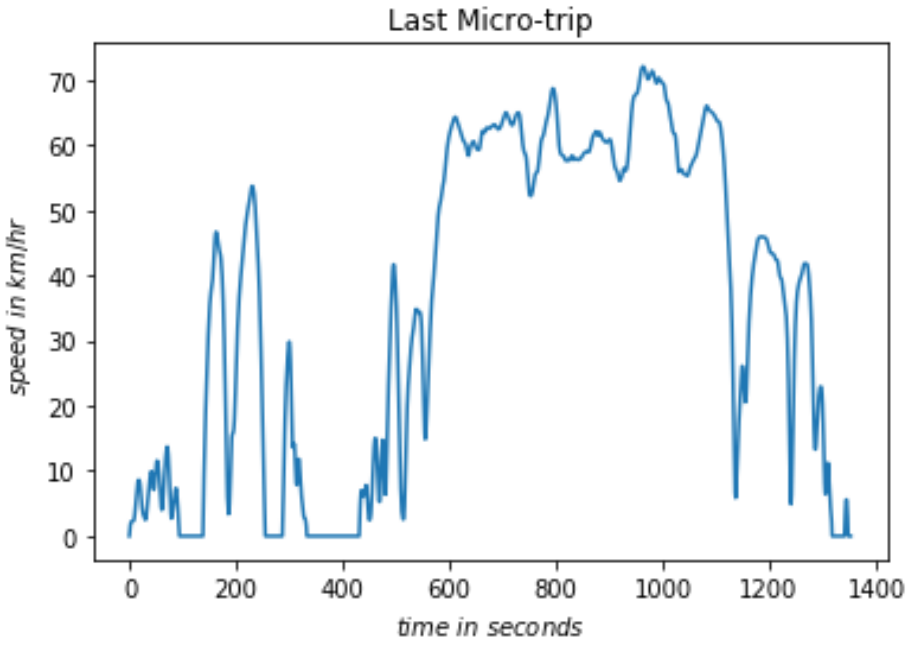



Figure 8. 4 Last of the five micro-trips extracted from the raw driving cycle in Figure 8.1

The quartiles and inter-quartile range of the trip length were calculated for removing the outliers using the built-in function in pandas. Code block to evaluate the quartiles of the trip length is shown in Figure 8.5.

```
Q1 = df1.length.quantile(0.25)
Q3 = df1.length.quantile(0.75)
IQR = Q3 - Q1
print(f" First Quartile: {Q1}, Third Quartile: {Q3}, \
      Inter-quartile Range: {IQR}")
```

```
First Quartile: 315.0, Third Quartile: 1172.0, Inter-quartile Range: 857.0
```

Figure 8. 5 Finding the quartiles for removing outliers

For the Driver-specific driving cycles, data augmentation was performed to artificially generate more data from the collected 30 driving cycles. Data was imported using pandas and functions were created for data augmentation. Firstly, random noise was created and added to the original driving cycle and then the resulted cycle was smoothed by scaling down infeasible acceleration and deceleration values. Figure 8.6 shows one of the 30 driving cycles collected. Figure 8.7 shows a random noise that was used to generate a

driving cycle from the driving cycle shown in Figure 8.6. Figure 8.8 shows the generated driving cycle.

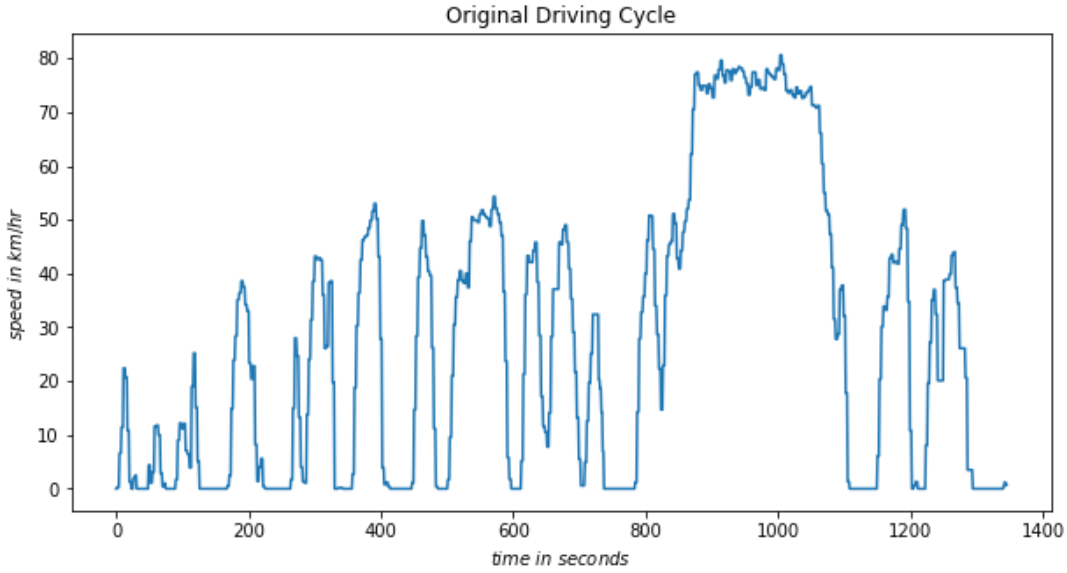



Figure 8. 6 One of the 30 collected Driver-specific driving cycles

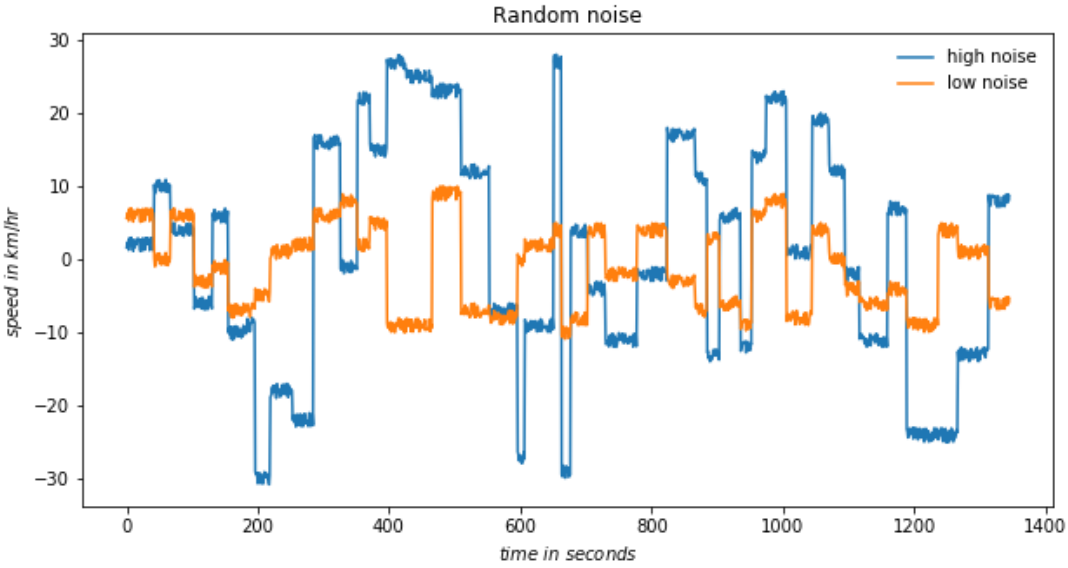



Figure 8. 7 Random noise generated for data augmentation of the driving cycle in Figure 8.6

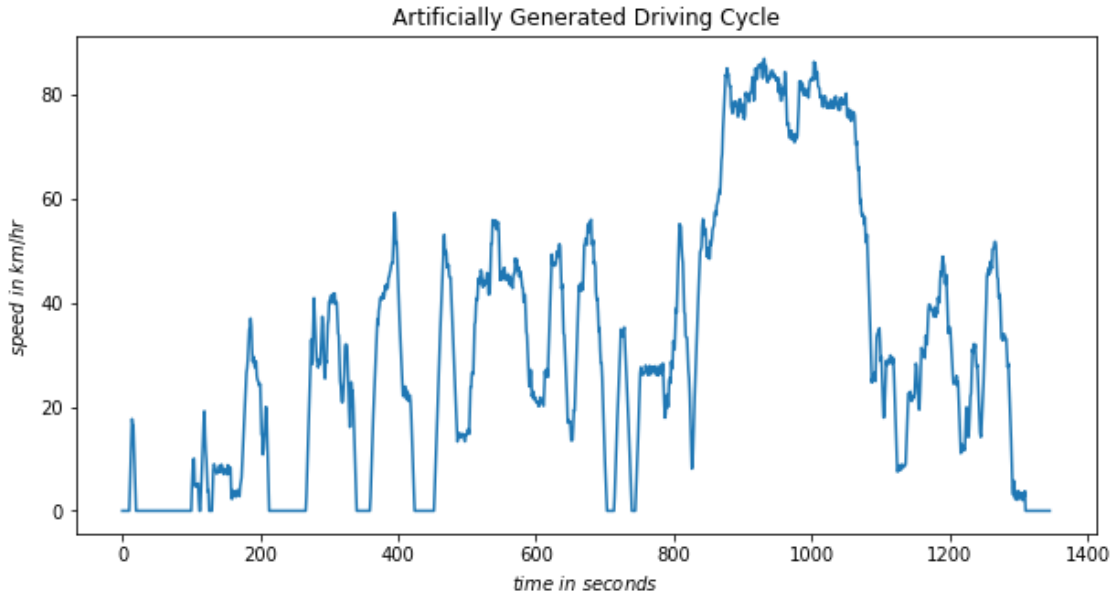



Figure 8. 8 Driving cycle generated from the cycle shown in Figure 8.6 using noise shown in Figure 8.7

The processed data frame was saved as a *.pkl* file (pickle file) using the function *pd.to_pickle("filename.pkl")*. Pickled files can be later imported in other script files by the function *pd.read_pickle("filename.pkl")*.

## 8.2 Data Abstraction

24 standard driving cycles from ADVISOR were selected and saved as *.csv* files using MATLAB. These standard driving cycles, along with 12 driving cycles selected from the CHTS data, were used to generate the library of cycle blocks. Apart from libraries used in the preprocessing, the following additional libraries shown in Figure 8.9 were used in data abstraction.

```
import numpy.polynomial.polynomial as poly
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import skfuzzy as fuzz
from sklearn.externals import joblib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.lines import Line2D
```

Figure 8. 9 Libraries used in data abstraction.

### 8.2.1 Feature Extraction

The driving cycles were divided into segments of 5 seconds. Then a quadratic polynomial was fitted on each of these segments to get the three polynomial coefficients for each of the segments. Numpy is the fundamental package for scientific computing in Python. The polynomial module in numpy was used to fit quadratic polynomial on the driving cycle segments. The function *polyfit()* was used to fit the polynomial and *polyval()* was used to generate curve from polynomial coefficients. An example of polynomial fitting and evaluating polynomial from the coefficients is shown in Figure 8.10 and Figure 8.11.

```
fig, axes = plt.subplots(1,2,figsize=(6,3))
axh = axes.reshape(-1)
sns.lineplot(df.iloc[i].time[26:31], df.iloc[i].speed[26:31], c="r", ax = axh[0])
axh[0].set_xlabel("$Time$ $in$ $Seconds$")
axh[0].set_ylabel("$Speed$ $in$ $kmph$")
axh[0].set_title("Speed Block 2")
t = list(range(0,5))
s = df.iloc[i].speed[26:31]
k = poly.polyfit(t, s, 2)
print(k)
ss=poly.polyval(t,k)
sns.lineplot(t, ss, c="r", ax = axh[1])
axh[1].set_title("Pattern Generated From Features")
axh[1].set_xlabel("$Time$ $in$ $Seconds$")
axh[1].set_ylabel("$Speed$ $in$ $kmph$")
plt.tight_layout()
```

Figure 8. 10 Code block that fits a quadratic polynomial on a driving cycle segment and generates speed from the polynomial coefficients.

```
[57.55714286  0.32571429 -0.17142857]
```

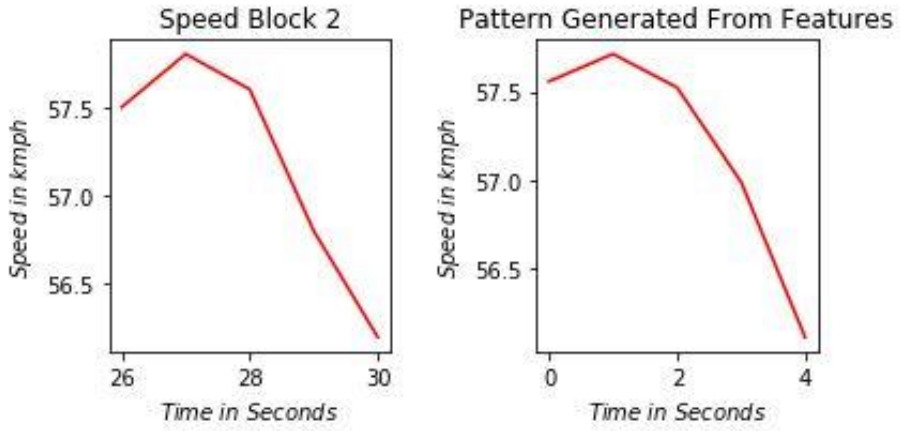



Figure 8. 11 Output from the code block in Figure 8.10

Feature extraction was performed for all the 7652 segments generated from the 36 standard driving cycles. This resulted in a datamatrix as shown in Figure 8.12. Fuzzy C-Means clustering algorithm was applied to this dataset to select 200 cycle blocks for driving cycle data abstraction.

Table 8. 1 Dataframe containing all 7652 cycle blocks from 30 standard cycles.

| | A0 | A1 | A2 |
|---|---|---|---|
| **0** | 0.000000 | 0.000000 | 0.000000 |
| **1** | 0.048571 | 0.042857 | 0.014286 |
| **2** | 1.368571 | 1.622857 | 1.614286 |
| **...** | ... | ... | ... |
| **7649** | 4.631429 | -0.582857 | 0.085714 |
| **7650** | 3.985714 | 0.358571 | -0.007143 |
| **7651** | 5.374286 | -0.158571 | -0.292857 |

7652 rows × 3 columns

The code block in Figure 8.13 shows FCM clustering of the 7652 cycle blocks into 200 clusters.

```
#Normalize the data using standardscaler
scaler = StandardScaler().fit(X)
x_scaled = scaler.transform(X)
#Apply Principal Component Analysis
pca = PCA().fit(x_scaled)
x_pca = pca.transform(x_scaled)

#Select the number of clusters "n"
n=200
#Perform FCM Clustering
centroids, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(x_pca.T, n, 2, error=0.005,\
                                                      maxiter=1000, init=None)
#Provide labels to each of the samples
labels = np.argmax(u, axis=0)
```

Figure 8. 12 Code block showing FCM Clustering applied on the 7652 cycle blocks.

Sci-kit learn is an open-source library built on NumPy, SciPy, and Matplotlib for simple and efficient data mining and data analysis. Sci-kit learn contains various modules for preprocessing, machine learning and other statistical applications. *StandardScaler()* function in the preprocessing module of sci-kit learn was used to normalize the data before applying FCM Clustering algorithm. *PCA()* function in the decomposition module of sci-kit learn was used for applying Principal Component Analysis (PCA).

FCM clustering algorithm was applied to the transformed data to cluster the 7652 cycle blocks into 200 clusters. SciKit-Fuzzy is a library containing a collection of fuzzy logic algorithms, written in Python. Cmeans is the Fuzzy C-Means clustering algorithm implemented in SciKit-Fuzzy. The input to the algorithm are data, a number of clusters, array exponentiation, stopping criteria error, maximum iterations, and initial fuzzy c-partitioned matrix. The function returns cluster centers, final fuzzy c-partitioned matrix, initial fuzzy c-partitioned matrix, final Euclidean distance matrix, objective function history, number of iterations run and final fuzzy partition coefficient. Here the samples are assigned a cluster having maximum membership value. The trained centroids can be used to predict the cluster of any unknown data. Module joblib was used to store the Python objects centroids, standard scaler and pca for later use. Python objects can be saved using joblib as shown below.

Save a Python object:

joblib.dump(Python Object, File Name)

Load a saved Python object:

Object = joblib.load(Python Object, File Name)

### 8.2.2 Block Representation

Driver-specific driving cycles and the driving cycles in the CHTS data were converted into block representation using the trained centroids in 8.2.1 as shown in Figure 8.13. Driving cycles were iteratively divided into segments of 5 seconds duration. All the segments from a single driving cycle were transformed by the StandardScaler and PCA in 8.2.1. The transformed data was then given to the function *cmeans_predict()* along with the trained centroids. Function outputs the final fuzzy c-partitioned matrix which contains the membership value of each of the segments to belong to each of the 200 cycle block clusters. Labels are assigned as per the maximum membership value. The sequence of labels then represents the block representation of the driving cycle.

```
#Applying Feature Extraction
df_new = extract_features(df_test)
Xtest=df_new1

# Normalize the data with the same scaler used for cycle block selection
x_scaled = scaler.transform(Xtest)
# Apply PCA with the same PCA used for cycle block selection
x_pca = pca.transform(x_scaled)
# Predict the cluster of input data using the trained centroids
#from cycle block selection
u, u0, d, jm, p, fpc = fuzz.cluster.cmeans_predict(x_pca.T, centroids,\
                                                   2, error=0.005, maxiter=1000)
predicted_labels = np.argmax(u, axis=0)
```

Figure 8. 13 Code block to convert the driving cycle into block representation.

## 8.3 Driving Cycle Pattern Recognition

Driving cycles in their block representation were imported for pattern recognition. TfidfVectorizer in the feature_extraction.text module in sci-kit learn was used for converting the categorical driving data into numerical data. As shown in Figure 8.14, TfidfVectorizer was fitted on *"main_cycle_text.txt*", which contains a sequence of cycle block labels of all the driving cycles in the dataset in text format. When this fitted vectorizer is used to transform a given driving cycle, it would count the number of occurrences of each cycle blocks scaled by their relative occurrence in the entire dataset.

```
with open(path1 + "main_cycle_text.txt") as input_file:
    cv = TfidfVectorizer(stop_words=[], ngram_range=(1, 1), \
                         max_features=50000, analyzer = "word").fit(input_file)
feature_names = cv.get_feature_names()
feature_len = len(feature_names)
```

Figure 8. 14 Code block showing TfidfVectorizer fitted onto all the driving cycles in the database.

In order to break symmetry, the TfidfVectorizer was applied separately on each half of the driving cycle and then normalized by the total number of cycle blocks in each half. Resulting features were concatenated horizontally. The Driver-specific driving cycles use 199 of the 200 available cycle blocks for block representation. On applying the vectorizer, this results in a number of features of 398. There are 1500 driving cycles in the driver-specific driving data collected from Victoria. Hence, for pattern recognition, the datamatrix has a size of 1500 rows by 398 columns.

Fuzzy C-Means clustering algorithm was applied to the driver-specific driving data. StandardScalar and PCA used to transform the data. First, three principal components were used to visualize the data. The transformed data then inputted to the cmeans algorithm in SciKit-Fuzzy. Clustering was done iteratively for a various number of clusters as shown in Figure 8.15. Respective fuzzy partition coefficient was plotted as shown in Figure 8.16 to obtain the optimum number of clusters, which was found to be 6 for Driver-specific driving cycles.

```
print("Trying various number of clusters...")
# Set up the loop and plot
fig1, axes1 = plt.subplots(4, 3, figsize=(8, 8))
fpcs = []
for ncenters, ax in enumerate(axes1.reshape(-1), 2):
    cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(x_pca.T,\
    ncenters, 2, error=0.005, maxiter=1000, init=None)
    fpcs.append(np.round(fpc,2))

#Plot FPC vs number of clusters
fig2, ax2 = plt.subplots(figsize=(10,5))
ax2.plot(np.r_[2:14], fpcs, marker="o", c="r")
ax2.set_xlabel("$Number$ $of$ $clusters$")
ax2.set_ylabel("$Fuzzy$ $partition$ $coefficient$")
ax2.axhline(xmin = 0, xmax = 0.4, y=np.max(fpcs), linestyle="--", \
            c="b", alpha=0.8)
ax2.axvline(ymin = 0, ymax = 0.99, x=6, linestyle="--", c="b", alpha=0.8)
ax2.set_title("Optimum Number of Clusters")
```

Figure 8. 15 Code block to find the optimum number of clusters

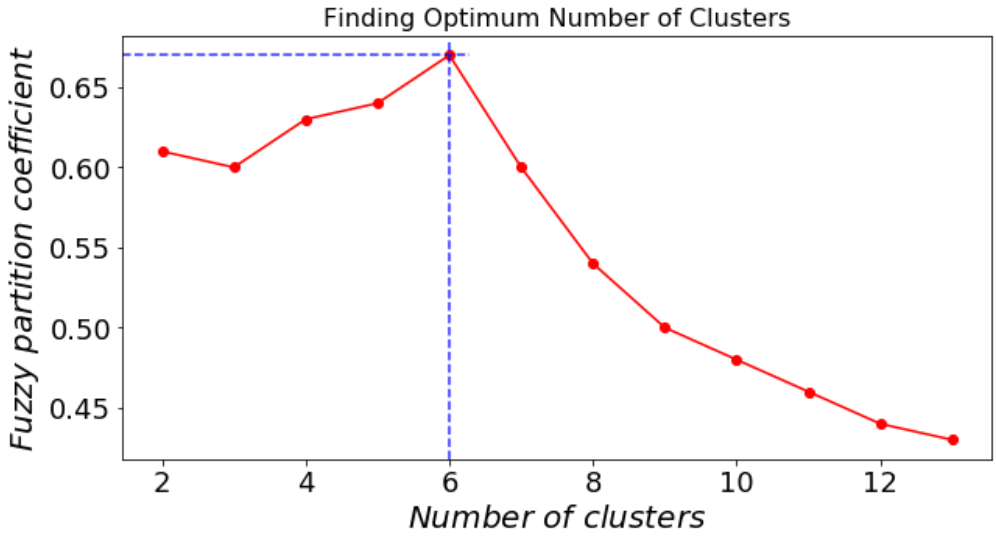



Figure 8. 16 Output from code in Figure 8.16 shows the optimum number of clusters for Driver-specific driving cycles

FCM clustering was applied to the Driver-specific driving cycles that grouped the data into 6 clusters. Every driving cycle in the dataset was then labeled as per the cluster number. The labeled dataset was saved for training an Artificial Neural Network that can predict the cluster of any given driving cycle in real-time. Also, the driving cycle with the highest

membership value from each of the clusters was selected to form representative driving cycles that represent the driver-specific unique driving patterns. The code block to select cluster center centers as representative driving cycles is shown in Figure 8.17. DP based off-line optimization is performed on these driving cycles.

```
# Create a membership_high column
# Assign highest membership value of each driving cycle
df2["membership_high"] = np.max(membership.T, axis=1)

#Find the highest membership value from each cluster
c0, c1, c2, c3, c4, c5 = df2.groupby("label").\
                            membership_high.max()
# Find the sample with highest membership from each cluster
dfc0 = df2[(df2.label==0) & (df2.membership_high==c0)]
dfc1 = df2[(df2.label==1) & (df2.membership_high==c1)]
dfc2 = df2[(df2.label==2) & (df2.membership_high==c2)]
dfc3 = df2[(df2.label==3) & (df2.membership_high==c3)]
dfc4 = df2[(df2.label==4) & (df2.membership_high==c4)]
dfc5 = df2[(df2.label==5) & (df2.membership_high==c5)]

# Save representative driving cycles to drive
file_path = path1+"cluster_centers\\"
df_clusters = [dfc0, dfc1, dfc2, dfc3, dfc4, dfc5]
for ind, df in enumerate(df_clusters):
    time = list(df.time.values)[0]
    speed = list(df.speed.values)[0]
    dfb = pd.DataFrame({"time": time, "speed": speed})
    dfb.to_excel(file_path + "driver_specific" + str(ind) + ".xlsx" )
```

Figure 8. 17 Finding cluster centers and saving to drive for off-line optimization.

### 8.4 Real-time Prediction

Single-layer feed-forward artificial neural networks were used to classify a given driving cycle into one of the identified representative driving cycles. The various modules in sci-kit learn was used to train and test the supervised machine learning model. Various modules used for implementing real-time pattern prediction are shown in Figure 8.18.

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.neural_network import MLPClassifier
import pdb
```

Figure 8. 18 Tools used for training and test ANNs.

MLPClassifier in the neural_network module of sci-kit learn was used to build the neural network model. Labeled driver-specific driving data from Victoria was imported and represented in the block representation. TfidfVectorizer was used to convert the categorical features to numerical for training the supervised model. Data was then split into training and cross-validation datasets, as shown in Figure 8.19, using the train_test_split function in the model_selection module in sci-kit learn.

```
# X is the dataframe containing block representation as a column
X = df.drop(columns=["label"], axis=1)
# Assign true labels to y
y = df.label
indices_main = list(range(0, df.shape[0]))
# Split dataset into training and cross-validation
df_train, df_valid, dfy_train, dfy_valid, train_ind_main, \
test_ind_main = train_test_split(X, y, indices_main, random_state=17)
```

Figure 8. 19 Code block to split the dataset into training and cross-validation datasets

For real-time energy management in PHEV's, the pattern prediction has to be made as early as possible after a trip begins. ANN will have access to data only in real-time. So it has to make a prediction based on what data is available to it in real-time. One ANN is not adequate for this task as the input data to the ANN is dynamic. Hence, a sequence of ANNs is trained to enable real-time pattern prediction.

Maximum trip length in driver-specific driving cycles is 1900 seconds. Hence, the maximum number of cycle blocks is 380. So a sequence of 380 ANNs were trained for real-time pattern prediction of Driver-specific driving cycles. First ANN is trained to predict using the information available only until 5 seconds into the trip. Second ANN is trained on data after 10 seconds of the trip. Likewise, $380^{th}$ ANN is allowed to use all the data available at the end of 1900 seconds.

A number of neurons in the hidden layer were tuned using GridSearchCV function in sci-kit learn. The improvement in prediction accuracy plateaued after 100 neurons in the hidden layer. Regularization parameter of 7 was used to prevent overfitting. Default values were used for all other parameters of the MLPClassifier. All the trained ANNs were stored in a Python dictionary. The sequential training of 380 ANNs is shown in Figure 8.20.

```
# Iterate through maximum number of cycle blocks
for block in range(1, block_limit):
    if block > 1:
        add_extra_blocks = True

    if add_extra_blocks:
        # Use all the cycle blocks until the current number of cycle
        # blocks elapsed, perform TfidfVectorizer.
        data1 = get_data(df_train, block)
        # Use first half of the cycle blocks until the current number of cycle
        # blocks elapsed, perform TfidfVectorizer.
        data2 = get_data(df_train, int(block/2))
        # concatenate both features
        data = data1.merge(data2, how="outer",on=data1.index).drop("key_0", axis=1)
    else:
        data = get_data(df_train, block)

    X = data
    # Train MLPClassifier
    clf = MLPClassifier(100, random_state=17, max_iter = 100, alpha=7)
    clf.fit(X, y)
    # Predict on training dataset
    y_pred = clf.predict(X)
    # Estimate and save training accuracy
    acc = np.round(accuracy_score(y, y_pred),4)*100
    accuracy.append(acc)
    # Add trained classifier to dictionary
    trained_clf[block] = clf
```

Figure 8. 20 Code block to sequentially train ANNs

The trained ANNs were tested on the cross-validation dataset iteratively. In the first iteration, only the first cycle block of each of the driving cycles in the cross-validation dataset was given to the first trained classifier in the trained classifier dictionary. The classifier then makes a prediction and the predicted value was compared with the true label and test accuracy for the first classifier was established. Similarly, all the classifiers were tested using the respective cycle blocks in the cross-validation dataset and the real-time prediction performance was plotted.

The function was created to demonstrate real-time pattern prediction. It takes in a driving cycle in its speed-time form and returns predicted label at every 5 seconds. In the first iteration, the function takes the first 5 speed data, convert it into block representation using the FCM trained centroids. Then it uses the first trained classifier and makes a prediction. In the second iteration, the function uses speed data from the first 10 seconds and makes a prediction using the second trained classifier. A plot of the sequence of predictions made by this function on one of the random driving cycles from the cross-validation data is shown in Figure 8.22.

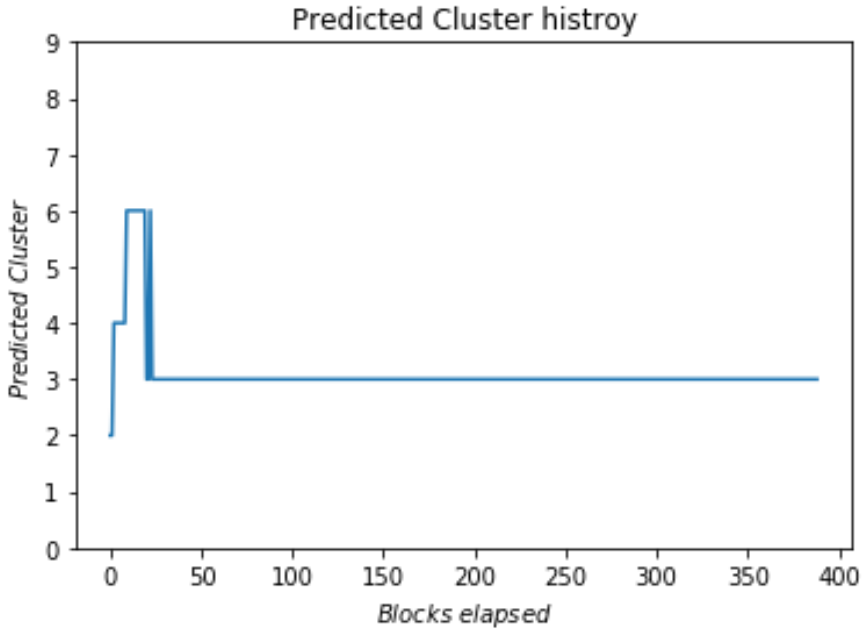



Figure 8. 21 Real-time predicted labels for a random driver-specific driving cycle

# 9 Test Results of the Driver Data Driven Optimal PHEV Energy Management

In the previous chapters, the introduced data clustering and pattern recognition methods have been tested using the acquired vehicle operation data from California and from Victoria. To apply these new intelligent system methods into real tests on driver adaptive intelligent energy management of PHEVs is beyond the scope of this research. However, to better demonstrate the benefits of the newly introduced methods, simulation results from driver adaptive intelligent energy management are included in this chapter. This test using the results from the introduced intelligent system has been conducted by Dr. Yanbiao Feng, a post-doctoral fellow at the University of Victoria. The tests were made using the MATLAB/Simulink models of the UVic EcoCAR2 that was retrofitted from a 2013 Chevrolet Malibu production vehicle, which was converted by replacing its original powertrain with a 4WD series-parallel multiple-regime PHEV as shown in Figure 9.1.

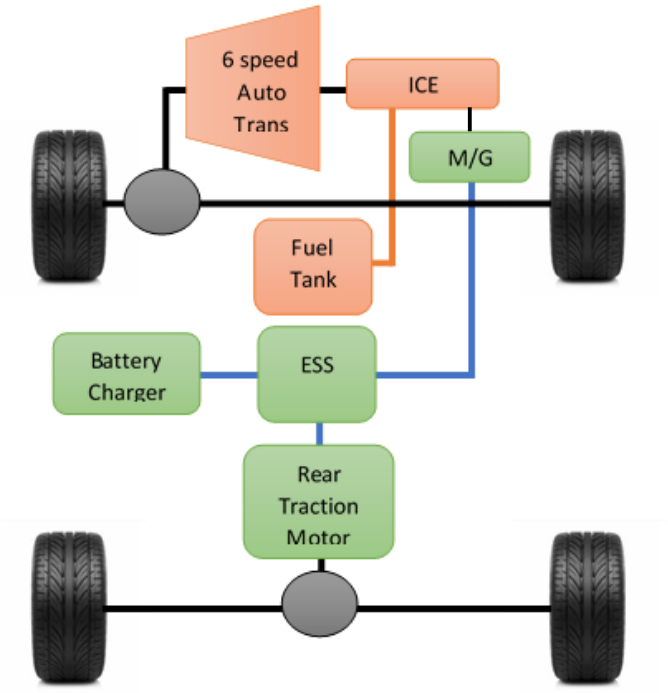



Figure 9. 1 Powertrain architecture of UVic EcoCAR2

UVic EcoCAR2 was modeled in MATLAB Simulink as per [35]. The objective of the test is to control the powertrain control and energy management parameters so that the vehicle

follows the required driving cycle while minimizing fuel consumption, emissions and maintaining battery SOC. The powertrain control parameters are the torque output from the ICE and the two electric motors. The energy management control parameters are the battery SOC and the rate of degradation of battery SOC. The test is conducted with two different powertrain control and energy management strategies. Firstly, a rule-based controller given in [35] was used on the UVic EcoCAR2 on each of the 6 representative driving cycles from the driver-specif driving data from Victoria to get the baseline performance. Then Dynamic Programming (DP) was applied to obtain the global optimal powertrain control and energy management parameters. The results from the simulation of the rule-based and DP based controller on the Driver-specific driving cycles are shown in Table 9.1. It shows the theoretically maximum possible improvement that the intelligent controller can achieve over the rule-based controller. The results confirm that there is a scope for improving the fuel consumption by about 30% on an average on the driver-specific data collected from Victoria. This improvement can be achieved by applying the intelligent powertrain control and energy management strategy.

Table 9. 1 Projected improvements in fuel consumption

| | Initial SOC | RULE-BASED CONTROLLER | | DP OFF-LINE CONTROLLER | | Improvement in Fuel Consumption (%) |
|---|---|---|---|---|---|---|
| | | Fuel Consumption (L/100km) | Final SOC | Fuel Consumption (L/100km) | Final SOC | |
| **Victoria Cycle 1** | 0.30 | 14.07 | 0.29 | 9.55 | 0.30 | 32.08 |
| **Victoria Cycle 2** | 0.30 | 18.53 | 0.31 | 12.54 | 0.30 | 32.32 |
| **Victoria Cycle 3** | 0.30 | 13.20 | 0.27 | 9.56 | 0.30 | 27.63 |
| **Victoria Cycle 4** | 0.30 | 18.99 | 0.31 | 13.95 | 0.30 | 26.52 |
| **Victoria Cycle 5** | 0.30 | 16.76 | 0.30 | 11.52 | 0.30 | 31.26 |
| **Victoria Cycle 6** | 0.30 | 16.80 | 0.30 | 12.36 | 0.30 | 26.44 |

Improvement in fuel consumption by the intelligent controller depends on two aspects. Firstly, how fast the trained neural networks can associate the test driving cycle to the correct representative driving cycle. As illustrated in Chapter 7, the prediction performance of neural networks for Driver-specific driving cycles is about 80% in the first 160 seconds of the trip. So optimal control parameters for energy management can be selected within the first 160 seconds at 80% accuracy. Secondly, the improvement in fuel consumption by

the intelligent controller depends on the similarity in patterns in the test driving cycle and the respective representative driving cycle. Fuzzy partition coefficient (FPC) is used as the measure of cluster quality. As discussed earlier, FPC can take a value between 1 and $1/C$ where $C$ is the number of clusters that the data is partitioned into. For the driver-specific driving data collected from Victoria, the fuzzy partition coefficient at optimum partition is found to be 0.67. Respective minimum possible FPC value is 0.17. Hence, the partition is more towards a hard partition than a total fuzzy partition. This shows the high cluster quality. The same is confirmed in Figure 7.3 which shows 6 distinct clusters in the driver-specific data. The samples within the same cluster are closely packed whereas the different clusters are well separated. This shows that the driving cycles within the same cluster have similar patterns that are different from that of driving cycles from other clusters. This is validated by visually inspecting random driving cycles from each of the 6 clusters, as shown in Annexure B4. Hence, the intelligent energy management strategy has a high potential to reach the global optimal solution on the driver-specific driving cycles.

However, for the California driving data, the clusters are not well separated. FPC value is found to be 0.45 and the respective minimum possible FPC value is 0.33. This shows the low clustering quality and the lack of clustering tendency in the dataset. So the driving cycles in the same cluster may not be very similar to apply the same optimal powertrain controller and energy management strategy. Also, the neural networks are comparatively slower in predicting the correct representative driving cycle. Hence, intelligent powertrain control and energy management strategy may not be suitable for CHTS dataset. For best results, the intelligent controller has to be implemented on driver-specific data. This enables the intelligent controller to adapt to the driving behavior of an individual driver. Thus each driver will have an energy management strategy that is custom-designed to suit his/ her driving behavior.

Even for dataset similar to CHTS data, the proposed energy management can be modified to suit overlapping clusters. One option is to use the Fuzzy Partition Coefficient to identify lack of well-separated clusters and use ECMS for energy management. Here, instead of estimating the global optimal control parameters using DP, optimal equivalence factor can be evaluated for each of the representative driving cycles. For real-time implementation, the respective optimal equivalence factor shall be used.

If the resultant clusters are not well separated and the trained neural networks fail to predict the pattern, energy management strategy shall be switched from the intelligent controller to a standby controller such as a rule-based controller. This decision shall be taken automatically by the onboard computer in the vehicle based on a flag that represents if the intelligent controller shall be effective. This flag shall be triggered at the cloud computing platform if the FPC is above a threshold and if the neural networks are able to predict the correct pattern within a predefined time and with acceptable accuracy.

# 10 Conclusion and Future Work

An intelligent, adaptive, driver-specific energy management strategy for power control in a hybrid electric vehicle is introduced in this research work. The intelligent controller makes use of advanced machine learning algorithms, global optimization algorithms and the power of cloud computing platforms and cloud-based storage systems. As per the intelligent controller, each driver will have his/ her own custom controller that adapts to the driver's change in driving behavior and road conditions.

Data from a single vehicle are collected and stored in the onboard computer. At pre-defined intervals, the collected data is uploaded onto a cloud-based storage and computing platform, where an unsupervised machine learning model, a supervised machine learning model, and a global optimization algorithm are deployed. Firstly, the FCM algorithm is applied to the collected data to get representative driving cycles. Then DP based optimization is done for each of the identified representative driving cycles to get global optimal control parameters for energy management in HEV. Lastly, a sequence of neural networks is trained that could classify any trip in real-time to one of the identified representative driving cycles. The trained machine learning model, the representative driving cycles and their respective optimal control parameters are downloaded to the onboard computer in the vehicle for real-time implementation of the proposed intelligent controller. As soon as a trip begins, the trained neural networks associate the current trip to one of the previously identified representatives driving cycles, for which the optimal control parameters are already available. Based on the predicted representative driving cycle, the respective optimal control parameters are selected for energy management. As the trip progresses, the trained neural networks receive more data and they make a more informed prediction and updates the predicted representative driving cycle in real-time. Accordingly, the control parameters are also selected.

Data collected from a single driver revealed distinct patterns by applying FCM clustering algorithm. Also, ANNs were successfully trained to predict the patterns in real-time to support optimal energy management. Thus the proposed strategy can be very effective when implemented driver-specific.

For the CHTS data, due to the lack of distinct clusters, the proposed intelligent controller may not be suitable with a global optimization technique. The proposed intelligent controller shall be made more robust to accommodate data that has overlapping clusters. One of the possible solution using ECMS is briefly mentioned in Chapter 9. Another solution is to use the membership information of the FCM algorithm. Since the FCM algorithm is very robust in handling overlapping data, a threshold on membership value may be introduced while clustering. All the samples with membership values below the threshold may be clustered together as outliers. During real-time implementation, if the neural networks predict the current trip as an outlier, instead of using the intelligent controller, the energy management strategy may be rolled back to a rule-based controller.

In data abstraction, a block representation of the driving cycle is introduced. This enables the driving cycles to be represented as a sequence of strings. In future work, driving cycles in this sequence of string representation may be used to train a Recurrent Neural Network (RNN) that could predict the patterns in real-time based on past driving patterns. This information may be used to improve the energy management strategy proposed in [18].

# References


[1] Jouzal, J, V. Masson-Delmotte, O. Cattani, G. Dreyfus, S. Falourd, G. Hoffmann, B. Minster, J. Nouet, J. M. Barnola, J. Chappellaz, H. Fischer, J. C. Gallet, S. Johnsen, M. Leuenberger, L. Loulergue, D. Luethi, H. Oerter, F. Parrenin, G. Raisbeck, D. Raynaud, A. Schilt, J. Schwander, E. Selmo, R. Souchez, R. Spahni, B. Stauffer, J. P. Steffensen, B. Stenni, T. F. Stocker, J. L. Tison, M. Werner and E. W. Wolff, "Orbital and Millennial Antarctic Climate Variability over the Past 800,000 Years," *Science,* vol. 317, no. 5839, pp. 793-797, 2007.

[2] Petit, J. R, J. Jouzel, D. Raynaud, N. I. Barkov, J. M. Barnola, I. Basile, M. Bender, J. Chappellaz, J. Davis, G. Delaygue, M. Delmotte, V. M. Kotlyakov, M. Legrand, V. Lipenkov, C. Lorius, L. Pépin, C. Ritz, E. Saltzman and M. Stievenard, "Climate and atmospheric history of the past 420,000 years from the Vostok Ice Core, Antarctica," *Nature,* vol. 399, pp. 429-436, 1999.

[3] Organisation Internationale des Constructeurs d'Automobiles, "OICA," 2019. [Online]. Available: http://www.oica.net/man-made-co2-emissions/. [Accessed March 2019].

[4] Organisation Internationale des Constructeurs d'Automobiles, "OICA," 2019. [Online]. Available: http://www.oica.net/category/production-statistics/2018-statistics/. [Accessed March 2019].

[5] B. Baumann, G. Rizzoni and G. Washington, "Mechatronic design and control of hybrid electric vehicles," *IEEE/ASME Transactions on Mechatronics,* 2000.

[6] G. Rizzoni and S. Onori, "Energy Management of Hybrid Electric Vehicles: 15 Years of Development at the Ohio State University," *Oil Gas Sci. Technol* , vol. 70, no. 1, pp. 41-54, 2015.

[7] National Renewable Energy Laboratory, "Transportation Secure Data Center," 2015. [Online]. Available: www.nrel.gov/tsdc. [Accessed December 2018].

[8] A. Panday and H. O. Bansal, "A Review of Optimal Energy Management Strategies for," *International Journal of Vehicular Technology,* vol. 2014, no. Article ID 160510, 2014.

[9] A. Brahma , Y. Guezennec and G. Rizzoni, "Dynamic optimization of mechanical/electrical power flow in parallel hybrid electric vehicles," in *5th International Symposium on Advanced Vehicle Control,* Ann Arbor, MI, 2000.

[10] R. Wang and S. M. Lukic, "Dynamic programming technique in hybrid electric vehicle optimization," in *2012 IEEE International Electric Vehicle Conference,* Greenville, SC, 2012.

[11] H. Lee, C. Kang, Y. -I. Park and S. W. Cha, "Study on Power Management Strategy of HEV using Dynamic Programming," *World Electr. Veh. J.,* vol. 8, pp. 274-280, 2016.

[12] W. Ximing, H. Hongwen, S. Fengchun and Z. Jieli, "Application Study on the Dynamic Programming Algorithm for Energy Management of Plug-in Hybrid Electric Vehicles," *Energies, MDPI, Open Access Journal,* vol. 8(4), pp. 1-20, 2015.

[13] G. Paganelli, Conception et commande d'une chaıˆne de traction pour ve´hicule hybride paralle`l ether-mique et e´ lectr, Valenciennes, France: PhD Thesis, Universite´ de Valenciennes, 1999.

[14] C. Musardo, G. Rizzoni, Y. Guezennec and B. Staccia, "A-ECMS: An Adaptive Algorithm for Hybrid Electric Vehicle Energy Management," *European Journal of Control - EUR J CONTROL,* vol. 10, no. 3166, pp. 509-524, 2005.

[15] G. Rizzoni and S. Onori, "Energy Management of Hybrid Electric Vehicles: 15 Years of Developmentat the Ohio State University," *Oil Gas Sci. Technol ,* vol. 70, no. 1, pp. 41-54, 2015.

[16] B. Gu and G. Rizzoni, "An adaptive algorithm for hybrid electric vehicle energy management based on driving pattern recognition," in *Proceedings of the ASME International Mechanical Engineering Congress and Exposition*, 2006.

[17] J. Wang, Y. Huang, H. Xie and G. Tian, "Driving Pattern Recognition and Energy Management for Extended Range Electric Bus," in *2014 IEEE Vehicle Power and Propulsion Conference (VPPC)*, Coimbra, 2014.

[18] H. Hongwen, J. Zhang and G. Li, "Model Predictive Control for Energy Management of a Plug-in Hybrid Electric Bus," *Energy Procedia,* vol. 88, pp. 901-907, 2016.

[19] A. K. Jain and R. C. Dubes, Algorithms for data clustering, Prentice Hall.

[20] A. Fotouhi and M. Montazeri-Gh, "Tehran driving cycle development using the k-means clustering," *Scientia Iranica,* vol. 20, no. 2, pp. 286-293, 2013.

[21] J. C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters," *Journal of Cybernetics,* vol. 3, no. 3, pp. 32-57, 1973.

[22] C. Zeynel and Y. Figen, "Comparison of K-Means and Fuzzy C-Means Algorithms on Different Cluster Structures," *Journal of Agricultural Informatics,* vol. 6, no. 3, pp. 13-23, 2015.

[23] Z. Dai, D. Niemeier and D. Eisinger, "Driving cycles: a new cycle-building method that better represents real-world emissions," 2008.

[24] K. B. Wipke, M. R. Cuddy and S. D. Burch, "ADVISOR 2.1: a user-friendly advanced powertrain simulation using a combined backward/forward approach," *IEEE Transactions on Vehicular Technology,* vol. 48, no. 6, pp. 1751-1761, 1999.

[25] L. A. Zadeh, "Fuzzy sets," *Information and Control,* vol. 8, no. 3, pp. 338-353, 1965.

[26] C. J. Bezdek, R. Ehrlich and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Computers & Geosciences,* vol. 10, no. 2-3, pp. 191-203, 1984.

[27] M. Wes, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, 2010.

[28] E. Travis E and Oliphant, A guide to NumPy, USA: Trelgol Publishing, 2006.

[29] v. d. W. Stéfan, C. S. Chris and V. Gaël , "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in Science & Engineering,* vol. 13, pp. 22-30, 2011.

[30] P. Fabian , V. Gaël , G. Alexandre, M. Vincent, T. Bertrand , G. Olivier , B. Mathieu , P. Peter , W. Ron , D. Vincent , V. Jake , P. Alexandre , C. David , B. Matthieu , P. Matthieu and D. Édouard , "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[31] scikit-fuzzy development team, "skfuzzy 0.2 docs; skfuzzy v0.2 docs," 2012. [Online]. Available: https://pythonhosted.org/scikit-fuzzy/. [Accessed August 2018].

[32] E. A. Zanty, "Determining the number of clusters for kernelized fuzzy C-means algorithms for automatic medical image segmentation," *Egyptian Informatics Journal,* vol. 13, pp. 39-58, 2012.

[33] E. Trauwaert, "On the meaning of Dunn's partition coefficient for fuzzy clusters," *Elsevier B.V.,* vol. 25, no. 2, pp. 217-242, 1988.

[34] P. Fabian, V. Gaël , G. Alexandre , M. Vincent , T. Bertrand , G. Olivier , B. Mathieu , P. Peter , W. Ron , D. Vincent , V. Jake , P. Alexandre , C. David , B. Matthieu , P. Matthieu and D. Édouard , "Scikit-learn: Machine Learning in Python," *JMLR,* vol. 12, pp. 2825-2830, 2011.

[35] R. Cheng, "Modeling and Simulation of Plug-in Hybrid Electric Powertrain System for Different Vehicular Applications," Masters Thesis, University of Victoria, Vcitoria, 2016.

# Appendix A: Program User's Manual

## A1 Major Software Packages Used

### Python

A python is an object-oriented, high level, interpreted, open-source programming language.
Documentation: *https://docs.python.org/3/*

### Pandas

pandas is an open-source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Documentation: *http://pandas.pydata.org/pandas-docs/stable/*

### Numpy

NumPy is the fundamental package for scientific computing with Python.

Documentation: *https://www.numpy.org/devdocs/*

### SciKit-Fuzzy

Scikit-Fuzzy is a collection of fuzzy logic algorithms intended for use in the SciPy Stack, written in the Python computing language.

Documentation: https://pythonhosted.org/scikit-fuzzy/api/api.html

### Sci-kit Learn

Sci-kit learn is an open-source Python library for data preprocessing, data analytics and machine learning.

Documentation: https://scikit-learn.org/stable/documentation.html

### Matplotlib

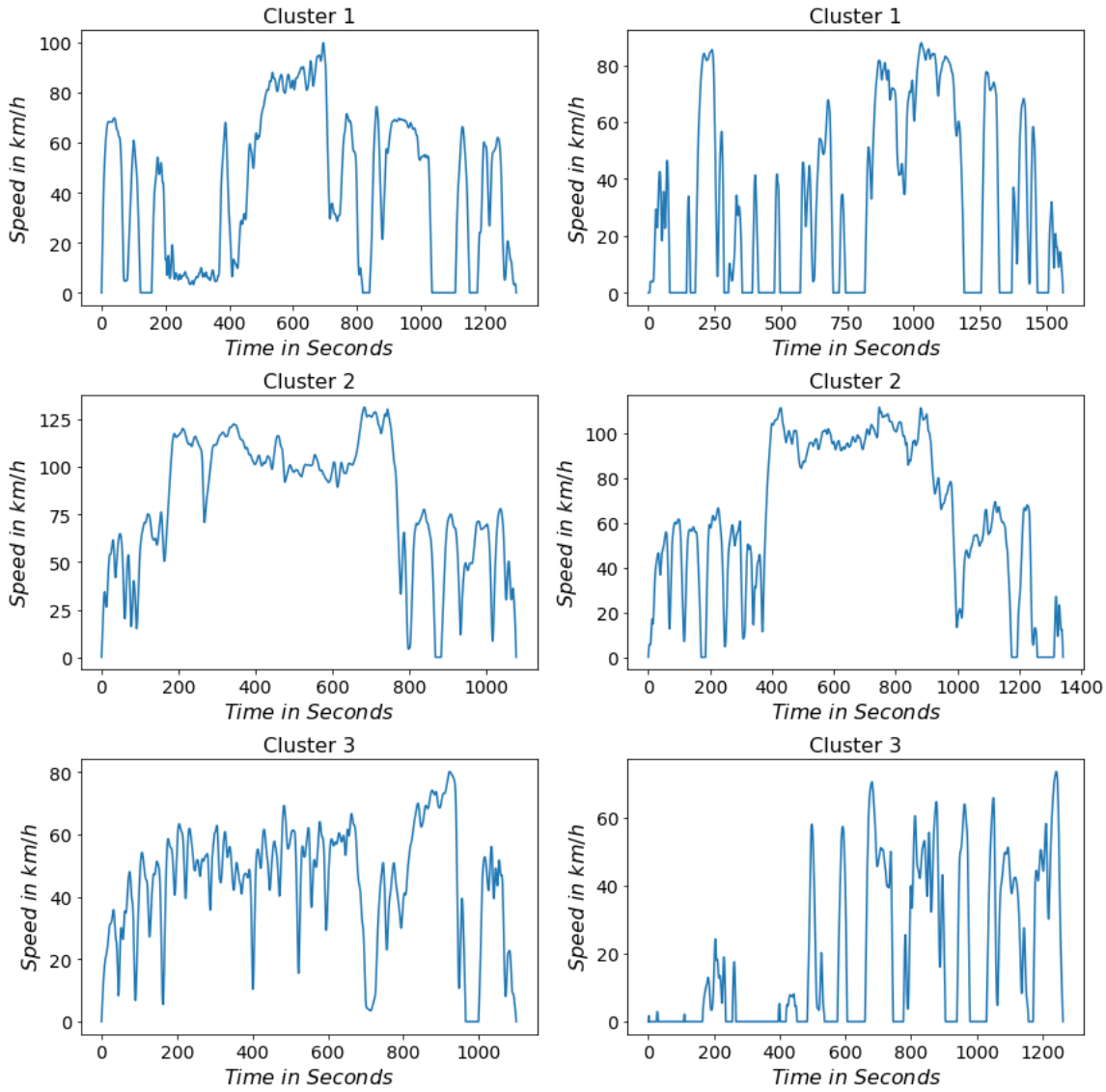
Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Documentation: *https://matplotlib.org/users/index.html*

# Appendix B: Example Driving Cycles

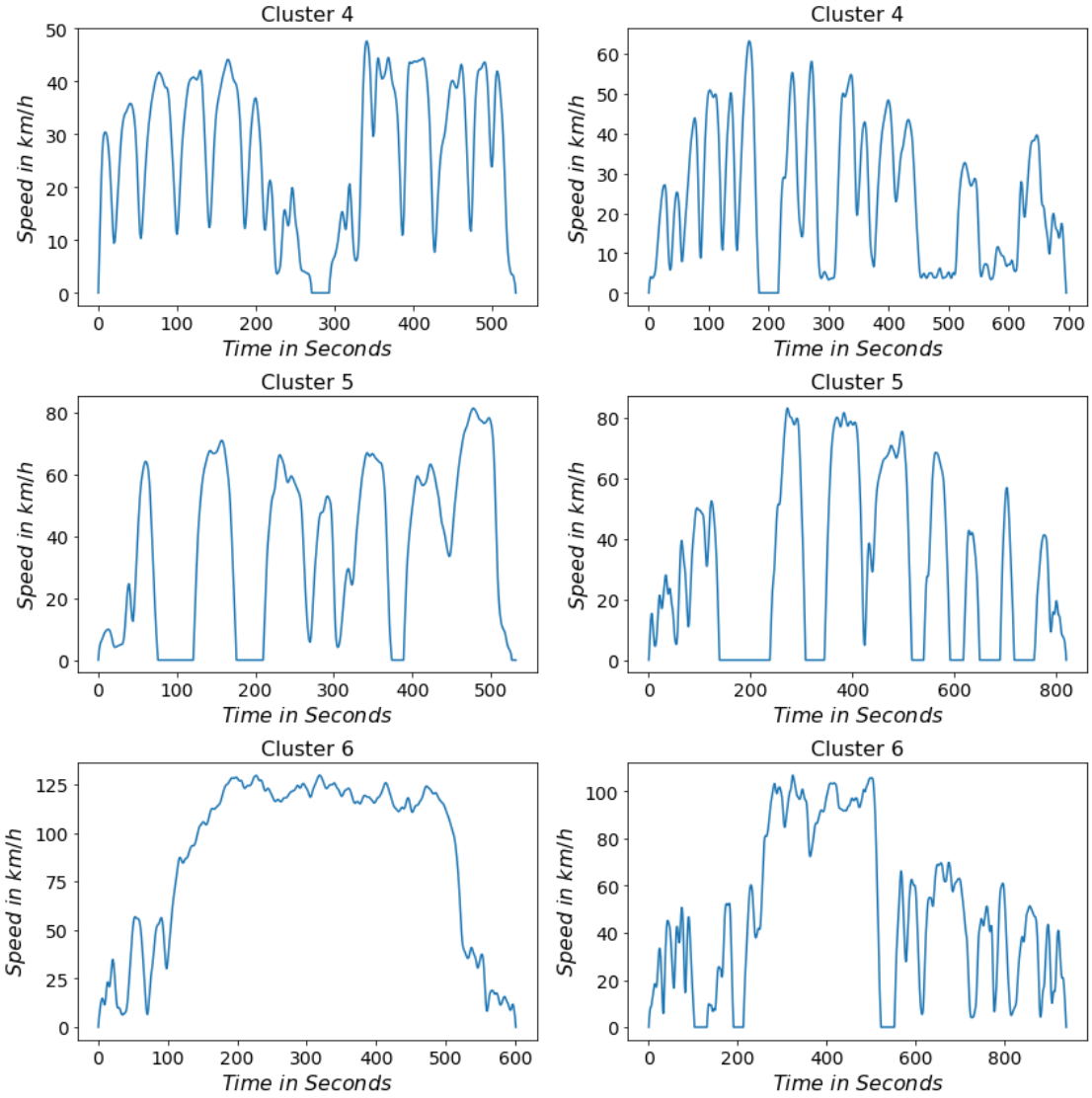
## B1 Long Trip Driving Cycles

Randomly selected 2 driving cycles from each of the 3 clusters in California long trip driving cycles.



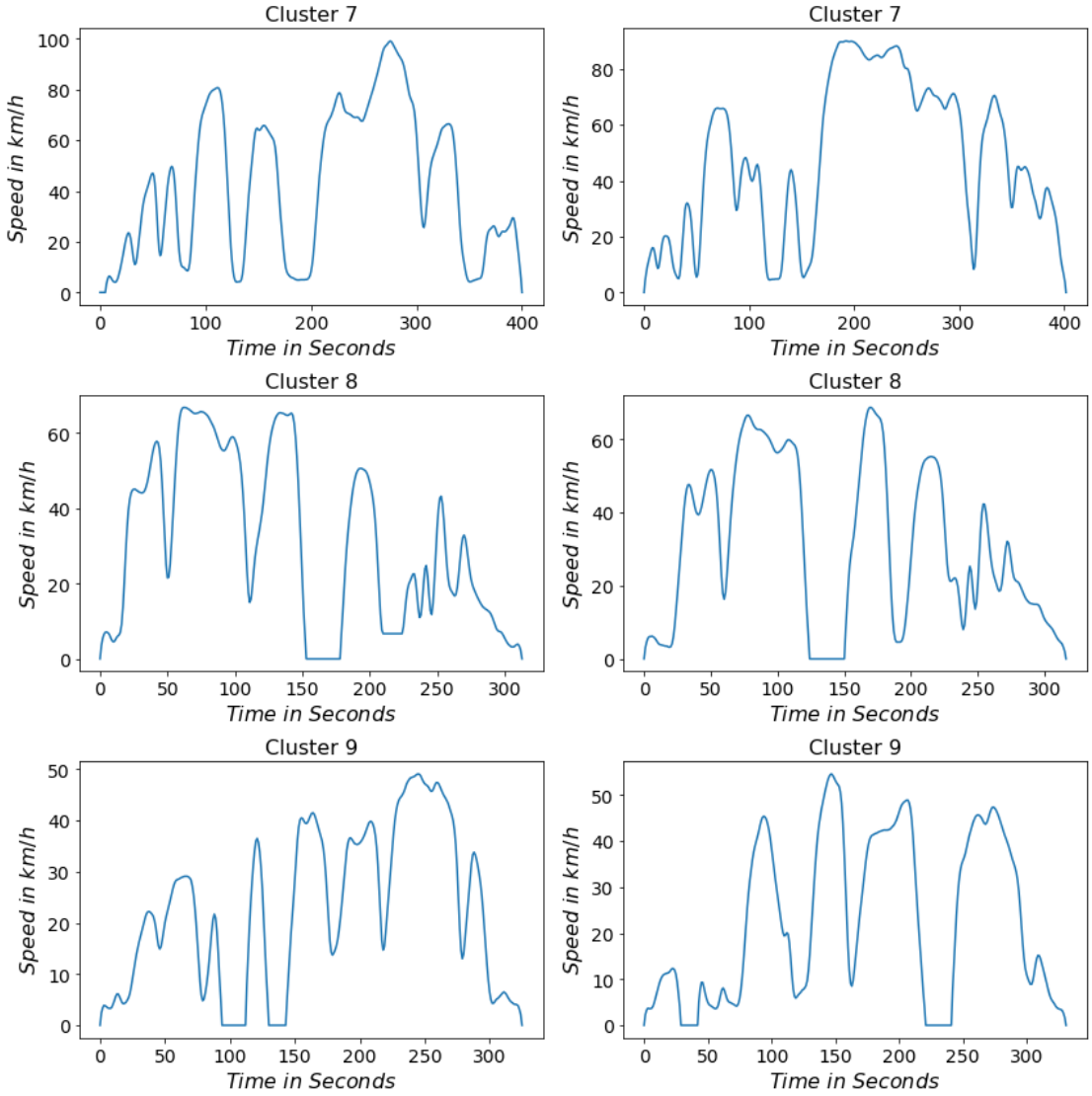
## B2 Medium Trip Driving Cycles

Randomly selected 2 driving cycles from each of the 3 clusters in California medium trip driving cycles.



## B3 Short Trip Driving Cycles

Randomly selected 2 driving cycles from each of the 3 clusters in California short trip driving cycles.



## B4 Driver-specific driving cycles

Randomly selected 2 driving cycles from each of the 6 clusters in Driver-specific driving cycles.