Machine Learning for Identification and Optimal Control of Advanced Automotive Engines

by

Vijay Manikandan Janakiraman

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2013

Doctoral Committee:

Professor Dionissios N. Assanis, Co-Chair
Professor Long Nguyen, Co-Chair
Professor Jeffrey L. Stein, Co-Chair
Professor IIya V. Kolmanovsky
Associate Research Scientist Stani V. Bohac

सहजं कर्म कौन्तेय सदोषमपि न त्यजेत् ।
सर्वारम्भा हि दोषेण धूमेनाग्निरिवावृताः ॥

Every endeavor is covered with defects or problems, just like fire is covered by smoke. One should not give up his/her duties even if such work is full of defects or problems .

-Quote from Bhagavad Gita

This thesis is dedicated to my parents, Janakiraman and Dhanakotti, my brother, Shanmugha Karthik and my fiancée, Priyadharshini

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my parents, Janakiraman and Dhanakotti, without whose blessings and love, I would not have traveled so far.

The research presented in this document would not be possible without the guidance and support of my advisors Prof. Dennis Assanis and Prof. XuanLong Nguyen. It was Prof. Assanis who instilled in me, the confidence to perform independent research. Prof. Assanis has a unique ability to "see the big picture", connect concepts across multiple domains, and focus on solving high impact problems. I would like to thank him for the motivation, freedom and direction that he provided me throughout the course of my doctoral studies.

I am extremely thankful to Prof. XuanLong Nguyen for the enormous technical guidance that I received, particularly in machine learning. The numerous discussions with Prof. Nguyen helped me appreciate the concepts behind machine learning more clearly and also identify a deep interest in the subject. The advice and support that I received from him have been of catalytic importance for the shaping of my current endeavors in this field.

I would also like to thank the other committee members, Prof. Jeffrey Stein, Prof. Ilya Kolmanovsky and Dr. Stani Bohac for their critical feedback and valuable comments. Their insightful advice has been crucial towards creating a solid foundation for the work presented in this dissertation.

I would like to acknowledge the financial support provided by Robert Bosch Inc. I would like to thank, in particular, Alan Mond, Li Jiang and Jeff Sterniak for their

constant encouragement and support. I would like to thank Jeff again for setting up the engine, debugging, performing experiments and answering technical questions.

I would like to thank Dr. Stani Bohac and Dr. Jason Matrz for helping me schedule the experiments and handling administrative tasks in the lab. I also thank the powertrain control group students, Shyam Jade and Jacob Larimore for setting up the control experiments and providing me with experimental data.

Finally, I would like to acknowledge all my friends and colleagues at the University of Michigan and the W.E. Lay Automotive lab for their interaction and support whenever I needed. They made my experience in Ann Arbor an enjoyable one. Thank you all!

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

LIST OF FIGURES ix

LIST OF TABLES xiii

LIST OF ABBREVIATIONS xv

ABSTRACT xviii

CHAPTER

I. Introduction 1

1.1 HCCI Engine - Background 1
1.2 Physics Based versus Data Based Modeling 2
1.3 Motivation 4
1.4 Research Goals 5
1.5 Contributions 5
1.6 Organization of Thesis 6

II. Machine Learning of Dynamic Systems - Background and Formulations 9

2.1 Preliminaries 9
2.1.1 Classification Learning 10
2.1.2 Regression Learning 10
2.1.3 Supervised Training 11
2.1.4 Over-fitting 11
2.1.5 Risk Minimization and Regularization 12
2.2 Learning the HCCI Dynamic Data 14
2.2.1 Modeling Tasks 16
2.3 Artificial Neural Networks 17

2.3.1 Multi Layer Perceptron 18
2.3.2 Radial Basis Network 20
2.4 Support Vector Classification 21
2.5 Support Vector Regression 24
2.6 Extreme Learning Machines 27
2.6.1 Difference between Neural Networks and Extreme Learning Machines 28
2.6.2 ELM - Mathematical Background 28
2.7 Comparison of Learning Algorithms 31

III. HCCI Engine - System, Experiment Design and Data Collection 33

3.1 HCCI Engine - Background 33
3.1.1 Modeling Challenges 35
3.2 Experimental Setup 36
3.3 Steady State Experiments and Mapping 39
3.4 Open Loop Transient Experiments 42
3.5 Closed Loop Transient Experiments 44

IV. Development of HCCI Engine Models using Regression Learning 45

4.1 Motivation 45
4.2 Data Processing 46
4.3 Model Selection 48
4.3.1 ANN 49
4.3.2 SVM 50
4.3.3 ELM 51
4.4 Model Evaluation 53
4.4.1 One-step Ahead Prediction 54
4.4.2 Multiple-step Ahead Prediction 58
4.5 Model Development using Closed-loop Experimental Data . . 65
4.6 Predictions at Different Engine Speeds 65
4.6.1 Interpolation Model Approach 69

V. Modeling the HCCI Operating Envelope using Class Imbalance Learning 76

5.1 Motivation and Problem Statement 76
5.2 Classification Algorithms 79
5.2.1 Class Imbalance Learning 80
5.2.2 Logistic Regression 80
5.2.3 Support Vector Machines 82
5.2.4 Extreme Learning Machines 82

5.3 HCCI Engine and Data Processing 83
5.3.1 HCCI System and Experimentation 84
5.3.2 HCCI Instabilities 84
5.3.3 Data Preprocessing and Labeling 86
5.4 Model Development 88
5.4.1 Model Selection 89
5.4.2 Prediction Results 97

VI. Stable Online Learning Algorithms for Extreme Learning Machines and Application to the HCCI Engine System 102

6.1 Motivation and Problem Statement 102
6.2 Existing Methods and Limitations 103
6.2.1 Online Sequential ELM (OS-ELM) 104
6.3 Lyapunov Based Algorithm (L-ELM) 106
6.3.1 Algorithm Derivation 106
6.3.2 Stability Analysis 111
6.3.3 Simulation Results 112
6.3.4 Stability Advantage of L-ELM 119
6.4 Stochastic Gradient Based ELM Algorithm 121
6.4.1 Algorithm Derivation 123
6.4.2 Stability Analysis 125
6.4.3 Simulation 1: Online Classification and Class Imbalance Learning 127
6.4.4 Simulation 2: Online Regression Learning 129
6.5 Online Regression Learning for HCCI Engine 131
6.6 Online Envelope Learning for HCCI Engine 136

VII. Controls Development - Model Predictive Control using Extreme Learning Machine Models 140

7.1 MPC Formulation using Extreme Learning Machines 140
7.1.1 Calculation of System Matrices 142
7.1.2 MPC Optimization Problem 143
7.2 HCCI Optimal Control 147
7.2.1 Fast Quadratic Programming 148
7.2.2 Simulations 150
7.3 MPC with Online Model Adaptation 154

VIII. Conclusions and Future Work 164

8.1 Summary of Research 164
8.2 Some Precautions in Using Computational Learning 169
8.3 Future Work 171

APPENDIX 173
A.1 Loss functions 174
A.2 Logistic Regression 175
A.3 Levenberg-Marquardt back-propagation algorithm 176

BIBLIOGRAPHY 179

LIST OF FIGURES

Figure

2.1	Figure showing a binary classification problem.	10
2.2	Figure showing a regression problem.	11
2.3	Figure showing supervised learning process.	12
2.4	Figure showing model over-fitting.	13
2.5	Series-Parallel architecture.	15
2.6	Parallel Architecture.	16
2.7	Neural network model structure.	18
2.8	Figure showing classification decision boundary separating two classes of data (marked as stars and spheres), SVM margin and support vectors.	23
2.9	Figure showing the working of support vector regression model.	27
2.10	Extreme Learning Machine model structure.	29
3.1	A schematic of the HCCI engine setup and instrumentation (only relevant instrumentation is shown).	37
3.2	HCCI engine pressure trace showing cycle definition, actuator ranges of intake valve opening (IVO), exhaust valve closing (EVC), start of injection (SOI). The crank angle at data recording are also shown.	38
3.3	Steady state operating region of the HCCI engine.	40
3.4	Predictions of the ELM based steady state model of the HCCI engine.	41
3.5	A subset of experimental data showing the A-PRBS inputs and the measured engine outputs. Misfire cycles are shown in dotted rectangles.	43
3.6	A subset of closed loop experimental data showing the A-PRBS inputs and the measured engine outputs.	44
4.1	A subset of open loop experiments showing A-PRBS inputs and HCCI engine outputs. Misfires are shown using rectangular dotted lines.	47
4.2	Comparison of 200 step ahead prediction for IMEP by MLP, RBN and linear models with actual engine data.	60
4.3	Comparison of 200 step ahead prediction for CA50 by MLP, RBN and linear models with actual engine data.	61
4.4	Comparison of 200 step ahead prediction for maximum pressure rise rate (R_{max}) by MLP, RBN and linear models with actual engine data.	61

4.5 Comparison of 200 step ahead prediction for EAFR by MLP, RBN and linear models with actual engine data. 62
4.6 Comparison of IMEP (engine output and SVR prediction). 63
4.7 Comparison of CA50 (engine output and SVR prediction). 63
4.8 Comparison of Maximum rate of pressure rise (engine output and SVR prediction). 64
4.9 Comparison of Lambda (engine output and SVR prediction). 64
4.10 Input trajectories for simulating the models for performing multi-step ahead predictions. 67
4.11 Comparison of IMEP predictions by SVM, ANN and ELM models with the experimental engine data. 67
4.12 Comparison of CA50 predictions by SVM, ANN and ELM models with the experimental engine data. 68
4.13 Comparison of R_{max} predictions by SVM, ANN and ELM models with the experimental engine data. 68
4.14 Comparison of EAFR predictions by SVM, ANN and ELM models with the experimental engine data. 68
4.15 Prediction summary of the nonlinear ELM model at 1600 RPM. . . 70
4.16 Prediction summary of the nonlinear ELM model at 1800 RPM. . . 71
4.17 Closed loop experiments by varying engine speed between 1600 and 1800 RPM. This data set is used for validating the interpolation model approach. 72
4.18 Validation of the interpolation model for IMEP and CA50 predictions. The predictions of 1600 RPM model and 1800 RPM models do not perform well when the engine speed is varied. 73
4.19 Validation of the interpolation model for P_{max} and R_{max} predictions. The predictions of 1600 RPM model and 1800 RPM models do not perform well when the engine speed is varied. 74
4.20 Validation of the interpolation model for engine torque and EAFR predictions. The predictions of 1600 RPM model and 1800 RPM models do not perform well when the engine speed is varied. 75
5.1 A-PRBS inputs and outputs showing misfire regions. 85
5.2 Illustration showing labeling of unstable observations. 87
5.3 Illustration showing labeling of stable observations. 88
5.4 Sensitivity plot for TPR, TNR and Total Accuracy with scaling factor f for cost-sensitive SVM. 94
5.5 Sensitivity plot for TPR, TNR and Total Accuracy with scaling factor f for cost-sensitive ELM. 95
5.6 Prediction results of the cost-sensitive ELM. The color code indicates model prediction - green (and red) indicate stable (and unstable) prediction by the model. 99
5.7 Prediction results of the cost-sensitive SVM. The color code indicates model prediction - green (and red) indicate stable (and unstable) prediction by the model. 100

5.8 A small subset of prediction results of the cost-sensitive ELM and SVM showing CA50, IMEP and one input variable to compare predictions in perspective to input variables. The green points indicate stable operation while red points indicate unstable operation. 101
6.1 Comparison of parameter evolution for the L-ELM with OS-ELM and the linear models for the simple scalar system 114
6.2 Comparison of multi-step ahead predictions for the L-ELM with OS-ELM, O-ELM and the linear models for the simple scalar system . . 115
6.3 Comparison of parameter evolution for the L-ELM with OS-ELM and the linear models for the more complex system 118
6.4 Comparison of multi-step ahead predictions for the L-ELM with OS-ELM, O-ELM and the linear models for the more complex system . 119
6.5 The ill-conditioning of OS-ELM as more number of hidden neurons (n_h) are added compared to bounded parameter evolution of L-ELM. 120
6.6 Comparison of parameter evolution for the online learning models. The recursive least squares based linear and ELM models exhibit an ill-conditioning problem which results in undesirable parameter growth. The parameters of the Lyapunov and Stochastic Gradient ELM are always bounded. 134
6.7 Comparison of multi-step ahead predictions of IMEP of the online models. The color codes are as follows - black:OS-ELM, blue:SG-ELM, green:L-ELM, red:linear and grey:experimental. 135
6.8 Comparison of multi-step ahead predictions of IMEP, CA50, P_{max}, R_{max}, Torque and EAFR by all online models. The color codes are as follows - black:OS-ELM, blue:SG-ELM, green:L-ELM, grey:experimental.137
6.9 Prediction results of the SG-ELM algorithm showing CA50, IMEP and one input variable (fueling) for 2 unseen data sets. The color code indicates model prediction - green (and red) indicate stable (and unstable) prediction by the model. The dotted line in the IMEP plot indicates misfire limit, dotted ellipse in CA50 plot indicates high variability instability mode while dotted rectangle indicates a wrong predictions by model. 139
7.1 MPC Framework 148
7.2 State trajectories of the HCCI engine model using MPC control. . . 152
7.3 Control trajectories of the HCCI engine model using MPC control. The upper and lower limits of each actuator is shown in dotted red. 153
7.4 State trajectories of the HCCI engine model (with noise) using MPC control. 155
7.5 Control trajectories of the HCCI engine model (with noise) using MPC control. The upper and lower limits of each actuator is shown in dotted red. 156
7.6 Framework showing model predictive control with online learning. . 157
7.7 Poor control by MPC due to inaccuracies in the ELM model. 157

7.8 Tracking performance of MPC controller with an without online adaptation of the models. The plots compare tracking performance of IMEP and CA50. 159
7.9 Optimal control trajectories of MPC controller with an without online adaptation of the models. 160
7.10 Tracking performance of MPC controller with OS-ELM adaptation. The model parameters grows unbounded resulting in unstable MPC control. 161
7.11 Control trajectories of MPC controller with OS-ELM adaptation. . 162
7.12 Tracking performance of MPC controller with L-ELM and SG-ELM adaptation. 163
A.1 A comparison of the loss functions of the algorithms used in this paper with the baseline 0-1 error standard. 175

LIST OF TABLES

Table

2.1 Comparison of algorithm properties for suitability to the HCCI engine problem. 32
3.1 Specifications of the experimental HCCI engine 37
4.1 The minimum and maximum values of regression variables x and y determined based on expert knowledge for HCCI conditions. 48
4.2 The optimal values of number of hidden neurons (n_h), regularization coefficient (λ in MLP and σ_h in RBN) and system order (n_o) determined using cross-validation. Here E_{val} represents the minimum validation error in the grid search. 50
4.3 The optimal values of system order (n_o=n_u=n_y, assumed to be the same), the cost parameter C, kernel parameter ω and SVR parameter ν determined using cross-validation from the range of listed values. . 52
4.4 The optimal values of number of hidden neurons (n_h), regularization coefficient (λ) and system order (n_o) determined using cross-validation. Here E_{val} represents the minimum validation error in the grid search. 53
4.5 Summary of prediction performances of MLP and RBN models. The results of linear regression model is also included as a baseline. Here n_m and n_p represents the number of memory units and number of model parameters required for prediction. The training and testing errors are for one-step ahead prediction while MSAP RMSE indicates the root mean squared error for multiple-step ahead prediction. The minimum error among linear, MLP and RBN models is highlighted in bold. 55
4.6 Performance comparison of SVR for modeling IMEP, CA50, R_{max} and EAFR. Here, n_m and n_p represents number of memory units and number of parameters (support vectors) required by the models. . . 57
4.7 Performance comparison of ANN, SVM and ELM models. Here RMSE refers to root mean squared error, OSAP and MSAP refer to one-step ahead prediction and multi-step ahead prediction respectively. The minimum error models are highlighted bold. 66
4.8 Modeling summary for experiments at 1600 RPM and 1800 RPM. . 69

5.1 Grid search results for ELM model selection for the regular ELM, ELM with under-sampling and ELM with over-sampling (The models resulting in lowest total accuracy is highlighted in bold). 91
5.2 Grid search results for SVM model selection for the regular SVM, SVM with under-sampling and SVM with over-sampling (The models resulting in lowest total accuracy is highlighted in bold). 92
5.3 Grid search results for Cost-sensitive SVM and Cost-sensitive ELM models (The models resulting in lowest total accuracy is highlighted in bold). 93
5.4 Cost-sensitive ELM models with different random initialization of input layer parameters 95
5.5 Summary of results for SVM and ELM models for all cases (Regular model, under-sampling, over-sampling and cost-sensitive). The results of the linear models (logistic regression and linear least squares) are also compared. The value of hyper-parameters and number of model parameters n_p are also included for every model 96
6.1 Performance comparison of L-ELM with OS-ELM and the linear models for the simple scalar system 115
6.2 Performance comparison of L-ELM with OS-ELM and the linear models for the more complex system 117
6.3 Benchmark data sets used for classification and class imbalance learning. 128
6.4 Performance comparison of OS-ELM and SG-ELM in terms of training time, total positive rate (TPR), total negative rate (TNR), average accuracy and geometric mean accuracy. The results of a recursive least squares based linear model is also compared to measure the nonlinearity in the data sets. 130
6.5 Benchmark data sets used for regression learning. 130
6.6 Performance comparison of OS-ELM and SG-ELM in terms of training time and generalization error. The results of a recursive least squares based linear model is also compared to measure the nonlinearity in the data sets. 132
6.7 Performance comparison of the nonlinear online models (OS-ELM, L-ELM and SG-ELM) for the regression learning problem. A baseline linear model and an offline trained ELM model (O-ELM) are also used for comparison. 135
6.8 Performance comparison of the nonlinear models (OS-ELM and SG-ELM) for the online class imbalance learning problem. A baseline linear model and an offline trained ELM model (O-ELM) are also used for comparison. 138
7.1 Tracking performance of the MPC controller with online adaptation using L-ELM, SG-ELM and OS-ELM compared against MPC with no adaptation. 158

LIST OF ABBREVIATIONS

HCCI - Homogeneous charge compression ignition

SI - Spark ignited

CI - Compression ignited

NO_X - Nitrogen oxides

ECU - Engine control unit

OSAP - One step ahead prediction

MSAP - Multiple steps ahead prediction

NARX - Nonlinear auto regressive model with exogenous input

ANN - Artificial neural networks

MLP - Multi-layer perceptron

RBN - Radial basis network

SVM - Support vector machines

SVR - Support vector regression

ELM - Extreme learning machines

LM - Levenberg-Marquardt back-propagation algorithm

EGR - Exhaust gas recirculation

VVT - Variable valve timing

FM - Fuel mass injected per cycle

IVO - Intake valve opening

EVC - Exhaust valve closing

SOI - Start of injection

IMEP - Indicated mean effective pressure (indicator for work output)

CA50 - Crank angle at 50 % mass fraction burned (indicator for combustion phasing)

P_{max} - Maximum pressure

R_{max} - Maximum rate of pressure rise

EAFR - Equivalent air-fuel ratio

NVO - Negative valve overlap

BDC - Bottom dead center

cTDC - combustion Top dead center

eTDC - exhaust Top dead center

A-PRBS - Amplitude modulated pseudo-random binary sequence

DOE - Design of experiments

RMSE - Root mean squared error

n_h - Number of hidden neurons

λ - Regularization coefficient

LS - Least squares

LR - Logistic regression

CIL - Class imbalance learning

TP - Total positive observations

TPR - Total positive rate

TN - Total negative observations

TNR - Total negative rate

OS-ELM - Recursive least squares based ELM algorithm

L-ELM - Lyapunov based ELM algorithm

SG-ELM - Stochastic gradient based ELM algorithm

O-ELM - Offline ELM (batch learning) algorithm

Llin - Linear Lyapunov based estimation algorithm

RLS - Recursive least squares

SGD - Stochastic gradient descent

MPC - Model predictive control

QP - Quadratic programming

ABSTRACT

Machine Learning for Identification and Optimal Control of Advanced Automotive Engines

by

Vijay Manikandan Janakiraman

Co-Chairs: Professor Dennis N. Assanis, Professor XuanLong Nguyen and Professor Jeffrey Stein

The complexity of automotive engines continues to increase to meet increasing performance requirements such as high fuel economy and low emissions. The increased sensing capabilities associated with such systems generate a large volume of informative data. With advancements in computing technologies, predictive models of complex dynamic systems useful for diagnostics and controls can be developed using data based learning. Such models have a short development time and can serve as alternatives to traditional physics based modeling. In this thesis, the modeling and control problem of an advanced automotive engine, the homogeneous charge compression ignition (HCCI) engine, is addressed using data based learning techniques. Several frameworks including design of experiments for data generation, identification of HCCI combustion variables, modeling the HCCI operating envelope and model predictive control have been developed and analyzed. In addition, stable online learning algorithms for a general class of nonlinear systems have been developed using extreme

learning machine (ELM) model structure. The main contributions of this research can be listed as follows

1. A machine learning framework to perform nonlinear identification of the HCCI engine is developed. Advanced machine learning algorithms such as recurrent neural networks, support vector machines, extreme learning machines are applied to the HCCI engine system.

2. Dynamic simulators of HCCI engine variables such as indicated mean effective pressure (IMEP), combustion phasing (CA50), maximum pressure and maximum pressure rise rate, equivalent air-fuel-ratio etc. are developed using transient experimental data.

3. A model of the transient stable operating envelope is developed for HCCI combustion using class imbalance learning. The model predicts if a future HCCI combustion cycle is stable/unstable, given the history of engine measurements.

4. Stable online learning algorithms are developed using Lyapunov and stochastic gradient descent methods. The algorithms use an extreme learning machine model structure and can be used for online system identification of a general class of nonlinear systems.

5. An online learning framework is developed for HCCI that can be used to develop engine simulators online (in real-time). In addition, a model of the stable operating envelope is developed online using the stochastic gradient descent algorithm.

6. A linearized model predictive control (MPC) framework for the nonlinear HCCI engine is developed using extreme learning machine models. Further, integration of the developed online learning algorithms to the MPC framework is performed enabling online adaptation of the engine model for parameter variations.

CHAPTER I

Introduction

1.1 HCCI Engine - Background

In recent years, the requirements on automotive performance, emissions and safety have become increasingly stringent. In spite of advanced concepts entering the industry, achieving simultaneous benefits in fuel economy, emissions and cost still remain an arduous task. HCCI engines shifted the spotlight from traditional spark ignited (SI) and compression ignited (CI) engines owing to its ability to reduce emissions and fuel consumption significantly [1, 2, 3]. The fuel lean mixtures allow HCCI to operate with a higher compression ratio similar to diesel engines resulting in high thermal efficiency [4]. Also, un-throttled operation improves volumetric efficiency. The fuel and oxidizer are premixed which results in a cleaner combustion and hence reduced emissions [4]. A characteristic feature of HCCI combustion is that the peak in-cylinder temperatures are low resulting in low nitrogen oxides (NO_X) emissions [5]. The homogeneous mixtures result in reduced soot emissions [4]. However, in spite of its known advantages, HCCI combustion poses several challenges for implementation.

Control of HCCI combustion is a major challenge for automotive application. Several factors contribute to the challenge including the absence of a direct trigger for combustion, narrow operating range and high sensitivity to disturbances. To address the issue, advanced model based control methods are common where the control

actions are often made using a predictive model of the engine [6, 7, 8]. Typically, a nonlinear high order model of the engine is first developed using physics, capturing the cycle-by-cycle dynamics of HCCI. The model is then approximated and a low order linear model is used to derive control laws that are applied in real time. The performance of the controller is highly dependent on the performance of the models used. Hence, developing robust and accurate predictive models of HCCI combustion is of extreme importance.

1.2 Physics Based versus Data Based Modeling

Mathematical models can be developed using two approaches - a first principle approach or physics based and identification approach or data based. Sometimes, a combination of the two may be used. A physics based approach may be very useful in terms of analyzing the controller with the insight of the underlying physics especially when building a controller for the prototype for the first time. However, such an approach has its limitations. The controller may be conservative and valid in a small range of operation because the model parameters are usually calibrated using steady state experimental data and the models include several ideal behavior assumptions. Also, sometimes, the first principle models tend to be too simple to describe the real systems to the necessary degree of precision necessary to meet the increasing performance requirements [9]. Further, such models require extensive calibration and frequently they cannot be used for control design, as they are too complex for on-line use [9]. This motivates the search for an alternative approach to model development and controls design for HCCI.

A natural alternative to the first principles based controls development is the identification approach which involves determining the model structure and parameters using experimental data. For simple systems, a linear or close to linear models such as linear parameter varying models can be used. However, for strong nonlinear

systems, linear models fail to describe the system behavior to the required level of accuracy. In such cases, nonlinear models such as neural networks (NN) are very popular for engine systems. Identification models for HCCI are not common and literature is scarce. A subspace based identification was the only reported work [**?**] where linear models were developed for HCCI model predictive control. The approach demonstrates identification based control for HCCI but is too simplistic to operate in a small region. Several implementations of NN for nonlinear system modeling and control have been reported in the literature [10, 11, 12]. Automotive implementations include identification of a diesel locomotive engine using NN [13], virtual sensors to predict specific quantities like NO_X [14], air-fuel ratio [15] and diagnostics [16] etc. The prime advantages of data based models over physics based models can be listed as follows.

1. Data based models do not make idealistic assumptions about the system.

2. Data based models capture the complete input-output behavior including dynamics associated with sensors and actuators.

3. Data based models like neural nets have a parallelization capability for fast implementation on board using less computation and memory [17].

4. Online adaptation of data based models is straight forward, with the structure of the model initially identified.

5. Development time and associated costs are typically less.

However, a poorly developed model can diminish the advantages of the data driven approach if not completely making it infeasible. For instance, an over-fitted model gives absurd predictions of the system, an improperly designed algorithm or careless application might result in wasted time and cost, usage of the models in a region that is not appropriate for the model may lead to poor performance if not absurd predictions.

Thus, it is important to analyze the data, obtain possible expert knowledge from the system and systematically identify and apply suitable algorithms using theoretical understanding of the algorithms. The need for model development from data and the complexity involved is increasing and varies between different application domains. A dedicated field of research combining ingredients from statistics, computer science and engineering has been created for performing data based inference and decision making and is widely known as machine learning.

1.3 Motivation

In spite of the advantages listed in the previous section, machine learning is mostly at the stage of research and not in the mainstream of automotive production yet. One of the reasons is the black-box nature of such models. Other, perhaps more compelling reasons are business conservatism and existing/legacy applications [18]. However, with advanced systems entering the industry to meet performance requirements, the complexity of the system increases which necessitates the capabilities of data driven approaches be included to augment/replace existing conventional approaches. For instance, developing a physics based model for emissions is a challenging task for first principle based approach but can be addressed by using data driven models [19]. A similar situation is the detection of engine misfires onboard which is solved using a neural network system and is currently in production [18]. The spectrum of challenging problems that can be solved using data driven models is increasing and it becomes important to develop and analyze methodologies and algorithms that can perform data based learning for complex systems in a simple, efficient and stable manner.

In addition to the above, there is a need to develop fast and accurate predictive models for HCCI combustion. As mentioned earlier, HCCI control is model-based and a physics based approach might be insufficient for optimal engine operation. For

instance, HCCI engine control between set points determined from offline steady-state values may be too conservative for competing objectives such as fuel economy and transient emissions. The capabilities of a data based learning approach could pave way towards an effective engine control for HCCI.

1.4 Research Goals

One of the goals of this thesis is to gain first insights in the novel application of advanced machine learning algorithms to the HCCI engine problem in developing accurate dynamic models in a fast, simple, efficient and stable manner. Using the developed models, it is also aimed to derive model based optimal control solutions for HCCI operation.

In addition, using the insights gained using HCCI engine data, the second goal of this thesis is to develop efficient algorithms for online model adaptation and optimal control for a general class of non-linear systems with system level constraints such as limited computation (as in the HCCI engine ECU) as well as with a guarantee on convergence and stability.

1.5 Contributions

The main contributions of this research can be listed as follows

1. A machine learning framework to perform nonlinear system identification of the HCCI engine is developed. Advanced machine learning algorithms such as neural networks, support vector machines, extreme learning machines are applied to the HCCI engine system for the first time in the literature.

2. Dynamic simulators of HCCI engine variables such as indicated mean effective pressure (IMEP), combustion phasing (CA50), maximum pressure and maxi-

mum pressure rise rate, equivalent air-fuel-ratio etc. are developed using transient experimental data.

3. A stable operating envelope model for HCCI combustion in transients is developed using class imbalance learning. The model predicts if a future HCCI combustion cycle is stable/unstable given the present and past engine measurements, and can be used for predicting engine misfires and high variability combustion behavior. The transient boundary prediction is a novel application in the automotive literature.

4. Stable online learning algorithms are developed for nonlinear systems using Lyapunov and stochastic gradient descent methods. The algorithms use an extreme learning machine model structure and can be used for online system identification of a general class of nonlinear systems.

5. An online learning framework was developed for HCCI that can be used to develop engine simulators online (in real-time). A model of stable operating envelope is developed using the online imbalanced classification algorithm. The framework can also correct the offline identified models by processing experimental data and adapting model parameters on-the-fly.

6. A linearized model predictive control (MPC) framework for the nonlinear HCCI engine is developed using extreme learning machine models. Further, integration of the online learning algorithms to the MPC framework is performed enabling online correction of the engine model for parameter variations.

1.6 Organization of Thesis

The upcoming chapters of this thesis are organized as follows.

Chapter - 2: Machine Learning of Dynamic Systems - Background and Formulations

This chapter touches upon some preliminaries on machine learning and system identification, the state-of-the-art machine learning algorithms that are used in this research - background, literature, mathematical formulations etc. of artificial neural networks, support vector machines and extreme learning machines for classification and regression tasks.

Chapter - 3: HCCI Engine - System, Experiment Design and Data Collection

In this chapter, the considered HCCI system is described along with the relevant sensor variables, experimental setup and experiment design for steady state and transient experiments. The data collection procedure to obtain sufficiently rich information for system identification is described.

Chapter - 4: Development of HCCI Engine Models using Regression Learning

This chapter details the development of identification models (or system simulators) from experimental data. Since this task is performed offline, the models are referred to as offline models. The key topics include data processing, model structure learning, model training, validation and evaluation in terms of prediction accuracy, memory requirement and potential for online learning.

Chapter - 5: Modeling the HCCI Operating Envelope using Class Imbalance Learning

In this chapter, a model for the stable operating boundary of HCCI engine is developed using experimental data. Addressing the insufficient data for unstable class, a class imbalance learning is evaluated to choose between data re-sampling and cost-sensitive methods. Further, a misfire prediction model for the HCCI engine is demonstrated.

Chapter - 6: Stable Online Learning Algorithms for Extreme Learning Machines and Application to the HCCI Engine System

This chapter introduces two stable online learning algorithms developed for extreme learning machine models namely the Lyapunov based algorithm and the stochastic gradient based algorithm. The algorithms are evaluated both on benchmark data sets as well as on the HCCI engine system.

Chapter - 7: Controls Development - Model Predictive Control using Extreme Learning Machine Models

Finally, the machine learning models are used to develop an optimal controller using a model predictive control approach. The effectiveness of the method is demonstrated in simulation. Further, in order to compensate for inaccurate identification models, an online adaptation using the online learning algorithms is presented.

Chapter - 8: Conclusions and Future Work

The conclusions of this research, some precautionary lessons learned in using black-box models and possible future directions are included in this chapter.

CHAPTER II

Machine Learning of Dynamic Systems - Background and Formulations

In this chapter, the HCCI system identification task is considered as a machine learning problem and the related tools and techniques are discussed. Three state-of-the-art learning algorithms including artificial neural networks, support vector machines and extreme learning machines are introduced. The literature, mathematical formulation and technical details of the learning algorithms are summarized.

2.1 Preliminaries

Consider the following data set which is used for machine learning

$$\{(x_1, y_1), ..., (x_N, y_N)\} \in \big(\mathcal{X}, \mathcal{Y}\big), \tag{2.1}$$

where N denotes the number of training samples, $\mathcal{X}$ denotes the space of the input features and $\mathcal{Y}$ denotes the predictor labels. In simple terms, the task of developing inference using the above data is referred to as training or learning. Data could be labeled or unlabeled and the nature, availability of labels differentiate the learning problem in hand. For instance, if $\mathcal{Y}$ takes integer values {1,2,3,..} then the problem is referred to as classification and if $\mathcal{Y}$ takes real values, it becomes a regression problem.

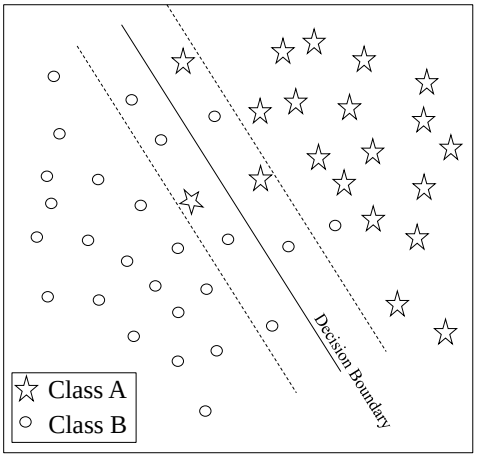


Figure 2.1: Figure showing a binary classification problem.

Before introducing the algorithms considered in this work, the important terminology involved in machine learning and model development is briefed below.

2.1.1 Classification Learning

In a classification task, the goal of learning is to approximate a function that forms a decision boundary between different classes of data. If there are two classes of data, it is called a binary classification problem whereas if the problem involves multiple classes of data, it is called a multi-class classification. Typically labels for a binary classification problem takes values in {-1,+1} or {0,+1} while it takes values in {1,2,3,..} for a multi-class classification. An example of a binary classification problem is shown in Fig. 2.1.

2.1.2 Regression Learning

In a regression problem, the labels take real values and are suitable for function approximations. An illustration of a regression problem can be shown in Fig. 2.2.

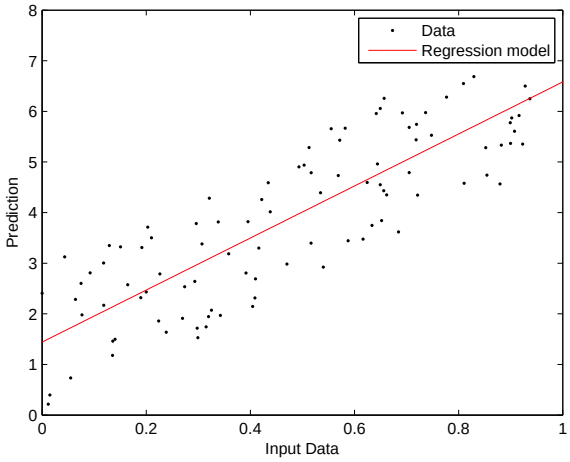


Figure 2.2: Figure showing a regression problem.

2.1.3 Supervised Training

The task of developing predictive models of HCCI engine using learning algorithms falls into the category of supervised learning. Given a set of experimental data as in (2.1), the task of developing inference from data that is labeled is referred to as supervised learning. Supervisory learning involves a supervisor that directs the training process based on availability of model's performance relative to the system. For instance, consider the system and model in Fig. 2.3 where an error metric can be defined based on the data labels and the model predictions which can be used to train the model. Typically, an optimization problem is solved by minimizing the defined error metric. The model when trained, can be used to simulate the system for unknown inputs.

2.1.4 Over-fitting

As mentioned previously, supervised training determines the model parameters by solving an optimization problem over the training data set. As a result, the model can memorize the training data too well. Since the data is always noisy, the over trained models capture both the underlying phenomena as well as the noise. This

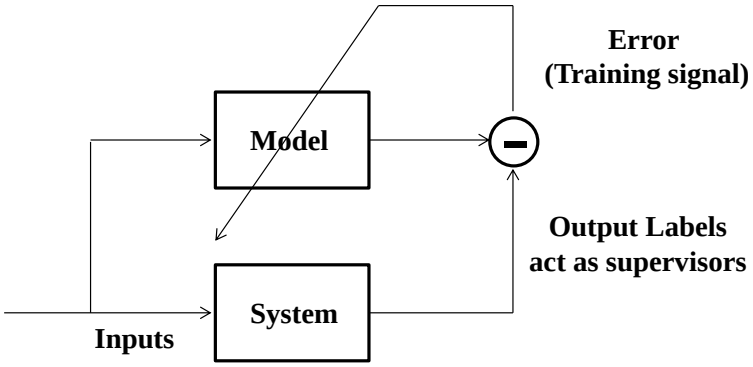


Figure 2.3: Figure showing supervised learning process.

is highly undesirable as an over trained model cannot be used for predictions. This phenomenon is commonly referred to as over-fitting and can be shown in Fig. 2.4.

Over-fitting may be avoided/reduced either by stopping the training process early or by reducing the model complexity. Early stopping typically involves dividing the available data set into a training set and a validation set. As the model is trained, it is evaluated on the validation set so that any over learning of the training set is indicated by a poor performance on the validation set. The evaluation on the unseen validation set is a check for the generalization capability of the model which is the ultimate goal of learning. Early stopping is a heuristic approach that works fairly well but in this work, a more systematic approach to avoid over-fitting using regularization to achieve the right model complexity is employed.

2.1.5 Risk Minimization and Regularization

The ability of a trained model to generalize to any unseen data set is referred to as generalization. A generalizable model can be used as an emulator for the system and is a major requirement in this work. Let $P(x, y)$ be an unknown probability distribution that is to be modeled. Training involves minimizing an expected error metric, often referred to as a risk function, over the distribution $P(x, y)$. Thus, the

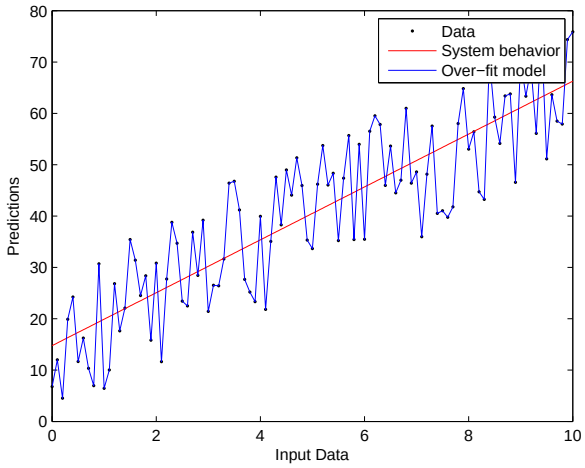


Figure 2.4: Figure showing model over-fitting.

goal of training is to minimize the expected risk

$$E_{exp}(f) = \int L(f(x), y) dP(x, y) \tag{2.2}$$

where $f(x)$ is the function to be identified and $L(f(x), y)$ is a loss function, typically a squared loss or mean squared error (see appendix A.1 for other loss functions considered in this work). However, in practice, it is not possible to solve the above equation and a proxy for expected risk, referred to as the empirical risk, can be used to perform training. With limited data availability, the empirical risk makes training from data possible. Let the data (x_i, y_i), $i = 1, 2, .., N$ be sampled from the unknown joint probability distribution $P(x, y)$. The empirical risk E_{emp} is given by

$$E_{emp}(f) = \frac{1}{N} \sum_{i=1}^{N} L(f(x_i), y_i). \tag{2.3}$$

A detailed explanation of the concepts covered in this section can be found in [20].

Over-fitting can be thought as occurring because of availability of extra degrees of freedom in the model structure that can capture undesirable noise. Such a model is considered complex and typically consists of several extra parameters (an over-

parameterized model). Regularization is a process in which the extra parameters are suppressed by moving their values close to zero to give a simpler model in spite of being over-parameterized. This can be achieved by minimizing the norm of parameters, referred to as the structural risk function. Hence, in a learning process, a combination of the empirical risk and structural risk is minimized as shown below

$$\min \left\{ \frac{1}{N} \sum_{i=1}^{N} L(f(x_i), y_i) + \lambda \|f(.)\|_p \right\} \tag{2.4}$$

where λ is the regularization coefficient and p the dimension of the norm space $\|.\|$.

2.2 Learning the HCCI Dynamic Data

In the HCCI modeling problem, both the inputs and the outputs of the engine are available as sensor measurements and hence supervised learning can be employed. The HCCI engine is a nonlinear dynamic system and sensor measurements represent discrete time sequences. The input-output behavior can be modeled using a nonlinear auto regressive model with exogenous input (NARX) [21] as follows

$$y(k) = f_{NARX}[u(k-1), .., u(k-n_u), y(k-1), .., y(k-n_y)] \tag{2.5}$$

where $u(k) \in \mathbb{R}^{u_d}$ and $y(k) \in \mathbb{R}^{y_d}$ represent the inputs and outputs of the system respectively, k represents the discrete time index, $f_{NARX}(.)$ represents the nonlinear function mapping specified by the model, n_u, n_y represent the number of past input and output samples required (order of the system) while u_d and y_d represent the dimension of inputs and outputs respectively. Let x represent the augmented input vector obtained by appending the input and output measurements from the system.

$$x = [u(k-1), .., u(k-n_u), y(k-1), .., y(k-n_y)]^T \tag{2.6}$$

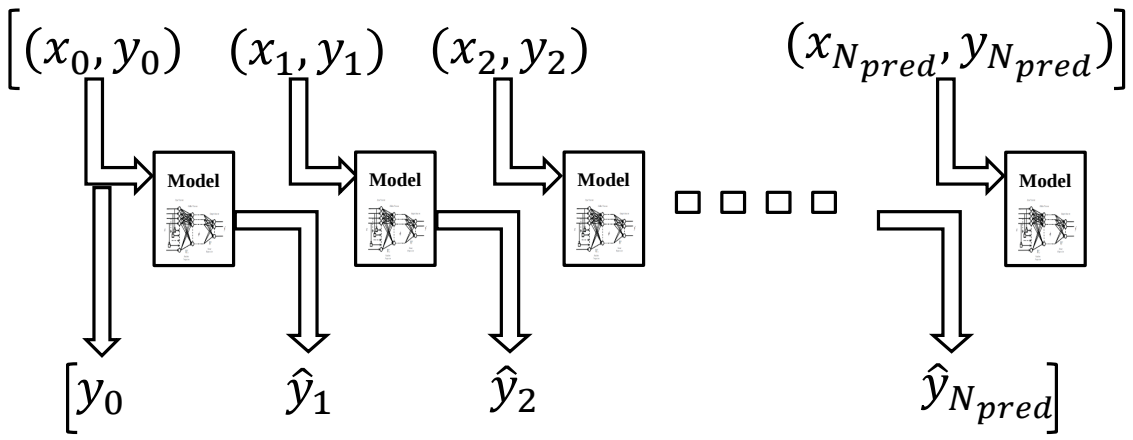


Figure 2.5: Series-Parallel architecture.

The input measurement sequence can be converted to the form of training data

$$\{(x_1, y_1), ..., (x_N, y_N)\} \in (\mathcal{X}, \mathcal{Y}) \tag{2.7}$$

where N denotes the number of training samples, $\mathcal{X}$ denotes the space of the input features (Here $\mathcal{X} = \mathbb{R}^{u_d n_u + y_d n_y}$ and $\mathcal{Y} = \mathbb{R}$ for regression and $\mathcal{Y} = \{+1, -1\}$ for a binary classification). The above conversion of system measurements to training data is a natural definition for a series-parallel model architecture and the models can be used for a one-step ahead prediction (OSAP) i.e., given a set of measurements until time index k, the model predicts the output at time $k+1$ (see equation (2.8) and Fig. 2.5). A parallel architecture on the other hand can be used to perform multiple step ahead predictions (MSAP) by feeding back the predictions of the OSAP model in a recurrent manner (see equation (4.2) and Fig. 2.6). The series-parallel and parallel architectures are well explained in [22].

$$\hat{y}(k+1) = \hat{f}_{NARX}[u(k), .., u(k - n_u + 1), y(k), .., y(k - n_y + 1)] \tag{2.8}$$

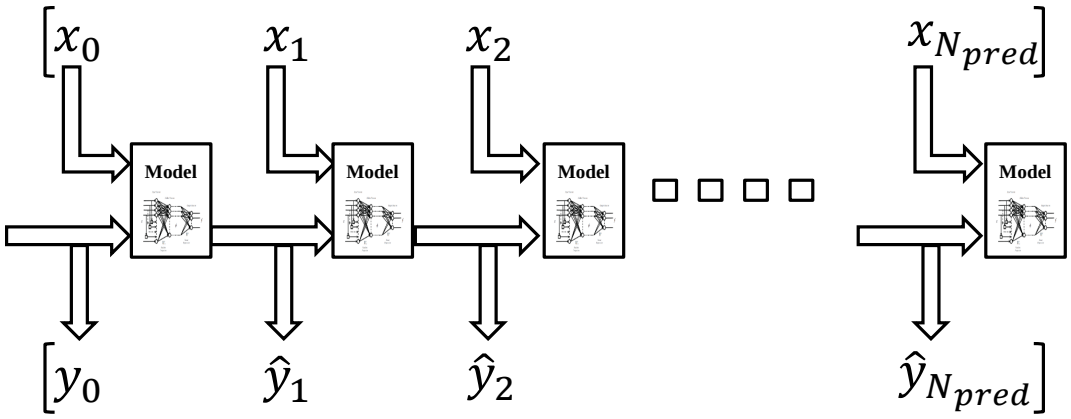


Figure 2.6: Parallel Architecture.

$$\hat{y}(k+n_{pred}) = \hat{f}_{NARX}[u(k+n_{pred}-1),..,u(k-n_u+n_{pred}), \\ \hat{y}(k+n_{pred}-1),..,\hat{y}(k-n_y+n_{pred})] \quad (2.9)$$

The OSAP model is used for training as existing simple training algorithms can be used and once the model becomes accurate for OSAP, it can be converted to a MSAP model in a straightforward manner. The MSAP model can be used for making long term predictions useful for predictive control discussed in chapters IV and VII.

2.2.1 Modeling Tasks

For the HCCI engine identification, models predicting engine performance variables such as IMEP, CA50 etc. (see chapter IV) that take real values are required. Such models can be obtained by solving a regression problem. On the other hand, for determining the operating envelope of HCCI (see chapter V), decision surfaces predicting binary valued outputs can be obtained by solving a classification task.

The goal of machine learning is to model the underlying relationship between x and y by minimizing a risk function $R(w)$ with respect to the model parameters w.

$$R(w) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{y}_i(x,w)) + \frac{\lambda}{2} w^T w \quad (2.10)$$

Here, $R(w)$ has two components - the empirical risk minimizing the training error and the structural risk minimizing the model parameters in order to obtain a simple model that generalizes well. L represents a loss functional and $\hat{y}(x, w)$ represents the model prediction, whose structures are given by the learning algorithm. For instance, regression problems have a squared loss function while classification problems make use of a hinge loss or logistic loss (see appendix A.1). The intuition behind loss function is as follows. For a regression problem, squared errors along positive and negative directions are penalized equally resulting in a function close to the true function. Classification problems on the other hand maximizes the margin of separation ($y\hat{y}$) between the data classes. A squared loss would penalize a rightly classified data more (a positive margin in Fig. A.1) while a hinge loss or logistic loss do not penalize the rightly classified data as much and hence efficient in classification problems. The learning algorithms applied for the HCCI engine problem is summarized in the following sections.

2.3 Artificial Neural Networks

Motivated by the superior learning ability of the human brain, Artificial Neural Networks (ANN) were developed to solve early machine learning problems [23]. ANNs are constructed by interconnecting a number of simple decision making elements called perceptrons. Each perceptron is parameterized by a weight, a bias and a nonlinear activation function and the parameters are determined by solving a nonlinear optimization problem that minimizes the risk function (2.11). ANNs employ a squared loss function which makes it more suitable for a regression problem. Two popular models of ANNs are considered in this work including Multi Layer Perceptron and Radial Basis Network models. The general structure of ANN models can be shown in Fig. 2.7 where some of the inputs are delayed using an unit delay component

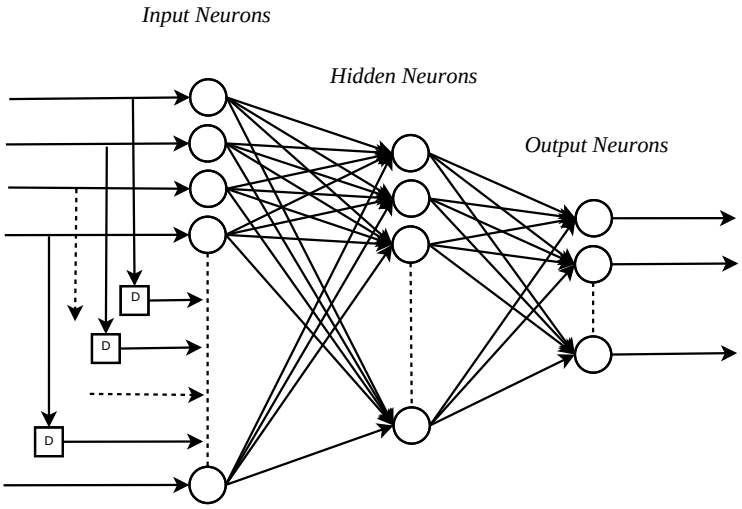


Figure 2.7: Neural network model structure.

D to mimic the dynamic behavior of the system.

$$R(w) = \frac{1}{N}\sum_{i=1}^{N} \|y_i - \hat{y}_i(x, w)\|^2 + \frac{\lambda}{2}\|w\|^2 \tag{2.11}$$

for $i = 1, .., N$.

2.3.1 Multi Layer Perceptron

The Multi Layer Perceptron (MLP) model is constructed using several layers of interconnected perceptron nodes which are connected in a feed-forward manner between the input and the output layer. MLPs are known as universal function approximators as they can approximate any smooth function to an arbitrary accuracy [24, 25]. The network considered in this thesis consists of an input layer, an output layer and a hidden layer. The predicted output from the network can be expressed as

$$\hat{y}_{MLP} = f_2[b_2 + W_2^T f_1\{b_1 + W_1^T(x)\}] \tag{2.12}$$

$$f_1(z) = \frac{2}{1 + e^{-2z}} - 1 \tag{2.13}$$

where f_1 and f_2 represent the activation functions of the hidden and the output layers respectively. The functions f_1 and f_2 return outputs of the same dimension as that of their inputs. A tangential sigmoid function is used to represent f_1 (see equation (2.13)) while a linear function is used to represent f_2. The parameters b_1 and b_2 represent the bias terms of the hidden and output layers while W_1 and W_2 represent the weights of the connection between input-hidden layers and hidden-output layers respectively. The number of input and output neurons corresponds to the dimensions of $\mathcal{X}$ and $\mathcal{Y}$ respectively.

The MLP is trained using standard back-propagation via the Levenberg-Marquardt (LM) algorithm [26] which is more efficient compared to other available gradient based training algorithms such as the conjugate gradient and the variable learning rate back-propagation [27]. The sensitivity of the cost function (2.11) with respect to the network weights are determined and weights updated so that the risk functional (2.11) is minimized (see appendix A.3). Batch training is done where the entire training data is repeatedly presented to the network. Batch training is more efficient compared to incremental training but requires all the data to be available for an off-line training. The activation functions (sigmoid and linear) take finite values for the entire range of network operation and hence the MLP model requires a relatively less number of parameters to map complex functions [28]. However training could become complicated involving heavy computation making the process slow for large data sets. The MLP is a nonlinear regression problem and training with an iterative algorithm like the LM, usually finds a local minima. Global optimization methods such as simulated annealing [29, 30, 31], genetic algorithms [32, 29, 33] etc can be used in training but convergence is usually slow and may not suit problems with large input dimensions.

2.3.2 Radial Basis Network

The Radial Basis Network (RBN) model is a two layered network where the connections are feed-forward between the input and the output layers similar to the MLP model. The nodes are connected directly between the input and the hidden layers (i.e, with unit weights). The hidden layer neurons have a gaussian activation function f_3 (see equation (2.15)) that determines the excitation level of the neurons depending on how close the input data is located with respect to the center (μ_h) of the neuron. Hence a notion of location and similarity is introduced in the RBN model resulting in local learning. The output layer is linear and hence the output is a linear combination of the activation levels of the hidden neurons. Hence when the centers of the activation functions are fixed, the training reduces to determining the hidden-output layer weights using linear least squares. This usually makes training in RBN models fast compared to MLP and is an important distinction between the two models. The predicted output and the activation function of the RBN model can be expressed as

$$\hat{y}_{RBN} = b_4 + W_4^T f_3\{x\} \tag{2.14}$$

$$f_{3_h}(z) = exp\Big(- \frac{\|z - \mu_h\|^2}{2\sigma_h^2}\Big) \tag{2.15}$$

where μ_h with the same dimension of x while $\sigma_h \in \mathbb{R}$ are the center and the spread of the gaussian function for a given hidden neuron h. The spread σ_h is constant for all hidden neurons and is considered as a hyper-parameter and is not updated in the training process.

RBN models are also considered universal approximators [34, 35, 36] and are popular in regression and classification applications. The training of RBN model involves two tasks - identifying the neuron centers (μ_h) and determining the hidden to output layer weights (W_4). Several means of finding the neuron centers include randomly selecting locations from the input data, unsupervised learning of the input

structure using clustering [37, 38, 39], orthogonal least squares [40, 41] among others. The hidden-output layer weights are determined using linear least squares. In this paper, the RBN model is trained using the Matlab subroutine where at every training pass, a new hidden neuron is created at the location of the the input vector that results in reducing the network error the most. The above process is repeated until the error reaches a specified goal or the limit for maximum number of hidden neurons is reached.

2.4 Support Vector Classification

Support Vector Machines (SVM) are state-of-the-art methods for classification, regression and other estimation problems in machine learning. For classification, SVM training involves determining the boundary that maximizes the margin between the data classes based on a hinge loss [42]

$$L_{hinge}(w, b) = max(0, 1 - y\hat{y}(x, w)). \tag{2.16}$$

The margin of a binary classifier can be defined as

$$m_c = y\hat{y}(x, w). \tag{2.17}$$

A positive margin indicates that both y and $\hat{y}(x, w)$ have the same sign, or the classifier prediction is right while a negative margin indicates misclassification. SVM learning translates to finding the optimal model parameters (w^*, b^*) by solving the following optimization problem

$$\min_{w,b,\zeta_i} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \zeta_i \tag{2.18}$$

$$\text{subjected to} \begin{cases} y_i[\langle w, \phi(x_i)\rangle + b] \geq 1 - \zeta_i \\ \zeta_i \geq 0 \end{cases} \tag{2.19}$$

for $i = 1, .., N$. Here ζ_i represents the slack variable for data observation i, C represents the cost penalty hyper-parameter. The slack variables ζ_i are required in order to allow for misclassifications in a noisy overlapping data set that cannot be completely separated by a linear decision boundary. The input vectors x are mapped onto a higher dimensional subspace by the function ϕ. By making this transformation, the nonlinear data is aligned linearly in a high dimensional space where SVM finds a linear optimal margin separating hyperplane. The transformation is performed implicitly using kernel matrix $K(x_i, x_j) = [k(x_i, x_j)]_{i,j}$ where $k(x_i, x_j)$ could be any function satisfying Mercer's condition [42]. The gaussian kernel function (equation (2.20)) is used in this work for classification tasks. More details on SVM formulation can be found in [42].

$$k(x_i, x_j) = e^{-\omega\|x_i - x_j\|^2}, \omega > 0 \tag{2.20}$$

The convex constrained optimization problem in equation (2.18) is in the primal form, and the variables w, b and ζ_i are referred to as primal variables. The primal problem is converted to a dual formulation in equation (2.21) and solved for the dual variables α_i.

$$\max_{\alpha_i} \left\{ -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j K(x_i, x_j) + \sum_{i=1}^{N} \alpha_i \right\} \tag{2.21}$$

$$\text{subjected to} \begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \end{cases} \tag{2.22}$$

for $i = 1, .., N$. The SVM hypothesis is given by

$$\hat{y}(x) = sgn\left(\sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b \right) \tag{2.23}$$

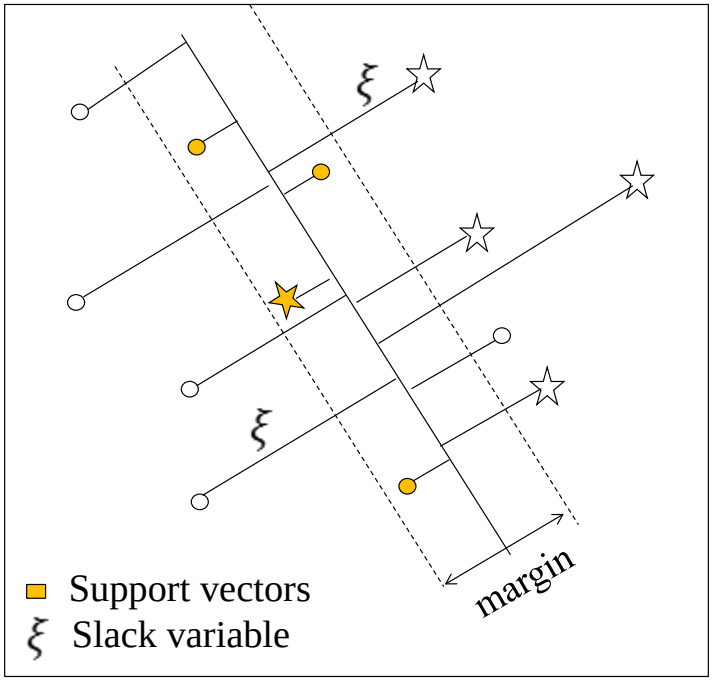


Figure 2.8: Figure showing classification decision boundary separating two classes of data (marked as stars and spheres), SVM margin and support vectors.

where

$$sgn(x) = \begin{cases} 1 & x > 0 \\ -1 & x \leq 0. \end{cases} \tag{2.24}$$

The solution to the dual optimization problem yields the lagrange multipliers α_i for each data observation and the SVM model is a weighted combination of the kernel function values weighted by α_i. For a large data set, the number of α, x, y to be stored is very large. However, it should be noted that most of the α_i are zero and only the data corresponding to the non-zero α_i may be stored for predictions. Such a representation is considered sparse and the non-zero α_i data observations are referred to as support vectors. The support vectors are the data observations that determine the decision boundary (see Fig. 2.8). The presence of the other data observations are immaterial to the decision boundary and hence the SVM model.

The above SVM formulation is not designed for an imbalanced data set where the majority class data outnumbers the minority class data. A cost-sensitive version

of the SVM algorithm is used in such cases, where the cost penalty parameter C in equation (2.18) is modified to minimize the slack variables of the minority class data more, compared to the majority class data [43, 44]. In this modification, the decision boundary moves towards distinguishing the minority data well. The cost modification can be performed as follows

$$C_i = \begin{cases} \frac{C*f}{r} & \text{majority class data} \\ C & \text{minority class data} \end{cases} \tag{2.25}$$

where r represents the ratio of number of majority class data to number minority class data and f represents a scaling factor to be tuned for a given data set. Learning of imbalanced class data is performed in chapter V.

2.5 Support Vector Regression

The Support Vector Regression (SVR) [45] was developed as an extension to SVM classification algorithms. The SVR model approximates the given input-output data by forming an error boundary (error tube) around the data and solving a similar optimization problem as SVM. The ν-SVR model is considered for regression, as the tradeoff between model complexity and accuracy (controlled by ν) can be tuned to the required accuracy and sparseness [46, 47]. Sparseness can be defined as the ratio of the number of support vectors to the total number of data observations in the model. The following ϵ-insensitive loss function is used in this work. The ϵ-insensitive loss penalizes data such that any data that falls within the error tube of size ϵ has a zero error while the data falling outside is penalized linearly with its distance from the ϵ tube.

$$L(y-\hat{y})_\epsilon = \begin{cases} 0 & \text{if } \mid y-\hat{y} \mid \leq \epsilon \\ \mid y-\hat{y} \mid -\epsilon & \text{otherwise} \end{cases} \tag{2.26}$$

The goal of SVR training is to determine the optimal model parameters (w^*, b^*) by solving the following optimization problem

$$\min_{w,b,\epsilon,\zeta_i,\zeta_i^*} \frac{1}{2} w^T w + C(\nu\epsilon + \frac{1}{n}\sum_{i=1}^{N}(\zeta_i + \zeta_i^*)) \tag{2.27}$$

$$\text{subjected to} \begin{cases} y_i - (\langle w, \phi(x_i)\rangle + b) \leq \epsilon + \zeta_i \\ (\langle w, \phi(x_i)\rangle + b) - y_i \leq \epsilon + \zeta_i^* \\ \zeta_i, \zeta_i^*, \epsilon \geq 0 \end{cases} \tag{2.28}$$

for $i = 1, .., N$. It should be noted that the slack variables take values of zero when the points lie inside the error tube. Also, separate slack variables ζ and ζ^* are assigned for points lying outside the error tube on either side of the function. The above optimization problem is usually referred as the primal problem and the variables w, b, ζ, ζ^* and ϵ are the primal variables. In the above formulation (2.27), ϵ is considered as a variable to be optimized along with the model parameters. This allows ν to set a lower bound on the fraction of data points used in parameterizing the model [48] and hence by tuning ν one can achieve a tradeoff between model complexity (sparseness) and accuracy. A value of ν close to unity tries to shrink the ϵ tube and reduce sparseness (all data points become support vectors) while reducing ν close to zero will result in a sparse model (very few data points are used in model parametrization) with possible under-fitting. Such a flexibility is the prime reason for selecting the ν-SVR algorithm in this work.

On applying the first order optimality conditions, we can convert the primal problem (equation (2.27) and (2.28)) to the following dual optimization problem

$$\max_{\alpha_i,\alpha_i^*} \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)y_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(x_i, x_j) \tag{2.29}$$

$$\text{subjected to} \begin{cases} \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) = 0 \\ \sum_{i=1}^{N} (\alpha_i^* + \alpha_i) \leq \nu C \\ 0 \leq \alpha_i \leq \frac{C}{N} \\ 0 \leq \alpha_i^* \leq \frac{C}{N} \end{cases} \tag{2.30}$$

for $i = 1, .., N$. Solving the dual problem yields α_i and α_i^* giving the following SVR model

$$\hat{y}(x) = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) K(x_i, x) + b \tag{2.31}$$

where ϵ and b can be determined using (2.28). This is the well known SVR model and the following are some known properties. The parameter w can be completely described as a linear combination of functions of the training data (x_i) determined as support vectors. Unlike the SVM algorithm, the support vectors of a SVR algorithm constitutes all the observations that fall outside the error tube. The model is independent of the dimensionality of $\mathcal{X}$ and the sample size n and the model can be described by dot products between the data. The function ϕ is not required to be known but any kernel function that satisfies the Mercer's condition such as radial basis functions (2.32), polynomial (2.33) and sigmoidal functions (2.34) can be used. In this work, along with the above kernels, the linear kernel (pure inner product) (2.35) is also used.

$$K(x_i, x_j) = e^{-\omega \|x_i - x_j\|^2}, \omega > 0 \tag{2.32}$$

$$K(x_i, x_j) = (\omega \langle x_i, x_j \rangle + c_0)^{deg}, \omega > 0 \tag{2.33}$$

$$K(x_i, x_j) = tanh(\omega \langle x_i, x_j \rangle + c_0), \omega > 0 \tag{2.34}$$

$$K(x_i, x_j) = \langle x_i, x_j \rangle = x_i^T x_j \tag{2.35}$$

The structure and working of the SVR model can be seen in Fig. 2.9. The data operation inside the dotted rectangle is implicitly performed using the kernel functions.

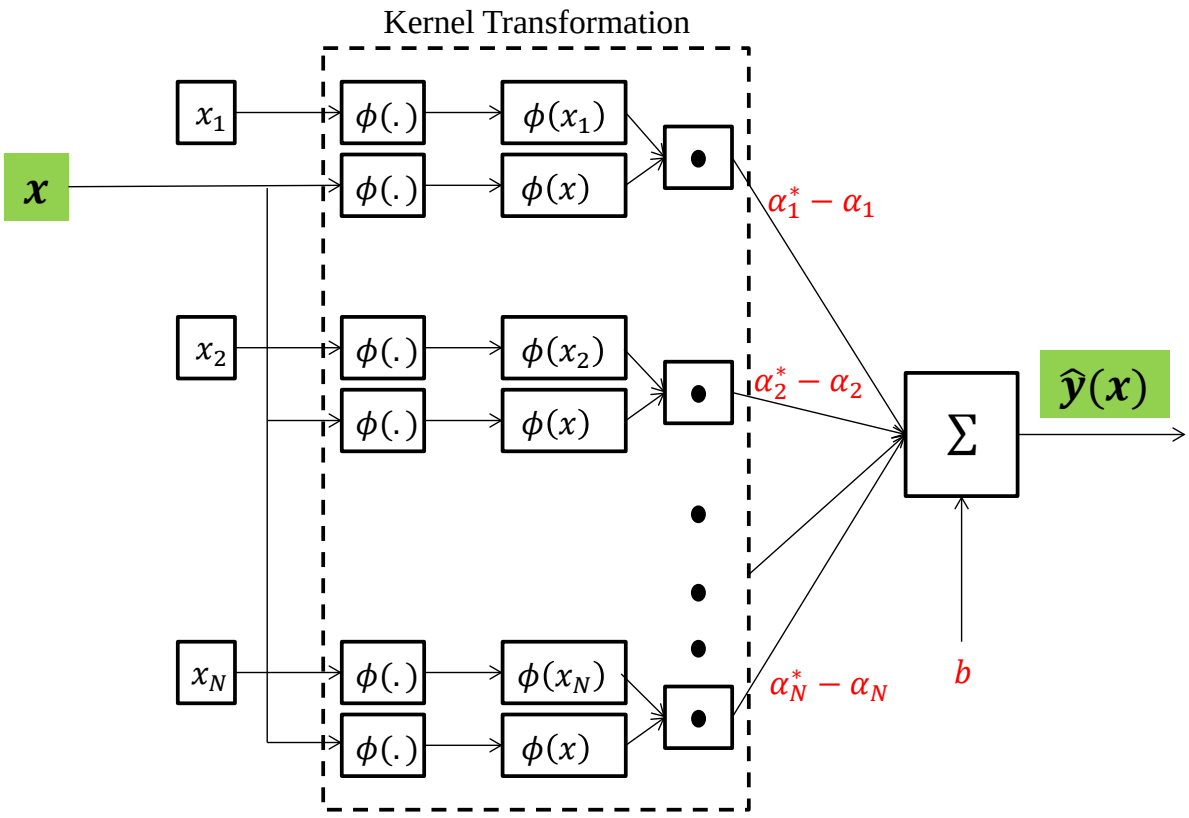


Figure 2.9: Figure showing the working of support vector regression model.

2.6 Extreme Learning Machines

Extreme Learning Machine (ELM) is an emerging learning paradigm for multi-class classification and regression problems [49, 50]. The highlight of ELM is that the training speed is extremely fast (or training is computationally efficient). The key enabler for ELM's training speed is the random assignment of input layer parameters which do not require adaptation to the data. In such a setup, the output layer parameters can be determined analytically using least squares. Some of the attractive features of ELM [49] include the universal approximation capability of ELM, the convex optimization problem of ELM resulting in the smallest training error without getting trapped in local minima, closed form solution of ELM eliminating iterative

training, better generalization capability of ELM.

2.6.1 Difference between Neural Networks and Extreme Learning Machines

ELM models have the exact same structure of a traditional neural network (see Section 2.3) but there are subtle differences that make the ELM models significantly powerful compared to ANNs. In a traditional ANN model, both the input layer parameters and the output layer parameters are determined in the training process. As a result, a nonlinear least squares approach is required that has limitations in slow iterative training and landing in local minima. ELM overcomes these limitations by decoupling the tuning of the hidden layer parameters from the main training process. As the input layer parameters are randomly assigned, the training involves determining the output layer parameters only. The input data is projected onto a high dimension space using the hidden neurons and the randomly initialized parameters where the nonlinear data is transformed to a linear pattern. This effect is similar to using kernel functions in SVM modeling. Finally, the high dimensional linear data is used to model the outputs using a linear least squares method. Since training involves a linear least squares and the objective function is convex, the solution obtained by ELM is extremely fast and is a global optimum. The ELM structure can be seen in Fig. 2.10.

2.6.2 ELM - Mathematical Background

ELM training involves solving the following optimization problem

$$\min_{W} \left\{ \|HW - Y\|^2 + \lambda \|W\|^2 \right\} \tag{2.36}$$

$$H^T = \psi(W_r^T x(k) + b_r) \in \mathbb{R}^{n_h \times 1} \tag{2.37}$$

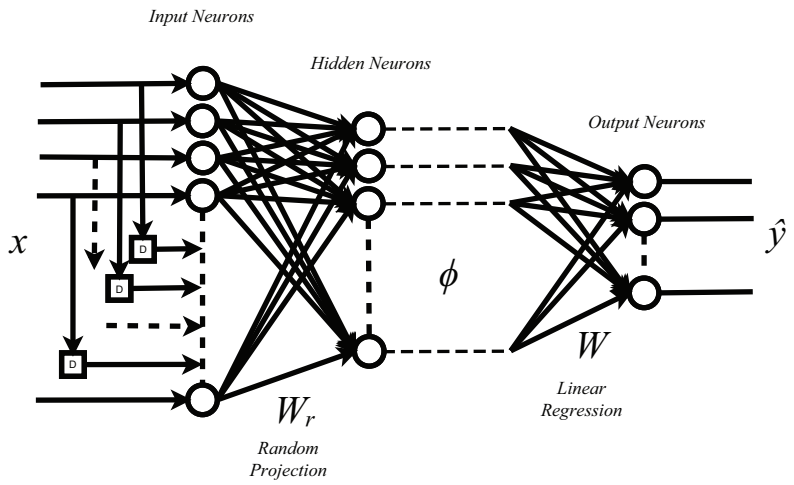


Figure 2.10: Extreme Learning Machine model structure.

where λ represents the regularization coefficient, Y represents the vector of outputs, ψ represents the hidden layer activation function (a sigmoidal function takes the same structure as (5.2)) and W_r, W represents the input and output layer parameters respectively. Here, n_h represents the number of hidden neurons of the ELM model, H represents the hidden layer output matrix.

A prominent feature of ELM is that the nonlinear optimization problem is reduced to a linear parameter estimation. This reduction is made possible by the random assignment of the input layer parameters. The matrix W_r consists of randomly assigned elements that maps the input vector to a high dimensional feature space while $b_r \in \mathbb{R}^{n_h}$ is a bias component assigned in a random manner similar to W_r. The number of hidden neurons determine the dimension of the transformed feature space. The elements can be assigned based on any continuous random distribution [50] and remains fixed during the learning process. The intuition behind the random parametrization of ELM is as follows. By assigning random weights as described above, along with an activation function, a regressor ϕ with several functional combinations (with different order terms) of the input feature vector x is obtained. For

a complex problem, more hidden neurons are required which can be thought of as introducing more complex feature mappings. When the regressor ϕ dimension is sufficiently high with sufficient inclusion of higher order (nonlinear) features, it can be mapped to the outputs linearly. Hence the training reduces to a single step calculation given by equation (2.38). The ELM decision hypothesis can be expressed as in equation (7.3) for classification and equation (2.41) for regression. It should be noted that the hidden layer and the corresponding activation functions give a nonlinear mapping of the data, which if eliminated, becomes a linear least squares (Linear LS) model and is considered as one of the baseline models in chapter V. It is not difficult to observe from equation (2.36) that a squared loss function is employed and hence ELM works well for regression but the classification performance may be inferior to SVM (see results of chapter V).

$$W^* = \left(H^T H + \lambda I\right)^{-1} H^T Y \tag{2.38}$$

$$f(x) = sgn\left(W^T[\psi(W_r^T x + b_r)]\right) \tag{2.39}$$

where

$$sgn(x) = \begin{cases} 1 & x > 0 \\ -1 & x \leq 0. \end{cases} \tag{2.40}$$

$$f(x) = W^T[\psi(W_r^T x + b_r)] \tag{2.41}$$

For classification problems, the above ELM formulation is not designed to handle imbalanced or skewed data sets. As a modification to weigh the minority class data more, a simple weighting method can be incorporated in the ELM objective function (2.36) as

$$\min_W \left\{(HW - Y)^T \Gamma (HW - Y) + \lambda W^T W\right\} \tag{2.42}$$

$$\Gamma = \begin{bmatrix} \gamma_1 & 0 & . & . & 0 \\ 0 & \gamma_2 & . & . & 0 \\ . & . & . & . & 0 \\ 0 & 0 & . & . & \gamma_N \end{bmatrix}$$

$$\gamma_i = \begin{cases} 1 & \text{majority class data} \\ r.f & \text{minority class data} \end{cases} \tag{2.43}$$

where Γ represents the weight matrix, r represents the ratio of number of majority class data to number minority class data and f represents a scaling factor to be tuned for a given data set. This results in the training step given by equation (5.11) and hypothesis given by equation (5.12).

$$W^* = \left(H^T \Gamma H + \lambda I\right)^{-1} H^T \Gamma Y \tag{2.44}$$

$$f(x) = sgn\left(W^T[\psi(W_r^T x + b_r)]\right) \tag{2.45}$$

where

$$sgn(x) = \begin{cases} 1 & x > 0 \\ -1 & x \leq 0. \end{cases} \tag{2.46}$$

2.7 Comparison of Learning Algorithms

In this section, the algorithms discussed in this chapter are compared and contrasted against each other. Table 2.1 summarizes the main features that are used for evaluation of the algorithms for efficiency and applicability to the HCCI engine problem. It can be seen that the ANN models have a non-convex optimization problem and hence requires iterative tuning methods. The optimal solution is a local minimum and hence is generally not a very efficient algorithm. The SVM models have a convex optimization problem but implementation is not simple requiring iterative solvers and

more importantly, requires significant memory and are not usually suitable for online learning. The ELM models, on the other hand, solves a convex optimization problem and typically using a least squares method resulting in a single step convergence to the global optimal solution. Further, it is highly attractive for the HCCI engine problem as it can handle multiple outputs in a straightforward manner and also can perform efficient online learning as discussed in future chapters.

Table 2.1: Comparison of algorithm properties for suitability to the HCCI engine problem.

	ANN	SVM	ELM
Nature of optimization problem	Non convex	convex	convex
Optimization type	Combined gradient and newtons (iterative)	Coordinate descent (iterative)	least squares (non-iterative)
Nature of optimal solution	Local minimum	Global minimum	Global minimum
Handling multiple outputs	Direct	Indirect (several single output models)	Direct
Online Learning	Simple to Complex	Complex	Simple
Memory required	Less (Parameteric)	More (Non-Parameteric)	Less to more (Parameteric)

CHAPTER III

HCCI Engine - System, Experiment Design and Data Collection

In this chapter, an overview of HCCI combustion and the relevant modeling challenges are given with sufficient details. The experimental setup, design of experiments and sensor measurements for both open loop and closed loop experiments are summarized. The experiments are designed for collecting suitable data for developing dynamic models of HCCI combustion in the chapters to follow.

3.1 HCCI Engine - Background

Majority of conventional automotive engines fall into two categories - spark ignited (SI) engines and compression ignited (CI) engines. In a spark ignited engine, the fuel and air are premixed and combustion is initiated using a spark. The load is controlled using a throttle which regulates the quantity of air that enters the cylinders. The throttled operation reduces the volumetric efficiency of the engine. However, the fuel and air are premixed to form a near-homogeneous mixture that results in a cleaner combustion with low hydrocarbon emissions. The temperature of combustion is relatively high resulting in high nitrous oxides (NO_X) emissions. The CI combustion, on the other hand, operates un-throttled and thus achieves a higher efficiency. The load

is controlled by regulating fuel injection quantity eliminating the need for a throttle. The combustion is initiated using a spray of fuel injected when the in-cylinder air is at a high temperature and pressure. The air-fuel mixture is heterogeneous resulting in high hydrocarbon emissions and particulate matter. Similar to the SI combustion, the peak temperatures in CI combustion is high leading to high (NO_X) emissions.

HCCI combustion attempts to retain the benefits of SI and CI combustion and eliminate their shortcomings in terms of efficiency and emissions. In a HCCI engine, the fuel and air are premixed to form a homogeneous mixture similar to the SI engine and compressed to autoignition similar to a CI engine. Unlike the SI or CI engines, the HCCI engine does not have a direct trigger for ignition. Hence the properties of the gas mixtures before combustion dictates the combustion behavior. In order to keep the peak combustion temperatures low, large volumes of exhaust gas is used. The process of using exhaust gas from the previous combustion cycles is commonly referred to as exhaust gas recirculation (EGR). Consequently, HCCI engines result in a cleaner combustion (homogeneous mixtures), low (NO_X) emissions (high EGR) and high efficiency (un-throttled operation and close to constant volume type combustion). Owing to these attractive features, HCCI engines have been studied in detail both in theory as well as in practical implementation. For further reading on SI, CI and HCCI engines, the reader is referred to [51, 52].

Several means of charge preparation strategies are employed for achieving HCCI including heating or pre-compressing intake air [3, 53], varying compression ratio [54], varying Exhaust gas recirculation (EGR) using variable valve timing (VVT) [55, 8]. However, in this thesis, a recompression strategy based HCCI engine is considered where the EGR from the previous cycle is retained and re-compressed for use in the next cycle [8]. The role of EGR is twofold - it helps in raising the temperature of the fuel-air mixture making it easy for autoignition and it controls the temperature rise within the cylinder. EGR is chemically inert to combustion and acts as a heat

sink absorbing heat energy keeping the peak temperatures to a reasonable value in order to reduce NO_X formation. EGR can be varied using a variable valve actuation (VVA) by closing the exhaust valve appropriately so that the desired volume of EGR is retained.

3.1.1 Modeling Challenges

HCCI is characterized by several complex phenomena such as gas transport, chemical kinetics, heat transfer, gas mixing etc. The absence of a combustion initiator such as a spark make HCCI combustion very sensitive to the mixture properties and physical conditions at which the engine operates [56]. Fundamental HCCI research has shown that the temperature and concentrations of mixture components at intake valve closing (IVC) play a major role in determining the auto-ignition phenomenon in HCCI combustion [1, 57, 58]. It is practically not feasible to measure mixture concentrations and in-cylinder temperatures dynamically. Hence these quantities are typically modeled using simplified physics and experimental correlations [59]. Also there exists a cycle-to-cycle coupling in residual affected HCCI as the exhaust gases from the previous cycle are reused [60, 59, 61, 57]. Hence temperature and concentrations of the combustion products from the previous cycle affect the combustion behavior during the present cycle. The knowledge of combustion reactions, heat release behavior, heat transfer, flow dynamics are required to model the residual properties. As noted earlier, high-fidelity models are able to capture the HCCI behavior accurately but cannot be used for controls purposes. Typically, a control oriented simplified model is used which demands long development time and associated cost in addition to being conservative.

The goal of this research is to model the HCCI behavior using data based learning. Although the engine prototype is available, experimentation and data collection are not straightforward owing to the following reasons. A direct measurement of key

quantities of HCCI combustion is not possible. Dynamically measuring in-cylinder temperatures and mixture concentrations is infeasible or very expensive in the time scale required for transient engine operation [62]. Also, the system is highly nonlinear and has a narrow region of stable operation [63, 64, 65, 66, 67] which makes design of experiments a challenging task. The HCCI instabilities and stable operating boundary are detailed in chapter V. The sensitive nature of HCCI combustion coupled with a narrow region of stable operation makes it extremely challenging to obtain transient data that contains rich information about the system for identification. In addition to the above, practical factors pose further challenges in identification and control including limited available memory and computation on the engine ECU. Noisy measurements and features with high variability increases further complication in selecting a robust identification method. However, if the identification process is made systematic for the HCCI system and benefits proved, it could be a powerful approach for HCCI engine control which is one of the objectives of this work.

3.2 Experimental Setup

The system considered in this work is a four cylinder gasoline HCCI engine with a variable valve timing system whose specifications are listed in Table 3.1. A schematic of the experimental setup and relevant instrumentation is shown in Fig. 3.1. A snapshot of the cylinder pressure trace of one combustion cycle along with valve events and fuel injection events are shown in Fig. 3.2. During every combustion cycle, three distinct events can be defined - the crank angle at intake valve opening (IVO), crank angle at exhaust valve closing (EVC) and crank angle at start of fuel injection (SOI). It should be noted that the fuel mass (FM in mg/cyc) is injected in a region between EVC and IVO defined as the negative valve overlap (NVO). The valve events are measured in degrees after exhaust top dead center (deg eTDC) while SOI is measured in degrees after combustion top dead center (deg cTDC).

Table 3.1: Specifications of the experimental HCCI engine

Engine Type	4-stroke In-line
Fuel	Gasoline
Displacement	2.0 L
Bore/Stroke	86/86 mm
Compression Ratio	11.25:1
Injection Type	Direct Injection
Valvetrain	Variable Valve Timing with hydraulic cam phaser having 119 degree constant duration defined at 0.25mm lift, 3.5mm peak lift and 50 degree crank angle phasing authority
HCCI strategy	Exhaust recompression using negative valve overlap

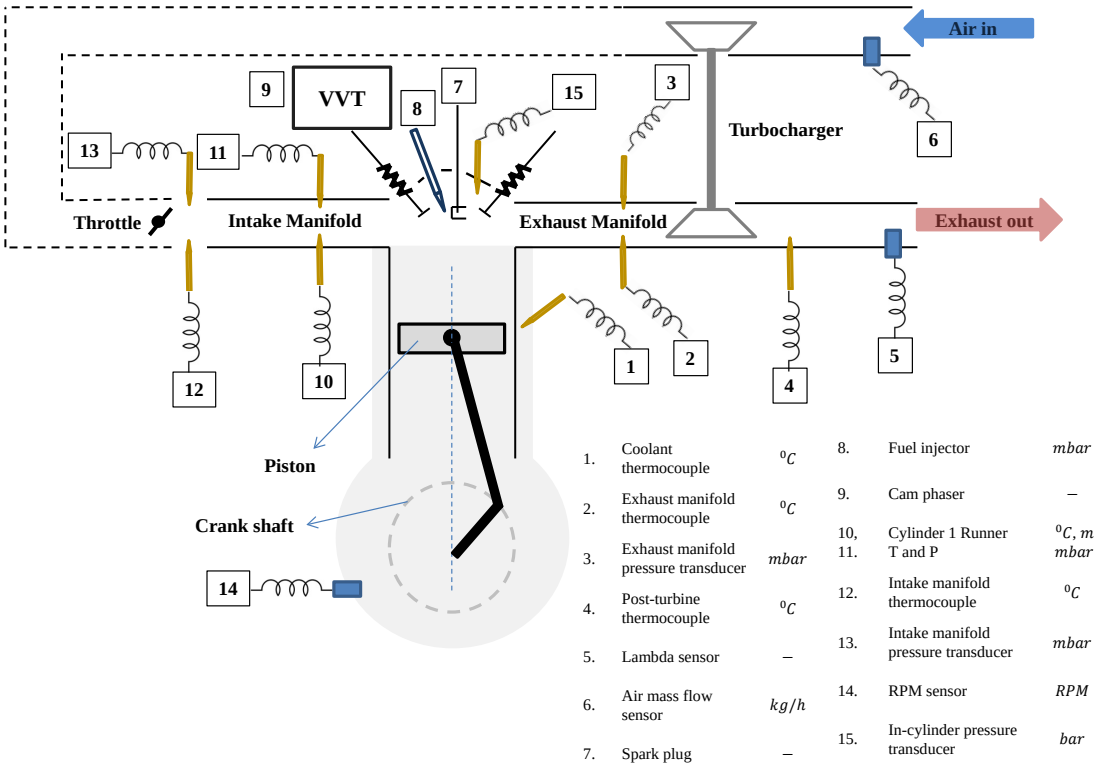


Figure 3.1: A schematic of the HCCI engine setup and instrumentation (only relevant instrumentation is shown).

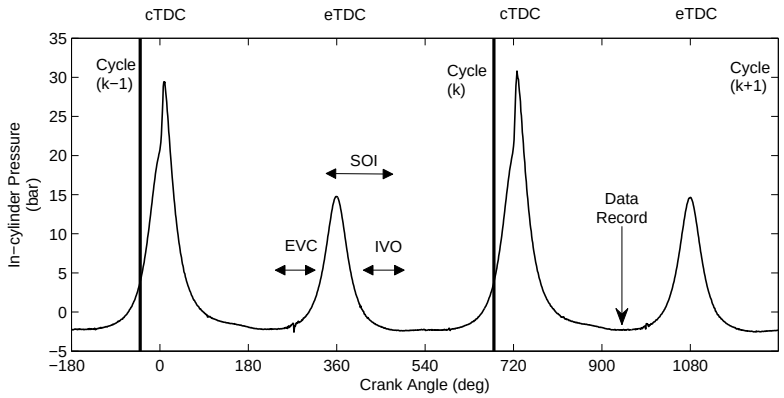


Figure 3.2: HCCI engine pressure trace showing cycle definition, actuator ranges of intake valve opening (IVO), exhaust valve closing (EVC), start of injection (SOI). The crank angle at data recording are also shown.

As mentioned earlier, HCCI is achieved by auto-ignition of the gas mixture in the cylinders. The fuel is injected early (during NVO of previous cycle) and given sufficient time to mix with air forming a homogeneous mixture. A large fraction of the exhaust gas from the previous combustion cycle is retained to elevate the temperature and hence the reaction rates of the fuel and air mixture. The variable valve timing capability of the engine enables trapping suitable quantities of exhaust gas in the cylinder. The engine is operated with natural aspiration, i.e., all experiments in this work involve no or negligible boosting.

For the HCCI identification process, the following measurable quantities are considered as precursors or indicators for the previously mentioned key quantities that cannot be measured directly.

1. The temperature (T_{in}), pressure (P_{in}) and flow rate ($\dot{m}_{in}$) at intake manifold,
2. The exhaust gas temperature (T_{ex}) and exhaust manifold pressure P_{ex},
3. Engine coolant temperature (T_c),
4. Controllable quantities such as FM, IVO, EVC and SOI,

5. Equivalent air to fuel ratio (EAFR) defined as

$$EAFR = \frac{(A/F)}{(A/F)_s} \tag{3.1}$$

where A/F = mass of air per cycle/mass of fuel per cycle and $(A/F)_s = (A/F)$ at stoichiometric condition and

6. Indicators of combustion behavior such as combustion phasing measured as the crank angle at 50% mass fraction burned (CA50), combustion work output measured by indicated or net mean effective pressure (IMEP or NMEP), peak pressure of a cycle (P_{max}) and combustion roughness indicated by maximum rate of pressure rise (R_{max}).

The measurement hardware include a high speed data acquisition system which records IMEP, CA50, P_{max} and R_{max} while the low speed data acquisition system records the other variables listed above. The time scales are different for each measurement and a data post processing is performed to convert the data to a cycle based sampling rate. For many of the experiments performed in this work, the ETAS rapid prototyping hardware is used to code complex algorithms for experiment design and for model learning.

3.3 Steady State Experiments and Mapping

In this section, steady state experiments are performed to develop a map of the feasible HCCI operation. A design of experiments (DOE) matrix is generated by uniformly spanning the input space of the engine (engine speed, IVO, EVC, SOI, FM). The engine is taken to steady state and the minimum and maximum load extremes are reached by tuning FM. Care is taken to operate the engine within constraints such as being lean (EAFR > 1), low combustion roughness ($R_{max} < 6$ bar/deg) and

avoid late burn or misfire (CA50 $<$ 12 before TDC). The space of permissible input combinations for stable HCCI in steady state is obtained using the DOE operating region. The steady state operating region of the HCCI engine can be shown in Fig. 3.3. It should be noted that the engine is capable of running in HCCI beyond the selected speed range.

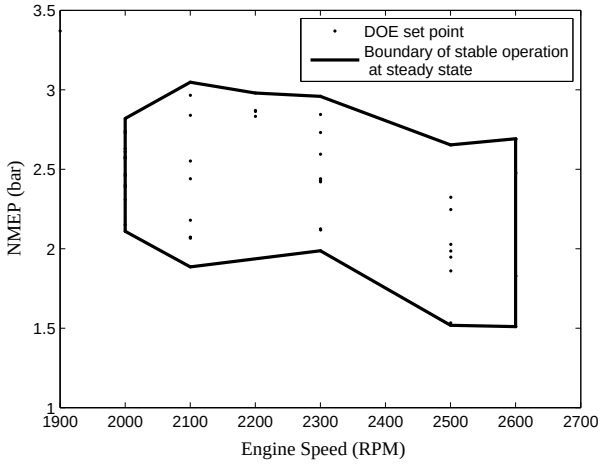


Figure 3.3: Steady state operating region of the HCCI engine.

The purpose of developing a steady state map is to assist the design of transient experiments for collecting step response data for system identification. The data from the steady state DOE is used to develop a steady state map of the engine. A nonlinear extreme learning machine based model is trained to predict engine variables as shown in Fig. 3.4. A sigmoidal model with 15 hidden units is built taking engine speed, IVO, EVC, SOI and FM as inputs. About 55 observations are used for training while the model is evaluated on 23 unseen test data. The model predicts with a mean squared error of 0.0656. The modeling procedure for extreme learning machines are discussed in chapter IV. It can be seen from Fig. 3.4 that the model predicts the HCCI engine variables reasonably well and the model can be used for transient experiment design.

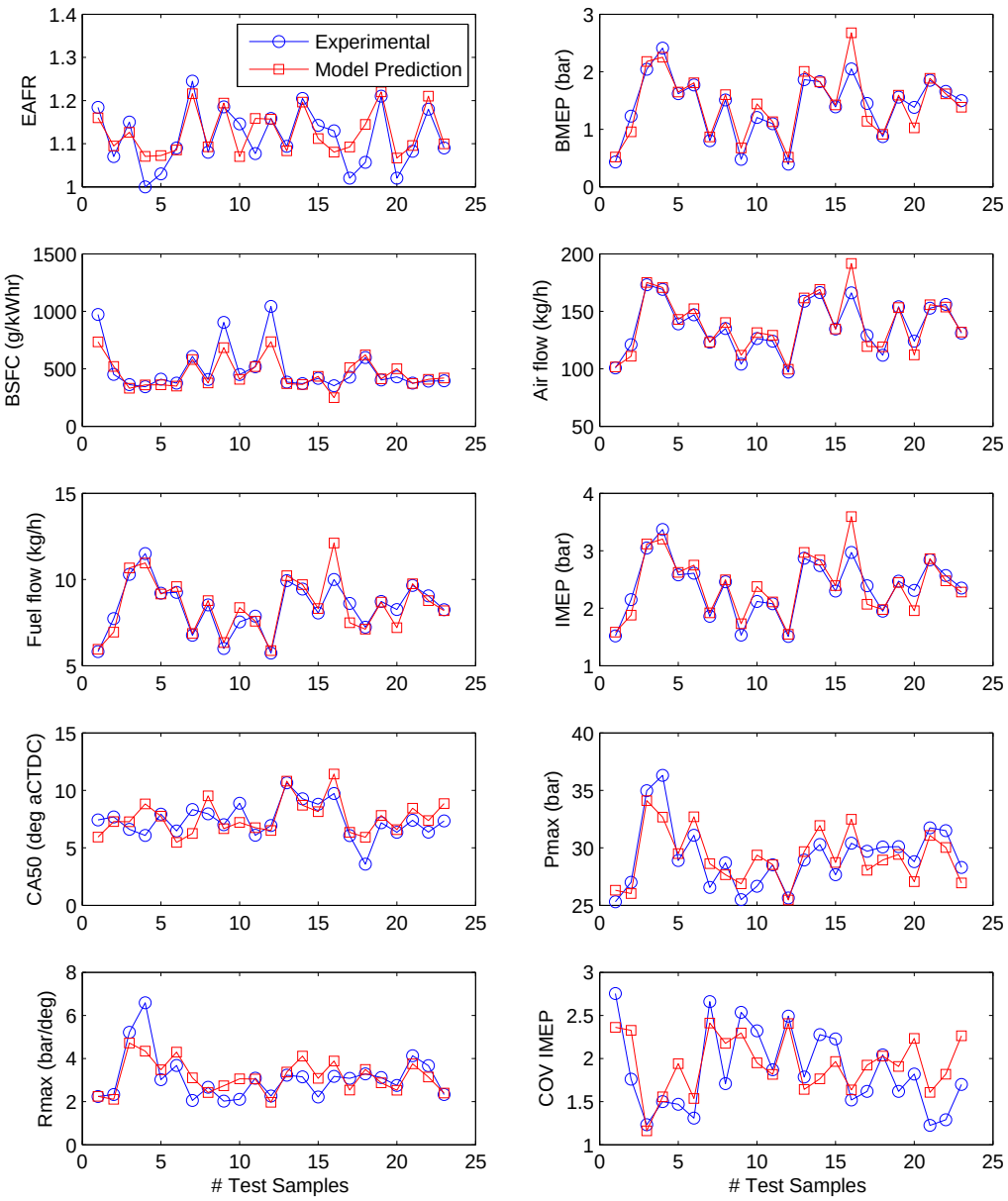


Figure 3.4: Predictions of the ELM based steady state model of the HCCI engine.

3.4 Open Loop Transient Experiments

The goal of this task is to record transient data corresponding to step inputs to the HCCI engine at a constant speed. For system identification of nonlinear systems, typically, an amplitude modulated pseudo-random binary sequence (A-PRBS) is employed to excite the system at several amplitudes and frequencies to obtain sufficiently rich data. The design of excitation signal must satisfy persistent excitation condition as well as exciting the system at all frequencies and amplitudes [68]. For this purpose, the A-PRBS design is employed which has shown to perform well for nonlinear system identification [19].

HCCI has a narrow operating region and a large input excitation close to unstable regions tend to knock, misfire the engine or operate on limit cycles and makes the transient DOE a challenging task. For this purpose, prior knowledge about the system is used to eliminate excitation signals that lead to instability. The steady state DOE model developed in the previous section is used to eliminate the unstable input combinations in the A-PRBS matrix of inputs. Every input from the A-PRBS DOE matrix is used to simulate operating constraints such as R_{max}, CA50 and EAFR and checked for feasibility. Only if the input is found feasible, it is commanded on the experimental engine.

It should be noted that the steady state DOE filtered inputs are valid only for steady state operation and might not guarantee stable transients. Hence as a means of precaution against running the engine in an unstable manner and to avoid restarting the measurements, a simple feedback was created, which attempts a particular input combination and if found to be unstable, quickly skips to the next combination in the A-PRBS sequence. As a first attempt, the CA50 was considered the feedback signal. During a small time window, any input combination that resulted in a CA50 above 11 (found by observing the CA50 during misfires) is immediately skipped, and the engine is run on the next combination in the sequence [69]. Several experiments were

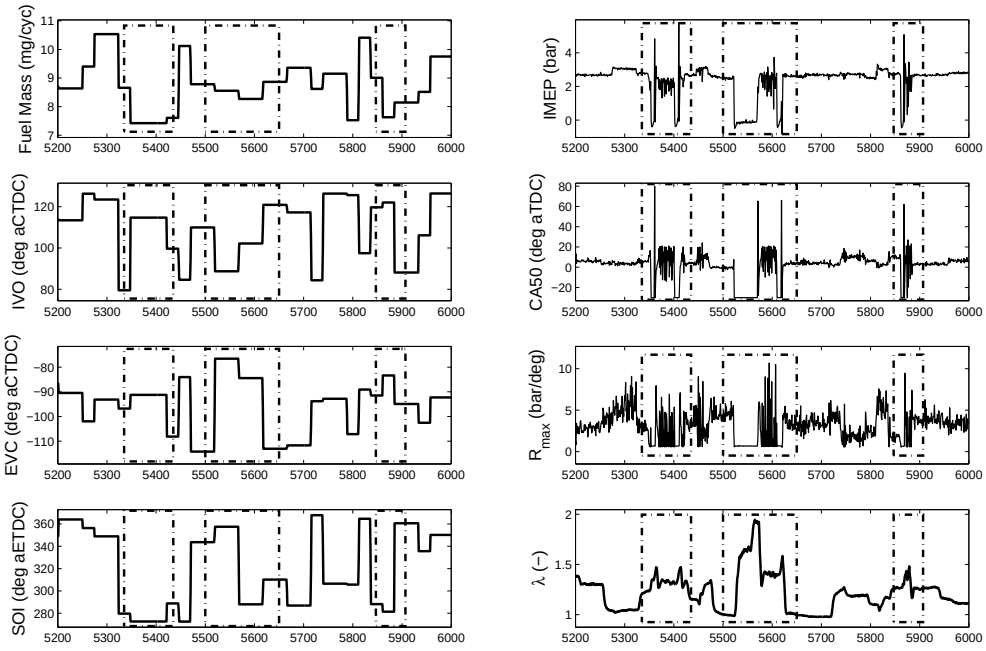


Figure 3.5: A subset of experimental data showing the A-PRBS inputs and the measured engine outputs. Misfire cycles are shown in dotted rectangles.

conducted at constant speeds of 2500-3000 RPM and naturally aspirated conditions. A subset of the data collected from the engine is shown in Fig. 3.5. It can be observed that in spite of eliminating the unstable excitations using a steady state model, engine misfires still occur. This is a major drawback of the open loop experiments where the engine data sets are corrupted with misfire data. A careful post-processing is required to eliminate the misfire and post-misfire cycles [70] before the data can be used for modeling. A more simpler approach is to use a feedback controller and perform the transient experiments in closed-loop. However, this requires the availability of a closed-loop feedback controller.

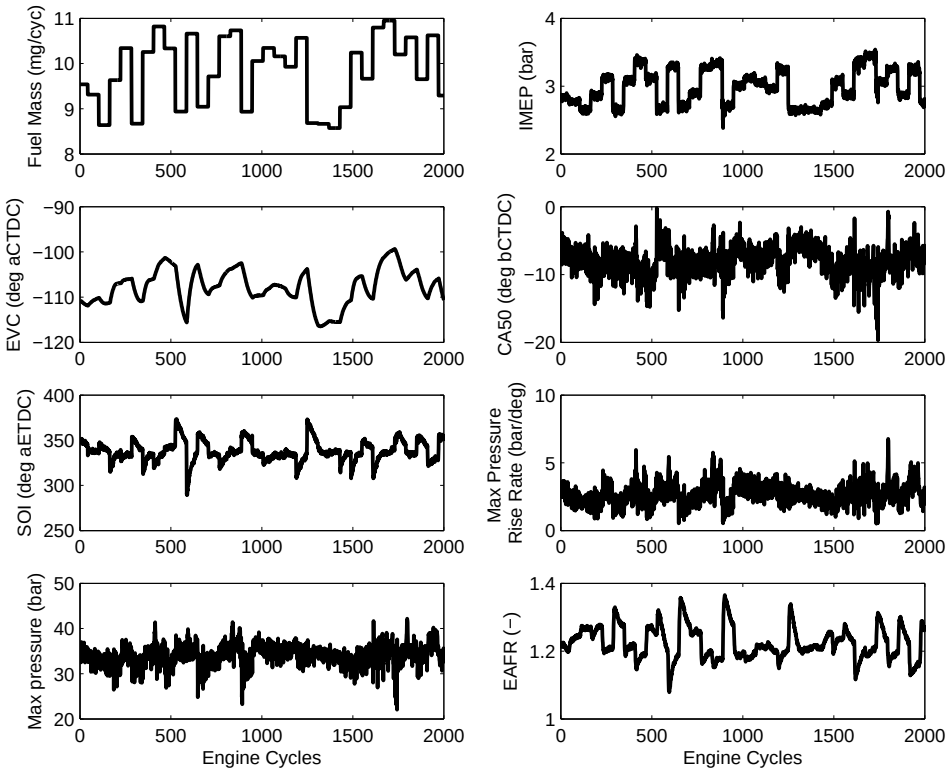


Figure 3.6: A subset of closed loop experimental data showing the A-PRBS inputs and the measured engine outputs.

3.5 Closed Loop Transient Experiments

In this section, the above transient experiments are performed with a feedback controller developed for HCCI [71]. In a similar manner, a DOE matrix is created that varies reference trajectories of IMEP and CA50 in a uniform random manner (A-PRBS pattern). The controller is commanded the designed reference trajectories and the engine response recorded. The closed loop experiments benefit from stable HCCI operation but the excitations are limited to the controller's capability. Several experiments at engine speeds between 1600 and 1800 RPM are conducted. A subset of the closed loop data can be shown using Fig. 3.6. It can be seen that the data corresponds to stable HCCI at a lean operation (EAFR above 1.2). Also, the EVC and SOI are derived from the controller which doesn't follow the A-PRBS pattern.

CHAPTER IV

Development of HCCI Engine Models using Regression Learning

In this chapter, experimental data from the HCCI engine is used to develop predictive dynamic models of HCCI combustion suitable for use in model based predictive control. The above task is not straightforward and a systematic procedure using statistical machine learning techniques is detailed in this chapter. A set of base models (also referred to as offline models) are developed using experimental HCCI engine data. The goal is to develop dynamic simulators that are capable of performing predictions for several steps ahead in time. This chapter is organized as follows. The motivation and goal of offline learning is summarized followed by appropriate data processing for HCCI experimental data. The machine learning task is performed in two parts - model structure learning and model training. Finally, the developed models are evaluated for one-step ahead and multi-step ahead predictions.

4.1 Motivation

For the model-based controls development considered in this research, it is important to obtain predictive models that mimic the HCCI combustion. In particular, the models are aimed to have a short development time, high accuracy, low computa-

tional demand and a capability for dynamic system simulations be performed onboard the engine ECU. This motivates the development of an HCCI engine simulator using machine learning based identification. It is not trivial to perform this task using advanced learning algorithms and a framework is necessary to develop predictive models from HCCI transient data.

4.2 Data Processing

For any machine learning task, the data needs to be appropriately scaled and centered. This includes normalization of all data, e.g. to lie typically between -1 and +1, to ensure model parameters are of the same order for numerical stability. For the HCCI system, additional preprocessing needs to be done before the data becomes meaningful for learning. The unstable nature of HCCI and designing experiments to reduce unstable excitations has been discussed in Chapter III. However, it is not possible to eliminate misfires completely during the open loop experiments and this data needs to be removed before it is used for learning.

A subset of the HCCI data with misfires (boxed in dotted lines) is shown in Figure 5.1. During a misfire, majority of the fresh air that enters the cylinder appears at the exhaust as there is no combustion. Hence the value of λ increases indicating high concentration of air and less fuel. Also, the IMEP (indicator for work output) drops to zero as no positive work is done. The value of CA50 drops to -30 (set by the measurement system during no combustion) while R_{max} drops to 0 as there is no significant variations in pressure. During a regular combustion event, some of the exhaust gas is trapped by the NVO for the next cycle but when a misfire occurs, there is not enough exhaust gas in the NVO trapped mixture for the subsequent cycles. Hence even though the sequence following the misfire cycles is stable, it may not represent a regular HCCI behavior. Hence along with the misfire data, some of the post-misfire data are eliminated.

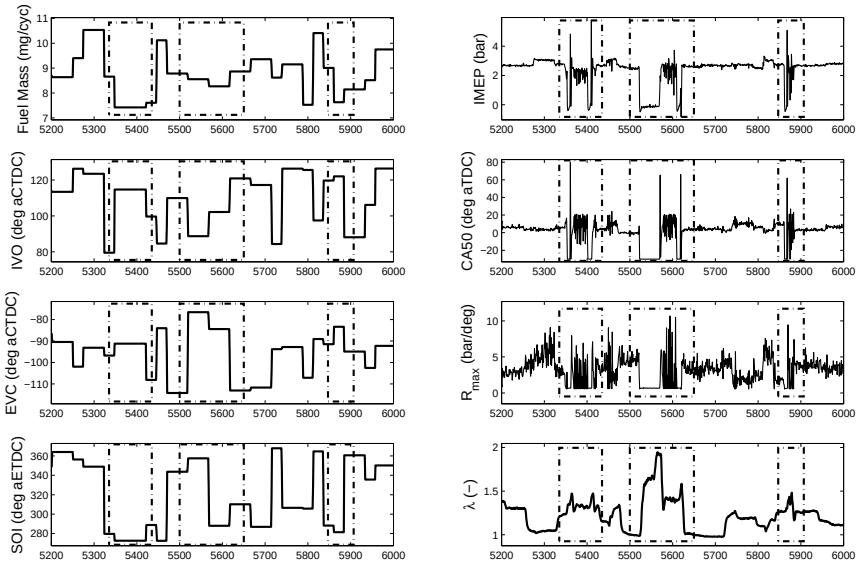


Figure 4.1: A subset of open loop experiments showing A-PRBS inputs and HCCI engine outputs. Misfires are shown using rectangular dotted lines.

When chunks of data is removed from a time series, the time connection is lost and the data set cannot be considered as continuous time series for learning. To overcome this problem, the time series data is converted to static data and then inappropriate data observations corresponding to misfires are removed as follows.

The combustion variables can be used to determine the validity of regular HCCI operation. For instance, a misfire can be identified using a combination of IMEP and CA50 (refer Chapter V). The identification model structure (2.5) and regressor definition (2.6)) allows time series data to be converted to feature vectors to be mapped on to the output variable. Hence, after the regressor conversion, the data can be treated as static (no time connection between observations). It is for the same reason that it becomes possible to use static regression models for time series data. The static data $\{(x_1, y_1), ..., (x_N, y_N)\}$ can be filtered using expert knowledge about the engine. The minimum and maximum limits of the vectors x and y are used to identify and eliminate data that does not represent HCCI combustion. The

minimum and maximum limits of the variables are shown in Table 4.1. The data is normalized based on the variable limits to lie between -1 and +1. Any pair (x,y) whose normalized value lies outside -1 and +1 is considered inappropriate data and removed from the data set. It should be noted that the above data processing is required for data collected in open loop where the controller is not used. For the closed loop experiments, misfires do not occur and the above processing is not required.

Table 4.1: The minimum and maximum values of regression variables x and y determined based on expert knowledge for HCCI conditions.

		Min	Max			Min	Max
FM	mg/cyc	6.9	11.2	P_{in}	bar	0.85	1.4
IVO	aETDC	78	128	P_{ex}	bar	1	1.2
EVC	aETDC	-119	-69	$\dot{m}_{in}$	kg/h	91	370
SOI	aCTDC	270	380	IMP	bar	0.5	4
T_c	^oC	89	93	CA50	aCTDC	1	11
T_{in}	^oC	49	70	R_{max}	bar/deg CA	0	6
T_{ex}	^oC	349	440	λ	-	1	2

4.3 Model Selection

The processed data set is used for model development using algorithms described in Chapter II. The modeling task involves two stages - model structure learning (tuning of hyper-parameters) and model training (tuning of parameters). For the model structure learning, a cross-validation based approach is presented which makes no priori assumptions about the model or the distribution of the data. The cross-validation process involves the following steps. First, the data is divided into three disjoint sets - training set, validation set and testing set. A matrix of hyper-parameter combinations are made and for each combination, the model is trained using the training set and prediction performance evaluated using the validation set. The optimal hyper-parameters are selected as the combination that resulted in the lowest validation error. Finally the model is evaluated with the optimal hyper-parameters using the

testing set for generalization performance using a root mean squared error metric as given by (4.1). The hyper-parameters and their effect on learning behavior of the models considered are briefed below.

4.3.1 ANN

The hyper-parameters of ANNs include number of hidden neurons (n_h), regularization coefficient (λ), system order ($n_o = n_u = n_y$) for MLP model and number of hidden neurons (n_h), gaussian spread parameter (σ_h), system order ($n_o = n_u = n_y$) for the RBN model. The number of hidden neurons (n_h) determines the model complexity, i.e, as more hidden neurons are added, the model has more degrees of freedom to fit the underlying function and if increased, might result in over-fitting the training data. In order to have a simpler model, a regularization term is included in the objective function (2.11) whose importance is determined by λ. A large value of λ forces the parameters to remain small and close to zero. Hence even if the model is over-parameterized, several of them move close to zero reducing the effect of the excess parameters. However, the relative importance given to reduce the mean squared error is also reduced. Hence a large value of λ reduces over-fitting but may result in a high bias under-fit model. However, a right combination of n_h and λ gives a right balance between accuracy and having a simple model.

The system order n_o affects both over-fitting as well as dimension of the feature vectors. A large value of n_o increases the dimension of $\mathcal{X}$ (see equation (2.6)). Increase in the input feature dimension might complicate the learning task by increasing computational demand, training time and might require more training data to handle the additional complexities in the additional dimensions. Also, a large n_o might result in redundant features and poor generalization performance. The gaussian spread parameter (σ_h) indicates the closeness of the data to the given neuron centers. A large σ_h increases the spread of the gaussian function. Hence at any given location,

Table 4.2: The optimal values of number of hidden neurons (n_h), regularization coefficient (λ in MLP and σ_h in RBN) and system order (n_o) determined using cross-validation. Here E_{val} represents the minimum validation error in the grid search.

	Hyper-parameter	NMEP	CA50	R_{max}	EAFR	Parameter Range
MLP model	n_h	8	10	10	8	{2,..,20}
	λ	0.0001	1	0.01	0.01	{0.0001,0.01,0.1,1,10}
	n_o	2	3	2	4	{1,2,3,4,5}
	E_{val}	0.0316	0.2492	0.2220	0.01	-
RBN model	n_h	200	200	200	120	{2,..,200}
	σ_h	10	1	1	10	{0.0001,0.01,0.1,1,10}
	n_o	3	2	2	4	{1,2,3,4,5}
	E_{val}	0.0346	0.2484	0.2278	0.01	-

the activation functions overlaps significantly resulting in a global effect. If inappropriate, a large σ might add more bias and less variance in predictions. A full grid search was performed over chosen combinations of n_o, n_h, λ for MLP and n_o, n_h, σ for RBN and the combination that had the minimum validation error was chosen as the optimal hyper-parameters. Table 4.2 lists the best combination of hyper-parameters for IMEP, CA50, R_{max} and $EAFR$ which had the minimum validation errors.

4.3.2 SVM

The hyper-parameters of SVM include cost penalty of slack variables (C), kernel parameter (ω), sparseness coefficient (ν), system order ($n_o = n_u = n_y$) for regression and cost penalty (C), kernel parameter (ω), system order ($n_o = n_u = n_y$) for classification problems. The cost penalty C in equation (2.18) influences the effect of outliers. A high value of C tries to push the value of ζ to zero resulting in the decision boundary pushed closer to that particular observation. If the observation is an outlier, a high C pushes the boundary close to the outlier and models an incorrect decision boundary. The sparseness coefficient (ν) affects the size of the error tube created by SVR (see equation (2.27)). A high value of ν shrinks the error tube (ϵ moves close to zero) resulting in an aggressive fitting. If the data is very noisy and with high

amplitudes, an aggressive error tube could result in several of the noisy data to fall outside the error tube ending up as support vectors and the representation might be less sparse. Further, the other components of the objective function in (2.27) is given less importance resulting in possible over-fitting. The kernel parameter (ω) has the same effect of gaussian spread parameter in ANN.

A full grid search was performed over the chosen combinations of n_o, C, ω, ν and the combination that had the minimum validation error was chosen as the optimal hyper-parameters. Table 4.3 lists the best combination of hyper-parameters for IMEP, CA50, R_{max} and EAFR which had the minimum validation errors. The training times were very long for polynomial kernel models while the training time for linear, sigmoidal and gaussian kernels were comparable. Hence the models with linear, sigmoidal and gaussian kernels were retrained with the optimal hyper-parameters on the entire training data set while the ones with polynomial kernels were retrained on a subset of the training set.

4.3.3 ELM

The hyper-parameters of ELM include number of hidden neurons (n_h), regularization coefficient (λ), system order ($n_o = n_u = n_y$) similar to ANNs. In addition, the randomized input layer parameters (W_r and b_r) can be considered as hyper-parameters. By varying the random initialization, different network complexity, accuracy and generalization levels can be achieved. However, there is no realizable effect on one initialization over the other as long as the initializations are done based on a uniform probability distribution [50]. The effect of different initializations is not considered in this chapter.

A full grid search was performed over all possible combinations of n_o, n_h, λ in the selected range and the combination that had the minimum validation error was chosen as the optimal hyper-parameters. Table 4.4 lists the best combination of hyper-

Table 4.3: The optimal values of system order (n_o=n_u=n_y, assumed to be the same), the cost parameter C, kernel parameter ω and SVR parameter ν determined using cross-validation from the range of listed values.

Kernel Type	Hyper-Parameter	IMEP	CA50	R_{max}	EAFR	Range
Linear	n_o	2	5	5	4	{1,2,3,4,5}
	C	1	0.1	0.1	100	{0.01,0.1,1,10,100}
	ν	0.2	0.6	0.4	0.2	{0.1,0.2,..,1}
	E_{val}	0.0727	0.2637	0.2597	0.0162	-
Polynomial of degree 2	n_o	2	3	4	4	{1,2,3,4,5}
	ω	1	1	1	1	{0.001,0.01,0.1,1,10,100}
	C	1	1	10	1	{0.01,0.1,1,10,100}
	ν	0.4	0.4	0.3	0.5	{0.1,0.2,..,1}
	E_{val}	0.0663	0.2793	0.3098	0.0155	-
Polynomial of degree 3	n_o	2	2	1	3	{1,2,3,4,5}
	ω	0.5	0.2	1	0.5	{0.01,0.1,0.2,..,1,5,10}
	C	0.5	0.5	1	1.5	{0.01,0.1,0.2,...,2}
	ν	0.5	0.4	0.4	0.3	{0.1,0.2,..,1}
	E_{val}	0.0670	0.2690	0.2583	0.0168	-
Sigmoidal	n_o	2	5	5	3	{1,2,3,4,5}
	ω	0.01	0.01	0.01	0.01	{0.001,0.01,0.1,1,10,100}
	C	1	1	1	1	{0.01,0.1,1,10,100}
	ν	0.6	0.4	0.4	0.4	{0.1,0.2,..,1}
	E_{val}	0.0692	0.3093	0.3054	0.0214	-
Gaussian	n_o	2	3	1	2	{1,2,3,4,5}
	ω	0.1	0.1	1	0.1	{0.001,0.01,0.1,1,10,100}
	C	1	1	1	1	{0.01,0.1,1,10,100}
	ν	0.3	0.4	0.4	0.3	{0.1,0.2,..,1}
	E_{val}	0.0611	0.2525	0.2183	0.0199	-

parameters for IMEP, CA50, R_{max} and $EAFR$ which had the minimum validation errors.

Table 4.4: The optimal values of number of hidden neurons (n_h), regularization coefficient (λ) and system order (n_o) determined using cross-validation. Here E_{val} represents the minimum validation error in the grid search.

Kernel Type	Hyper-Parameter	IMEP	CA50	R_max	EAFR	Range
Sigmoidal	n_h	65	90	100	70	{5,..100}
	λ	0.1	0.01	0.01	0.01	{0.001,0.01,0.1,1,10,100}
	n_0	2	2	3	3	{1,2,3,4}
	E_{val}	0.0582	0.1257	0.1497	0.0257	-
Gaussian	n_h	50	55	70	60	{5,..100}
	λ	0.01	0.01	0.001	0.01	{0.001,0.01,0.1,1,10,100}
	n_0	2	3	2	3	{1,2,3,4}
	E_{val}	0.0596	0.1265	0.1527	0.0305	-

4.4 Model Evaluation

The models developed in this chapter are intended for use in a predictive control framework where real time predictions are made in the engine ECU. In order to evaluate the prediction capability of the models, a root mean squared error (RMSE) given by equation (4.1) is used. Other evaluation criteria include number of registers required to store sensor measurements, number of parameters of the model and potential for online adaptation. The models are trained using a series-parallel architecture [22] and one-step ahead prediction performance is evaluated on the unseen testing data set. Finally, the model predictions are evaluated for n_{pred} steps ahead in time using a separate input sequence. Both the testing set as well as the input sequence for multi-step ahead predictions are never seen by the model during training. This gives a good measure of the long term predictions as well as the generalization capability

of the models to unseen situations.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{y_d}(y_j^i - \hat{y}_j^i)^2} \tag{4.1}$$

4.4.1 One-step Ahead Prediction

The one-step ahead predictions (OSAP) of the models are evaluated and discussed in this section. The OSAP performance can be used to evaluate the models in a series-parallel architecture. Although the ultimate design is a parallel architecture for long term predictions, the OSAP performance can be used to evaluate the efficiency of model training. The different variants of the ANN and SVR models (with different activation functions in ANN and with different kernel functions in SVM) are analyzed for prediction performance using the open loop experimental data.

4.4.1.1 ANN Model Predictions

It can be observed from Table 4.5 that both MLP and RBF networks are able to learn the HCCI combustion dynamics to a good accuracy for one-step-ahead prediction. As a baseline, a linear regression model is selected and trained using the data set after performing cross-validation based hyper-parameter selection similar to the neural models. All three model structures - MLP, RBN and Linear regression have a similar order of accuracy for the considered engine variables. It can be seen that for most of the engine variables, the MLP models achieves a minimum testing error. Also, the MLP model results in small system orders compared to the other two architectures. The number of parameters of MLP models are significantly less compared to RBN models because RBN models store the centers of the radial basis neurons. It should be noted that the linear regression surprisingly performs well for one-step-ahead prediction for the considered problem. However, a large system order is required for the models indicating that the data has to remain in a high dimension

Table 4.5: Summary of prediction performances of MLP and RBN models. The results of linear regression model is also included as a baseline. Here n_m and n_p represents the number of memory units and number of model parameters required for prediction. The training and testing errors are for one-step ahead prediction while MSAP RMSE indicates the root mean squared error for multiple-step ahead prediction. The minimum error among linear, MLP and RBN models is highlighted in bold.

		NMEP	CA50	R_{max}	EAFR
MLP model	Training Error	0.0332	0.2335	0.2175	0.0361
	Testing Error	**0.0374**	**0.2466**	**0.2300**	0.0374
	n_m	22	33	33	22
	n_p	131	161	171	131
	MSAP RMSE	**0.0671**	**0.2804**	**0.2638**	0.0742
RBN model	Training Error	0.0346	0.2375	0.2074	0.0300
	Testing Error	0.0424	0.2551	0.2373	**0.0361**
	n_m	22	33	22	22
	n_p	2401	1921	2401	2401
	MSAP RMSE	0.1311	0.3918	0.3076	**0.0707**
Linear Regression	Training Error	0.0400	0.2702	0.2423	0.0693
	Testing Error	0.0412	0.2778	0.2369	0.0686
	n_m	55	55	55	11
	n_p	19	18	21	9
	MSAP RMSE	0.1510	0.3401	0.2841	0.1375

feature space in order for linear models to capture the underlying behavior. More crucially, the linear models did not perform well under multi-step-ahead predictions (see MSAP error in Table 4.5 and Figures 4.2-4.5) and hence linear models are found to be unsatisfactory for identifying the combustion behavior of HCCI. The nonlinear models (MLP and RBN) perform well for MSAP with the MLP models winning by a slight margin over the RBN models.

An important comparison between MLP and RBN models can be made with respect to the memory required (n_m) and the total number of parameters (n_p) used to fit the data. Memory in this context is referred to as the space required to store the measurement signals (like FM, SOI, NMEP, CA50 etc.) for prediction on-board in the engine control unit. The total number of parameters can be thought of as the static memory space in the ECU that reads the model parameters. The memory requirement is given by $n_m = u_d n_u + y_d n_y$ while the number of parameters is given by $n_p = n_i n_h + n_h + n_h y_d + y_d$ for MLP network and $n_p = n_i n_h + n_h y_d + y_d$ for RBN models. However, it should be noted that the number of hidden neurons, n_h is very large in case of RBN i.e., to achieve similar performance levels of MLP network, RBN requires extremely large number of parameters to fit the data. With further increase in the dimension of the signals and the number of observations, the size of the RBN model can increase significantly which can limit the RBN models from being implemented on the engine ECU. It should be noted that the RBN model is relatively faster to train. However the training time is not considered as a metric for comparison as training can be afforded to be done off-line as presented in this chapter. Hence the MLP model is considered suitable for the HCCI identification problem both from prediction accuracy and storage perspectives.

4.4.1.2 SVM Model Predictions

Similar to the ANN evaluations, the SVR models are simulated with the unseen test data set and performance of the models are measured using an RMSE metric as in (4.1). Table 4.6 compares the performance of the models in terms of training and testing RMSE, memory units and the number of parameters required. The number of parameters represent the sparsity as controlled by the parameter ν. Several possible kernel functions (see Section 2.5) are evaluated for the HCCI engine data and its suitability evaluated.

Table 4.6: Performance comparison of SVR for modeling IMEP, CA50, R_{max} and EAFR. Here, n_m and n_p represents number of memory units and number of parameters (support vectors) required by the models.

Kernel Type		IMEP	CA50	R_{max}	EAFR
Linear	Training RMSE	0.0686	0.2621	0.2571	0.0173
	Testing RMSE	0.0735	0.2678	0.2555	0.0173
	n_m	20	50	50	50
	n_p	73502	330825	280335	168212
Polynomial of degree 2	Training RMSE	0.0693	0.2451	0.2398	0.0127
	Testing RMSE	0.0663	0.2793	0.3098	**0.0155**
	n_m	20	30	40	40
	n_p	46486	70455	79200	125488
Polynomial of degree 3	Training RMSE	0.0707	0.2651	0.2494	0.0141
	Testing RMSE	0.0671	0.2691	0.2583	0.0173
	n_m	20	20	10	30
	n_p	57068	45012	22517	56265
Sigmoidal	Training RMSE	0.0742	0.3082	0.3103	0.0245
	Testing RMSE	0.0693	0.3093	0.3055	0.0245
	n_m	20	50	50	30
	n_p	218372	220330	279400	192456
Gaussian	Training RMSE	0.0632	0.2396	0.2079	0.0173
	Testing RMSE	**0.0608**	**0.2525**	**0.2181**	0.0173
	n_m	20	30	10	20
	n_p	113652	149787	69861	102982

It can be seen from Table 4.6 that all the considered kernel models capture the dynamics of IMEP, CA50, R_{max} and EAFR to a reasonable accuracy but with different

memory and storage requirements. Gaussian kernel outperforms all the other kernels in terms of both achieving maximum accuracy as well as with relatively low memory and storage requirements. This can also be seen in Table 4.3 where gaussian kernel identifies the system with a relatively lower order (n_o) for all the response variables considered. The performance of all the kernels are comparable for IMEP with similar memory requirements. Indeed, the linear kernel uses less parameters compared to the gaussian kernel. This may be attributed to the simpler mechanism behind IMEP which strongly depends on the fuel mass injected [72] and can be potentially identified using linear models. It should be noted that the polynomial kernel models are tested on a smaller data set and hence the number of parameters are low. Overall, the gaussian kernel is chosen as appropriate for modeling the HCCI engine behavior.

4.4.2 Multiple-step Ahead Prediction

In order to observe the multi-step-ahead prediction capability of the models, a completely separate data set is used where only the input trajectories are given to the model along with the initial conditions of the outputs (delay initial conditions). The OSAP models (series-parallel architecture) are converted to a MSAP form (parallel architecture) by feeding back the predictions of the OSAP model in a recurrent manner as follows

$$\hat{y}(k+n_{pred}) = \hat{f}_{NARX}[u(k+n_{pred}-1),..,u(k-n_u+n_{pred}), \\ \hat{y}(k+n_{pred}-1),..,\hat{y}(k-n_y+n_{pred})] \quad (4.2)$$

where k indicates the present time index. An output feedback is made in the network to create a parallel architecture as shown in Figure 2.6. The MSAP performance of the considered models are discussed as follows.

4.4.2.1 ANN Model Predictions

The figures 4.2, 4.3, 4.4 and 4.5 compares the 200-cycle predictions of the MLP and RBN models for unseen input trajectories. In each plot, four different input trajectories are presented and the predictions compared. The engine variables are plotted every combustion cycle and each plot shows the predictions for about 8 seconds of engine data. The RMSE for the multi-step ahead predictions are included in Table 4.5.

It can be seen from Figures 4.2-4.5 and Table 4.5 that the linear models do not perform in MSAP as well as they did for OSAP. The reason could be insufficient capability of the linear models to capture nonlinear system behavior. Also, the model output is fed back for subsequent predictions in a recurrent manner. It is important that the model possesses sufficient robustness and any poor predictions not affect the future predictions in a significant manner. The linear models appear to be lacking this feature compared to the nonlinear models. Another reason could be over-fitting as the linear models fits the data with a high order when low order models [7, 8] were found sufficient for the HCCI variables.

It is also surprising to observe that the RBN models do not perform well at several operating conditions including Figure 4.2 (top left subplot between 100 and 200 cycles, top right subplot between 0 and 50 cycles), Figure 4.3 (bottom left subplot between 50 and 120 cycles) etc. A possible reason could be the local nature of the approximation captured by RBN. The local nature of RBNs might result in over-fitting (inability to generalize) if there are not enough training data in certain regions of interest. MLP models on the other hand perform reasonably well compared to the other two architectures. Both the steady-state and transient behavior are well captured. The good performance of MLP models may be attributed to the global nature of the sigmoidal activation function. Hence it is concluded that MLP models are suitable for the considered HCCI modeling problem in terms of both one-step-

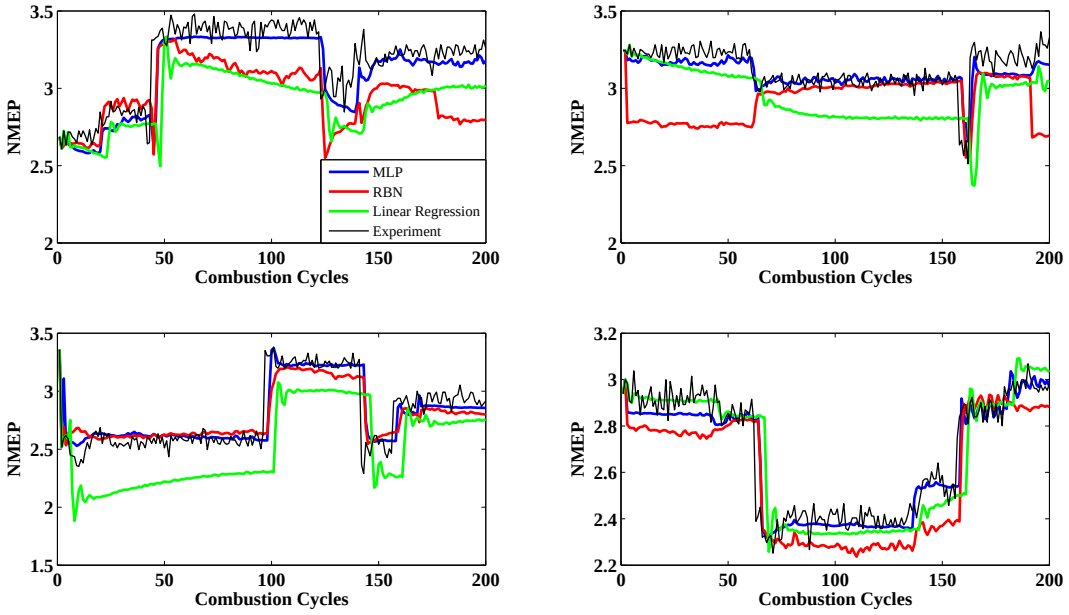


Figure 4.2: Comparison of 200 step ahead prediction for IMEP by MLP, RBN and linear models with actual engine data.

ahead and multi-step-ahead predictions.

4.4.2.2 SVM Model Predictions

Similar to the neural models, the OSAP SVR models are converted to MSAP models for performing long term dynamic simulations. For MSAP evaluations, the gaussian kernel models chosen from the previous section has been used. The MSAP RMSE for the gaussian models are 0.0598, 0.2980, 0.2383 and 0.0636 for IMEP, CA50, R_{max} and EAFR respectively. The 200 cycle ahead predictions of IMEP, CA50, R_{max} and EAFR are compared against measured data from the engine in Figures 4.6 - 4.9. Each figure shows predictions of an output variable for four different cases as validations in different operating conditions. It can be observed from the figures that the models are able to predict the HCCI dynamics to a good accuracy and can be used as engine simulators. It can also be observed that both the steady state values and the transients are well captured by the models except for a few regions where

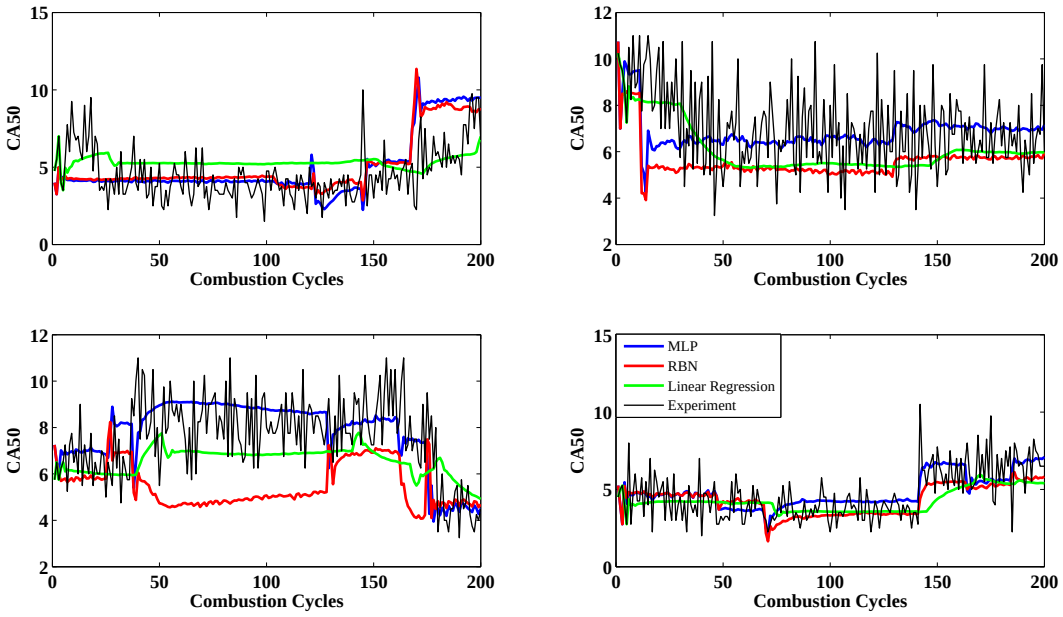


Figure 4.3: Comparison of 200 step ahead prediction for CA50 by MLP, RBN and linear models with actual engine data.

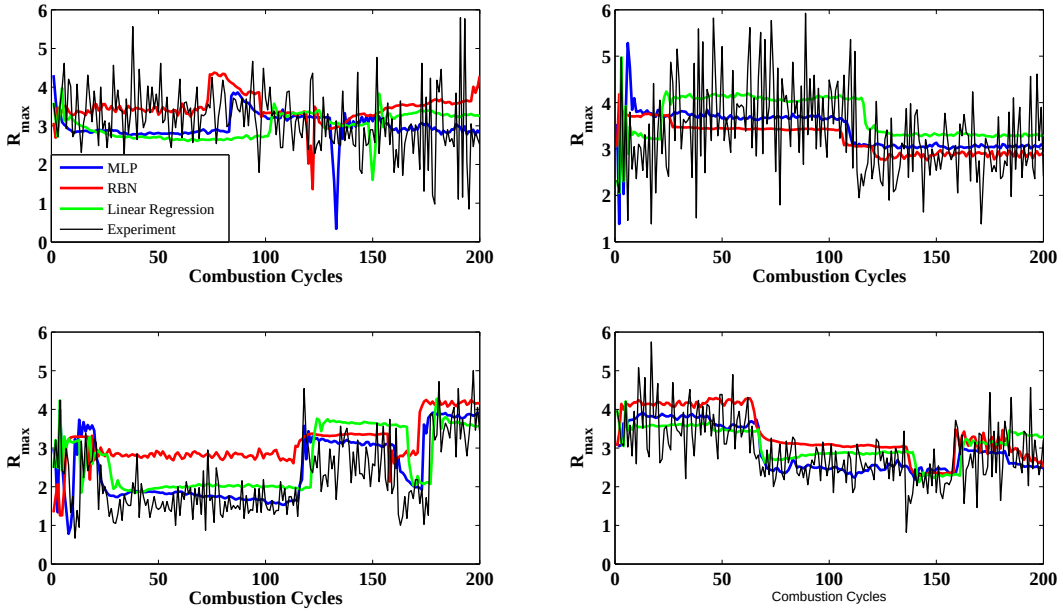


Figure 4.4: Comparison of 200 step ahead prediction for maximum pressure rise rate (R_{max}) by MLP, RBN and linear models with actual engine data.

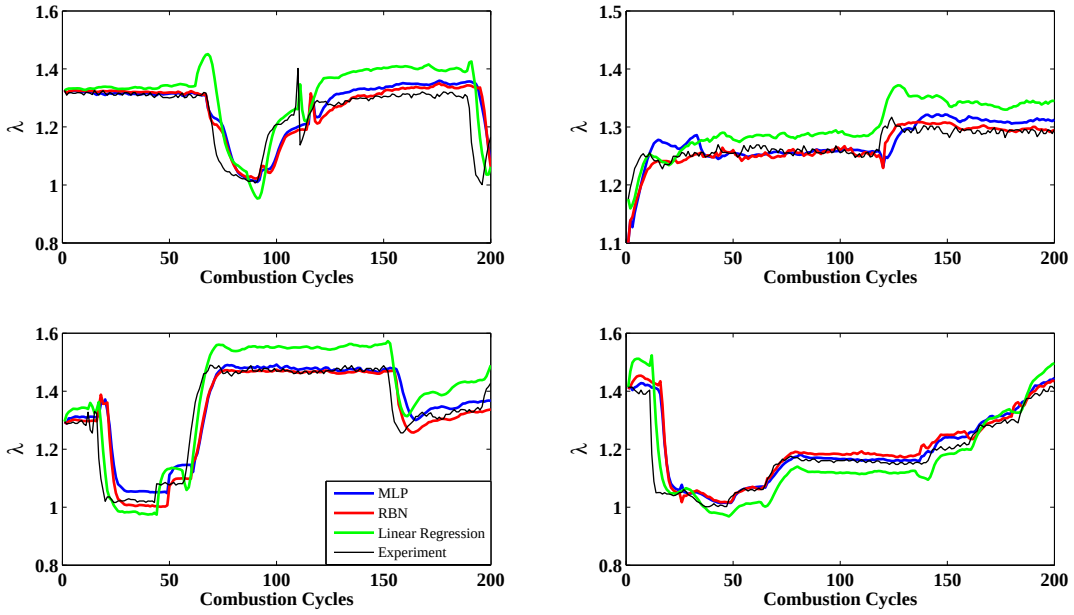


Figure 4.5: Comparison of 200 step ahead prediction for EAFR by MLP, RBN and linear models with actual engine data.

there is a bias offset owing to poor approximations. Lack of excitations near such input combinations could be a reason for the bad predictions of the model.

The MSAP accuracies of SVR and ANN models are similar with SVR models winning by a slight margin. However, it can be noted from Tables 4.5 and 4.6 that the number of parameters required by the SVR models are significantly high compared to the neural models (100-1600 times more) because of its non-parametric nature, i.e., the number of support vectors becomes a fraction of the total data set and if the data set is large, the number of support vectors can increase dramatically. This forms a major drawback of the SVR method although it solves a convex optimization problem and solution is robust. Overall, the SVR models with gaussian kernels are well suited for HCCI identification but might be computationally expensive for engine ECU application.

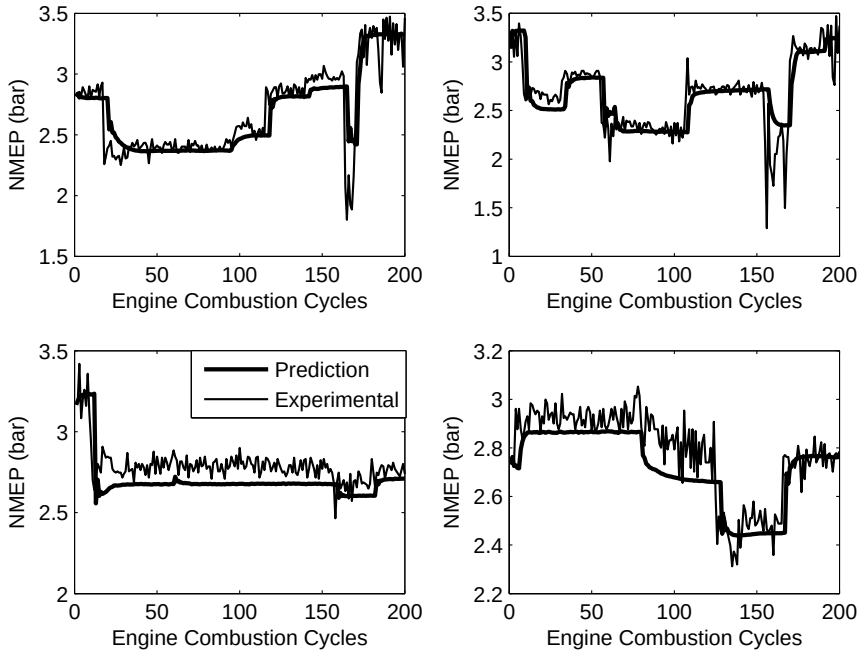


Figure 4.6: Comparison of IMEP (engine output and SVR prediction).

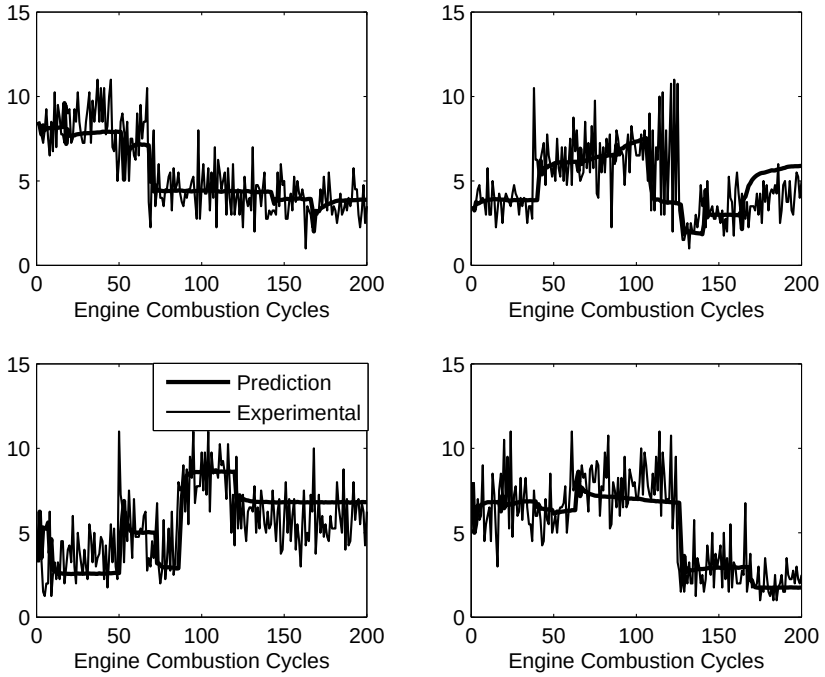


Figure 4.7: Comparison of CA50 (engine output and SVR prediction).

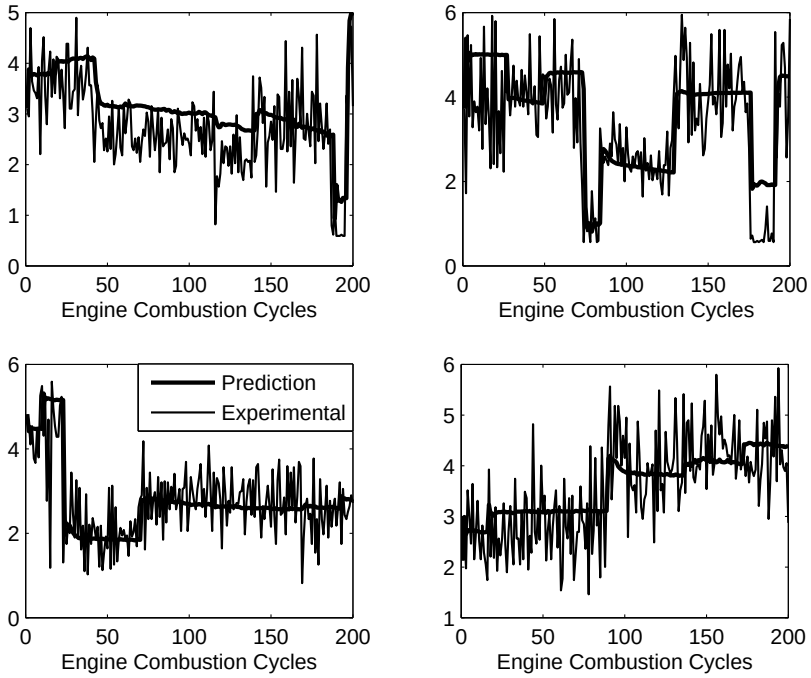


Figure 4.8: Comparison of Maximum rate of pressure rise (engine output and SVR prediction).

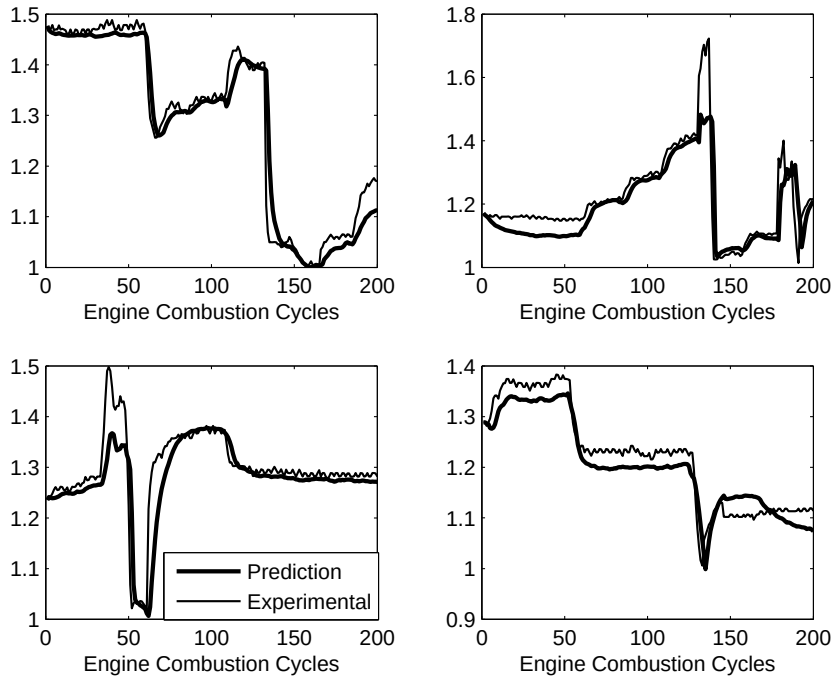


Figure 4.9: Comparison of Lambda (engine output and SVR prediction).

4.5 Model Development using Closed-loop Experimental Data

In this section, the data set obtained from closed-loop experiments (with the controller ON, see chapter III) is used to develop models for controls development. The three models considered in this chapter (ANN, SVM and ELM) are used to build models for IMEP, CA50, R_{max} and EAFR. Table 4.7 summarizes the performance of the considered models. It can be observed that the ELM models outperform the ANN and SVM models both in terms of prediction accuracy and number of parameters used. In spite of being less complex and possible local minima, the ANN models perform similar to the ELM models. However, such high accuracies are not always guaranteed with ANN models as ANN training might find a local minima for a different data set. It should be noted that the obtained results for ELM models correspond to a give set of random parameters of W_r and b_r. A different set of random parameters would result in different accuracy and convergent behavior. In this study, the matrices W_r and b_r are considered as model hyper-parameters and are chosen from a set of 20 different randomizations using cross-validation.

The multi-step ahead predictions are summarized in Figures 4.11-4.14. The models are simulated using a predefined input sequence as shown in Figure 4.10. It is clear that the ELM and ANN predictions are close to the experimental data while the SVR models have a slight offset as indicated in the RMSE values in Table 4.7

4.6 Predictions at Different Engine Speeds

In the above model building task, the experiments were performed at a constant engine speed and machine learning models were developed to simulate unknown inputs at that particular speed. In this section, the modeling task is extended to include multiple engine speeds. Ideally, engine speed has to be considered as an input excitation during the experiment design phase, but owing to the absence of the steady

Table 4.7: Performance comparison of ANN, SVM and ELM models. Here RMSE refers to root mean squared error, OSAP and MSAP refer to one-step ahead prediction and multi-step ahead prediction respectively. The minimum error models are highlighted bold.

Model		IMEP	CA50	R_{max}	EAFR
ANN model	OSAP RMSE	0.0587	0.1329	0.1508	0.0288
	MSAP RMSE	**0.0505**	0.1356	0.1614	0.0481
	#parameters	651	101	101	101
	n_h	65	10	10	5
	λ	0.1	0.01	0.01	0.01
	n_o	2	2	2	2
SVM model	OSAP RMSE	0.0749	0.1357	0.1624	0.0432
	MSAP RMSE	0.0789	0.1384	0.1718	0.0886
	#parameters	4776	19072	19088	28596
	ω	0.01	0.01	0.01	0.001
	C	1	1	10	1
	ν	0.1	0.2	0.2	0.2
	n_o	1	2	2	3
ELM model	OSAP RMSE	0.0582	0.1257	0.1497	0.0257
	MSAP RMSE	0.0536	**0.1284**	**0.1573**	**0.0407**
	#parameters	650	900	1400	980
	n_h	65	90	100	70
	λ	0.1	0.01	0.01	0.01
	n_o	2	2	3	3

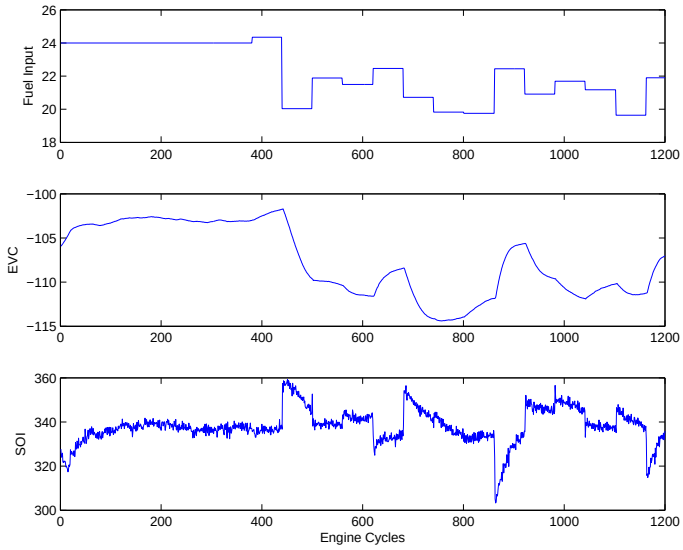


Figure 4.10: Input trajectories for simulating the models for performing multi-step ahead predictions.

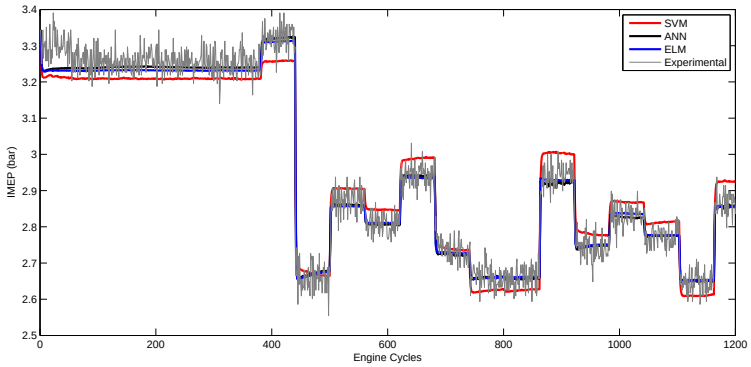


Figure 4.11: Comparison of IMEP predictions by SVM, ANN and ELM models with the experimental engine data.

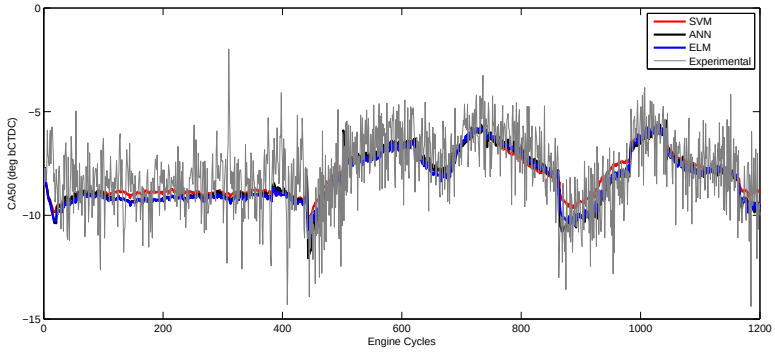


Figure 4.12: Comparison of CA50 predictions by SVM, ANN and ELM models with the experimental engine data.

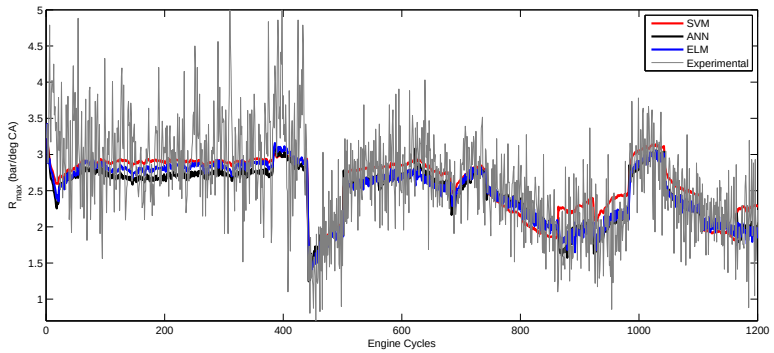


Figure 4.13: Comparison of R_{max} predictions by SVM, ANN and ELM models with the experimental engine data.

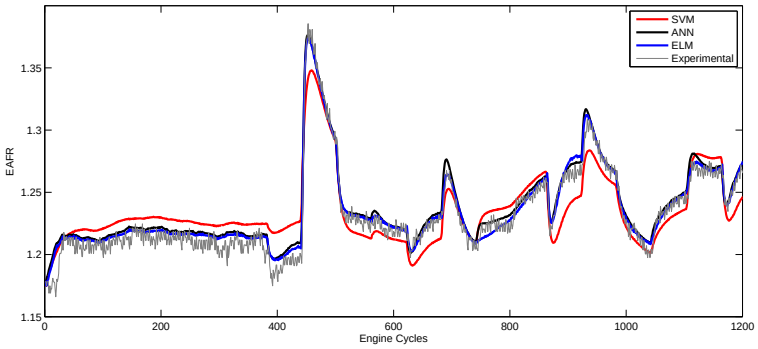


Figure 4.14: Comparison of EAFR predictions by SVM, ANN and ELM models with the experimental engine data.

Table 4.8: Modeling summary for experiments at 1600 RPM and 1800 RPM.

Experiments	1600 RPM	1800 RPM
train data size	10000	12000
test data size	4000	5000
n_h	180	135
n_o	2	2
λ	0.1	0.001
OSAP RMSE	0.1074	0.1085
MSAP RMSE	0.1111	0.1092

state engine dynamometer to perform speed transients, the following approach is used. For demonstration purposes, experimental data at two engine speeds are considered. Closed loop identification experiments are performed at 1600 and 1800 RPM and ELM models are developed in a similar manner as decried in this chapter. The specifications and predictions results are listed in Table 4.8 and multi-step ahead predictions are shown in Figures 4.15 and 4.16 for 1600 and 1800 respectively.

4.6.1 Interpolation Model Approach

The above models perform well for the corresponding engine speeds. However, they are not appropriate to be used for predictions at a different engine speed. To handle this problem, an interpolation based approach is employed. For any speed between 1600 and 1800 RPMs, the predictions are obtained by a linear interpolation between the two models. A different experiment is conducted on the engine by manually varying engine speed between the specified RPMs as shown in Fig. 4.17. The validation of the approach is shown in Figures 4.18 to 4.20. It can be verified that if the original models are used for speeds between 1600 and 1800 RPM, the predictions are bad with an offset. The MSAP errors for the 1600 RPM model, 1800 RPM model and the interpolation models are 1.3968, 1.4146 and 1.3245 respectively which indicates that the interpolation model gives a superior predictions for the intermediate speed data. It can be observed from the prediction figures that when the

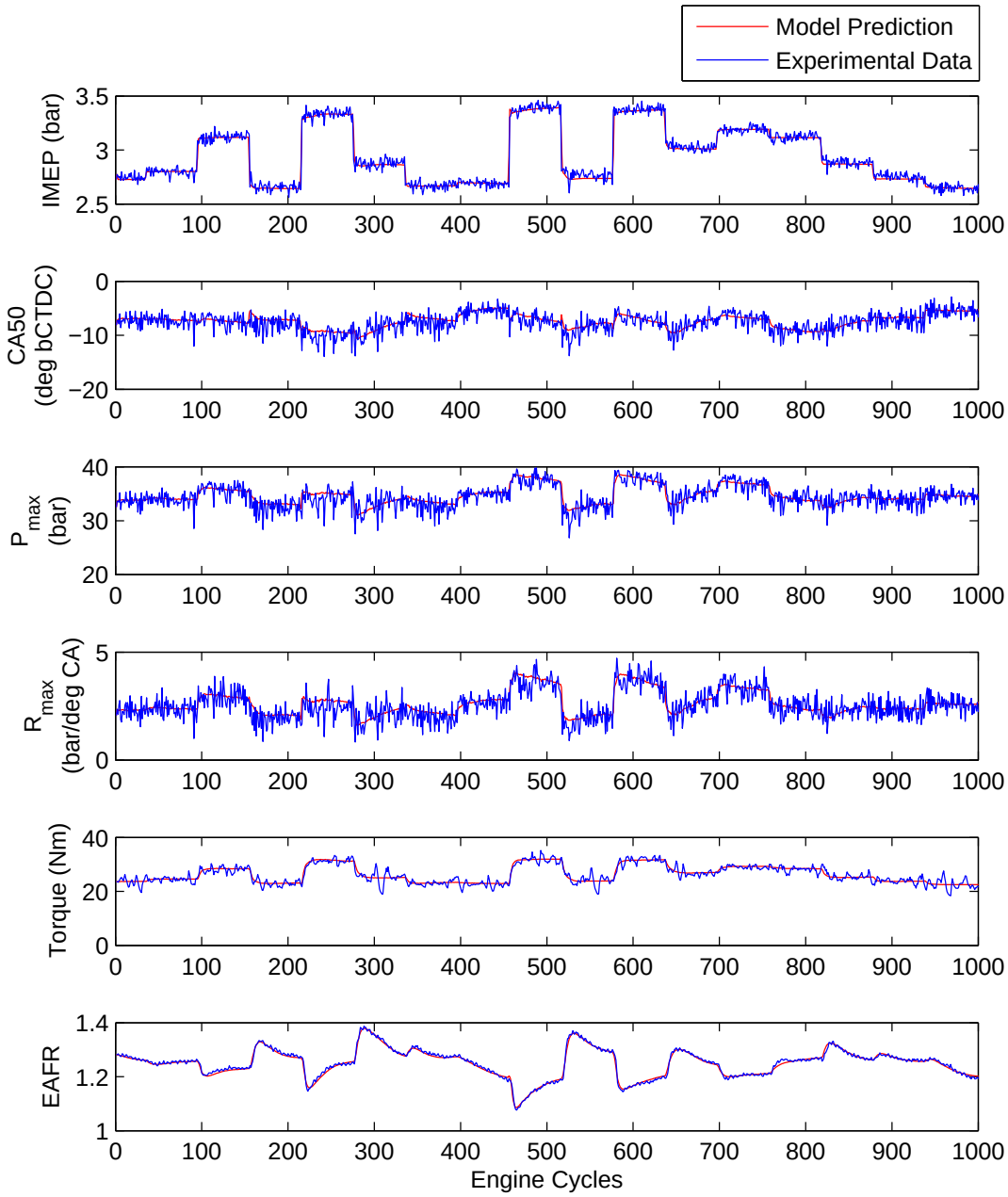


Figure 4.15: Prediction summary of the nonlinear ELM model at 1600 RPM.

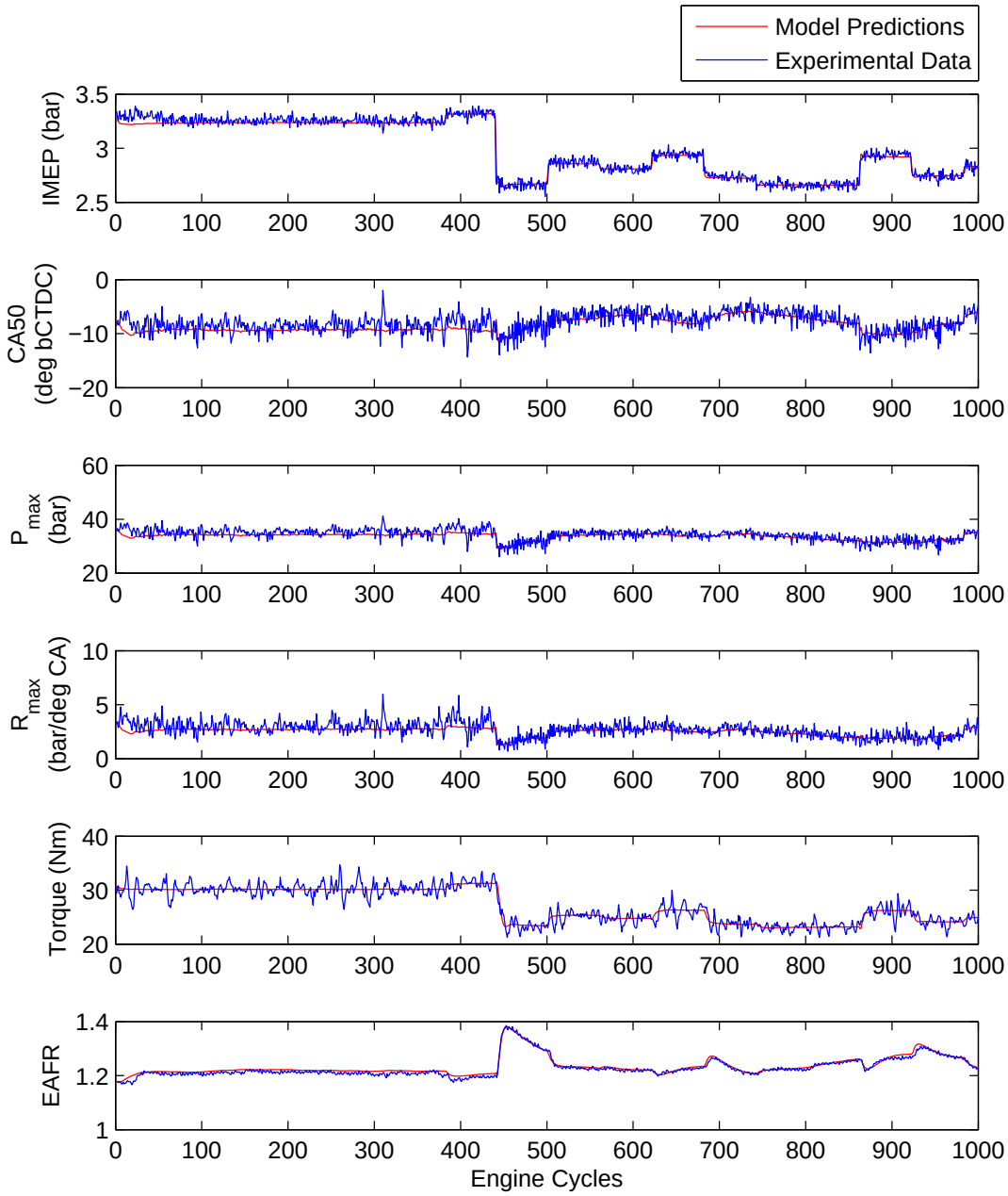


Figure 4.16: Prediction summary of the nonlinear ELM model at 1800 RPM.

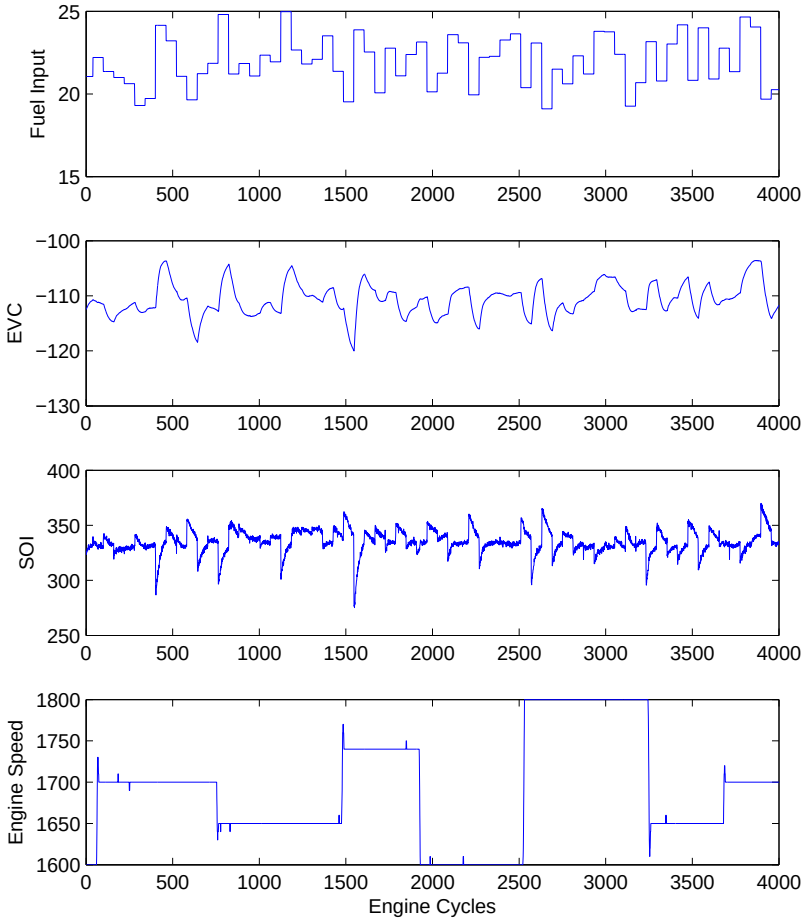


Figure 4.17: Closed loop experiments by varying engine speed between 1600 and 1800 RPM. This data set is used for validating the interpolation model approach.

engine speed is 1600 RPM (see cycles 2000-2500), the predictions of the 1600 RPM model is preferred and the solid red plot matches the blue curve which is close to the experimental values. Similar behavior is observed when the engine speed is 1800 RPM. For the intermediate speeds, the interpolated results are plotted. The approach approximates speed related behavior as a linear model and further work needs to be done to determine the appropriate relationship.

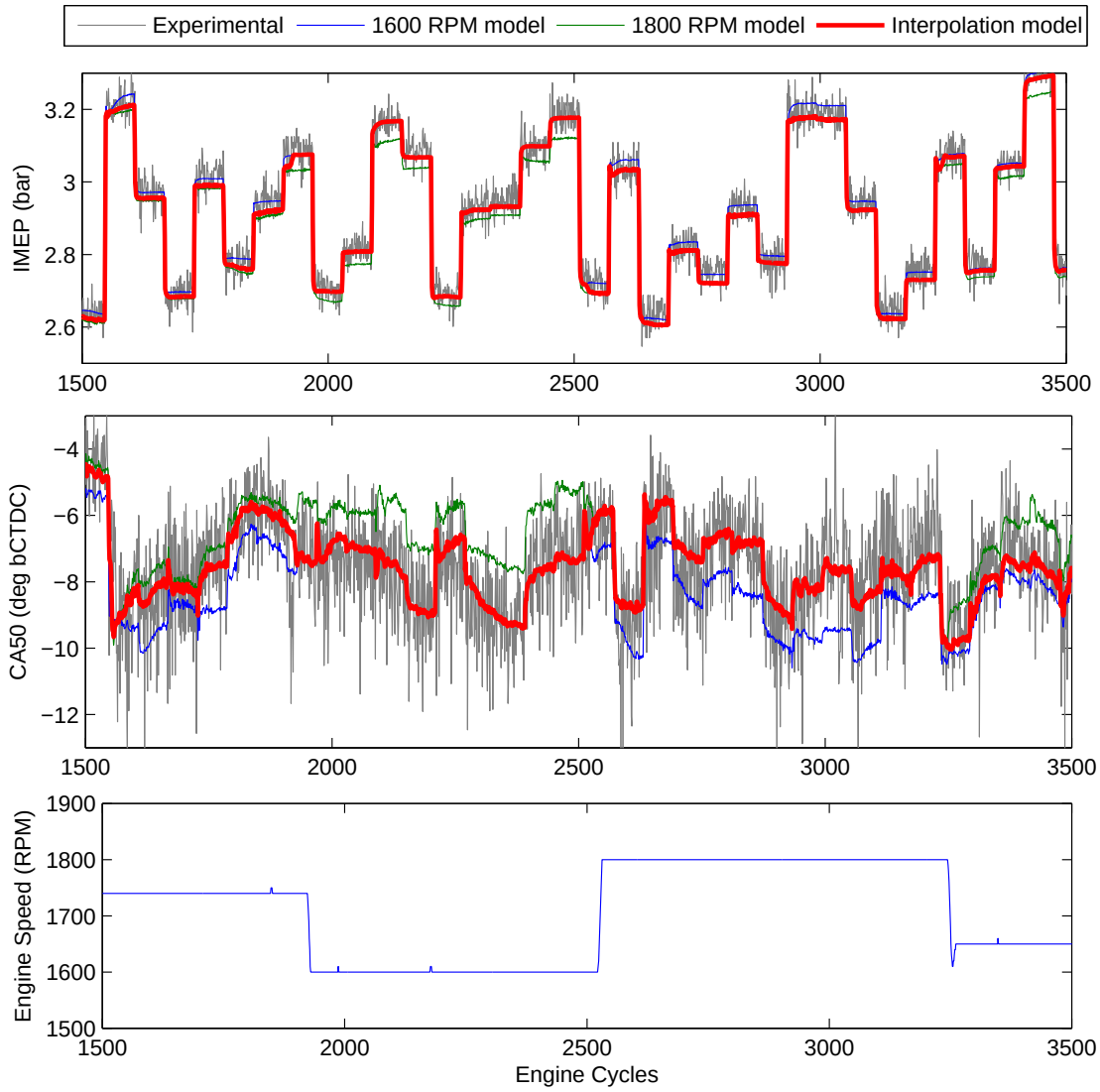


Figure 4.18: Validation of the interpolation model for IMEP and CA50 predictions. The predictions of 1600 RPM model and 1800 RPM models do not perform well when the engine speed is varied.

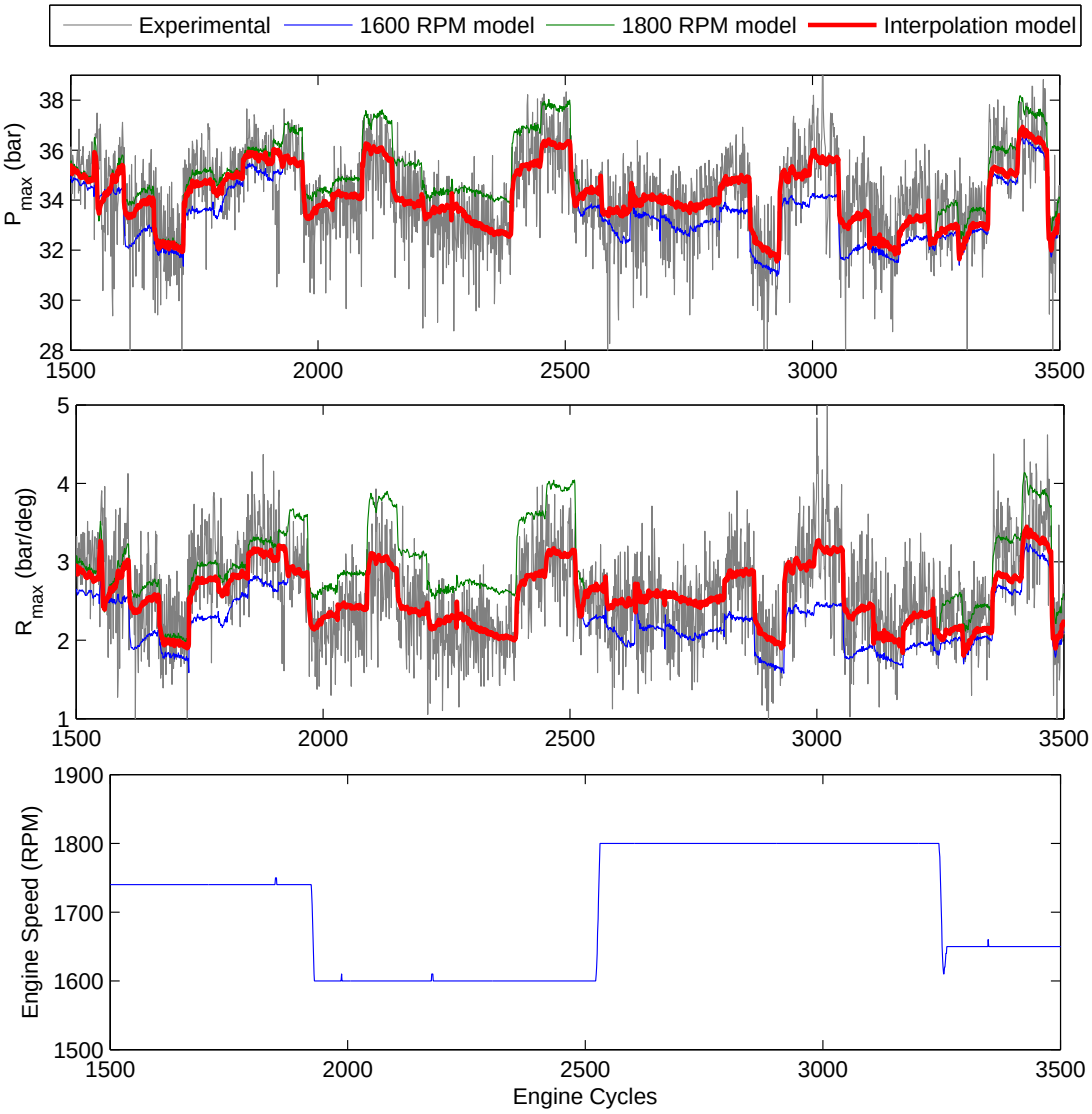


Figure 4.19: Validation of the interpolation model for P_{max} and R_{max} predictions. The predictions of 1600 RPM model and 1800 RPM models do not perform well when the engine speed is varied.

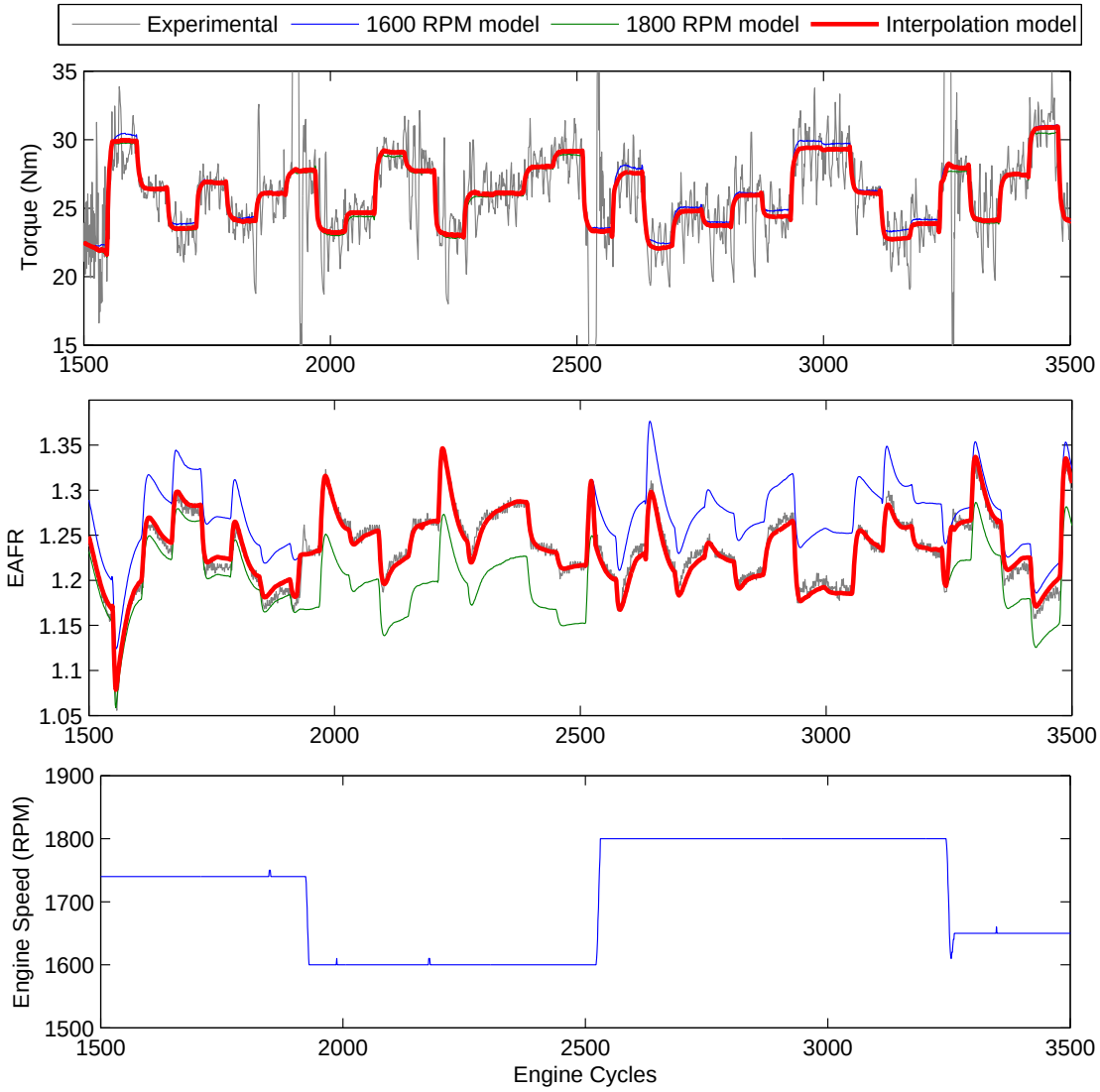


Figure 4.20: Validation of the interpolation model for engine torque and EAFR predictions. The predictions of 1600 RPM model and 1800 RPM models do not perform well when the engine speed is varied.

CHAPTER V

Modeling the HCCI Operating Envelope using Class Imbalance Learning

In the previous chapter, an application of machine learning to HCCI engines was described. Dynamic simulators were developed for engine variables including IMEP, CA50, R_{max}, P_{max}, EAFR, Torque etc. In this chapter, another novel application of machine learning is introduced. HCCI engines have a limited stable operating region owing to system and operating level constraints. For reasons mentioned in this chapter, it becomes important to identify the stable operating envelope of HCCI especially during transient operation. Such a task is extremely challenging when approached from a first principles based approach but adds to an excellent demonstration for a machine learning approach to engine modeling.

5.1 Motivation and Problem Statement

HCCI engines have been studied in the last decade owing to their ability to reduce emissions and fuel consumption significantly compared to traditional spark ignition and compression ignition engines [1, 2, 3]. The highly efficient operation of HCCI is achieved using advanced control strategies such as exhaust gas recirculation (EGR) [73], variable valve timings (VVT) [7], intake charge heating [74] among others. As

a consequence, complex manipulation of the system results in a highly nonlinear behavior [75] and narrow region of stable operation [76, 77].

In order to develop controllers and operate the system in a stable manner, it is imperative that the operating envelope of the system be determined. In general, the operating envelope can be defined as a region in the input space (of permissible values of system actuators for different thermal conditions of the engine) that results in a stable operation of the engine. Knowledge of the operating envelope is crucial for designing efficient controllers for the following reasons. The developer can get insights on the actuator extremes (for example, minimum and maximum fuel injection rates at a given speed and load condition) of the engine especially during transients, for instance, when the cylinder wall temperatures are changing [73]. Such information can be used to enforce constraints on the control variables for desired engine operation. Also, the operating envelope model can act as a filter to perform system identification by eliminating excitations that might lead the system to be unstable. Further, the model can be used to alarm the onboard diagnostics if the engine is about to misfire [78] owing to changes in system or operating conditions.

As mentioned earlier, HCCI engines are very complex systems involving chemical kinetics and thermal dynamics which requires high-fidelity modeling using numerical simulations for capturing accurate combustion behavior [79, 80, 81, 56]. Such an approach is computationally expensive particularly when there are several influencing variables. The situation is worsened by the transient effects, i.e., the variables along with its time history affects the system behavior. The operating envelope that depends on the system variables along with its time history [73] can be considered a dynamic system, and capturing this time varying behavior using conventional methods becomes intractable. Hence an approach using machine learning is considered in this chapter where time series data from the sensors are used to model the operating envelope of the HCCI engine.

The problem of identifying the operating envelope using experimental data reduces to a classification problem. As discussed in future sections, the labeling of stable operating data is well defined while instability is chosen with respect to misfires and high variability combustion. As a result, the problem can be posed as a binary classification and a decision boundary can be modeled using existing classification algorithms. A support vector machine algorithm was used to identify the operating envelope of a GDI engine [82] but the boundary was assumed to be a static system and time history was not considered resulting in a simple binary classification problem. However, for HCCI engines whose combustion behavior is influenced by EGR from previous cycles, the importance of considering the time history of measurements becomes significant [73]. Designing a classifier based on dynamic HCCI data is one of the objectives of this chapter. Also, the experimental data set consists of a large subset of stable class data with limited unstable class data, as misfiring the engine is undesirable for the emission control hardware, a regular classification might not be an appropriate solution since the decision boundary would be biased to one class of data, resulting in over-fitting. Therefore, the other objective of this work is to perform imbalanced class learning on the HCCI engine data. The following two approaches have been compared

1. Heuristic re-sampling of data: apply preprocessing methods such as under-sampling and over-sampling of data to get a balanced data set.

2. Cost-sensitive approach: modify the objective function of the learning system to weigh the minority class data more heavily.

Three classification models including support vector machines (SVM), extreme learning machines (ELM) and logistic regression (LR) have been developed. The models are compared for generalization accuracy, storage required in the engine ECU and potential for online learning.

This chapter is organized as follows. A brief background on the classification algorithms along with cost-sensitive modifications is given in Section 5.2. The HCCI engine experiments and data processing are briefed in Section 5.3 with the envelope modeling and prediction results discussed in Section 5.4.

5.2 Classification Algorithms

Consider the data set $\{(x_1, y_1), ..., (x_N, y_N)\} \in (\mathcal{X}, \mathcal{Y})$ where $\mathcal{X}$ denotes the space of the input features (let $\mathcal{X} = \mathbb{R}^n$) while $\mathcal{Y}$ takes values in {-1,+1} and N denotes the number of observations. The goal of the classification algorithm is to model the underlying boundary separating the data by minimizing a risk function $R(w)$ with respect to the model parameters w.

$$R(w) = \frac{1}{N}\sum_{i=1}^{N} L(y_i - \hat{y}_i(x|w)) + \frac{\lambda}{2} w^T w \tag{5.1}$$

Here, $R(w)$ has two components - the empirical risk minimizing the training error and the structural risk minimizing the model parameters, L represents a loss functional and $\hat{y}(x|w, b)$ represents the model prediction, whose structure is given by the learning algorithm (see following subsections). The algorithms considered in this study include logistic regression (a linear model), support vector machines and extreme learning machines (nonlinear models). Each algorithm is unique in formulation, loss function used, convergence rates, computation demand, prediction accuracy and potential for online learning. However, the main criteria used for evaluation in this study are prediction accuracy, number of parameters used for modeling and potential for online implementation for the HCCI engine system.

The HCCI classification problem involves identifying the boundary separating the input space that result in a stable or unstable operation. Also, when the engine misfires, the excitation command is changed to attempt a stable operation [70]. This

results in a class imbalance learning problem as the number of unstable class data is significantly smaller than the number of stable class data.

5.2.1 Class Imbalance Learning

Class imbalance learning (CIL) is encountered during situations when the number of instances of one class is very different from the number of instances in others. In a binary classification problem, the class where the number of observations is large, is referred to as the majority class (labeled +1) while the other class is referred to as the minority class (labeled -1). Imbalanced data sets need careful attention as machine learning (typically an optimization problem) causes the decision boundary to be favorable to the majority class data while ignoring the minority class data [83, 44].

Several solutions have been proposed to handle CIL problems including re-sampling the data where the minority class can be duplicated to be in proportion with the majority class (referred to as over-sampling) or some majority class data is removed to match proportions with the minority class (referred to as under-sampling). Although both sampling methods aim to artificially obtain a balanced data set, under-sampling is prone to loss of majority class information while over-sampling is prone to overfitting [83, 44]. Algorithm level modifications are also common which include cost-sensitive learning that weights the minority class data more than the majority class data in the optimization objective function. Other methods such as adjusting the decision threshold, one-class learning etc. are available in literature, but this work is restricted to the under-sampling, over-sampling and cost-sensitive methods.

5.2.2 Logistic Regression

Logistic regression (LR) is a classical linear classifier that proves to be effective especially for large data set problems owing to its computational efficiency. LR makes

use of a logistic function given by equation (5.2) which confines the output of the function to lie between 0 and 1. Unlike the linear regression model which solves a least squares problem with a squared loss function, LR solves a nonlinear optimization problem using a logistic loss function (see Figure A.1 in appendix A.1). The logistic loss function is particularly attractive for classification because the algorithm does not penalize the correctly classified points (at large positive margin in Figure A.1) as much as the squared loss improving convergence.

$$\psi(x) = \frac{1}{1 + e^{-x}} \tag{5.2}$$

The conditional probability of estimating y from x can be expressed in terms of the model parameters $\beta = [\beta_0 \quad \beta_1]^T$ as

$$P(Y = y|X = x) = \frac{1}{1 + e^{-y(\beta_1^T x + \beta_0)}} \tag{5.3}$$

where X and Y represent the input and output random variables. The goal of logistic regression is to determine β such that $P(Y|X, \beta)$ is maximized using the following optimization problem (see appendix A.2)

$$\beta^* = \arg\min_{\beta} \sum_{i=1}^{N} \log\left(1 + e^{-y(\beta_1^T x + \beta_0)}\right) \tag{5.4}$$

The equation (5.4) is nonlinear in β and can be solved by simple iterative methods [84]. The LR decision hypothesis is given by

$$f(x) = sgn(\beta_1^T x + \beta_0) \tag{5.5}$$

where

$$sgn(x) = \begin{cases} 1 & x > 0 \\ -1 & x \leq 0. \end{cases} \tag{5.6}$$

5.2.3 Support Vector Machines

Support Vector Machines (SVM) involve determining the boundary that maximizes the margin between the data based on a hinge loss

$$L_{hinge}(w, b) = max(0, 1 - yf(x)), \tag{5.7}$$

where $yf(x)$ gives the margin [42]. More details on SVM modeling can be found in chapter II. It should be noted that the regular SVM formulation is not designed for an imbalanced data set where the majority class data outnumbers the minority class data. A cost-sensitive version of the SVM algorithm is used in such cases, where the cost penalty parameter C in equation (2.18) is modified to weigh more to the penalties of the minority class data compared to the majority class data [43, 44]. All implementations of SVM are done using LibSVM [43]. The cost modification can be performed as follows

$$C_i = \begin{cases} C & \text{majority class data} \\ C(r \times f) & \text{minority class data} \end{cases} \tag{5.8}$$

where r represents the ratio of number of majority class data to the number of minority class data and f represents a scaling factor to be tuned for a given data set.

5.2.4 Extreme Learning Machines

The ELM formulation can be found in chapter II as well. In order to handle imbalanced class data, i.e., to modify the algorithm to weigh the minority class data

more, a simple weighting method can be incorporated in the ELM objective function (2.36) as

$$\min_{W} \left\{ (HW - Y)^T \Gamma_{imb} (HW - Y) + \lambda W^T W \right\} \tag{5.9}$$

$$\Gamma_{imb} = \begin{bmatrix} \Gamma_{imb_1} & 0 & . & . & 0 \\ 0 & \Gamma_{imb_2} & . & . & 0 \\ . & . & . & . & 0 \\ 0 & 0 & . & . & \Gamma_{imb_N} \end{bmatrix}$$

$$\Gamma_{imb_i} = \begin{cases} 1 & \text{majority class data} \\ r \times f & \text{minority class data} \end{cases} \tag{5.10}$$

where Γ_{imb} represents the weight matrix, r represents the ratio of number of majority class data to number minority class data and f represents a scaling factor to be tuned for a given data set. This results in the training step given by equation (5.11) and decision hypothesis given by equation (5.12).

$$W^* = \left(H^T \Gamma_{imb} H + \lambda I \right)^{-1} H^T \Gamma_{imb} Y \tag{5.11}$$

$$f(x) = sgn\left(W^T [\psi(W_r^T x + b_r)] \right) \tag{5.12}$$

5.3 HCCI Engine and Data Processing

For the purpose of identifying the stable operating envelope of HCCI engine, transient experiments are performed by exciting the engine and recording time sequences of engine variables similar to the open loop experiments in chapter III. In this section, the HCCI engine system and experiments performed are briefly explained followed by a methodology of labeling the data suitable for classification.

5.3.1 HCCI System and Experimentation

The system can be controlled using precalculated inputs such as injected fuel mass (FM in mg/cyc), crank angle at intake valve opening (IVO), crank angle at exhaust valve closing (EVC), crank angle at start of fuel injection (SOI). Other important physical variables that influence the performance of HCCI combustion include intake manifold temperature T_{in}, intake manifold pressure P_{in}, mass flow rate of air at intake $\dot{m}_{in}$, exhaust gas temperature T_{ex}, exhaust manifold pressure P_{ex}, coolant temperature T_c, equivalent fuel-air ratio (EAFR) etc. The engine performance metrics are given by combustion phasing indicated by the crank angle at 50% mass fraction burned (CA50), combustion work output indicated by net mean effective pressure (NMEP). Both CA50 and NMEP are determined from the high speed in-cylinder pressure measurements. The above variables at present time instant k along with their time histories are considered as inputs to the model (see Section 5.4, equation (5.15)). The data is pre-processed and labeled to identify stable and unstable observations as explained in Section 5.3.3.

5.3.2 HCCI Instabilities

A subset of the data collected from the engine is shown in Figure 5.1 where it can be observed that for some combinations of the inputs (left figures), the HCCI engine misfires (seen in the right figures where IMEP drops below 0 bar). HCCI operation is limited by several phenomena that lead to undesirable engine behavior. As described in [85], the HCCI operating range is conceptually constrained to a small region of permissible unburned (pre-combustion) and burned (post-combustion) charge temperature states. As previously noted, sufficiently high unburned gas temperatures are required to achieve ignition in the HCCI operating range without which complete misfire will occur. If the resulting combustion cannot achieve sufficiently high burned gas temperatures, commonly occurring in conditions with low fuel to diluent ratios or

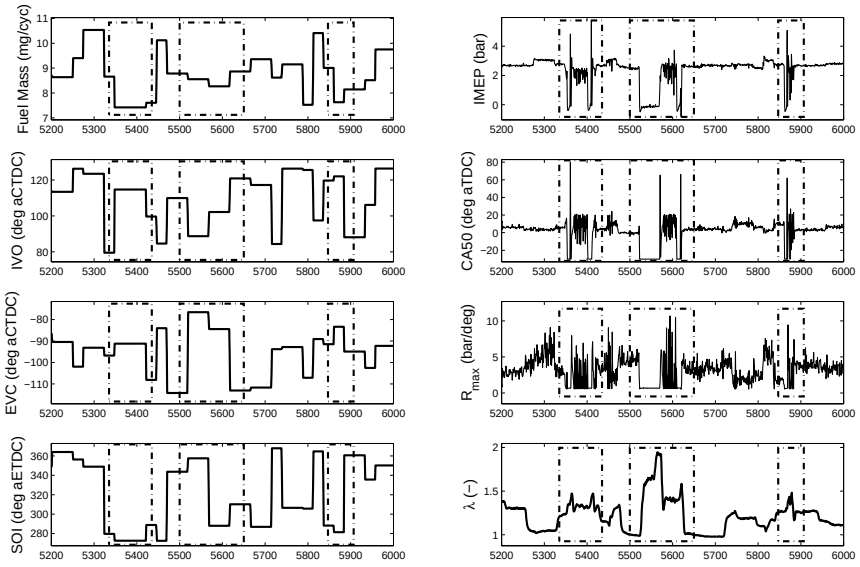


Figure 5.1: A-PRBS inputs and outputs showing misfire regions.

late combustion phasing, various degrees of quenching can occur resulting in reduced work output and increased hydrocarbon and carbon monoxide emissions. Under some conditions, this may lead to high cyclic variation due to the positive feedback loop existing through the trapped residual gas [86, 87]. Operation with high burned gas temperature, although stable and commonly reached at higher fueling rates where the fuel to diluent ratio is also high, yields high heat release and thus pressure rise rates that may pose challenges for engine noise and durability constraints. A discussion of the temperatures at which these phenomena occur may be found in [85].

In this work, the considered instabilities include those modes with high cyclic variability and those with complete misfire characterized by zero work output that can be readily identified through the two aforementioned cylinder pressure-based combustion features. The other phenomena could be included with the availability of additional sensing capability or analysis methods, e.g. fast response Flame Ionization Detection exhaust sampling equipment and detailed combustion noise analysis. Finally, it must be noted that control of these burned and unburned gas states, and

therefore the potential for undesirable combustion cycles, in a recompression HCCI engine is very much a function of the engine control variables. For instance, the EVC timing will determine the trapped residual mass that will be present in the upcoming cycle, while the IVO affects both the mass of incoming air and the state of the charge during the compression stroke leading up to the autoignition. The combination of IVO and EVC (see Fig. 3.2) define a negative valve overlap (NVO) period where exhaust gas from the previous cycle is trapped and compressed. A larger NVO period would necessarily yield a higher trapped residual mass that would tend to increase the charge temperature and advance CA50. Likewise, the timing and mass of the fuel injection event can significantly impact the charge temperature by changing the thermodynamic properties and air-fuel ratio of the charge present during NVO. The relatively high temperatures present during NVO can even lead to reactions of the fuel that will impact the temperature and chemical composition of the charge. Successful combustion of charge with a higher FM will tend to yield higher residual gas temperatures, thereby advancing CA50 in the following cycle. Likewise an earlier SOI in NVO will tend to increase charge temperatures and reduce the ignition delay of the charge, thereby advancing CA50. As such, an improper combination of control inputs (IVO, EVC, FM and SOI) in HCCI engines has the potential to shift operation from stable combustion to combustion with excessive heat release rates, high cyclic variability or misfires in a single cycle.

5.3.3 Data Preprocessing and Labeling

As mentioned earlier, the goal of this work is to classify the input space as stable (future HCCI cycles are stable) or unstable (future HCCI cycles misfire or have high variability). For this purpose, every data observation is labeled as follows. If either of the following two conditions are met, then the data at time instant k is labeled to be unstable (see Fig. 5.2) with a label value -1.

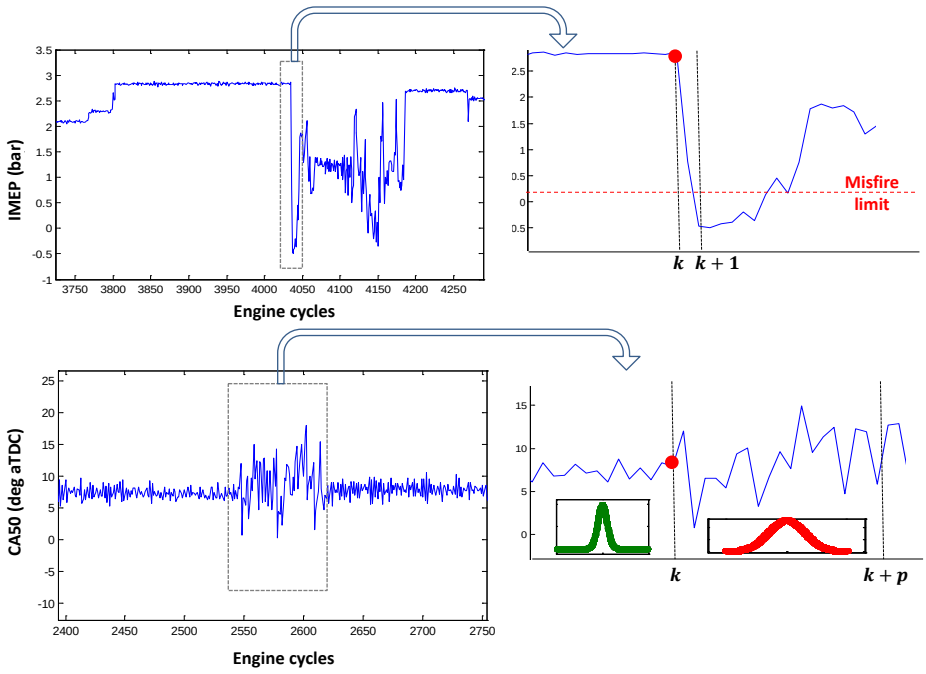


Figure 5.2: Illustration showing labeling of unstable observations.

1. an input (control inputs and past engine measurements up to an order of N_h) at cycle k results in an IMEP of less than 0.1 bar (chosen misfire limit) for any cylinder at cycle $k+1$.

2. an input at cycle k results in a high variance of CA50 (any cylinder) for cycles $k+1$ to $k+p$.

Only the first unstable data is considered in a sequence of unstable measurements. The labeling of stable data is as follows. A window of $N_w = 2p$ combustion cycles is considered (see Fig. 5.3). If the data at cycle k is obtained as a result of stable operation in the past p cycles as well as results in stable operation in the next p cycles, it is labeled stable with a label value of $+1$. If the time history of N_h is considered, then the data at cycle k along with the previous N_h samples are considered as inputs. If N_h is a large value, then the window length N_w can be increased accordingly. In this study, N_w and N_h correspond to 10 and 2 respectively.

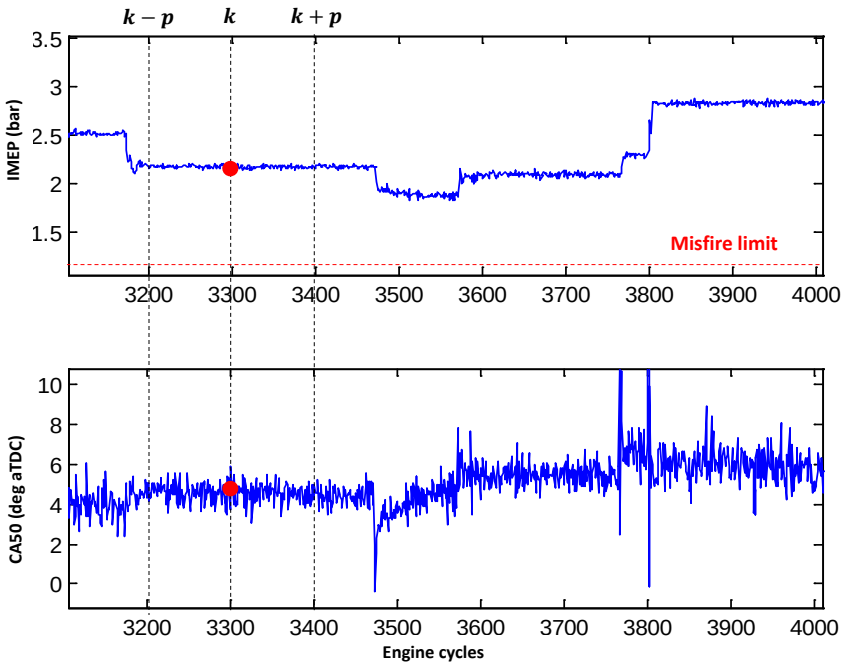


Figure 5.3: Illustration showing labeling of stable observations.

5.4 Model Development

The HCCI operating envelope is a function of the engine control inputs and engine variables such as temperatures and pressures. Also, the envelope is a dynamic system and hence a predictive model requires the measurement history up to an order of N_h. The dynamic classifier model can be given by

$$\hat{y}_{k+1} = sgn(f(x_k)) \tag{5.13}$$

where

$$sgn(x) = \begin{cases} 1 & x > 0 \\ -1 & x \leq 0. \end{cases} \tag{5.14}$$

where $\hat{y}_{k+1}$ indicates model prediction for the future cycle $k+1$, f can take any structure depending on the learning algorithm and x_k is given by

$$x_k = [IVO, EVC, FM, SOI, T_{in}, P_{in}, \dot{m}_{in}, T_{ex}, P_{ex}, T_c, FA, IMEP, CA50]^T \quad (5.15)$$

at cycle k upto cycle $k - N_h + 1$.

5.4.1 Model Selection

In this section, classification algorithms are developed based on Linear Regression (LS), Logistic regression (LR), SVM and ELM models. SVM and ELM models have variants based on performing under-sampling, over-sampling or no-sampling (regular) on the data set or the cost-sensitive version. The linear models (LR and LS) are compared as baselines and have their respective variants. The engine measurements and their time histories (defined by x_k) are considered inputs while the stability labels are considered outputs. The measured variables such as FM, IVO, EVC, SOI, T_c, T_{in}, P_{in}, $\dot{m}_{in}$, T_{ex}, P_{ex}, NMEP, CA50 and FA along with 2 cycles of history constitute the feature vector (input dimension $n = 39$). The measurement set consists of about 17000 observations out of which about 6400 observations are sampled as training set while about 10200 observations sampled as testing set. The ratio of number of majority class data to number minority class data (r) for the training set is 17.5 and for the testing set is 16.7.

For the class imbalance problem considered here, a typical error metric like the overall misclassification rate cannot be used as it would find a classifier that deemphasizes the minor class inaccuracies. Hence the following accuracy metric for skewed data sets is considered. Let TP and TN represent the total number of positive and negative class data classified correctly by the classifier. If N^+ and N^- represent the total number of positive and negative class data respectively, the true positive rate

(TPR) and true negative rate (TNR) and the total accuracy of the classifier can be defined as follows [88]

$$\begin{aligned} TPR &= \frac{TP}{N^+} \\ TNR &= \frac{TN}{N^-} \\ \text{Total Accuracy} &= \frac{TPR + TNR}{2}. \end{aligned} \tag{5.16}$$

Each of the considered models have a set of hyper-parameters (cost penalty C and kernel parameter σ for SVM while regularization coefficient λ and number of hidden neurons n_h for ELM) which needs tuning to suit the data set. A full grid search cross-validation is employed where the optimal combination of hyper-parameters are determined based on observed total accuracy of the classifier. The hyper-parameter tuning results are shown in Table 5.1 for ELM, Table 5.2 for SVM for the no-sampling and re-sampling cases. It can be observed that the total accuracy is generally high for SVM models compared to the ELM models. It can also be observed that by under-sampling or over-sampling the data, better accuracies can be achieved compared to the no-sampling case. Also both sampling methods give similar accuracy levels for both ELM and SVM models. However, an advantage of under-sampling can be realized in reduced computation as training is performed with a smaller subset of the training data.

Table 5.1: Grid search results for ELM model selection for the regular ELM, ELM with under-sampling and ELM with over-sampling (The models resulting in lowest total accuracy is highlighted in bold).

	Regular ELM					ELM with under-sampling					ELM with over-sampling				
	TPR					TPR					TPR				
n_h \ λ	0.01	0.1	1	10	100	0.01	0.1	1	10	100	0.01	0.1	1	10	100
10	0.995	0.995	0.995	0.996	1.000	0.909	0.909	0.912	0.906	0.878	0.925	0.925	0.925	0.925	0.919
30	0.994	0.994	0.995	0.995	0.997	0.917	0.916	0.918	0.923	0.896	0.934	0.934	0.934	0.936	0.936
50	0.995	0.995	0.995	0.995	0.998	0.917	0.918	0.925	0.948	0.915	0.930	0.931	0.932	0.935	0.951
70	0.996	0.996	0.996	0.996	0.996	0.936	0.937	0.931	0.936	0.948	0.944	0.945	0.946	0.946	0.946
90	**0.995**	0.995	0.995	0.995	0.996	0.915	**0.918**	0.924	0.929	0.927	0.938	0.938	0.939	**0.944**	0.942
	TNR					TNR					TNR				
n_h \ λ	0.01	0.1	1	10	100	0.01	0.1	1	10	100	0.01	0.1	1	10	100
10	0.333	0.333	0.327	0.258	0.000	0.732	0.734	0.737	0.771	0.743	0.716	0.716	0.716	0.722	0.757
30	0.387	0.389	0.387	0.366	0.160	0.714	0.719	0.730	0.732	0.732	0.727	0.725	0.729	0.735	0.727
50	0.430	0.430	0.423	0.407	0.294	0.771	0.773	0.752	0.735	0.727	0.773	0.773	0.771	0.755	0.724
70	0.426	0.423	0.413	0.404	0.351	0.771	0.770	0.775	0.748	0.704	0.739	0.739	0.739	0.758	0.740
90	**0.433**	0.430	0.420	0.405	0.356	0.789	**0.794**	0.784	0.768	0.755	0.763	0.768	0.773	**0.775**	0.775
	Total Accuracy					Total Accuracy					Total Accuracy				
n_h \ λ	0.01	0.1	1	10	100	0.01	0.1	1	10	100	0.01	0.1	1	10	100
10	0.664	0.664	0.661	0.627	0.500	0.821	0.821	0.825	0.839	0.811	0.820	0.820	0.820	0.824	0.838
30	0.691	0.692	0.691	0.680	0.579	0.815	0.818	0.824	0.828	0.814	0.831	0.830	0.832	0.835	0.832
50	0.712	0.712	0.709	0.701	0.646	0.844	0.846	0.838	0.842	0.821	0.852	0.852	0.851	0.845	0.838
70	0.711	0.709	0.705	0.700	0.674	0.853	0.853	0.853	0.842	0.826	0.842	0.842	0.842	0.852	0.843
90	**0.714**	0.712	0.707	0.700	0.676	0.852	**0.856**	0.854	0.848	0.841	0.850	0.853	0.856	**0.859**	0.858

Table 5.2: Grid search results for SVM model selection for the regular SVM, SVM with under-sampling and SVM with over-sampling (The models resulting in lowest total accuracy is highlighted in bold).

	Regular SVM					SVM with under-sampling					SVM with over-sampling				
	TPR					TPR					TPR				
C \ σ	0.01	0.1	1	10	100	0.01	0.1	1	10	100	0.01	0.1	1	10	100
0.1	1.000	0.996	1.000	1.000	1.000	0.928	0.933	0.767	0.120	0.106	0.966	0.923	0.917	**0.899**	0.993
1	0.996	0.996	0.994	0.998	1.000	0.965	0.906	0.910	0.782	0.588	0.932	0.931	0.962	0.989	0.999
10	0.996	0.995	0.990	0.997	0.999	0.916	0.915	**0.909**	0.792	0.618	0.933	0.951	0.976	0.996	0.999
100	0.996	0.990	**0.987**	0.996	0.999	0.924	0.927	0.896	0.793	0.618	0.935	0.966	0.983	0.996	0.999
500	0.995	0.988	0.985	0.996	0.999	0.925	0.917	0.892	0.793	0.618	0.945	0.971	0.983	0.996	0.999
	TNR					TNR					TNR				
C \ σ	0.01	0.1	1	10	100	0.01	0.1	1	10	100	0.01	0.1	1	10	100
0.1	0.000	0.374	0.000	0.000	0.000	0.632	0.735	0.931	0.998	0.998	0.667	0.820	0.913	**0.958**	0.162
1	0.397	0.423	0.552	0.108	0.054	0.668	0.825	0.923	0.967	0.987	0.792	0.884	0.814	0.221	0.082
10	0.423	0.444	0.645	0.145	0.080	0.802	0.882	**0.925**	0.958	0.975	0.822	0.817	0.732	0.167	0.080
100	0.430	0.567	**0.650**	0.165	0.080	0.833	0.886	0.915	0.954	0.975	0.848	0.763	0.642	0.165	0.080
500	0.436	0.627	0.637	0.165	0.080	0.853	0.882	0.910	0.954	0.975	0.833	0.724	0.623	0.165	0.080
	Total Accuracy					Total Accuracy					Total Accuracy				
C \ σ	0.01	0.1	1	10	100	0.01	0.1	1	10	100	0.01	0.1	1	10	100
0.1	0.500	0.685	0.500	0.500	0.500	0.780	0.834	0.849	0.559	0.552	0.816	0.871	0.915	**0.928**	0.577
1	0.696	0.710	0.773	0.553	0.527	0.817	0.866	0.916	0.875	0.787	0.862	0.907	0.888	0.605	0.540
10	0.709	0.720	0.818	0.571	0.539	0.859	0.899	**0.917**	0.875	0.797	0.878	0.884	0.854	0.581	0.539
100	0.713	0.779	**0.819**	0.580	0.539	0.879	0.906	0.905	0.874	0.797	0.892	0.865	0.812	0.580	0.539
500	0.716	0.808	0.811	0.580	0.539	0.889	0.899	0.901	0.874	0.797	0.889	0.847	0.803	0.580	0.539

Table 5.3: Grid search results for Cost-sensitive SVM and Cost-sensitive ELM models (The models resulting in lowest total accuracy is highlighted in bold).

Cost-sensitive SVM

TPR

C \ σ	0.001	0.01	0.1	1	10	100
0.1	0.000	0.000	0.000	0.000	0.000	0.000
1	0.984	0.972	0.921	0.918	**0.884**	0.764
10	0.995	0.994	0.993	0.988	0.996	0.999
100	1.000	0.999	0.993	0.987	0.996	0.999
500	1.000	0.997	0.989	0.985	0.996	0.999

TNR

	0.001	0.01	0.1	1	10	100
0.1	1.000	1.000	1.000	1.000	1.000	1.000
1	0.490	0.627	0.806	0.912	**0.961**	0.972
10	0.399	0.443	0.489	0.660	0.154	0.080
100	0.181	0.355	0.495	0.647	0.165	0.080
500	0.126	0.405	0.554	0.639	0.165	0.080

Total Accuracy

	0.001	0.01	0.1	1	10	100
0.1	0.500	0.500	0.500	0.500	0.500	0.500
1	0.737	0.800	0.863	0.915	**0.922**	0.868
10	0.697	0.718	0.741	0.824	0.575	0.539
100	0.591	0.677	0.744	0.817	0.580	0.539
500	0.563	0.701	0.771	0.812	0.580	0.539

Cost-sensitive ELM

TPR

n_h \ λ	0.01	0.1	1	10	100
10	0.909	0.909	0.909	0.907	0.901
30	0.924	0.924	0.923	0.927	0.927
50	0.925	0.925	0.926	0.930	0.936
70	0.939	0.939	0.939	0.934	0.931
90	0.932	0.932	**0.933**	0.936	0.930

TNR

	0.01	0.1	1	10	100
10	0.742	0.743	0.745	0.753	0.779
30	0.737	0.737	0.735	0.740	0.735
50	0.778	0.778	0.775	0.771	0.748
70	0.770	0.770	0.765	0.779	0.768
90	0.786	0.786	**0.794**	0.784	0.784

Total Accuracy

	0.01	0.1	1	10	100
10	0.825	0.826	0.827	0.830	0.840
30	0.830	0.830	0.829	0.833	0.831
50	0.852	0.851	0.850	0.851	0.842
70	0.854	0.854	0.852	0.857	0.850
90	0.859	0.859	**0.864**	0.860	0.857

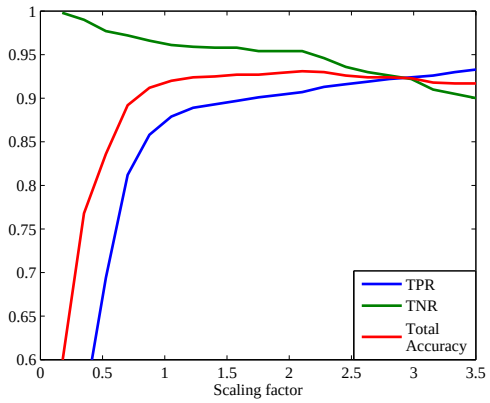


Figure 5.4: Sensitivity plot for TPR, TNR and Total Accuracy with scaling factor f for cost-sensitive SVM.

The model tuning for cost-sensitive SVM and ELM are summarized in Table 5.3. It can be observed that the cost-sensitive models find the decision boundary without re-sampling the data. Also, the total accuracy levels are slightly higher compared to the re-sampling methods. However, it can be observed that the TPR and TNR are not close to each other for both ELM and SVM models and the same can be observed for under-sampling and over-sampling cases too indicating that it could be a limitation in the data set to classify both classes to similar accuracy. By varying the scaling factor f, the boundary can be perturbed to suit the application which require either high TPR or high TNR. Sensitivity plots have been shown in Figure 5.4 and Figure 5.5 for SVM and ELM models respectively to observe the variation of total accuracy with the scaling factor. By understanding the sensitivity of the weight factors, an optimal weight for the minority class data (combination of r and f) can be determined.

As mentioned earlier, the SVM models have a better total accuracy compared to the ELM models. One reason could be that SVM minimizes a hinge loss while ELM minimizes a squared loss (as shown in Figure A.1) and hence better regularization performance. Another reason could be that the ELM models are too simple to identify the decision boundary accurately. A possibility is to add more hidden neurons.

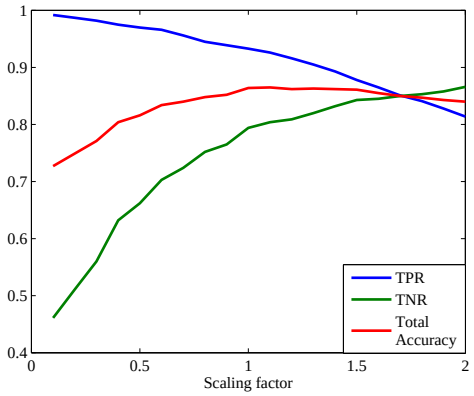


Figure 5.5: Sensitivity plot for TPR, TNR and Total Accuracy with scaling factor f for cost-sensitive ELM.

Table 5.4: Cost-sensitive ELM models with different random initialization of input layer parameters

#	TPR	TNR	Total Accuracy	λ	n_h	f
1	0.932	0.871	0.901	10.000	800.000	1.2
2	**0.921**	**0.879**	**0.900**	**10.000**	**600.000**	**1.3**
3	0.933	0.871	0.902	10.000	900.000	1.3
4	0.930	0.863	0.896	10.000	600.000	1.2
5	0.934	0.859	0.896	10.000	800.000	1.3
6	0.927	0.871	0.899	10.000	800.000	1.2
7	0.939	0.864	0.902	10.000	800.000	1.2
8	0.927	0.869	0.898	10.000	700.000	1.1
9	0.929	0.873	0.901	10.000	1000.000	1.3
10	0.928	0.866	0.897	10.000	900.000	1.3

Also, the ELM solution greatly depends on the random initialization of the input layer parameters (W_r and b_r). In an attempt to evaluate models with more hidden neurons and different random initializations of the input layer parameters, a further experiment was conducted and results summarized in Table 5.4. It can be observed that as more hidden neurons are added, the total accuracy of ELM model improves. Also, different randomization helps in finding a compact model at a given accuracy level (compare case 2 with case 9 where 400 additional neurons are required for a negligible improvement). Hence determining an efficient way of initialization of input layer parameters is required for ELM models and will be considered in the future.

Table 5.5: Summary of results for SVM and ELM models for all cases (Regular model, under-sampling, over-sampling and cost-sensitive). The results of the linear models (logistic regression and linear least squares) are also compared. The value of hyper-parameters and number of model parameters n_p are also included for every model

	SVM				ELM				
	Regular	Under-sampling	Over-sampling	Cost-sensitive	Regular	Under-sampling	Over-sampling	Cost-sensitive	Best ELM model
TPR	0.987	0.909	0.899	0.907	0.995	0.918	0.944	0.933	0.921
TNR	0.650	0.925	0.958	0.954	0.433	0.794	0.775	0.794	0.879
Total Accuracy	0.819	0.917	0.928	0.931	0.714	0.856	0.859	0.864	0.900
λ	-	-	-	-	0.010	0.100	10.000	1.000	10.000
n_h	-	-	-	-	90.000	90.000	90.000	90.000	600.000
f	-	-	-	2.104	-	-	-	0.909	0.769
C	100	10	0.1	1	-	-	-	-	-
σ	1	1	10	10	-	-	-	-	-
n_p	33696	16965	236691	120900	3690	3690	3690	3690	24600

	Logistic Regression			Linear LS			
	Regular	Under-sampling	Over-sampling	Regular	Under-sampling	Over-sampling	Cost-sensitive
TPR	0.995	0.911	0.928	0.996	0.941	0.955	0.875
TNR	0.441	0.791	0.786	0.389	0.704	0.699	0.828
Total Accuracy	0.718	0.851	0.857	0.692	0.822	0.827	0.852
n_p	40	40	40	40	40	40	40

5.4.2 Prediction Results

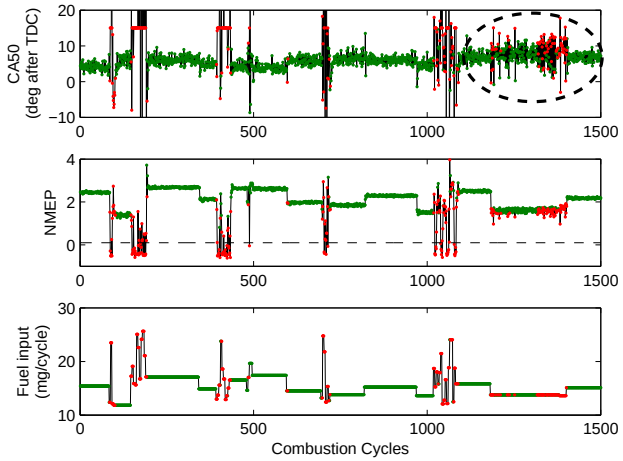
The models developed using SVM and ELM are compared against baseline linear models and the results summarized in Table 5.5. The case 2 model in Table 5.4 is considered as the best ELM model and included in the summary table. From the modeling results, it can be observed that both re-sampling methods (under-sampling and over-sampling) as well as cost-sensitive classification are suitable for the problem considered in this work. The nonlinear models result in better accuracies compared to the linear models indicating that the HCCI boundary is a nonlinear system and that nonlinear classification methods are necessary. However the cost paid for selecting a nonlinear model is the additional computation and memory required and the tradeoff can be evaluated specific to the application based on the importance of having accurate versus having low complexity models. For instance, the number of parameters required to identify the classification boundary for different models is summarized in Table 5.5. It is obvious that the decision boundary identified by linear models (under-parameterized) is very simple and does not capture the right behavior. SVM and ELM models on the other hand requires a large number of parameters in an attempt to capture more complex behavior. SVM is a non-parametric model and hence the number of parameters grows with number of training data. ELM on the other hand is a parametric model and hence the number of parameters are fixed and hence requires about 80% less number of parameters for a loss in accuracy of 3% compared to SVM. This is a major drawback of SVM for applications onboard the engine ECU which is limited in memory and computation.

For the engine problem considered here, it was observed that both SVM and ELM are capable of identifying the stable and unstable boundary of HCCI from experimental data. However, an important criteria for HCCI engine application is that the models must be able to be adapted online. As HCCI combustion is sensitive to other variables like engine speed, ambient temperature, pressure and humidity

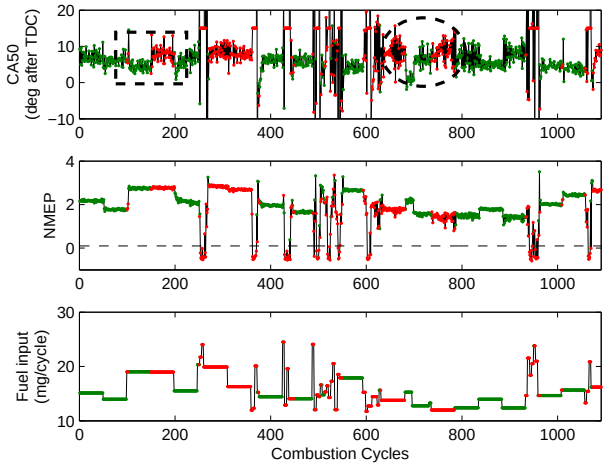
levels etc., the complexity of experiment design and size of data might increase. In addition, for reasons mentioned in chapter VI, a system that can learn and adapt online is of extreme importance. Hence even though SVM outperformed ELM in terms of accuracy, ELM requires relatively less parameters and is simple and efficient in implementation for on-line learning and chosen as suitable for the application in hand.

The most accurate models using ELM and SVM are used to make predictions on unseen engine inputs and predictions are summarized in Figure 5.6 and Figure 5.7 respectively while quantitative results are included in Table 5.5. Color codes are used to represent the predictions - a red marker on the plots indicate the model's prediction being "unstable" while a green marker indicates prediction being "stable". It can be seen from the plots that both models predict the HCCI instabilities well. The dotted line in the IMEP plot indicates misfire limit, dotted ellipse in CA50 plot indicates high variability instability mode while dotted rectangle indicates a wrong predictions by model.

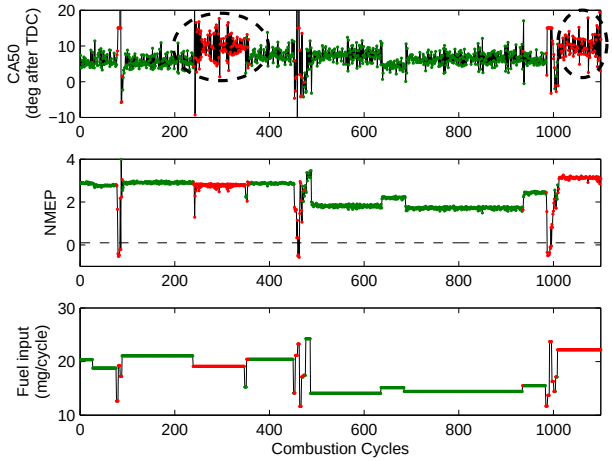
In order to get a closer look, subsections of the above figures are plotted in Figure 5.8a and Figure 5.8b for ELM and SVM respectively. It can be observed from Figure 5.8a that the input sample (at cycle 34) along with other input measurements is predicted unstable and rightly so, the following cycles are unstable as indicated by CA50 overshooting 10 degree after TDC. This is the primary goal of the proposed method that predicts if the engine operation is stable/unstable at time $k+1$, given the measurements up to time k. Hence using the model, any given input actuator setting and history of measurements of key combustion variables, it would be possible to predict if the subsequent set of combustion cycles misfire or not. A similar plot can be shown for SVM in Figure 5.8b which has a slightly different prediction compared to the ELM model. For instance, SVM predicts most of the region beyond cycle 34 to be unstable but ELM predicts to be stable between cycles 75 and 95. However, these



(a)

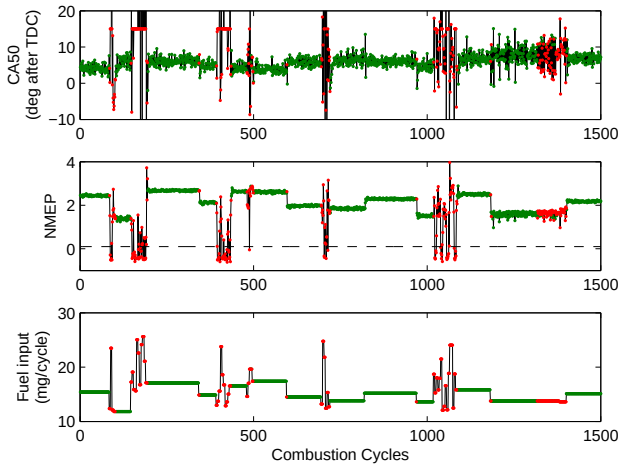


(b)

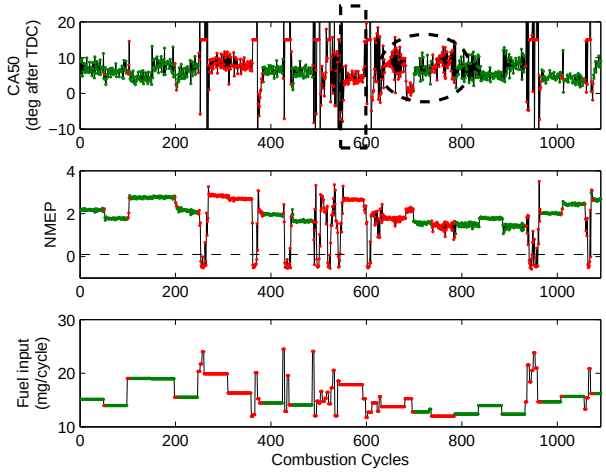


(c)

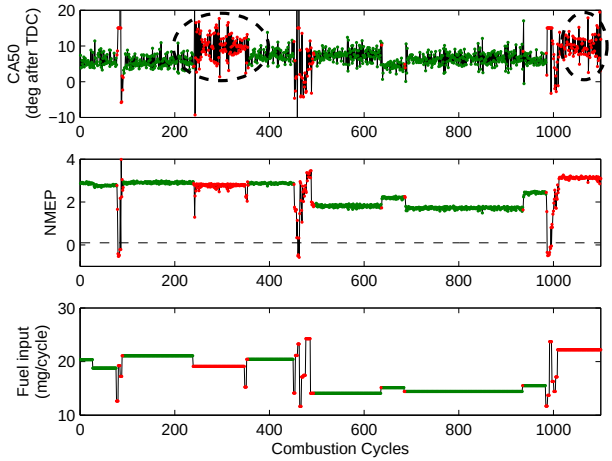
Figure 5.6: Prediction results of the cost-sensitive ELM. The color code indicates model prediction - green (and red) indicate stable (and unstable) prediction by the model.



(a)

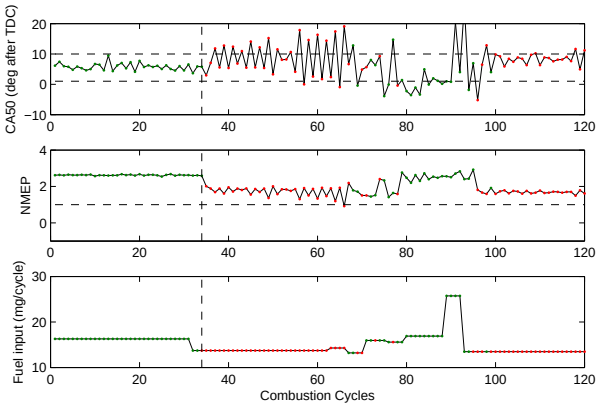


(b)

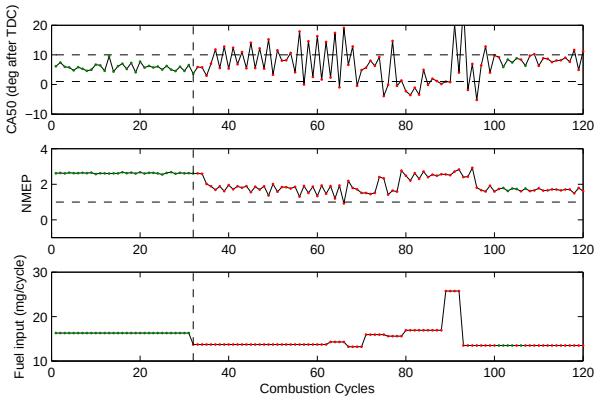


(c)

Figure 5.7: Prediction results of the cost-sensitive SVM. The color code indicates model prediction - green (and red) indicate stable (and unstable) prediction by the model.



(a) Cost-sensitive ELM



(b) Cost-sensitive SVM

Figure 5.8: A small subset of prediction results of the cost-sensitive ELM and SVM showing CA50, IMEP and one input variable to compare predictions in perspective to input variables. The green points indicate stable operation while red points indicate unstable operation.

cycles do not fall into completely stable or completely unstable and hence might be in between the two classes. Such predictions are to be expected from both models as training was not performed using a comprehensive data set that had a dense distribution of data in both classes.

CHAPTER VI

Stable Online Learning Algorithms for Extreme Learning Machines and Application to the HCCI Engine System

Identification models of HCCI engine variables were developed in the previous chapter using state-of-the-art machine learning algorithms and offline data processing. In this chapter, an online learning framework is developed for the HCCI engine. In the process, stable online learning algorithms including a Lyapunov based estimation law and a stochastic gradient based estimation law have been developed. The estimation laws are developed specifically for extreme learning machine models that are generic and can be applied for identification of any nonlinear system. Simple examples in the respective sections demonstrate the working of the algorithms while a suitable online learning algorithm has been identified for the HCCI system and a predictive model of HCCI is developed from streaming experimental data.

6.1 Motivation and Problem Statement

The availability of data for learning constitutes the primary difference between offline and online learning approaches. While computational demand is less for online learning owing to processing the data one-by-one, the accuracy of modeling or

prediction error convergence is a known issue with such methods. Online (sequential) learning is suitable for high volume/high velocity data sets, where the task is to process maximum data in a given time and make predictions whereas offline (batch) leaning is suitable for limited data sets where time and computation are not typically a concern. However, a tradeoff between the two can be analyzed to build powerful models.

For the HCCI engine case, online learning is necessary because of the following reasons. The models developed offline are valid only in the limited experimental conditions. For instance, the experiments are performed at a controlled ambient temperature, pressure and humidity conditions. As a result, the models developed are valid for the specified conditions and a vehicle application, for instance, would require the model based control to work in a wide range of climatic conditions. Hence, an online adaptation to learn the behavior of the system at new/unfamiliar situations is required. Also, since the models are developed directly from experimental data, they can perform poorly in certain operating regions where the density of experimental data is low. As more data becomes available in such regions, an online mechanism can be used to adapt to such data. Hence an online learning mechanism can have significant benefits both from a practical perspective as well as handling the high velocity sensor data.

The problem statement for online learning can be expressed as follows. Given a continuously operating system, the goal of the online learning algorithm is to process the data one-by-one and update the model parameters in a recursive manner to capture the underlying input-output relationship of the system.

6.2 Existing Methods and Limitations

There are several existing algorithms for online learning but most of them come with severe limitations. The early forms of online learning involved a gradient based

parameter update law based on first order information [89]. Owing to slow convergence of training error, second order information was suggested in the recursive LevenbergMarquardt algorithm [90]. Although convergence may be better than pure gradient based method, the solution is not guaranteed to achieve global minimum. Feed-forward networks with RBF nodes including resource allocation network [91], growing and pruning networks [92, 93] which had the ability to add hidden neurons for novel incoming data as well as remove insignificant nodes. However, these algorithms require several control parameters to be tuned and for large problems, the learning speed may be slow [94]. After the arrival of extreme learning machine models [49, 50], the OS-ELM algorithm [94] emerged as a powerful online learning algorithm with several impressive properties including processing data both one-by-one and chunk-by-chunk, discarding the data that is learned and only one tuning parameter, the number of hidden neurons. OS-ELM algorithm has been shown to perform better than the above mentioned algorithms for several benchmark data sets [94] and hence considered as one of the choices for the HCCI engine system.

6.2.1 Online Sequential ELM (OS-ELM)

The OS-ELM is a recursive version of the batch ELM algorithm. Training involves two steps - initialization step and sequential learning step. During the initialization step, a set of data observations (N_0) are required to initialize the H_0 and W_0 by solving the following optimization problem

$$\min_{W_0} \left\{ \|H_0 W_0 - Y_0\|^2 + \lambda \|W_0\|^2 \right\} \tag{6.1}$$

$$H_0 = [g(W_r^T x_0 + b_r)]^T \in \mathbb{R}^{N_0 \times n_h}. \tag{6.2}$$

The solution W_0 is given by

$$W_0 = K_0^{-1} H_0^T Y_0 \tag{6.3}$$

where $K_0 = H_0^T H_0$. Suppose given another new data x_1, the problem becomes

$$\min_{W_1} \left\| \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} W_1 - \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} \right\|^2 . \tag{6.4}$$

The solution can be derived as

$$\begin{aligned} W_1 &= W_0 + K_1^{-1} H_1^T (Y_1 - H_1 W_0) \\ K_1 &= K_0 + H_1^T H_1. \end{aligned}$$

Based on the above, a generalized recursive algorithm for updating the least-squares solution can be computed as follows

$$M_{k+1} = M_k - M_k H_{k+1}^T (I + H_{k+1} M_k H_{K+1}^T)^{-1} H_{k+1} M_k \tag{6.5}$$

$$W_{k+1} = W_k + M_{k+1} H_{k+1}^T (Y_{k+1} - H_{k+1} W_k) \tag{6.6}$$

where M represents the covariance of the parameter estimate.

In spite of its known advantages, an over-parameterized ELM suffers from ill-conditioning problem when recursive least squares type update is performed (as in OS-ELM). This results in poor regularization behavior [95, 96, 97, 98], which leads to an unbounded growth of the model parameters and unbounded model predictions. If decisions are made simultaneously based on the estimated model (as in case of adaptive control for instance [99]), it is vital for the parameter estimation to be stable so that model based decisions are valid. Hence a guarantee of stability and boundedness is of extreme importance. To address this issue, stable online learning algorithms based on Lyapunov stability theory and stochastic gradient descent are developed. The goal of the algorithms is to guarantee a bounded prediction from the estimation model.

Notable prior work include a Lyapunov approach applied for identification using radial basis function neural networks [9] and GLO-MAP models [10]. The parameter update in such methods involved complex gradient calculation in real time or first estimating a linear model and then estimating a nonlinear difference using orthonormal polynomial basis functions. The following section describes the developed online algorithms which take advantage of the convex optimization nature of ELM models and derive simple update laws along with a stability guarantee on the estimation.

6.3 Lyapunov Based Algorithm (L-ELM)

The L-ELM algorithm is designed for online regression learning of dynamic systems and time series (system identification). Data mining on dynamic systems has been as significant as on static systems but often the time connection in a dynamic system is usually not utilized. For instance, neural networks [21, 28] and support vector machines [70] have been used in modeling dynamic systems but algorithms designed for static data with an i.i.d assumption (data sampled from an independent and identical distribution) are used. The temporal aspects of the data useful for parameter estimation and decision making for dynamical systems are typically not taken into account. In this section, a Lyapunov based online learning algorithm is developed for dynamic systems that is guaranteed to have bounded model predictions.

6.3.1 Algorithm Derivation

Consider a general nonlinear discrete time dynamic system model representing the underlying phenomena of interest as follows.

$$z(k+1) \quad = \quad f(z(k), u(k)) \tag{6.7}$$

The system in (7.1) is assumed to satisfy the following assumptions in order for the given nonlinear system to be identified via a Lyapunov function [100].

1. The system is completely controllable and completely observable
2. The function f is continuously differentiable and satisfies f($\mathbf{0}$)=0. The function also satisfies global Lipschitz's condition [101] which guarantees existence and uniqueness of solution to the differential equation (7.1).
3. The function f is stationary; i.e., $f(.)$ does not depend explicitly on time.

Now, the model in (7.1) can be expressed as

$$\begin{aligned} z(k+1) &= A_L z(k) + g(z(k), u(k)) && (6.8) \\ &= A_L z(k) + W_*^T \phi(k) + \epsilon && (6.9) \end{aligned}$$

In the above expression, $z \in \mathbb{R}^n$, n representing the number of states, $g(z(k), u(k)) = f(z(k), u(k)) - A_L z(k)$ represents the system nonlinearity and assuming ELM can model $g(z(k), u(k))$ with an accuracy of ϵ using parameters W_*. The above assumption is valid in theory as ELM is an universal approximator [49]. The matrix A_L is included so as to maintain asymptotic stability of equation (6.9) and can be chosen as any matrix with eigen values in the unit circle [102]. The parametric model of the system can be constructed assuming the same structure as equation (6.9) by

$$\hat{z}(k+1) = A_L \hat{z}(k) + \hat{W}^T(k)\phi(k) \tag{6.10}$$

where $\hat{W}(k)$ represents the parameter estimate of W_* at time index k. The state error is given by the following

$$\begin{aligned} e(k+1) &= z(k+1) - \hat{z}(k+1) \\ &= A_L e(k) + \tilde{W}^T(k)\phi(k) + \epsilon && (6.11) \end{aligned}$$

where $\tilde{W}(k)$ represents the parameter error ($\tilde{W}(k) = W_* - \hat{W}(k)$). It should be noted that $\phi(k)$ is common to both system model and the parametric model which means the inputs and states of the plant are fed to the model indicating a series-parallel architecture [22].

A positive definite, decrescent and radially unbounded [103] function $V(e, \tilde{W}, k)$ (6.12) is used to construct the parameter update law given by equation in equation (6.13) (See Appendix for full derivation). Lyapunov functions are used to establish stability of nonlinear systems and a thorough mathematical treatment of Lyapunov based system identification can be found in [100]. Here Γ_L represents the gain of learning and P is a positive definite matrix.

$$V(e, \tilde{W}, k) = e^T(k)Pe(k) + \frac{1}{2}tr(\tilde{W}(k)\Gamma_L\tilde{W}^T(k)) \tag{6.12}$$

Taking the difference in V,

$$V(k+1) - V(k) = e^T(k+1)Pe(k+1) + \frac{1}{2}tr(\tilde{W}(k+1)\Gamma_L\tilde{W}^T(k+1)) - e^T(k)Pe(k) - \frac{1}{2}tr(\tilde{W}(k)\Gamma_L\tilde{W}^T(k))$$

$$= [A_L e(k) + \tilde{W}^T(k)\phi(k) + \epsilon]^T Pe(k+1) + \frac{1}{2}tr(\tilde{W}(k+1)\Gamma_L\tilde{W}^T(k+1)) - e^T(k)Pe(k) - \frac{1}{2}tr(\tilde{W}(k)\Gamma_L\tilde{W}^T(k))$$

$$= e^T(k)A_L^T Pe(k+1) + \phi(k)^T\tilde{W}(k)Pe(k+1) + \epsilon^T Pe(k+1) + \frac{1}{2}tr(\tilde{W}(k+1)\Gamma_L\tilde{W}^T(k+1)) - e^T(k)Pe(k) - \frac{1}{2}tr(\tilde{W}(k)\Gamma_L\tilde{W}^T(k))$$

$$= e^T(k)A_L^T P(A_L e(k) + \tilde{W}^T(k)\phi(k) + \epsilon) + \phi(k)^T \tilde{W}(k) P e(k+1) + \epsilon^T P e(k+1)$$
$$+ \frac{1}{2} tr(\tilde{W}(k+1)\Gamma_L \tilde{W}^T(k+1)) - e^T(k)Pe(k) - \frac{1}{2} tr(\tilde{W}(k)\Gamma_L \tilde{W}^T(k))$$

$$= e^T(k)A_L^T P A_L e(k) + e^T(k)A_L^T P \tilde{W}^T(k)\phi(k) + e^T(k)A_L^T P\epsilon$$
$$+ \phi(k)^T \tilde{W}(k) P e(k+1) + \epsilon^T P e(k+1) + \frac{1}{2} tr(\tilde{W}(k+1)\Gamma_L \tilde{W}^T(k+1))$$
$$- e^T(k)Pe(k) - \frac{1}{2} tr(\tilde{W}(k)\Gamma_L \tilde{W}^T(k))$$

Let $Q \in \mathbb{R}^{n \times n}$ be a positive definite matrix which satisfies the following discrete Lyapunov equation

$$A_L^T P A_L - P = -Q$$

$$V(k+1) - V(k) = -e^T(k)Qe(k) + e^T(k)A_L^T P\epsilon + \epsilon^T P e(k+1)$$
$$+ e^T(k)A_L^T P \tilde{W}^T(k)\phi(k) + \phi^T(k)\tilde{W}(k) P e(k+1) + \frac{1}{2} tr(\tilde{W}(k+1)\Gamma_L \tilde{W}^T(k+1))$$
$$- \frac{1}{2} tr(\tilde{W}(k)\Gamma_L \tilde{W}^T(k))$$

$$= -e^T(k)Qe(k) + e^T(k)A_L^T P\epsilon + \epsilon^T P e(k+1) + e^T(k)A_L^T P \tilde{W}^T(k)\phi(k) + \phi^T(k)\tilde{W}(k) P e(k+1)$$
$$+ \frac{1}{2} tr([\tilde{W}(k) + \Delta\tilde{W}(k)]\Gamma_L [\tilde{W}(k) + \Delta\tilde{W}(k)]^T) - \frac{1}{2} tr(\tilde{W}(k)\Gamma_L \tilde{W}^T(k))$$

$$= -e^T(k)Qe(k) + e^T(k)A_L^T P\epsilon + \epsilon^T P e(k+1)$$
$$+ e^T(k)A_L^T P \tilde{W}^T(k)\phi(k) + \phi^T(k)\tilde{W}(k) P e(k+1) + tr(\Delta\tilde{W}^T(k)\Gamma_L \tilde{W}(k))$$

By converting, $e^T(k)A_L^T P\tilde{W}^T(k)\phi(k) = tr(PA_L e(k)\phi^T(k)\tilde{W}(k))$ and

$$\phi^T(k)\tilde{W}(k)Pe(k+1) = tr(Pe(k+1)\phi^T(k)\tilde{W}(k))$$

$$\begin{aligned} V(k+1) - V(k) = & -e^T(k)Qe(k) + e^T(k)A_L^T P\epsilon + \epsilon^T Pe(k+1) \\ & + tr(\Delta\tilde{W}^T(k)\Gamma_L\tilde{W}(k) + PA_L e(k)\phi^T(k)\tilde{W}(k) + Pe(k+1)\phi^T(k)\tilde{W}(k)) \end{aligned}$$

By setting the terms in the trace to be zero, we get

$$0 \quad = \quad \Delta\tilde{W}^T(k)\Gamma_L\tilde{W}(k) \; + \; PA_L e(k)\phi^T(k)\tilde{W}(k) \; + \; Pe(k \; + \; 1)\phi^T(k)\tilde{W}(k)$$

$$\Delta\tilde{W}(k) \;\; = \;\; -\Gamma_L^{-T}\phi(k)[e(k+1) + A_L e(k)]^T P$$

Also, $\tilde{W}(k) = W_* - \hat{W}(k)$, the change in parameter can be given by

$$\Delta\hat{W}(k) \;\; = \;\; \Gamma_L^{-T}\phi(k)[e(k+1) + A_L e(k)]^T P$$

$$\begin{aligned} \hat{W}(k+1) &= \hat{W}(k) + \Delta\hat{W}(k) \\ &= \hat{W}(k) + \Gamma_L^{-T}\phi(k)[e(k+1) + A_L e(k)]^T P \end{aligned} \tag{6.13}$$

6.3.2 Stability Analysis

The Lyapunov based online learning algorithm is given by (6.13). Now, applying the update law, the change in Lyapunov function becomes

$$\begin{aligned}
\Delta V(k) &= -e^T(k)Qe(k) + e^T(k)A_L^T P\epsilon + \epsilon^T Pe(k+1) \\
&= -e^T(k)Qe(k) + e^T(k)A_L^T P\epsilon + \epsilon^T P(e(k) + \Delta e(k)) \\
&= -e^T(k)Qe(k) + e^T(k)A_L^T P\epsilon + \epsilon^T Pe(k) \\
&\leq -|\lambda_{max}Q|\|e(k)\|^2 + \|\epsilon\|\|P\|\|e(k)\| + \|\epsilon\|\|P\|\|A_L\|\|e(k)\| \\
&\leq 0 \text{ if } \|e(k)\| \geq \frac{\|\epsilon\|\|P\|\|I + A_L\|}{|\lambda_{max}Q|} = E_\epsilon \quad (6.14)
\end{aligned}$$

When the above condition is satisfied, equation (6.14) can be written as

$$\Delta V(k) \leq -|\lambda_{max}Q|\|e(k)\|^2 \leq 0 \quad (6.15)$$

For all $k > 0$, $V(k) - V(0) \leq 0$ or $V(k) \leq V(0)$. Also, from the positive definiteness of V, $V(e, \tilde{W}, k) > 0$ for all $[e, \tilde{W}, k]^T > 0$. Hence $0 < V(k) \leq V(0)$ for all $k > 0$. Hence $V(k) \in \mathbf{L}_\infty$, i.e.,

$$\lim_{k\to\infty} V(k) = V_\infty < \infty \quad (6.16)$$

$V(k)$ is a function of $e(k)$ and hence $e(k) \in \mathbf{L}_\infty$. This implies $\hat{z}(k) \in \mathbf{L}_\infty$ assuming the states of the plant $z(k)$ are bounded. Hence the states of the estimation model $\hat{z}(k)$ do not blow up eliminating possible errors during model based decision making. Also, from equation (6.15),

$$V(k+1) - V(k) \leq -|\lambda_{max}Q|\|e(k)\|^2$$

$$\sum_{k=0}^{\infty} V(k+1) - V(k) \leq -|\lambda_{max}Q| \sum_{k=0}^{\infty} \|e(k)\|^2$$

$$\Rightarrow \sum_{k=0}^{\infty} \|e(k)\|^2 \leq \frac{V(0) - V_\infty}{|\lambda_{max} Q|} < \infty \tag{6.17}$$

which implies $e(k) \in \mathbf{L}_2$. From (6.13), $(\hat{W}(k+1) - \hat{W}(k)) \in \mathbf{L}_2$. Using discrete time Barbalat's lemma [104],

$$\lim_{k \to \infty} e(k) = 0 \tag{6.18}$$

$$\lim_{k \to \infty} \hat{W}(k+1) = \hat{W}(k) \tag{6.19}$$

Hence, the adaptive law in (6.13) guarantees that the estimated output $\hat{z}(k)$ converges to the actual output $z(k)$ and the model parameters $\hat{W}$ converge to some constant values. The parameters converge to the true parameters W_* only under conditions of persistence of excitation [103]. For general nonlinear systems whose persistence of excitation cannot be guaranteed [105], a multi-step signal varying between extreme values of the inputs are typically used [28, 70]. Hence, intuitively, the proposed algorithm converges locally (according to the available excitation strength) to its estimates and as the system is excited more and more, moves towards its true values. Further, using boundedness of $V(k)$, $e(k) \in \mathbf{L}_\infty$ which guarantees that the online model predictions are bounded as long as the system output is bounded. As the error between the true model and the estimation model converges to zero, the estimation model becomes a one-step ahead predictive model of the nonlinear system.

6.3.3 Simulation Results

In this section, the developed algorithm is applied to two simple nonlinear dynamic systems and its working analyzed. The performance of the Lyapunov based algorithm (L-ELM) is compared against the baseline nonlinear OS-ELM algorithm. The results of linear identification models based on recursive least squares (RLS) and linear Lyapunov based method (LLin) are also compared to demonstrate that the

chosen examples are truly nonlinear. Also, the results of an offline batch learning ELM algorithm (O-ELM) is included to evaluate if the L-ELM trades off accuracy for computational simplicity.

The performance metrics used are one-step ahead prediction (OSAP) and multi-step ahead prediction (MSAP) root mean squared errors (RMSE). For MSAP, the models are converted from a series-parallel architecture to a parallel architecture [22] where the predictions of the model is fed back as inputs along with external excitation creating a recursive prediction over the prediction horizon. In this way, the evaluation of the model being close to the actual dynamic system can be performed. Such a metric is important in model based decision making applications [21, 28].

6.3.3.1 Example - I: Simple Scalar System

Consider the following single input single output (SISO) system

$$\begin{aligned} z(k+1) &= sin\big(\frac{z(k)^3}{10}\big) + \frac{u(k)^2}{10} \\ z(0) &= 0. \end{aligned} \tag{6.20}$$

Here the augmented vector becomes $x(k) = [z(k), u(k)]^T$. The inputs vary between -1 and +1 in an uniformly distributed multi step pseudorandom pattern to get information about majority of the operating regions of the system. The parameters of identification are chosen as follows: $A_L = 0.1$,$n_h = 5$,$Q = 1$ and $\Gamma_L = 10I_{5,5}$.

It can be seen from Fig. 6.1 that the parameters of the L-ELM appear to be converging to some steady state values which indicates that the nonlinear system is close to being identified. However, the OS-ELM parameters grow in an unstable manner and then converge back. Even though the parameters converge, the instability can be observed. This will be discussed more in Section 6.3.4. The linear models do not have sufficient degrees of freedom and oscillates without converging (The

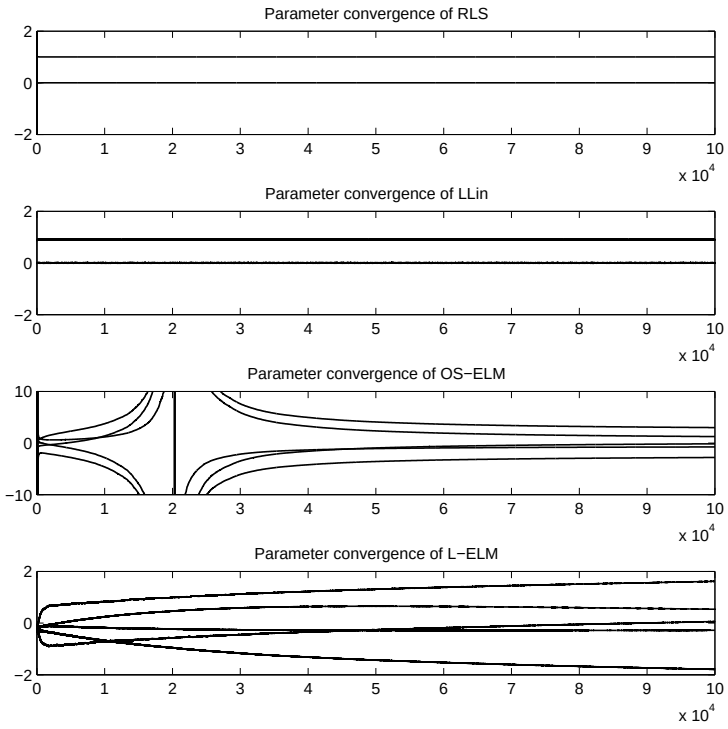


Figure 6.1: Comparison of parameter evolution for the L-ELM with OS-ELM and the linear models for the simple scalar system

oscillations are not seen because the time scale is very large, please refer to Fig. 6.3).

The multi-step ahead prediction performances of the models can be summarized in Fig. 6.2 respectively where the nonlinear models performed well. The linear models are clearly unsuitable for MSAP and cannot be used for dynamic simulations. The high RMSE values indicate the inability of linear models to capture the system behavior with limited number of degrees of freedom. The multi-step ahead prediction is done by feeding back the model predictions along with an input of the form $u(k) = 0.2 * sin\left(\frac{2\pi k}{50}\right) + 0.8 * cos\left(\frac{2\pi k}{50}\right)$ so that the input covers the region of operation of the models.

The prediction RMSE and the norm of the estimated parameters are listed in Table 6.1. It can be seen that the L-ELM achieves a lower norm of parameters compared

Table 6.1: Performance comparison of L-ELM with OS-ELM and the linear models for the simple scalar system

	OSAP RMSE	MSAP RMSE	$\|W\|$
RLS	0.0098	0.0494	0.981
Llin	0.0099	0.0556	0.834
OS-ELM	0.0059	0.0008	5.915
L-ELM	0.0028	0.0012	2.412
O-ELM	0.0012	0.0007	3.655

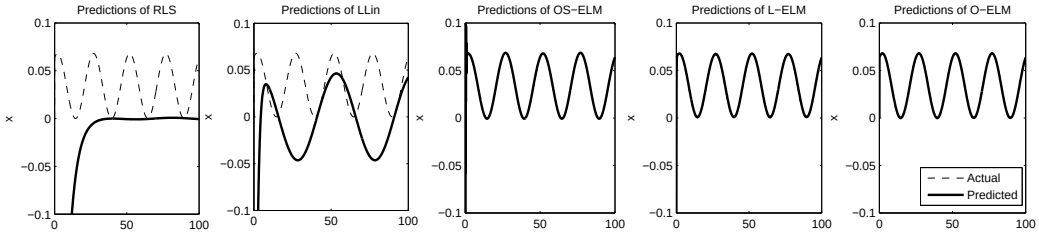


Figure 6.2: Comparison of multi-step ahead predictions for the L-ELM with OS-ELM, O-ELM and the linear models for the simple scalar system

to the OS-ELM and O-ELM which can be attributed to the stable learning method which bounds the parameter growth. The results of the offline (batch learning) ELM model indicates that the performance of the developed online learning algorithm is comparable to that of batch learning.

It should be noted that the observed results and discussion are for a given random initialization of the hidden layer parameters (W_r, b_r), a given initial conditions of the estimated parameters (W_0) and a given set of design parameters of the algorithm (A_L, Γ_L, P). A different gain value (Γ_L) for instance might lead to a different convergence behavior and might result in different prediction errors and different norm of parameters.

6.3.3.2 Example - II: A more complex system

In this section, a more complex example of a system with two states and two inputs are considered as follows

$$x_1(k+1) = sin\big(\frac{x_1(k)}{1+x_2^2(k)} + u_1(k)\big) \tag{6.21}$$

$$x_2(k+1) = cos\big(1 - \frac{x_1(k)x_2(k)}{1+x_2^2(k)} - u_2(k)\big) \tag{6.22}$$

$$x(0) = [0,0]^T \tag{6.23}$$

Similar to the previous case, the inputs vary between -1 and +1 independently in a multi-step pseudorandom pattern. The algorithm design parameters are chosen as follows: $A_L = 0.1I_{2,2}, n_h = 8, Q = I_{2,2}$ and $\Gamma_L = 10I_{8,8}$.

It can be seen from Fig. 6.3 that the parameters of the L-ELM do not converge completely indicating that either the number of hidden neurons is less or the identification time is insufficient to have sufficient excitations. Hence parameter oscillations are observed. Similarly, the parameters of the LLin model also oscillates indicating that the linear models do not have enough degrees of freedom to capture the nonlinear behavior.

An important observation to be made is that the Lyapunov method (for linear or nonlinear model) solves the error dynamics differential equation (6.11) and finds the solution to the differential equation (6.13). The use of PRBS type of signal forces the update law to reduce the error and updates the parameters to only adapt to the system locally given by that particular excitation. If the excitation signal had several frequencies simultaneously (sum of sinusoids), the convergence would be global as in case of linear system identification [103]. However, for nonlinear systems, sufficient excitation is never completely achieved [103, 21] indicating that the parameters locally adapt to particular excitations and if carried out for an infinite time, might achieve global convergence. This is an existing problem to all types of

nonlinear identification algorithms and is not addressed in this paper. Hence the reason for parameter oscillation for L-ELM for this example. A good trick is to slow down the local learning process using the gain parameter Γ_L so that aggressive local convergence is reduced and global convergence is made faster.

Table 6.2: Performance comparison of L-ELM with OS-ELM and the linear models for the more complex system

	OSAP RMSE	MSAP RMSE	$\|W\|$
RLS	0.147	0.67	0.92
Llin	0.128	0.69	0.83
OS-ELM	0.124	0.32	8.5
L-ELM	0.082	0.235	6.9
O-ELM	0.088	0.295	13.26

The prediction performance is summarized in Fig. 6.4 along with Table 6.2. The linear models have high RMSE values and hence not suitable for identification of the given nonlinear system. The multi-step ahead prediction is done by feeding back the model's predictions along with external inputs of the form $u(k) = [sin\big(\frac{2\pi k}{10}\big), cos\big(\frac{2\pi k}{10}\big)]^T$. The results of the offline ELM model is included to show that the performance of the developed algorithm is comparable to the batch learning model indicating that there is no compromise in accuracy by performing online learning using the Lyapunov method.

It can be seen from Table 6.2 that the norm of estimated parameters are smaller for the L-ELM indicating better generalization compared to OS-ELM and O-ELM. It should be noted that the offline models are developed and validated using the same data set (sub-sampled to reduce computation) on which the online learning was performed. In this example, the MSAP RMSE is lower for the L-ELM compared to the OS-ELM while in the previous example, the MSAP RMSE for OS-ELM was better indicating that a conclusion cannot be made as which method outperforms the other in terms of prediction accuracy. However, the prime advantage of the L-ELM

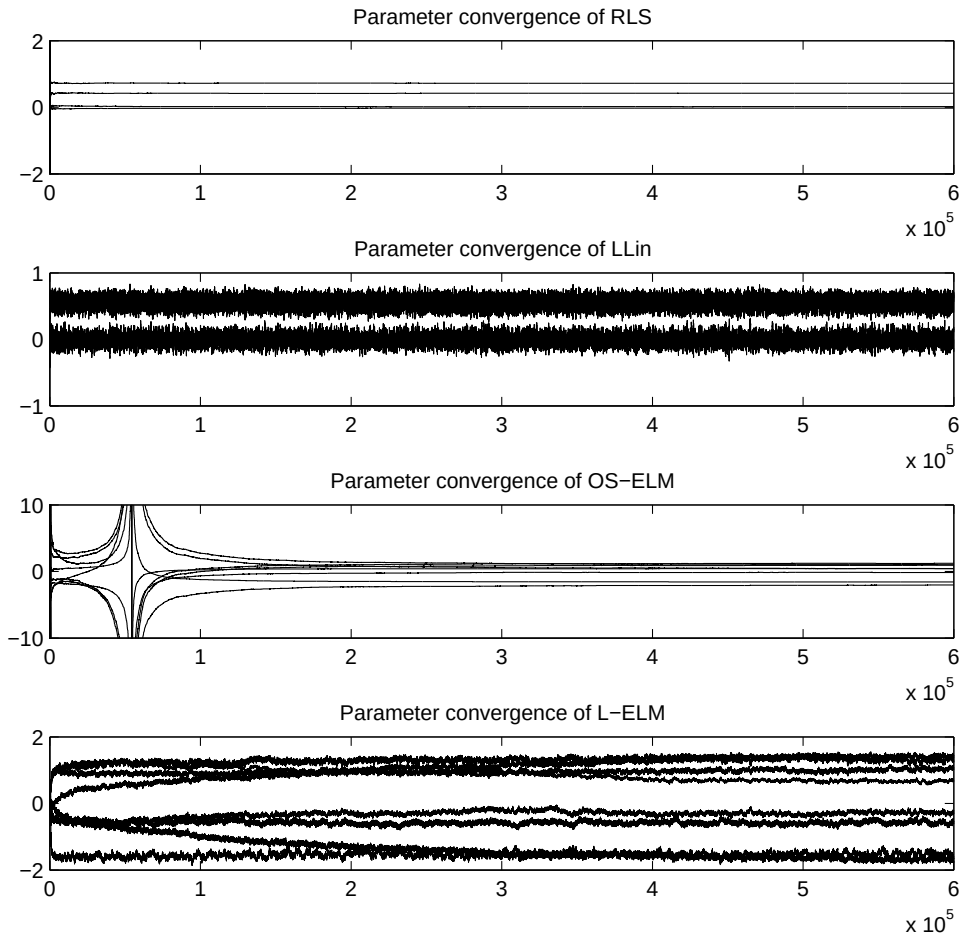


Figure 6.3: Comparison of parameter evolution for the L-ELM with OS-ELM and the linear models for the more complex system

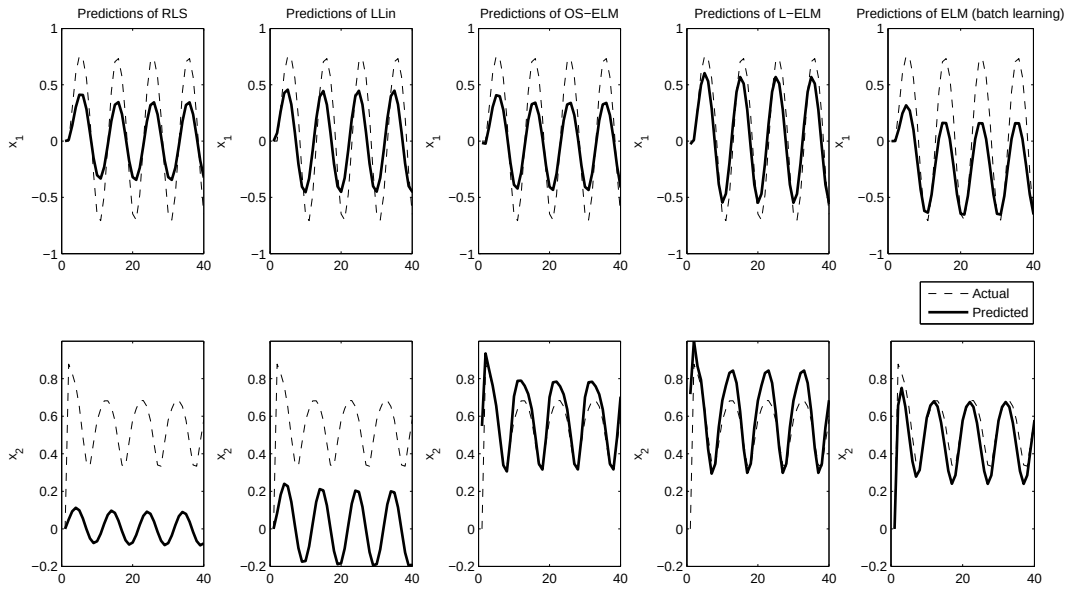


Figure 6.4: Comparison of multi-step ahead predictions for the L-ELM with OS-ELM, O-ELM and the linear models for the more complex system

comes from its stable parameter evolution which is briefed in the following subsection.

6.3.4 Stability Advantage of L-ELM

It has been reported that ELM might run into an ill-posed problem when it is over parameterized or when not properly initialized [95, 96, 97, 98]. A few attempts have been made to improve the regularization behavior or ELM [97] and OS-ELM [98]. However, when the data is being processed 1-by-1 (as in the case of system identification), the regularization improvement suggested by [98] was not found to improve the situation. Such an unstable parametric evolution can cause fatal problems when such online models are used in decision making. To address the issue, a Lyapunov based stable learning algorithm was developed for ELM models. The simulation of such a scenario is summarized in the Fig. 6.5 for the system considered in example II. It can be seen that with a small number of hidden neurons (n_h=3), the condition

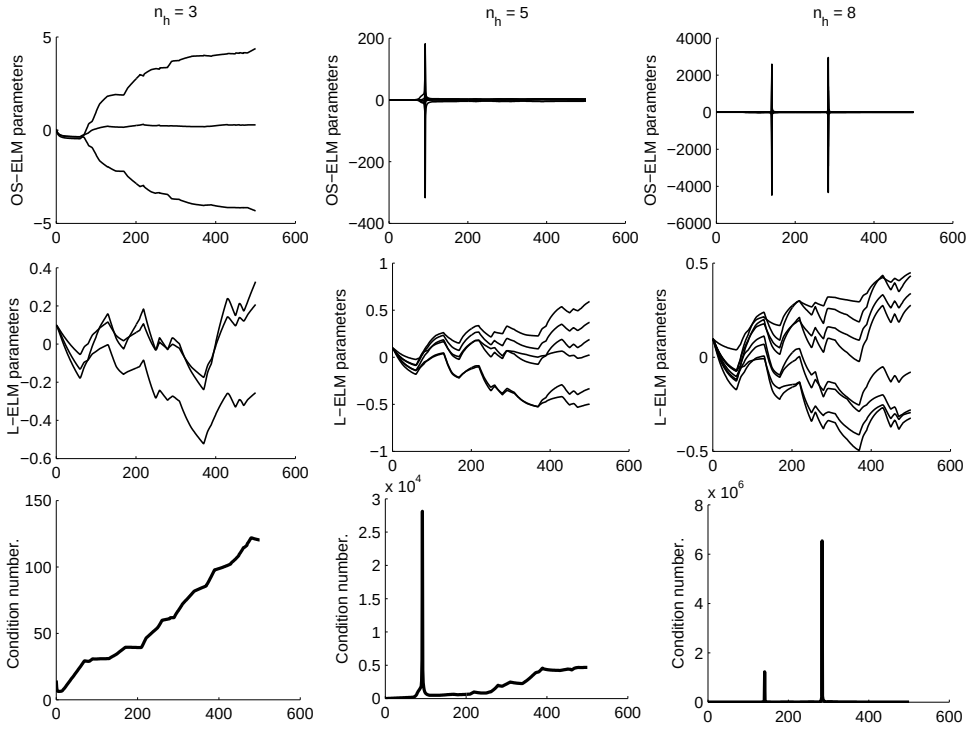


Figure 6.5: The ill-conditioning of OS-ELM as more number of hidden neurons (n_h) are added compared to bounded parameter evolution of L-ELM.

number (of matrix M in equation (6.5)) remains within reasonable values and the parameters do not diverge. However, as the number of hidden neurons increase, the condition number grows to a very high value creating an ill-conditioned least squares problem [95], the solution of which is absurd. For any general case, the solution of OS-ELM is never guaranteed to be stable. The Lyapunov based algorithm on the other hand, results in a model that is well regularized where parameter growth can be controlled using small values for the gain matrix Γ_L. The Lyapunov method gives a stability guarantee and performs well with no undesirable parameter growth even when the model is over-parameterized. Such a guarantee is necessary for control related applications. This shows the effectiveness of the method.

6.4 Stochastic Gradient Based ELM Algorithm

In this section, the stochastic gradient descent (SGD) based update is developed for the extreme learning machine models for both classification and regression. SGD methods have been popular for several decades for performing online learning but with severe limitations on poor optimization and slow convergence rates. However, only recently, the asymptotic behavior of SGD methods has been analyzed indicating that SGD methods can be very powerful in learning from large data sets [106, 107]. SGD type algorithms have been developed for Adaline networks, perceptron models, K-means, SVM and Lasso [106]. In this work, the SGD algorithm is applied for extreme learning machine models showing good potential for online learning of high velocity (streaming) data.

In any learning problem, there are three types of errors namely the approximation error, the estimation error and the optimization error [106]. As mentioned previously [chapter II), the expected risk $E_{exp}(f)$ and the empirical risk E_{emp} can be given by

$$\begin{aligned} E_{exp}(f) &= \int l(f(x), y) dP(x, y) \\ E_{emp}(f) &= \frac{1}{N} \sum_{i=1}^{N} l(f(x_i), y_i) \end{aligned}$$

Let $f^* = \text{argmin}_f E_{exp}(f)$ be the best possible prediction function. In practice, the prediction function is chosen from a family of parametric functions. Let $f^*_{\mathcal{F}} = \text{argmin}_{f \in \mathcal{F}} E_{exp}(f)$ be the best prediction function chosen from a parameterized family of functions $\mathcal{F}$. When a data set is available, the empirical risk becomes a proxy for the expected risk for a learning problem [20]. Let $\bar{f}^*_{\mathcal{F}} = \text{argmin}_{f \in \mathcal{F}} E_{emp}(f)$ be the solution that minimizes the empirical risk. However, the global solution is not typically obtained because of computational limitations and hence the solution of the learning problem is reduced to finding $\bar{f}_{\mathcal{F}} = \text{argmin}_{f \in \mathcal{F}} E_{emp}(f)$.

Following the above setup, the approximation error (E_{app}) is the error introduced in approximating the true function space with a family of functions $\mathcal{F}$, the estimation error (E_{est}) is the error introduced in optimizing over $E_{emp}(f)$ instead of $E_{exp}(f)$, the optimization error (E_{opt}) is the error induced as a result of stopping the optimization to $\bar{f}_{\mathcal{F}}$. The total error E_{tot} can be expressed as

$$\begin{aligned} E_{app} &= E_{exp}(f^*) - E_{exp}(f^*_{\mathcal{F}}) \\ E_{est} &= E_{exp}(f^*_{\mathcal{F}}) - E_{emp}(\bar{f}^*_{\mathcal{F}}) \\ E_{opt} &= E_{emp}(\bar{f}^*_{\mathcal{F}}) - E_{emp}(\bar{f}_{\mathcal{F}}) \\ E_{tot} &= E_{app} + E_{est} + E_{opt} \end{aligned}$$

The following observations are taken from the asymptotic analysis of SGD algorithms [106, 108].

1. The empirical risk $E_{emp}(f)$ is only a surrogate for the expected risk $E_{exp}(f)$ and hence an increased effort to minimize E_{opt} doesn't equate to better learning. In fact, if E_{opt} is very low, there is a good chance that the prediction function will over-fit the training data.

2. SGD are worst optimization algorithms (in terms of reducing E_{opt}) but they need a relatively less time to minimize expected risk. Therefore, in the large scale setup, when the limiting factor is computational time rather than the number of examples, SGD algorithms performs asymptotically better.

3. SGD results in faster convergence when the loss function has strong convexity properties.

ELM models have a squared loss function and when the hidden neurons are randomly assigned and fixed, the training translates to solving a convex optimization problem. Hence the ELM model can be a good candidate to perform SGD type

learning and hence the motivation for this study. The SGD based algorithm can be derived for the ELM models as follows.

6.4.1 Algorithm Derivation

Let (x_i, y_i) where $i = 1, 2, ..N$ be the streaming data in consideration. The data can be considered to be available to the algorithm one-by-one or artificially sampled one-by-one from a very large data set. Let the ELM empirical risk be defined as follows

$$\begin{aligned} J(W) &= \min_W \frac{1}{2} \sum_{i=1}^{N} \|y_i - W^T \phi_i\|^2 \\ &= \min_W \left\{ \frac{1}{2}\|y_1 - W^T\phi_1\|^2 + \frac{1}{2}\|y_2 - W^T\phi_2\|^2 + .. + \frac{1}{2}\|y_N - W^T\phi_N\|^2 \right\} \\ &= \min_W \left\{ J_1(W) + J_2(W) + .. + J_N(W) \right\}. \end{aligned} \tag{6.24}$$

Consider the objective for a data observation i,

$$\begin{aligned} J_i(W) &= \frac{1}{2} e_i^T e_i \\ &= \frac{1}{2}(y_i - W^T\phi_i)^T(y_i - W^T\phi_i) \\ &= \frac{1}{2} y_i^T y_i + \frac{1}{2}\phi_i^T W W^T \phi_i - \phi_i^T W y_i \\ \frac{\partial J_i}{\partial W}^T &= W^T\phi_i\phi_i^T - y_i\phi_i^T = (W^T\phi_i - y_i)\phi_i^T = -e_i\phi_i^T \\ \Rightarrow \frac{\partial J_i}{\partial W} &= -\phi_i e_i^T. \end{aligned} \tag{6.25}$$

In the regular gradient descent (GD) algorithm, the gradient of $J(W)$ is used to update the model parameters as follows.

$$\begin{aligned}
\frac{\partial J}{\partial W} &= \frac{\partial J_1}{\partial W} + \frac{\partial J_2}{\partial W} + .. + \frac{\partial J_N}{\partial W} \\
\Rightarrow \frac{\partial J}{\partial W} &= -\phi_1 e_1^T - \phi_2 e_2^T - .. - \phi_N e_N^T \\
W_{k+1} &= W_k - \Gamma_{SG} \frac{\partial J}{\partial W} \\
&= W_k + \Gamma_{SG}(\phi_1 e_1^T) + \Gamma_{SG}(\phi_2 e_2^T) + .. + \Gamma_{SG}(\phi_N e_N^T) \qquad (6.26)
\end{aligned}$$

where k is the iteration count, Γ_{SG} represents the step size or update gain matrix for the GD algorithm.

It can be seen from equation (6.26) that the parameter matrix W is updated based on gradients calculated from all the available examples. If the number of data observations is large, the gradient calculation can take enormous computational effort. The stochastic gradient descent algorithm considers one example at a time and updates W based on gradients calculated from (x_i, y_i) as shown in

$$W_{i+1} = W_i + \Gamma_{SG}(\phi_i e_i^T). \qquad (6.27)$$

From equation (6.26), it is clear that the optimal W is a function of gradients calculated from all the examples. As a result, as more data becomes available, W converges close to its optimal value in SGD algorithm. Processing data one-by-one significantly reduces the computational requirement and the algorithm is scalable to large data sets. More importantly, for the online learning task considered in this work, the SGD algorithm becomes a strong candidate.

In order to handle class imbalance learning, the algorithm in (6.27) can be modified by weighting the minority class data more. The modified algorithm can be expressed

as

$$W_{i+1} = W_i + \Gamma_{imb}\Gamma_{SG}(\phi_i e_i^T) \tag{6.28}$$

where $\Gamma_{imb} = r \times f$, r and f represent the imbalance ratio and the scaling factor that needs to be tuned.

6.4.2 Stability Analysis

The stability analysis of the SGD based ELM algorithm can be derived as follows. The ELM structure makes the analysis simple and similar to that of a linear gradient based algorithm [103].

As defined previously, the instantaneous estimation error can be expressed in terms of parametric error ($\tilde{W} = W_* - W$) as

$$\begin{aligned} e_i &= y_i - W^T\phi_i \\ &= W_*^T\phi_i - W^T\phi_i \\ &= \tilde{W}^T\phi_i \end{aligned} \tag{6.29}$$

where W_* represents true model parameters. Further, the parametric error dynamics can be obtained as follows.

$$\begin{aligned} \tilde{W}_{i+1} &= W_* - W_{i+1} \\ &= W_* - W_i - \Gamma_{SG}\phi_i e_i^T \\ &= \tilde{W}_i - \Gamma_{SG}\phi_i e_i^T \end{aligned} \tag{6.30}$$

Consider the following positive definite, decrescent and radially unbounded [103] Lyapunov function V

$$V(\tilde{W}) = tr(\tilde{W}^T\Gamma_{SG}^{-1}\tilde{W}) \tag{6.31}$$

where tr represents the trace of a matrix.

$$\begin{aligned} V(\tilde{W}_{i+1}) - V(\tilde{W}_i) &= tr(\tilde{W}_{i+1}^T \Gamma_{SG}^{-1} \tilde{W}_{i+1}) - tr(\tilde{W}_i^T \Gamma_{SG}^{-1} \tilde{W}_i) \\ &= tr((\tilde{W}_i - \Gamma_{SG}\phi_i e_i^T)^T \Gamma_{SG}^{-1}(\tilde{W}_i - \Gamma_{SG}\phi_i e_i^T)) - tr(\tilde{W}_i^T \Gamma_{SG}^{-1} \tilde{W}_i) \\ &= tr(-2\tilde{W}_i^T \phi_i e_i^T + e_i \phi_i^T \Gamma_{SG} \phi_i e_i^T) \\ &= tr(-2e_i e_i^T + e_i \phi_i^T \Gamma_{SG} \phi_i e_i^T) \\ &= -2e_i^T e_i + e_i^T e_i \phi_i^T \Gamma_{SG} \phi_i \\ &= -2e_i^T e_i + e_i^T \phi_i^T \Gamma_{SG} \phi_i e_i \\ &= -e_i^T M_{SG} e_i \end{aligned} \tag{6.32}$$

where $M_{SG} = 2 - \phi_i^T \Gamma_{SG} \phi_i$. It can be seen that $V_{i+1} - V_i \leq 0$ if $M_{SG} > 0$ or $2 - \phi_i^T \Gamma_{SG} \phi_i > 0$ or

$$0 < \lambda_{max}(\Gamma_{SG}) < 2 \tag{6.33}$$

When (6.33) is satisfied, $V(\tilde{W}) \geq 0$ is non-increasing in i and the limit

$$\lim_{k \to \infty} V(\tilde{W}) = V_\infty \tag{6.34}$$

exists. From (6.32),

$$\begin{aligned} V_{i+1} - V_i &= -e_i^T M_{SG} e_i \\ \sum_{i=0}^{\infty} V_{i+1} - V_i &= -\sum_{i=0}^{\infty} e_i^T M_{SG} e_i \\ \Rightarrow \sum_{i=0}^{\infty} e_i^T M_{SG} e_i &= V(0) - V_\infty < \infty \end{aligned} \tag{6.35}$$

$$\tag{6.36}$$

Also,

$$\sum_{i=0}^{\infty} e_i^T I e_i \leq \sum_{i=0}^{\infty} e_i^T M_{SG} e_i < \infty \tag{6.37}$$

when $M_{SG} > I$ or when

$$\lambda_{max}(\Gamma_{SG}) < 1. \tag{6.38}$$

Hence, when (6.38) is satisfied, $e_i \in \mathbf{L}_2$. From (6.27), $(W_{i+1} - W_i) \in \mathbf{L}_2 \cap \mathbf{L}_\infty$. Using discrete time Barbalat's lemma [104],

$$\lim_{i\to\infty} e_i = 0 \tag{6.39}$$

$$\lim_{i\to\infty} W_{i+1} = W_i \tag{6.40}$$

Hence, the SGD learning law in (6.27) guarantees that the estimated output $\hat{y}_i$ converges to the actual output y_i and the model parameters W converge to some constant values. The parameters converge to the true parameters W_* only under conditions of persistence of excitation [103]. Further, using boundedness of V_i, $e_i \in \mathbf{L}_\infty$ which guarantees that the online model predictions are bounded as long as the system output is bounded. As the error between the true model and the estimation model converges to zero, the estimation model becomes a one-step ahead predictive model of the nonlinear system.

In the next section, the developed stochastic gradient descent ELM algorithm is evaluated on several benchmark data sets addressing regression, classification as well as class imbalance learning problems. The SG-ELM algorithm is compared against the OS-ELM algorithm for generalization performance as well as computational time. A baseline linear classification algorithm based on recursive least squares is also used for comparison.

6.4.3 Simulation 1: Online Classification and Class Imbalance Learning

The following benchmark data sets are obtained from the UCI [109]. The shuttle data set contains 9 attributes all of which are numerical. Approximately 80% of the data belongs to class 1 which gives an imbalance ratio of 3.67. The skin data set

comprises of randomly sampled blue, green and red values (dimension 3) from face images of various age groups (young, middle, and old), race groups (white, black, and asian), and genders obtained from FERET database and PAL database. Total learning sample size is 245057; out of which 50859 is the skin samples and 194198 is non-skin samples. The imbalance ratio is about 3.8. The letter recognition data set consists of image information that is to be classified as one of the 26 English alphabets. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. The data is 16 dimensional and for the purpose of class imbalance learning, one of the class is considered as minority class while the rest is considered a majority class resulting in an imbalance ratio of about 24. The norm data set and the ring data sets are taken from KEEL repositories [110]. The data sets and their features are listed in Table 6.3.

Table 6.3: Benchmark data sets used for classification and class imbalance learning.

Data sets	imbalance ratio	feature dimension	Training data size	Testing data size
norm data	1	20	4660	2220
shuttle data	3.67	9	2320	55100
skin data	3.82	3	2400	242600
letter data	24.4	16	3600	16000
ring data	1.02	20	5076	2220

The performance of the algorithms are compared with respect to the training time in seconds, total positive rate (TPR), total negative rate (TNR), average accuracy and geometric mean accuracy. The error metrics consider class imbalance learning and are defined as in chapter V. It can be observed from Table 6.4 that the training time for SG-ELM is reduced by more than half compared to the OS-ELM. The main reason for the reduction in training time is because of the simple update for the SG-

ELM algorithm. The OS-ELM, on the other hand, updates both the parameter W and its covariance matrix M and so consuming more computational time and memory. The computation could be a vital requirement for large data sets and the SG-ELM algorithm can be a suitable candidate for large scale learning. For the imbalanced data sets, the scaling factor f was tuned by trial and error to obtain a low generalization error in terms of TPR, TNR and total accuracy. Also, from Table 6.4, it can be observed that the SG-ELM algorithm performs well for highly imbalanced data sets.

From the above simulation study, it can be observed that the accuracy of the SG-ELM algorithm is very comparable to that of the OS-ELM algorithm. Precisely, the SG-ELM algorithm can be trained faster with less computational demand to reach slightly inaccurate models compared to OS-ELM. The linear model achieves a high accuracy on the skin data set indicating that the separating boundary is close to linear. The nonlinear OS-ELM and SG-ELM models perform equally well. The other data sets are nonlinear for which the linear models are less accurate.

6.4.4 Simulation 2: Online Regression Learning

The following regression data sets are taken from UCI repository [109] as well. The auto MPG data set predicts the miles per gallon (MPG) of a given automobile based on 8 attributes, the concrete data set predicts the compressive strength of concrete from 8 features, wine quality data set contains data for predicting wine quality based on physicochemical tests while year prediction data set contains audio features and the corresponding release year of a song. The year data set is an example for high dimension learning problem. The delta data set data set is obtained from the task of controlling the ailerons of a F16 aircraft [110]. The data sets and their features are listed in Table 6.5.

A similar observation can be made using Table 6.6 for regression learning. The training time for SG-ELM is reduced by more than half compared to the OS-ELM

Table 6.4: Performance comparison of OS-ELM and SG-ELM in terms of training time, total positive rate (TPR), total negative rate (TNR), average accuracy and geometric mean accuracy. The results of a recursive least squares based linear model is also compared to measure the nonlinearity in the data sets.

Data sets	Algorithms	Training Time	TPR	TNR	Average Accuracy	Gmean Accuracy
norm data	SG-ELM	0.11	0.93	0.90	0.92	0.92
	OS-ELM	0.27	0.91	0.89	0.90	0.90
	Linear	0.25	0.89	0.87	0.88	0.88
shuttle data	SG-ELM	0.06	0.96	0.98	0.97	0.97
	OS-ELM	0.22	0.99	0.98	0.99	0.99
	Linear	0.10	0.97	0.69	0.83	0.82
skin data	SG-ELM	0.05	0.93	0.94	0.93	0.93
	OS-ELM	0.10	0.96	0.92	0.94	0.94
	Linear	0.09	0.88	1.00	0.94	0.94
letter data	SG-ELM	0.20	0.90	0.96	0.93	0.93
	OS-ELM	0.58	0.93	0.96	0.94	0.94
	Linear	0.40	0.85	0.92	0.89	0.89
ring data	SG-ELM	0.12	0.82	0.76	0.79	0.79
	OS-ELM	0.28	0.74	0.80	0.77	0.77
	Linear	0.25	0.64	0.64	0.64	0.64

Table 6.5: Benchmark data sets used for regression learning.

Data sets	feature dimension	Training data size	Testing data size
auto MPG	7	219	118
concrete	8	650	310
wine	11	4450	1950
year	90	1160	514000
delta	5	4270	2850

owing to their simple update law. Also, the generalization accuracy of SG-ELM is comparable to that of OS-ELM indicating that SG-ELM could be a suitable online learning algorithm for large regression problems. It has to be noted that for both the classification and regression problems, the same data and random matrix initializations were used for both the algorithms. The OS-ELM algorithm requires that the model initialized offline using a small data set [94] before performing sequential learning. However, careful tuning is required for both L-ELM and the SG-ELM algorithms which can be time consuming. The OS-ELM has a particularly attractive feature that is missing in the L-ELM and SG-ELM is that the OS-ELM updates the covariance matrix of the estimated parameter which indicates if the estimate has converged or not. This could be very important in a MPC framework to act as a condition to start making predictions based on the online model. This will be discussed more in the next section.

6.5 Online Regression Learning for HCCI Engine

In this section, the stable online learning algorithms namely L-ELM and SG-ELM are applied in identifying the HCCI engine variables online. The closed loop data at 1800 RPM is used for demonstration. The inputs to the model include control actuator signals namely fuel injection mass, fuel injection timing and exhaust valve opening. The outputs of interest include IMEP, CA50, P_{max}, R_{max}, Torque and EAFR. Three online learning schemes namely OS-ELM, L-ELM and SG-ELM are evaluated for prediction performance and training time. A baseline online linear RLS based model is also used to evaluate the nonlinear nature of the problem. Further, the offline ELM (O-ELM) model is compared as well to evaluate the efficiency of the online learning models in learning the HCCI behavior.

First, the learning performance of IMEP is studied in detail for better understanding of the working of the algorithms. An ELM model with 20 hidden units is

Table 6.6: Performance comparison of OS-ELM and SG-ELM in terms of training time and generalization error. The results of a recursive least squares based linear model is also compared to measure the nonlinearity in the data sets.

Datasets	Algotihms	Training Time	Testing Error
auto	SG-ELM	0.007	0.26
	OS-ELM	0.014	0.25
	Linear	0.011	0.41
concrete	SG-ELM	0.018	0.26
	OS-ELM	0.048	0.22
	Linear	0.034	0.35
wine	SG-ELM	0.091	0.28
	OS-ELM	0.201	0.25
	Linear	0.170	0.45
year	SG-ELM	0.273	0.24
	OS-ELM	0.572	0.24
	Linear	0.527	0.32
delta	SG-ELM	0.08	0.10
	OS-ELM	0.17	0.09
	Linear	0.13	0.10

considered with fixed randomized input layer parameters. About 13400 cycles of data is considered one-by-one as it is sampled by the engine ECU and model parameters updated in a sequential manner. After the training phase, the models are evaluated for the next 3400 cycles of data for one step ahead predictions. Further, to evaluate if the learned models represent the actual HCCI dynamics, the multi-step ahead prediction of the models are compared using about 1600 cycles of data.

The parameters of the models are tuned to obtain accurate models for the given data. As required by OS-ELM [94], the model is first initialized using about 200 cycles of data which gives an initial estimate of the model parameters W_0 and covariance matrix M_0 (see equations (6.5) and (6.6)). The parameters of L-ELM and SG-ELM are as follows. $A_L = 5 \times 10^{-4} \quad I_1$, $\Gamma = 0.025 \quad I_{20}$, $Q = 0.2 \quad I_1$, $\Gamma_{SG} = 0.007 \quad I_{20}$. The W for L-ELM and SG-ELM are initialized to zero.

On performing online learning, it can be observed from Fig. 6.6 that even after initializing the OS-ELM model, the parameters still grow owing to ill-conditioning but converge back as more data becomes available. However, the L-ELM and SG-ELM have bounded parameters owing to the stable nature of parameter updates. It can also be observed that the linear model too has the ill-conditioning problem because of the recursive least squares based method similar to OS-ELM. Also, the parameter values for L-ELM and SG-ELM remain small compared to the OS-ELM. This has a significant implication in the statistical learning theory [42]. A small norm of model parameters implies a simpler model which results in good generalization. However, the L-ELM and SG-ELM has a high testing error. This is attributed to the slow convergence of these algorithms compared to the OS-ELM. As observed earlier in Section 6.3 as well as in simulations, the Lyapunov based parameter estimate converges to the local excitation levels. Even for mild nonlinear systems (see Section 6.3.3), the L-ELM required several random step excitations for good convergence. However, the online learning mechanism is aimed to run along with the engine and

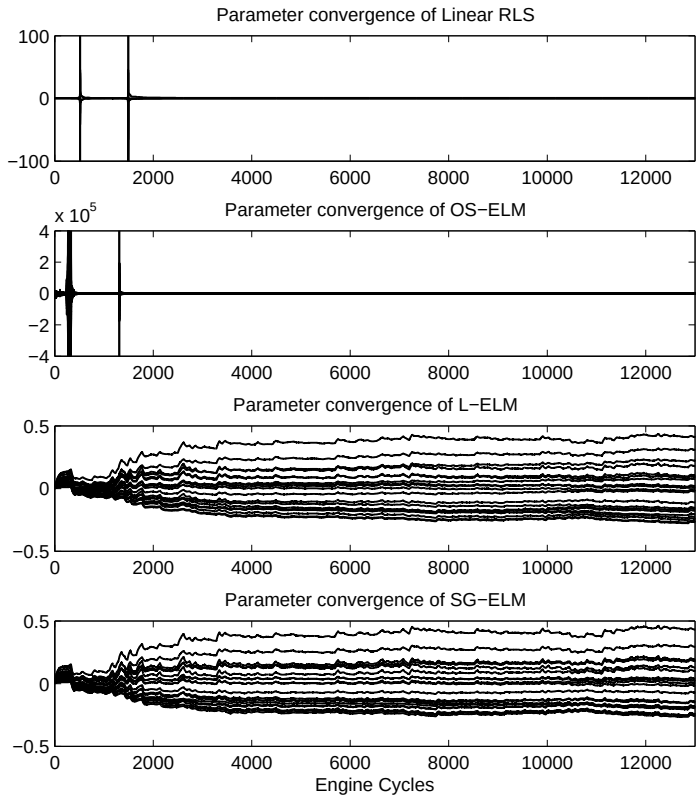


Figure 6.6: Comparison of parameter evolution for the online learning models. The recursive least squares based linear and ELM models exhibit an ill-conditioning problem which results in undesirable parameter growth. The parameters of the Lyapunov and Stochastic Gradient ELM are always bounded.

hence the slow convergence may not be an issue in a real application. The prediction results as well as training time for the online models are compared in Table 6.7. It can be observed that the computational demand for L-ELM and SG-ELM are significantly less compared to OS-ELM indicating faster learning. Although the linear model performs well for one-step ahead predictions, it fails to mimic the dynamic nature of the system as evaluated in multi-step ahead predictions. The OS-ELM, in spite of exhibiting the ill-conditioning problem, performs well both in OSAP and MSAP. The OS-ELM performance is the closest to the offline trained models that are expected to have learned better (as there is no limitation on computation or data). The L-ELM

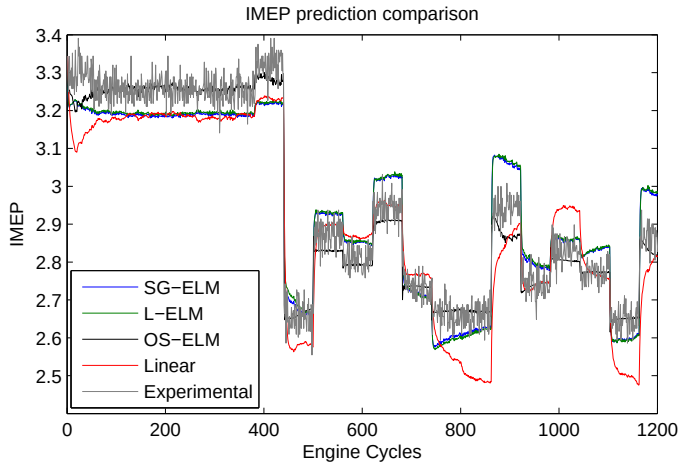


Figure 6.7: Comparison of multi-step ahead predictions of IMEP of the online models. The color codes are as follows - black:OS-ELM, blue:SG-ELM, green:L-ELM, red:linear and grey:experimental.

and SG-ELM are less accurate than the OS-ELM but with further continuation of the online learning task, is expected to improve. The multi-step ahead predictions of the models are compared in Fig. 6.7.

Table 6.7: Performance comparison of the nonlinear online models (OS-ELM, L-ELM and SG-ELM) for the regression learning problem. A baseline linear model and an offline trained ELM model (O-ELM) are also used for comparison.

	Training Time	OSAP RMSE	MSAP RMSE
Linear	0.415	0.081	0.132
OS-ELM	0.731	0.091	0.061
L-ELM	0.467	0.099	0.107
SG-ELM	0.344	0.100	0.105
O-ELM	-	0.060	0.053

Similarly, the online models are developed for CA50, P_{max}, R_{max}, Torque and EAFR. The predictions are compared in Fig. 6.8. This task is a case of multi-input multi-output modeling which adds some limitations of the L-ELM and SG-ELM methods. When the model complexity increases, the L-ELM and SG-ELM require more excitations for convergence, further decreasing the convergence rate. Also, for multiple variables with different noise characteristics, the L-ELM requires separate

tuning of the parameters A_L, Γ and Q. This is a time consuming task if done by trial and error. The OS-ELM on the other hand, requires much less tuning and extension to a multi-output task is straight forward. Further, an important advantage of the OS-ELM algorithm is as follows. The covariance matrix contains useful information on the parameter convergence levels. Several values close to zero would imply the corresponding parameters are close to convergence. This information can be used in real applications to identify if the online learning is completed and if the models can be used for prediction purposes. Such information may not be available with the L-ELM and SG-ELM algorithms.

6.6 Online Envelope Learning for HCCI Engine

In chapter V, a class imbalance learning was performed to identify the stable operating envelope of HCCI engine. In this section, this task is performed online. As noted previously, the HCCI operating envelope is a function of HCCI measurement variables like pressures and temperatures of the manifolds and the cylinders. In addition to the engine thermal conditions, HCCI operation depends on the ambient conditions including temperature, pressure and humidity [111]. Also, the envelope is a function of engine speed (which is held constant in the previous sections). In a vehicle level application, it is important to capture the effects of such constrained variables on the stabile operating boundary. Hence, there is a need for a framework that can adapt online.

The online learning problem translates to an online classification task for which the following algorithms are used namely, the OS-ELM algorithm and the SG-ELM algorithm. The L-ELM cannot be used for a classification task as a state space representation of the dynamic classifier (see equation (5.15)) is not easy to achieve. A RLS based linear classification algorithm is also used as a baseline comparison. A weighted classification version of the algorithms is developed to handle the class

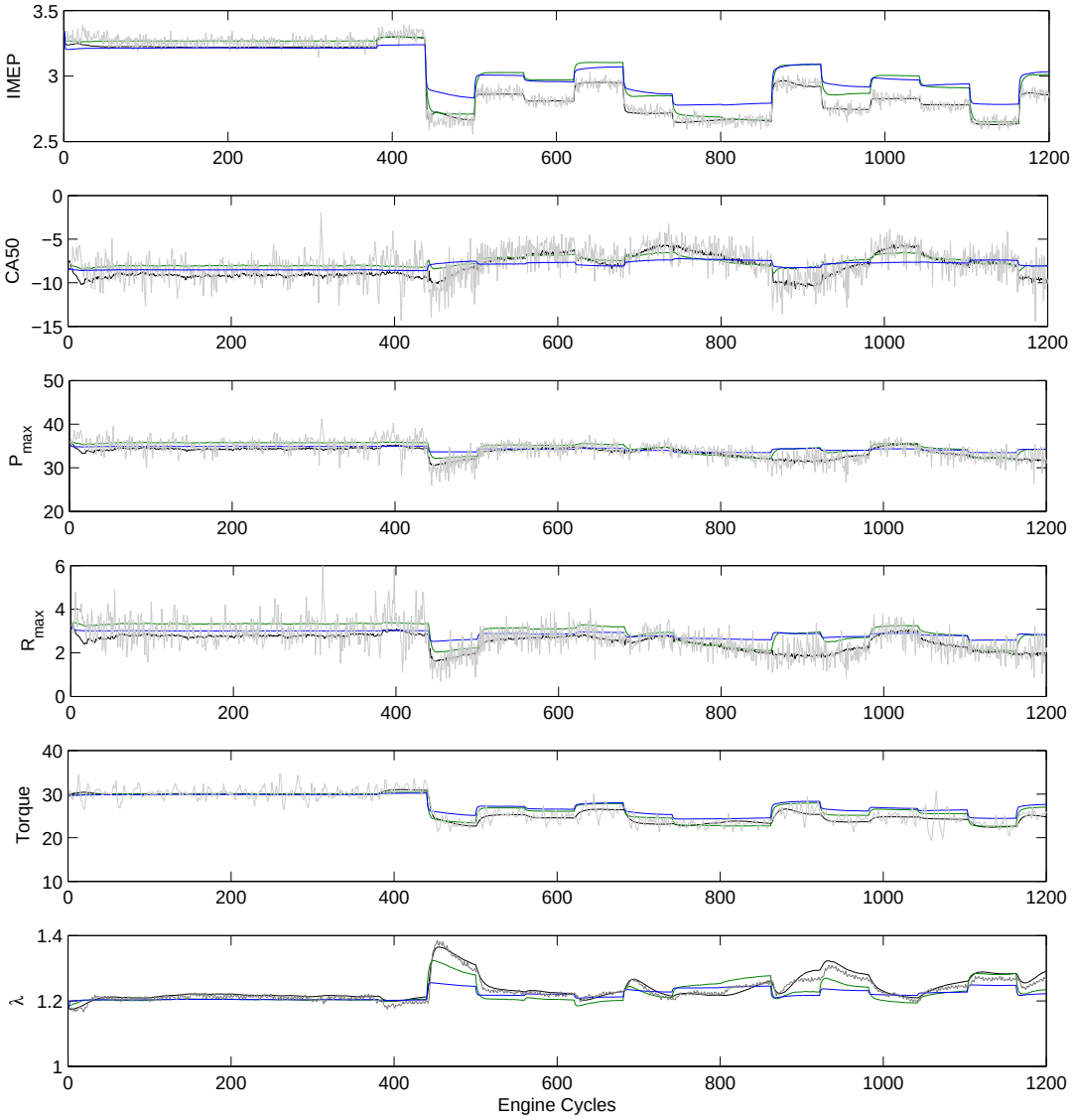


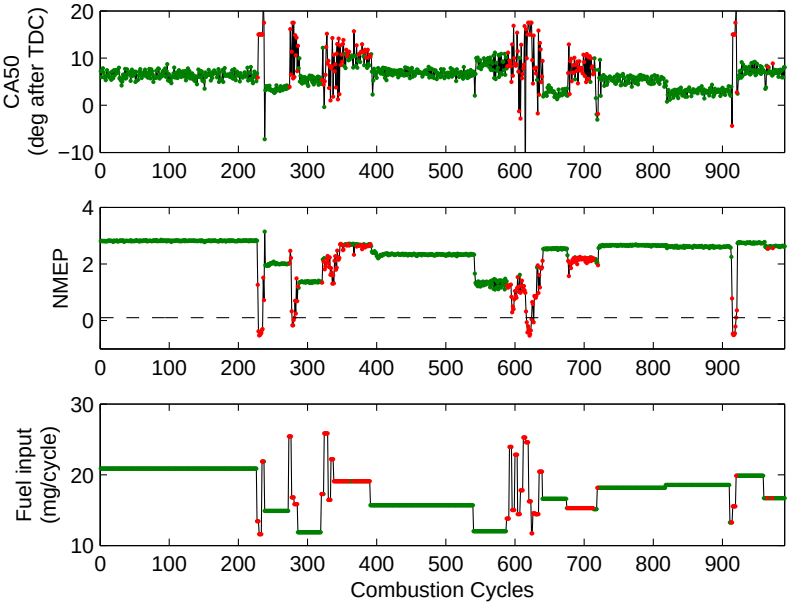
Figure 6.8: Comparison of multi-step ahead predictions of IMEP, CA50, P_{max}, R_{max}, Torque and EAFR by all online models. The color codes are as follows - black:OS-ELM, blue:SG-ELM, green:L-ELM, grey:experimental.

imbalance problem. A subset of the data set used in chapter V with a modified imbalance ratio used for offline classification has been used to simulate online learning by processing data one-by-one.

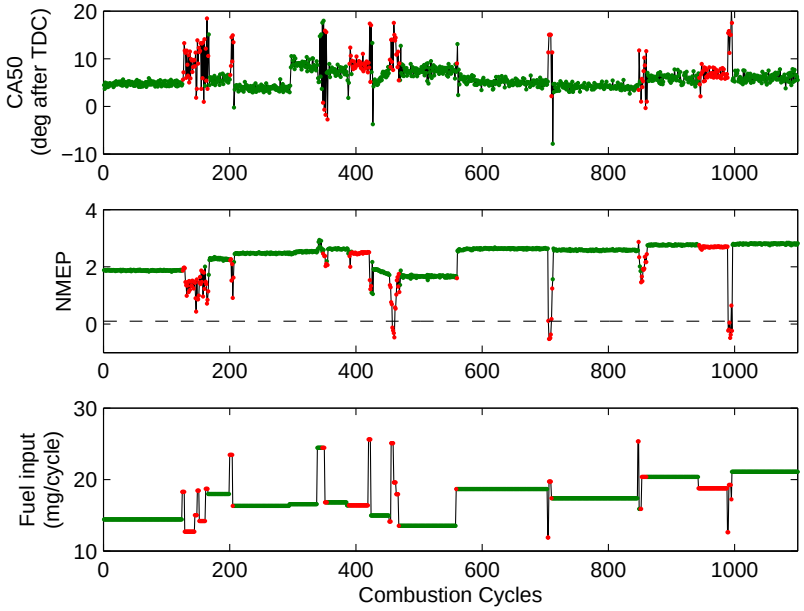
The engine data set is a 13 dimension problem with an imbalance ratio of about 5.3. Online learning is performed using about 14300 cycles of data streaming from the engine. The models are evaluated using the next 6200 cycles of data. An ELM model with 10 hidden neurons are used for all the nonlinear models. The parameters of SG-ELM take values as $\Gamma_{SG} = 0.001 \quad I_{10}$ and $f = 4$. About 140 cycles of data are used to initialize the parameters and covariance matrix of OS-ELM. After performing online learning, the prediction performance of the algorithms are evaluated using the unseen test data stream. Table 6.8 summarizes the computational time as well as the accuracies of prediction for the considered models. It can be observed that the online models perform well for the HCCI boundary identification problem. The problem is mildly nonlinear as linear models achieve a slightly inferior accuracy. Both OS-ELM and SG-ELM perform well and achieve results similar to an offline model indicating completeness of learning. The SG-ELM has significant advantages in terms of computational efficiency. The algorithm is simple and requires less than half of the time required to train OS-ELM. Further, for the considered classification problem, the prediction accuracy of SG-ELM is slightly better than OS-ELM indicating the suitability of SGD based online learning for the HCCI problem. The predictions of the online SG-ELM model is shown in Fig. 6.9.

Table 6.8: Performance comparison of the nonlinear models (OS-ELM and SG-ELM) for the online class imbalance learning problem. A baseline linear model and an offline trained ELM model (O-ELM) are also used for comparison.

Algorithms	Training Time	TPR	TNR	Average Accuracy	Gmean Accuracy
SG-ELM	0.29	0.89	0.81	0.85	0.85
OS-ELM	0.63	0.83	0.83	0.83	0.83
O-ELM	-	0.85	0.85	0.85	0.85
Linear	0.64	0.95	0.65	0.80	0.79



(a)



(b)

Figure 6.9: Prediction results of the SG-ELM algorithm showing CA50, IMEP and one input variable (fueling) for 2 unseen data sets. The color code indicates model prediction - green (and red) indicate stable (and unstable) prediction by the model. The dotted line in the IMEP plot indicates misfire limit, dotted ellipse in CA50 plot indicates high variability instability mode while dotted rectangle indicates a wrong predictions by model.

CHAPTER VII

Controls Development - Model Predictive Control using Extreme Learning Machine Models

In this chapter, a control framework is developed for HCCI engines that makes use of the machine learning models developed in the previous chapters. Although black-box models do not give any physical insight for controls development, can be used for evaluating optimal control trajectories as in a model predictive control (MPC) method. The MPC method is a powerful method especially for black box based models for computing optimal control in the presence of system level and operating level constraints [112, 9]. The ELM based models are preferred over the ANN and SVM based models for its ease of training and superior generalization capability (see chapter IV) and are used in designing MPC controller in this chapter.

7.1 MPC Formulation using Extreme Learning Machines

In this section, the MPC problem is formulated based on general predictive control principles [112, 113]. Let the state space representation of a discrete time nonlinear

model of the underlying phenomenon of interest be given by

$$\begin{aligned} z(k+1) &= f(z(k), u(k)) \\ y(k) &= g(z(k)) \end{aligned} \tag{7.1}$$

where $z \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^p$, k the time index, n, m and p represent the number of states, inputs and outputs respectively. $f(.)$ and $g(.)$ are continuously differentiable and globally Lipschitz [101] nonlinear functions modeled offline using ELM. Around any operating point (z_0, u_0), the models can be linearized using Taylor's series expansion as follows

$$\begin{aligned} z(k+1) &= f(z^0(k), u^0(k)) + A(z(k) - z^0(k)) + B(u(k) - u^0(k)) + \epsilon_{z,ho} \\ y(k) &= g(z^0(k)) + C(z(k) - z^0(k)) + \epsilon_{y,ho} \end{aligned}$$

$$\begin{aligned} A &= \left[\frac{\partial f}{\partial z}(z(k), u(k))\right]_{z^0(k), u^0(k)} \\ B &= \left[\frac{\partial f}{\partial u}(z(k), u(k))\right]_{z^0(k), u^0(k)} \\ C &= \left[\frac{\partial g}{\partial z}(z(k), u(k))\right]_{z^0(k), u^0(k)} \end{aligned}$$

where A, B and C represent the partial derivatives of the ELM models in the Taylor's expansion, $\epsilon_{z,ho}$ and $\epsilon_{y,ho}$ represent higher order terms. The linearized model can be further written as

$$\begin{aligned} z(k+1) &= Az(k) + Bu(k) + d_1(k) \\ y(k) &= Cz(k) + d_2(k) \end{aligned} \tag{7.2}$$

where

$$\begin{aligned} d_1 &= f(z^0(k), u^0(k)) - (Az^0(k) + Bu^0(k)) + \epsilon_{z,ho} \\ d_2 &= g(z^0(k), u^0(k)) - Cz^0(k) + \epsilon_{y,ho} \end{aligned}$$

The following subsections describe the calculation of the system matrices A, B and C followed by the MPC optimization problem formulation.

7.1.1 Calculation of System Matrices

The matrices A, B and C in equation (7.2) can be determined by exploiting the structure of the ELM model as follows. Let the augmented input vector to ELM be given by $x(k) = [u(k), z(k)]^T \in \mathbb{R}^{n+m}$. The matrices A, B and C can be determined by calculating the Jacobian of $f(.)$ and $g(.)$ with respect to the augmented input vector $x(k)$. The ELM model structure can be expressed as

$$\hat{z}(k+1) = W^T[\psi(W_r^T x(k) + b_r)] \tag{7.3}$$

where ψ represents the hidden layer activation function and $W_r \in \mathbb{R}^{n+m \times n_h}, W \in \mathbb{R}^{n_h \times n}$ represents the input and output layer parameters respectively. Here, n_h represents the number of hidden neurons of the ELM model, $\phi(k) = \psi(W_r^T x(k) + b_r) \in \mathbb{R}^{n_h \times 1}$ represents the hidden layer output matrix as termed in literature (see Fig. 2.10). The matrix W_r consists of randomly assigned elements that maps the input vector to a high dimensional feature space while $b_r \in \mathbb{R}^{n_h}$ is a bias component assigned in a random manner similar to W_r. The elements can be assigned based on any continuous random distribution [50] and remains fixed during the learning process. The number of hidden neurons determine the dimension of the transformed feature space and the hidden layer is equipped with a nonlinear activation function similar to traditional neural network architecture. In this paper, a sigmoidal activation function

is considered.

Note that the ELM model in (7.3) is defined for inputs and outputs normalized to lie between $[-1, +1]$. Expressing the model in (7.3) along with the normalization and de-normalization terms,

$$\begin{aligned} \hat{z}(k+1) &= z_{min} + \left(\frac{z_{max} - z_{min}}{2}\right)\left\{1 + W^T\phi(k)\right\} \\ \phi(k) &= \frac{1}{1 + e^{-\left\{W_r^T\left[2\left(\frac{x(k)-x_{min}}{x_{max}-x_{min}}\right)-1\right]+b_r\right\}}} \end{aligned} \tag{7.4}$$

Then, Jacobian matrix $\frac{\partial f}{\partial x}$ can be derived (ignoring the time index k) as

$$\frac{\partial f}{\partial x} = \left(\frac{z_{max} - z_{min}}{2}\right) W^T \frac{\partial \phi}{\partial x} \tag{7.5}$$

$$\frac{\partial \phi_i}{\partial x_j} = \frac{2W_r(j,i)e^{-\left\{W_{r,1}^T\left[2\left(\frac{x-x_{min}}{x_{max}-x_{min}}\right)-1\right]+b_{r,1}\right\}}}{(x_{max,j} - x_{min,j})\left(1 + e^{-\left\{W_{r,1}^T\left[2\left(\frac{x-x_{min}}{x_{max}-x_{min}}\right)-1\right]+b_{r,1}\right\}}\right)^2}$$

where $W_{r,i}$ represents the i^{th} column of W_r and $b_{r,i}$ represents the i^{th} element of b_r, $\frac{\partial \phi}{\partial x} \in \mathbb{R}^{n_h \times n+m}$. Since the augmented vector is defined as $x(k) = [u(k), z(k)]^T$, the matrices A and B can be extracted from the Jacobian $\frac{\partial f}{\partial x}$ as

$$\left[\frac{\partial f}{\partial x}\right]_{n \times n+m} = [B_{n \times m} | A_{n \times n}]. \tag{7.6}$$

A similar jacobian calculation can be done for $g(.)$ and also for other type of activation functions $\psi(.)$. The above calculations are algebraic and can be efficiently performed online.

7.1.2 MPC Optimization Problem

The goal of MPC is to force the system output $y(k)$ track a given reference $r(k)$ and also penalize any large excursion in the input signal $u(k)$. This is obtained by

solving the following optimization problem at every time instant k

$$J(k) = \sum_{j=1}^{N_y} \|r(k+j|k) - y(k+j|k)\|^2_{Q_1} + \sum_{j=0}^{N_u-1} \|\Delta u(k+j|k)\|^2_{Q_2} \tag{7.7}$$

$$\text{subjected to} \begin{cases} u_{min} \leq u(k+j|k) \leq u_{max} \\ \Delta u_{min} \leq \Delta u(k+j|k) \leq \Delta u_{max} \\ y_{min} \leq y(k+j|k) \leq y_{max} \end{cases} \tag{7.8}$$

where $r(k+j|k), y(k+j|k), \Delta u(k+j|k)$ represents the reference, system output and control increment respectively. The argument $(k+j|k)$ indicates the signal from time index k until $k+j$ being used for solving the optimization problem at time index k. Here $\Delta u(k+j|k) = u(k+j|k) - u(k-1+j|k)$, N_y and N_u are prediction horizon ($N_y \geq 1$) and control horizon ($0 < N_u \leq N_y$) respectively, $\|.\|_{Q_1}$ and $\|.\|_{Q_2}$ denote weighted Euclidean norm weighted by matrices Q_1 and Q_2 respectively. The minimum and maximum constraints for u, Δu and y are given as in equation (7.8).

By defining the following vectors,

$$\begin{aligned} Y(k) &= [y(k+1|k), y(k+2|k), .., y(k+N_y|k)]^T \\ \Delta U(k) &= [\Delta u(k|k), \Delta u(k+2|k), .., \Delta u(k+N_u-1|k)]^T \\ R(k) &= [r(k+1|k), r(k+2|k), .., r(k+N_y|k)]^T \end{aligned}$$

and calculating $y(k+j|k)$ recursively using equation (7.2), the vector $Y(k)$ can be expressed as

$$Y(k) = \mathcal{Z}(k)z(k) + \mathcal{U}(k)\Delta U(k) + \mathcal{V}(k)u(k-1) + \mathcal{D}_1(k)d_1(k) + \mathcal{D}_2(k)d_2(k) \tag{7.9}$$

where

$$\mathcal{U} = \begin{bmatrix} CB & . & . & 0 \\ CB + CAB & . & . & . \\ . & . & . & . \\ CB + .. + CA^{N_u-1}B & . & . & CB \\ CB + .. + CA^{N_u}B & . & . & CB + CAB \\ . & . & . & . \\ CB + .. + CA^{N_y-1}B & . & . & CB + .. + CA^{N_y-N_u}B \end{bmatrix}$$

$$\mathcal{V} = \begin{bmatrix} CB \\ CB + CAB \\ . \\ . \\ CB + CA^{N_y-1}B \end{bmatrix}, \mathcal{Z} = \begin{bmatrix} CA \\ CA^2 \\ . \\ . \\ CA^{N_y} \end{bmatrix}, \mathcal{D}_1 = \begin{bmatrix} C \\ C + CA \\ . \\ . \\ C + CA^{N_y-1} \end{bmatrix}, \mathcal{D}_2 = \begin{bmatrix} I_p \\ I_p \\ . \\ . \\ I_p \end{bmatrix}$$

Here, $Q_1 \in \mathbb{R}^{N_y p \times N_y p}, Q_2 \in \mathbb{R}^{N_u m \times N_u m}, Y \in \mathbb{R}^{N_y p \times 1}, \Delta U \in \mathbb{R}^{N_u m \times 1}, R \in \mathbb{R}^{N_y p \times 1}$, $\mathcal{U} \in \mathbb{R}^{N_u p \times N_u m}, \mathcal{V} \in \mathbb{R}^{N_y p \times m}, \mathcal{Z} \in \mathbb{R}^{N_y p \times n}, \mathcal{D}_1 \in \mathbb{R}^{N_y p \times n}, \mathcal{D}_2 \in \mathbb{R}^{N_y p \times p}$.

The optimization problem in equation (7.7) can now be expressed in vector form as

$$\min_{\Delta U(k)} \left\{ \Delta Y^T(k) Q_1 \Delta Y(k) + \Delta U^T(k) Q_2 \Delta U(k) \right\} \tag{7.10}$$

where

$$\Delta Y(k) = R(k) - Y(k) = R(k) - \mathcal{Z}(k) z(k) - \mathcal{U}(k) \Delta U(k) - \mathcal{V}(k) u(k-1) - \mathcal{D}_1(k) d_1(k) - \mathcal{D}_2(k) d_2(k)$$

Expressing as a quadratic programming problem, the above formulation reduces to

$$\min_{\Delta U(k)} \left\{ \frac{1}{2} \Delta U^T(k) W_1(k) \Delta U(k) + W_2^T(k) \Delta U(k) \right\} \tag{7.11}$$

$$\text{subjected to } E(k) \Delta U(k) \leq F(k) \tag{7.12}$$

where

$$
\begin{aligned}
W_1(k) &= 2[\mathcal{U}^T(k)Q_1\mathcal{U}(k) + Q_2] \\
W_2(k) &= -2\mathcal{U}^T(k)Q_1[R(k) - \mathcal{Z}(k)z(k) \\
&\quad -\mathcal{V}(k)u(k-1) - \mathcal{D}_1(k)d_1(k) - \mathcal{D}_2(k)d_2(k)] \\
E(k) &= [I_{N_u m} \; -I_{N_u m} \quad H \; -H \quad \mathcal{U}(k) \; -\mathcal{U}(k)]^T \\
F(k) &= \begin{bmatrix}
\Delta U_{max} \\
-\Delta U_{min} \\
U_{max} - U_{k-1} \\
-\{U_{min} - U_{k-1}\} \\
\\
Y_{max} - \mathcal{Z}(k)z(k) - \mathcal{V}(k)u(k-1) \\
-\mathcal{D}_1(k)d_1(k) - \mathcal{D}_2(k)d_2(k) \\
\\
-\{Y_{min} - \mathcal{Z}(k)z(k) - \mathcal{V}(k)u(k-1) \\
-\mathcal{D}_1(k)d_1(k) - \mathcal{D}_2(k)d_2(k)\}
\end{bmatrix} \\
H &= \begin{bmatrix}
I_m & 0 & . & . & 0 \\
I_m & I_m & . & . & 0 \\
. & . & . & . & . \\
. & . & . & . & . \\
I_m & I_m & . & . & I_m
\end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
U_{min} &= [u_{min} \quad u_{min}..u_{min}]^T \\
U_{max} &= [u_{max} \quad u_{max}..u_{max}]^T \\
\Delta U_{min} &= [\Delta u_{min} \quad \Delta u_{min}..\Delta u_{min}]^T \\
\Delta U_{max} &= [\Delta u_{max} \quad \Delta u_{max}..\Delta u_{max}]^T \\
U_{k-1} &= [u(k-1) \quad u(k-1)..u(k-1)]^T \\
Y_{min} &= [y_{min} \quad y_{min}..y_{min}]^T \\
Y_{max} &= [y_{max} \quad y_{max}..y_{max}]^T
\end{aligned}
$$

Here, I_m, I_p and $I_{N_u m}$ represents identity matrices in $\mathbb{R}^{m \times m}$, $\mathbb{R}^{p \times p}$ and $\mathbb{R}^{N_u m \times N_u m}$ respectively. The above problem can be solved using commercial quadratic programming solvers that are efficient [112]. The first value of the optimal control increment $\Delta U(k)$ is applied at the present time instant k and the optimization is solved again for the next time index. This is done to handle any model inaccuracies and external disturbances.

7.2 HCCI Optimal Control

In this section, the design and development of an optimal tracking controller for HCCI engine is discussed. The MPC framework is shown in Figure 7.1. An offline trained nonlinear model of the HCCI engine is used. The model is developed as a first order system with 20 hidden neurons. The dynamic equations of the model can

be expressed as in (7.1) where

$$\begin{aligned} z(k) &= [IMEP(k) \quad CA50(k) \quad P_{max}(k) \quad R_{max}(k) \quad T_b(k) \quad \lambda(k)]^T \in \mathbb{R}^6 \\ u(k) &= [rk(k) \quad evc(k) \quad soi(k)]^T \in \mathbb{R}^3 \\ y(k) &= [IMEP(k) \quad CA50(k)]^T \in \mathbb{R}^2 \end{aligned}$$

Using the engine model, the controller is designed using simulations.

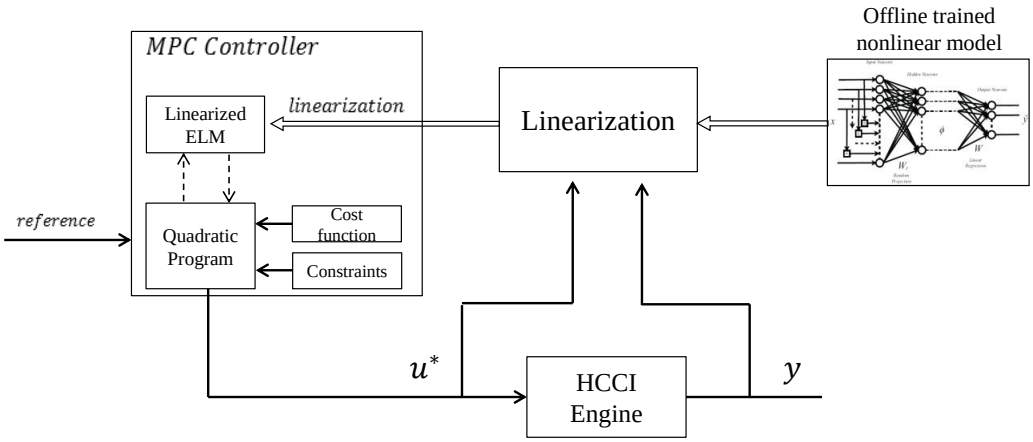


Figure 7.1: MPC Framework

7.2.1 Fast Quadratic Programming

The MPC problem involves solving a quadratic programming subproblem of the form given by equation (7.11). The generic quadratic programming solver demands computation and memory owing to iterative interior point, active sets or trust region approximations and becomes infeasible for implementation on the engine ECU. In order to reduce the computation and memory, an efficient and fast QP algorithm is adopted in this work [114]. The algorithm can be derived as follows.

Restating the MPC quadratic programming problem from (7.11),

$$\min_{\Delta U(k)} \left\{ \frac{1}{2} \Delta U^T(k) W_1(k) \Delta U(k) + W_2^T(k) \Delta U(k) \right\} \tag{7.13}$$

$$\text{subjected to } E(k)\Delta U(k) - F(k) \leq 0 \tag{7.14}$$

where $W_1(k)$ is a symmetric positive definite matrix and the objective function is strictly convex. The lagrangian dual problem can be formulated as

$$\max_{\lambda_L} \inf \left\{ \frac{1}{2}\Delta U^T(k) W_1(k) \Delta U(k) + W_2^T(k)\Delta U(k) + \lambda_L(E(k)\Delta U(k) - F(k)) \right\} \tag{7.15}$$

where $\lambda_L \geq 0$. Taking the derivative with respect to ΔU,

$$W_1(k)\Delta U(k) + W_2(k) + E^T(k)\lambda_L = 0. \tag{7.16}$$

The dual problem can be expressed as

$$\max_{\lambda_L} \left\{ \frac{1}{2}\Delta U^T(k) W_1(k) \Delta U(k) + W_2^T(k)\Delta U(k) + \lambda_L(E(k)\Delta U(k) - F(k)) \right\} \tag{7.17}$$

$$\text{subjected to } W_1(k)\Delta U(k) + W_2(k) + E^T(k)\lambda_L = 0 \tag{7.18}$$

$$y \geq 0 \tag{7.19}$$

Since $W_1(k)$ is positive definite, $W_1(k)^{-1}$ exists and equation (7.16) can be solved as follows

$$\Delta U^* = -W_1(k)^{-1}(W_2(k) + E^T(k)\lambda_L). \tag{7.20}$$

Following a direct substitution of (7.20) in (7.17), the problem becomes

$$\max_{\lambda_L} \left\{ \frac{1}{2}\lambda_L^T \Lambda_1 \lambda_L + \lambda_L^T \Lambda_2 - \frac{1}{2} W_2^T(k) W_1^{-1}(k) W_2(k) \right\} \tag{7.21}$$

$$\text{subjected to } y \geq 0 \tag{7.22}$$

where $\Lambda_1 = -E(k)W_1^{-1}(k)E^T(k)$ and $\Lambda_2 = -F(k) - E(k)W_1^{-1}(k)W_2(k)$. The problem

in (7.21) can be solved using a simple gradient ascent method as follows.

$$\lambda_{L_{k+1}} = \lambda_{L_k} + \lambda_{step}(\Lambda_1 \lambda_L + \Lambda_2). \tag{7.23}$$

In order to satisfy the constraint in (7.22), the following modification is made

$$\lambda_{L_{k+1}} = max(\lambda_{L_k} + \lambda_{step}(\Lambda_1 \lambda_L + \Lambda_2), 0) \tag{7.24}$$

where λ_{step} defines the step size. The solution of (7.24) gives the optimal λ_L^* which can be substituted in (7.20) to get the optimal MPC output ΔU^*. The strictly convex property of the MPC problem is made use of in solving the quadratic programming subproblem efficiently.

7.2.2 Simulations

In this section, the design of the MPC controller and tuning of parameters and controller gains have been discussed. The offline trained nonlinear model is considered as the true HCCI engine plant and controller is designed. The goal of the controller is to track the reference IMEP and CA50 trajectories with minimum error and with minimum change in control input (see (7.7)). The reference trajectory is designed offline depending on the allowable operating regions of HCCI and the valid region of the HCCI engine model. The step references are designed to vary between 2.6 and 3.2 bar IMEP and between -4 and -10 deg CA50 defined in degrees before combustion TDC. This almost covers the range defined by the experimental data used for training

the model. The MPC constraints are defined as follows.

$$\begin{aligned} u_{min} &= [19 \quad -121 \quad 272]^T \\ u_{max} &= [25 \quad -100 \quad 375]^T \\ \Delta u_{min} &= [-6 \quad -22 \quad -103]^T \\ \Delta u_{max} &= [6 \quad 22 \quad 103]^T \\ y_{min} &= [2.1 \quad -14]^T \\ y_{max} &= [3.55 \quad -2]^T \end{aligned}$$

At every simulation step, the HCCI model is linearized around sampled inputs and outputs of the plant. The linear model is used in the online MPC algorithm to determine the optimal control increment to be given to the plant. The prediction and control horizons take values $N_y = 3$ and $N_u = 3$ are tuned for minimum tracking error. Similarly, the gain matrices are tuned to be

$$Q_1 = \begin{bmatrix} 800 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 800 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 800 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}, Q_2 = I_{8\times 8}.$$

It should be noted that the gains are tuned for a very close tracking of the reference. In the experimental setup, the gains take smaller values to avoid noise amplification. The tracking performance of the MPC controller is shown in Fig. 7.2. It can be seen that the controller tracks the references with an accuracy of 0.036 and 0.192 for IMEP and CA50 respectively measured as root mean squared error (RMSE). The corresponding control trajectories can be seen in Fig. 7.3. The control trajectories fall within the actuator limits as defined by the constraints above.

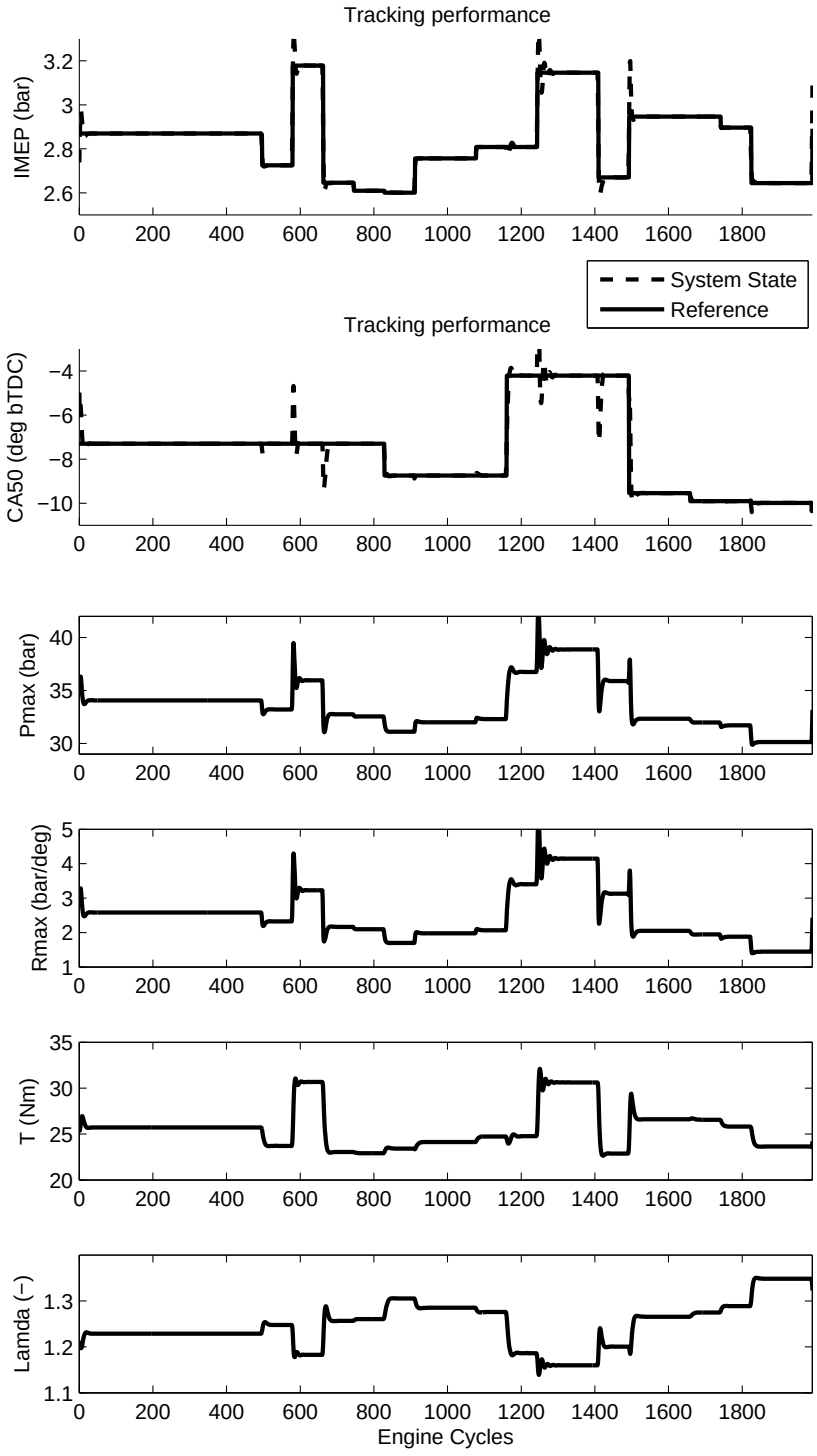


Figure 7.2: State trajectories of the HCCI engine model using MPC control.

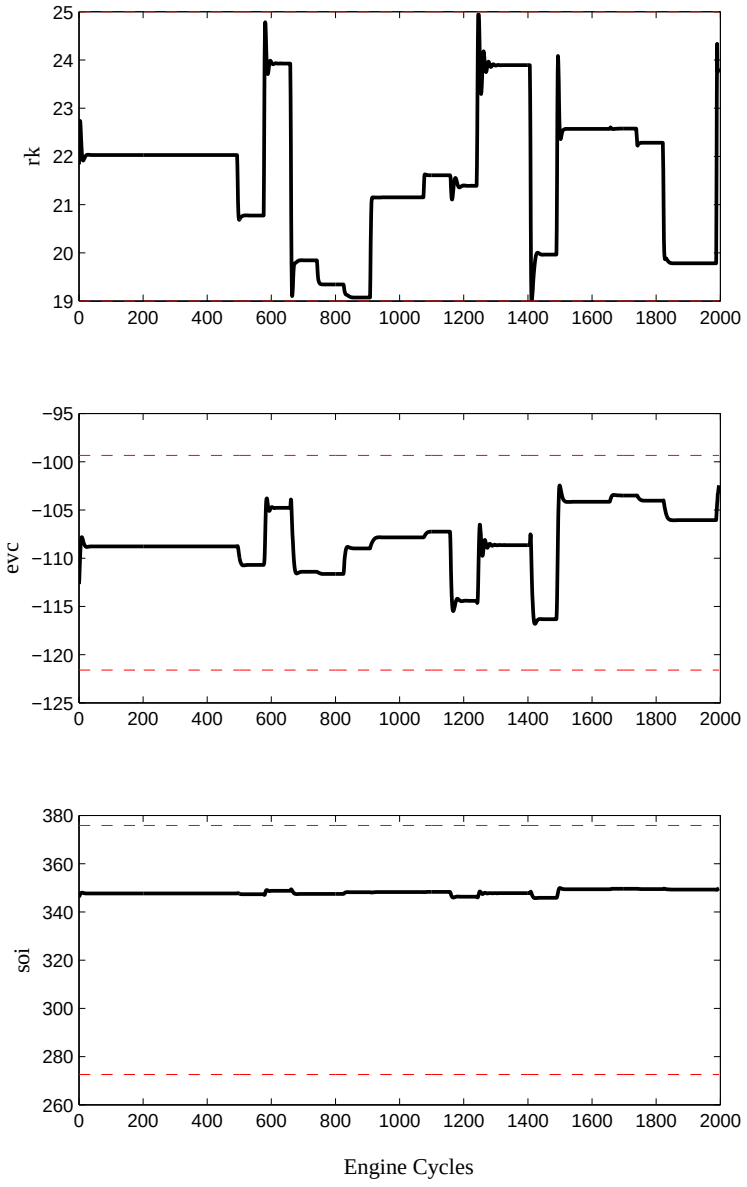


Figure 7.3: Control trajectories of the HCCI engine model using MPC control. The upper and lower limits of each actuator is shown in dotted red.

In order to evaluate the controller's robustness to noise, white noise signals of zero mean and variances 0.0011 and 2.34 are injected at the outputs of IMEP and CA50 respectively. The noise parameters are determined from the actual experimental data. The Figures 7.4 and 7.5 summarize the performance of the MPC controller.

7.3 MPC with Online Model Adaptation

In this section, the online learning algorithms developed for nonlinear systems are used in conjunction with the MPC algorithm to develop an adaptive MPC strategy for the HCCI engine control. The adaptive control framework can be shown in Fig. 7.6.

As mentioned earlier in chapter VI, online learning might be necessary to handle the following situations. The identification model developed using experimental data may involve inaccuracies owing to lack of experimental data at some operating regions. Also, the models are developed at constrained experimental conditions, for instance, at constant ambient temperature, pressure and humidity. Under such conditions, there might exist a slight difference between the engine's behavior and the model's prediction. Since the controller is model based, it is important to adapt the model for such parametric changes. In this demonstration, the online learning algorithms discussed in chapter VI is used to adapt the models online for parametric variations and simultaneously determining optimal control using MPC.

A parametric variation of 10% is considered in the offline estimated models. The variations introduces inaccuracies in the model which results in poor tracking by the MPC controller as shown in Fig. 7.7. Now, the online learning algorithms namely OS-ELM, SG-ELM and L-ELM are implemented to correct the inaccurate model. The parameter matrix W is adapted online and the updated matrix is used in MPC for linearizing the model and obtaining system matrices A, B and C. Using the adapted matrix, the MPC is performed in a similar manner. The parameters of L-

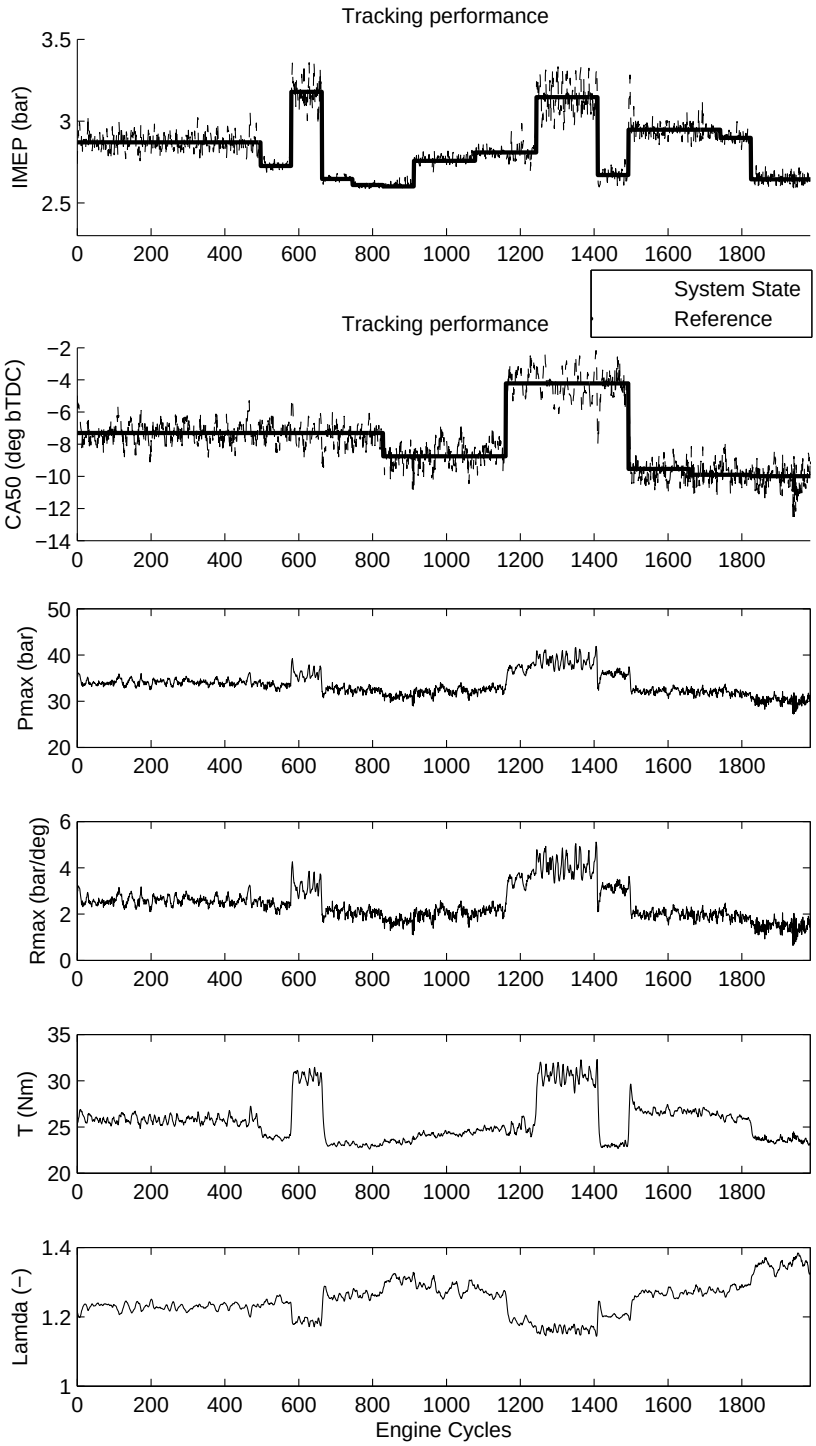


Figure 7.4: State trajectories of the HCCI engine model (with noise) using MPC control.

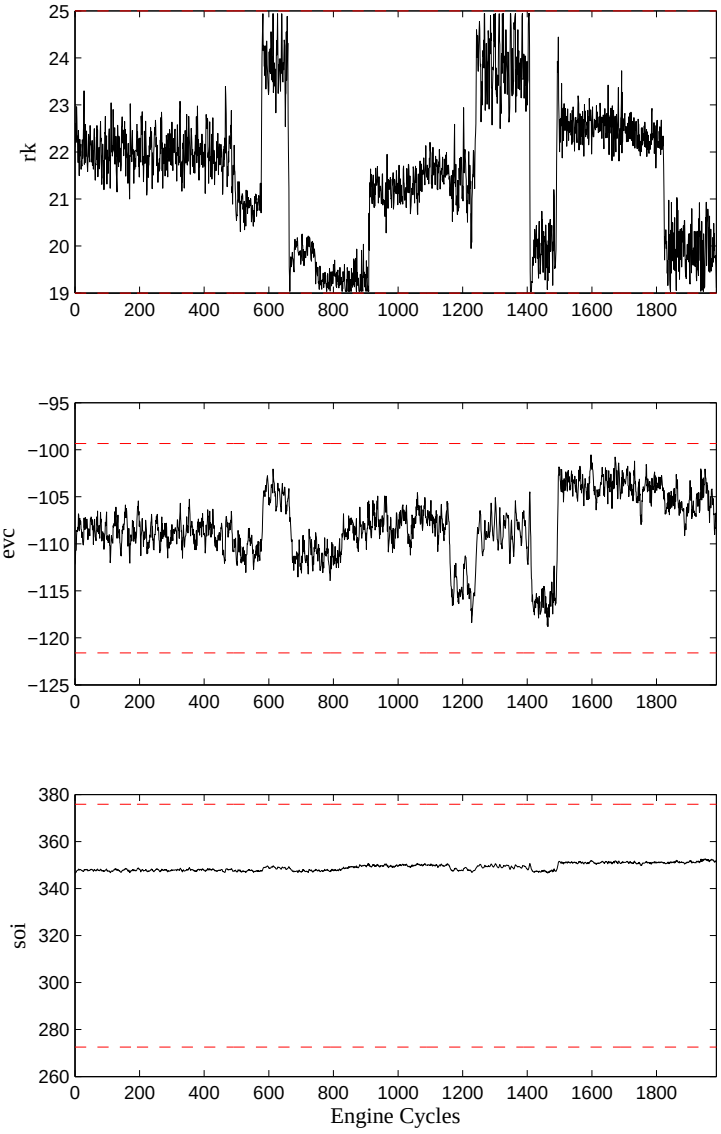


Figure 7.5: Control trajectories of the HCCI engine model (with noise) using MPC control. The upper and lower limits of each actuator is shown in dotted red.

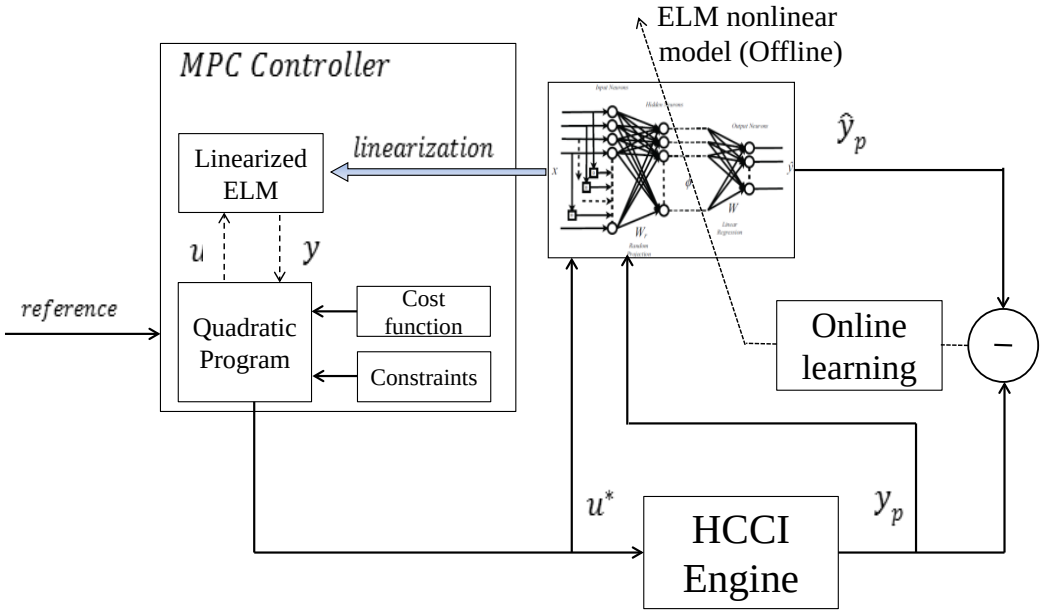


Figure 7.6: Framework showing model predictive control with online learning.

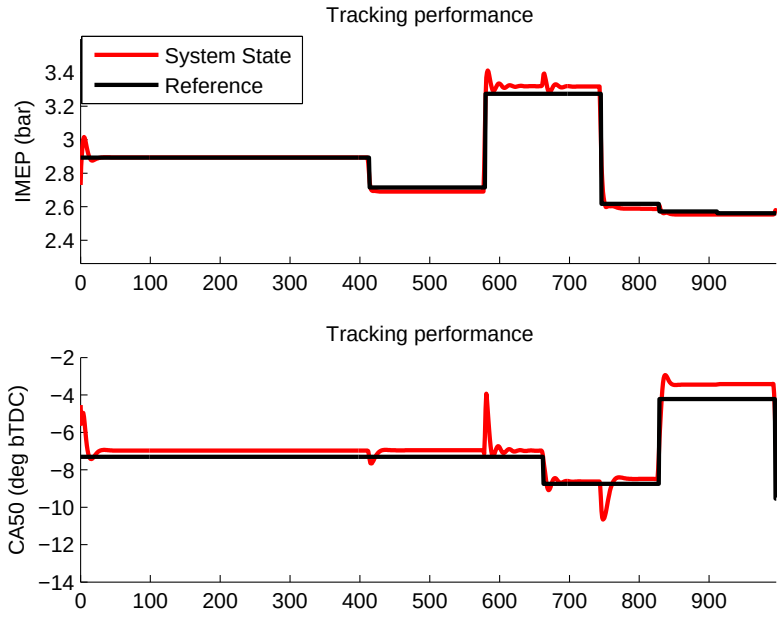


Figure 7.7: Poor control by MPC due to inaccuracies in the ELM model.

ELM and SG-ELM are as follows. $A_L = 1 \times 10^{-4} \quad I_6$, $\Gamma = 0.1 \quad I_{20}$, $Q = 3 \quad I_6$, $\Gamma_{SG} = 0.3 \quad I_{20}$. Compared to no adaptation, the MPC performs better for the cases with online adaptation as indicated by the tracking RMSE of IMEP and CA50 in Table 7.1. The MPC performance can be shown in Fig. 7.8 and the optimal control

Table 7.1: Tracking performance of the MPC controller with online adaptation using L-ELM, SG-ELM and OS-ELM compared against MPC with no adaptation.

	IMEP RMSE	CA50 RMSE
no adaptation	0.0268	0.8257
L-ELM	0.0034	0.3562
SG-ELM	0.0029	0.3537
OS-ELM	0.0088	0.3633

trajectories are shown in Fig. 7.9. It can be observed that, when online adaptation is done, the MPC tracks the reference with a better accuracy compared to the case without online learning. All three algorithms perform equally well with a slight accuracy advantage for SG-ELM as seen in Table 7.1.

The MPC simulation is extended for a long time period to evaluate the robustness of the algorithms in terms of stability in predictions. The behavior of L-ELM and SG-ELM were stable as expected. However, the OS-ELM suffered form the ill-conditioning problem (see chapter VI) and resulted in unstable parameter evolution which led to poor MPC tracking performance as shown in Figures 7.10 and 7.11. The performance of the SG-ELM and L-ELM are shown in Fig. 7.12.

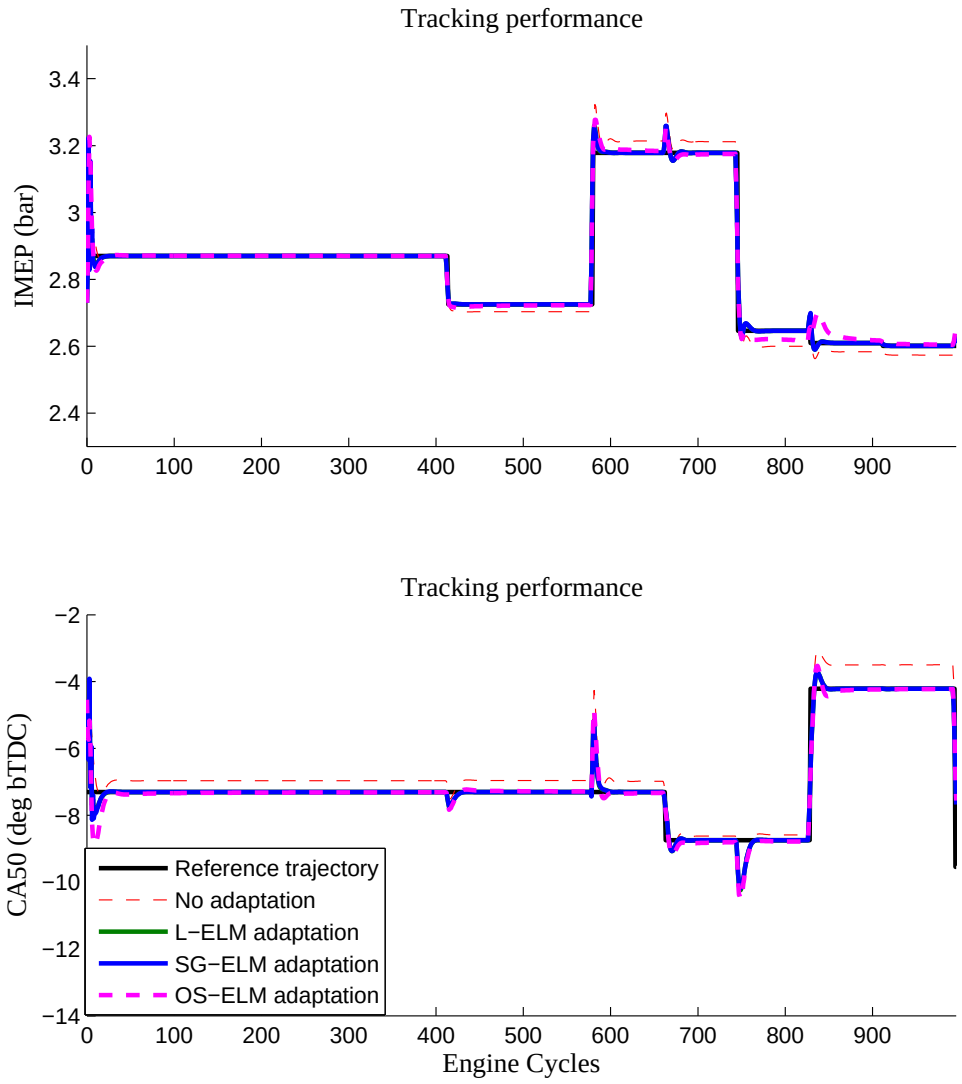


Figure 7.8: Tracking performance of MPC controller with an without online adaptation of the models. The plots compare tracking performance of IMEP and CA50.

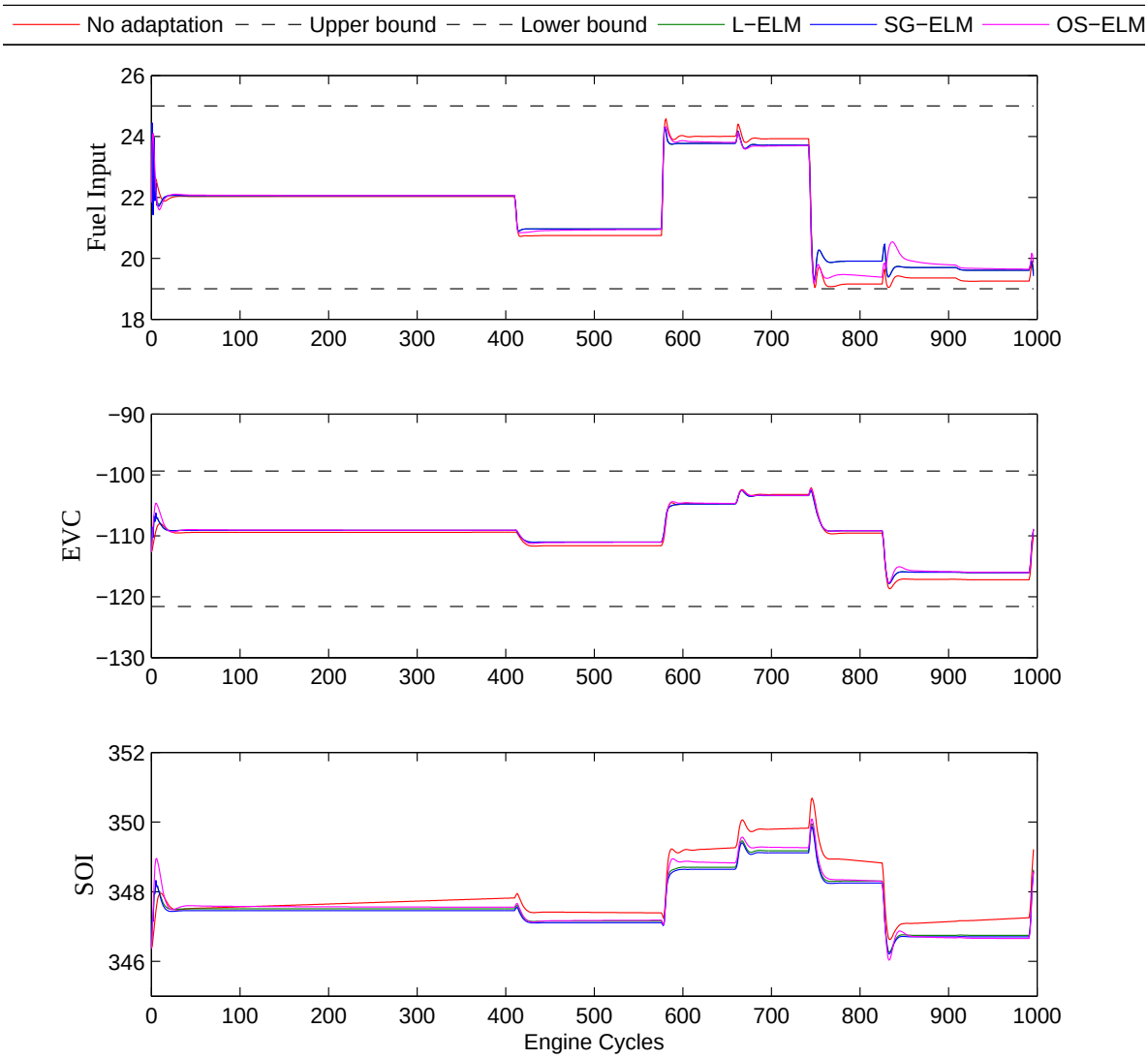


Figure 7.9: Optimal control trajectories of MPC controller with an without online adaptation of the models.

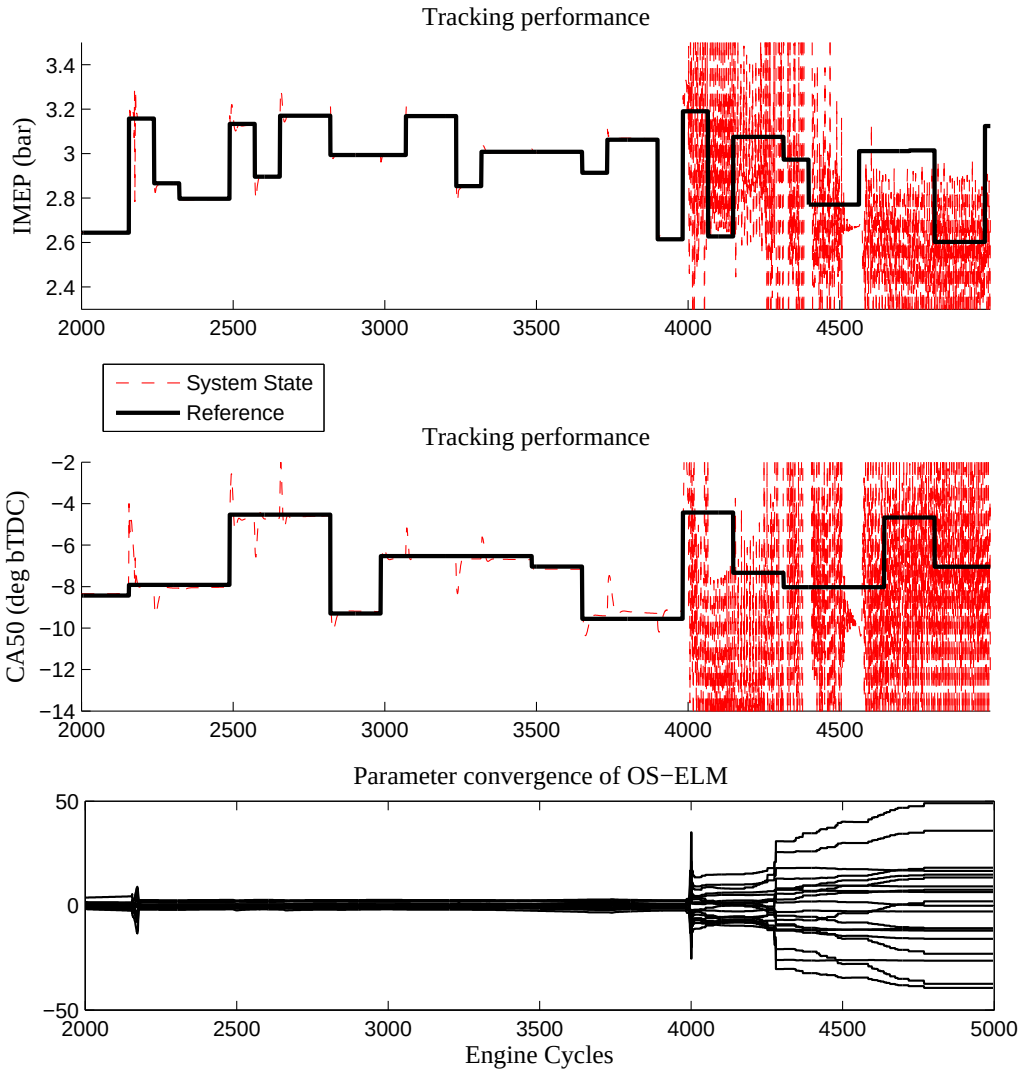


Figure 7.10: Tracking performance of MPC controller with OS-ELM adaptation. The model parameters grows unbounded resulting in unstable MPC control.

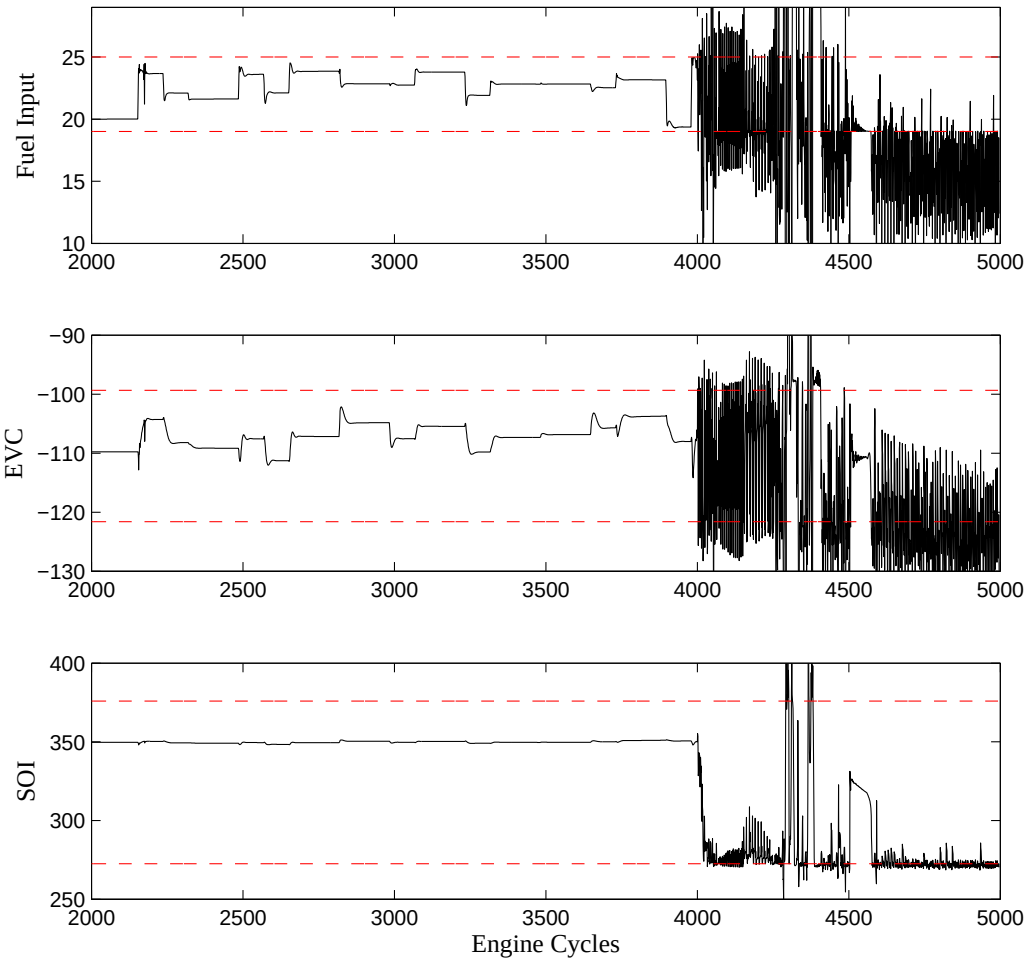


Figure 7.11: Control trajectories of MPC controller with OS-ELM adaptation.

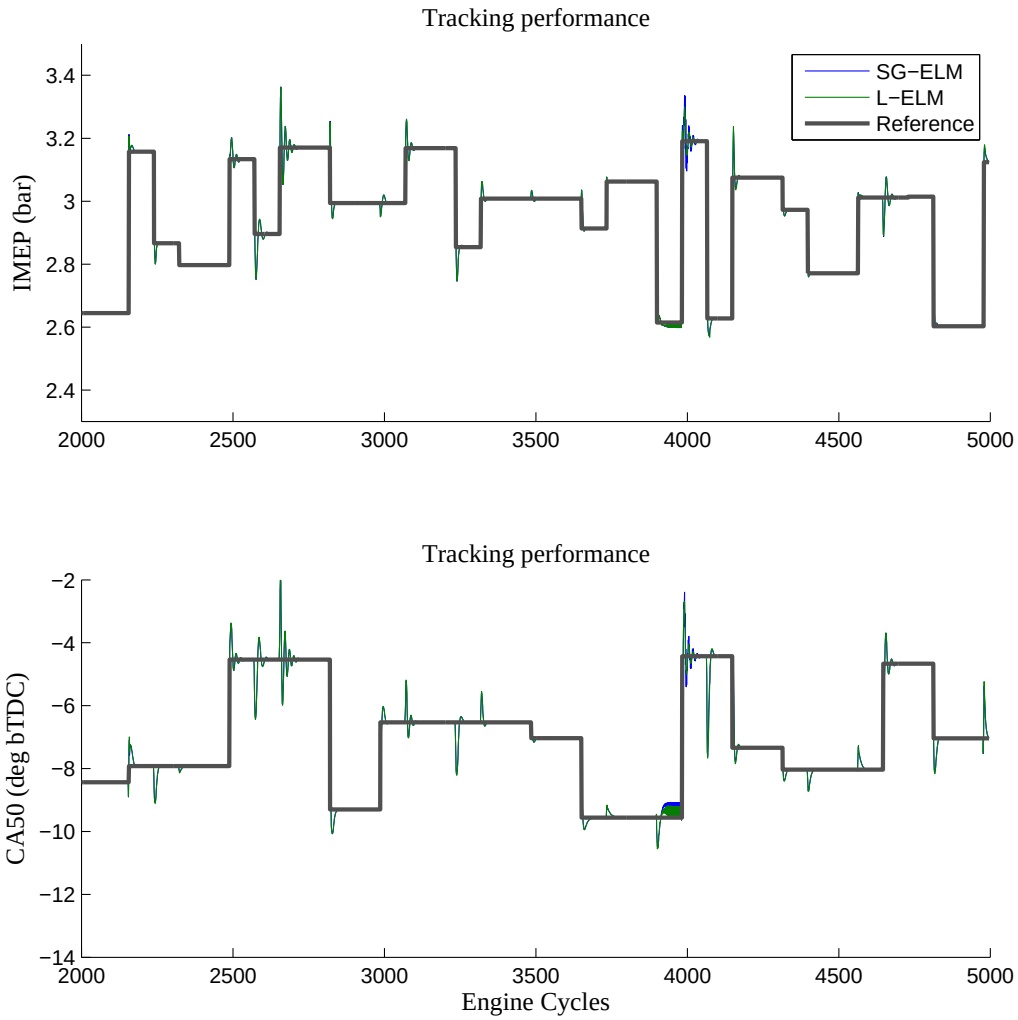


Figure 7.12: Tracking performance of MPC controller with L-ELM and SG-ELM adaptation.

CHAPTER VIII

Conclusions and Future Work

8.1 Summary of Research

This thesis develops novel algorithms and frameworks to solve some of the modeling and control problems encountered in a homogeneous charge compression ignition engine. Several insights have been generated with respect to experimentation, data processing, model selection and efficient learning from HCCI engine data. A machine learning modeling approach is considered in this work demonstrating its capabilities in terms of having a short development time, having high accuracies, solving complex problems, and possessing beneficial design properties like generalization and stability. Application frameworks for answering complex questions in HCCI modeling and control were developed. Novel online learning algorithms and model based optimal control algorithms were developed for any generic nonlinear system and application to the HCCI system demonstrated.

Chapter 2 introduces some key ideas in nonlinear system identification. Preliminaries on learning from dynamic data, supervised learning process, model architectures for one-step ahead and multi-step ahead predictions are briefed. A detailed description of the learning models considered in this work is included along with literature review and mathematical formulations. Although neural networks are effective for many applications, they are prone to finding a local optimal solution and

with a slow convergence rate which limits itself to relatively simpler tasks. SVM is designed to solve a convex optimization problem but the computation and memory required is high and becomes infeasible for the HCCI engine problem. ELM models on the other hand, are more efficient in terms of computation as well as optimality, are used as base model structures for the algorithms developed in this thesis.

A key component of system identification is to excite the system and obtain sufficiently rich data. The HCCI engine being both a nonlinear system and with a narrow stable region of operation creates a bottleneck for experiment design. The experimental setup and experiment design for identification of the HCCI engine is summarized in Chapter 3. A multi level random sequence is designed to obtain transient response of HCCI combustion. In order to reduce engine misfires, a steady state model was developed using extreme learning machines and used as a filter to eliminate unstable excitations during the transient experiments. Further, with a controller availability, the transient experiments were performed with a feedback controller in closed-loop.

Chapter 4 summarizes the frameworks for constructing HCCI engine simulators using three state-of-the-art machine learning algorithms namely neural networks, support vector machines and extreme learning machines. A systematic procedure is detailed for offline system identification of the HCCI engine variables including IMEP, CA50, P_{max}, R_{max}, engine torque, EAFR etc. Cross-validation is used to perform model structure learning and the model parameters are determined using established algorithms in literature. The summary of model evaluations is as follows. The ELM models outperform the ANN and SVM models both in terms of accuracy in one-step ahead predictions and multi-step ahead predictions. The ANN models call for a high computational overhead for calculating the Jacobian of high dimensional data sets. Similar computational overhead is required by the SVR models. Even though training time is not considered, the quick training by ELM enabled efficient model development and tuning thanks to the ELM model's random projection step which enabled

handling high dimensional data sets with ease. Further, ELM models can handle multiple outputs directly compared to ANN and SVM models which reduces training time significantly. An interpolation based approach is adopted for predicting HCCI engine behavior at different engine speeds and initial results are validated. Finally, the ELM models have a simple online adaptation algorithm (see chapter VI) suitable for onboard engine applications constrained by computational resources. Hence, ELM models are chosen as suitable model structures in this work.

Complex and highly sensitive systems such as HCCI engines have a narrow region of stable operation and it is important to gain knowledge about the stable operating envelope for diagnostics and controls development. In chapter 5, a novel solution using machine learning has been developed that predicts if the future HCCI combustion events are stable or not based on past and present measurements along with excitation inputs. The algorithm could potentially be used as a misfire prediction tool for HCCI combustion. A classification problem has been formulated and solved using linear and nonlinear methods such as logistic regression, linear regression, SVM and ELM. In order to handle the imbalance in class proportions in the experimental data, class imbalance modifications are made in the algorithms to enable cost-sensitive learning. Data re-sampling methods including under-sampling and over-sampling have been performed as well. Re-sampling methods are found to work well but cost-sensitive methods have a slightly better accuracy and do not modify data distributions. A modification to the ELM algorithm has been made by weighting the minority class data more to handle the imbalance in the data set. The cost-sensitive SVM classifier outperforms the other algorithms in terms of accuracy but requires a large fraction of the data to be stored for predictions, typical of non-parametric methods. ELM models result in an inferior accuracy compared to SVMs but preferred for their simplicity (less number of parameters) and potential for online learning.

The simulators developed in chapter 4 and 5 require online adaptation owing to

inaccuracies and parametric variations. In chapter 6, novel online learning algorithms using extreme learning machine models have been developed using a Lyapunov based approach and stochastic gradient based approach. The working of the algorithms are demonstrated using simple examples and benchmark data sets for classification, regression and nonlinear system identification. The algorithms have been applied to develop HCCI engine simulators online with the following conclusions. For simple HCCI behavior such as dynamics at a constant speed can be modeled with high accuracy using the developed online learning algorithms. The L-ELM and SG-ELM have a stable parameter evolution unlike the OS-ELM which runs into ill-conditioning problems. The computational requirements for L-ELM and SG-ELM are significantly less compared to OS-ELM which demonstrates its suitability in learning from large data sets and streaming data from engine sensors and with limited onboard ECU. OS-ELM, on the other hand, achieves superior convergence rates and also provides convergence indication using the covariance matrix but at the cost of possible instabilities in parameter evolution. If the online learning task is the only objective, OS-ELM can be considered as a suitable algorithm for HCCI online learning. However, if an adaptive control scheme is required as in chapter 7, where decisions are made on the fly using partially converged models, the L-ELM or SG-ELM may be a better choice.

In chapter 7, a model predictive control framework has been developed using machine learning models. The developed HCCI engine simulators using ELM models are used to simulate IMEP and CA50 for given input trajectories of FM, EVC and SOI. The MPC solves a real-time optimization problem involving objectives for tracking reference IMEP and CA50, reducing control effort and observing constraints on actuators. A fast quadratic programming has been applied for potential implementation of the developed MPC algorithm on the engine ECU. A simulation study has been conducted to demonstrate the working of the linearized MPC algorithm on a nonlinear HCCI engine simulator. Further, the online learning mechanisms developed in

chapter 6 has been used in conjunction with the MPC algorithm for online adaptation for model inaccuracies. A simplified demonstration shows that the L-ELM and SG-ELM algorithms are suited for the online adaptation against an OS-ELM algorithm which might run into ill-conditioning problems leading to instability in control.

The data based modeling framework developed in this thesis demonstrates a powerful alternative to the physics based modeling, which is the present state-of-the-art for HCCI engines and many other automotive systems. The framework can be easily adapted for modeling a different combustion engine. The nonlinear models used have a general structure and can be re-trained using a new data set with ease. The framework can be used to develop models very quickly and with good prediction capabilities which might rank this approach higher than the state-of-the-art for systems that are well understood and are in production. Further, HCCI operating envelope identification using transient data is completely novel and with further analysis and validation, can become a potential tool for commercial vehicles. The present state-of-the-art for HCCI control involve a conservative approach by using transient models parameterized using steady-state experimental data. The MPC based control demonstrated in this thesis has the potential to operate the engine over a wider operating range taking advantage of the nonlinear models. However, this has to be practically evaluated on the engine.

In comparison to the present state-of-the-art for online learning algorithm using ELM, the algorithms developed in this thesis (Lyapunov and Stochastic gradient descent methods) are superior in terms of stability of parameter evolution but inferior in terms of learning speed (parameter convergence). The application of such efficient algorithms for online parameter adaptation is novel for the HCCI engine problem. Although the developed algorithms perform well over the existing state-of-the-art algorithms for the benchmark problems and HCCI system, extensive studies are required to fully understand the pros and cons of the developed algorithms.

8.2 Some Precautions in Using Computational Learning

The approach considered in this research demonstrates the power of advanced data based models in solving complex modeling problems. However, such models rarely give any insight into the underlying physics. An inappropriate use of the approach can create more problems than benefits and is a major reason for the lack of adoption in several disciplines. Below are some precautions and qualitative observations from this work that might be useful for data based model development.

1. Pre- and post-processing of data - Data processing is important to achieve efficient models from data. This include elimination of inappropriate data, scaling and normalization to improve numerical stability of algorithms. Also, when the input data dimension is large, preprocessing methods such as principal component analysis can be used for feature selection with possible benefits in reducing model complexity and improving generalization.

2. Use of expert knowledge - Expert knowledge or domain understanding must be included whenever possible at every stage of model development. This include use of appropriate sensor information for the process to be modeled/controlled. Experiment design is a key component and expert knowledge can be used to design excitation signals for the concerned region of operation. Expert knowledge can also be used to determine the right number of signal orders and sampling rates to be used. The data can contain slow signals with slow dynamics and fast dynamics and an appropriate order selection is vital for capturing the right system behavior.

3. Appropriate model structure - A major problem in a complete black-box type modeling is to determine an appropriate model structure. Expert knowledge can be and must be used if known but during most situations, it is important to move bottom-up in terms of complexity. For instance, a simple linear model must be

the first attempt and only when the performance needs are not satisfied, one should resort to nonlinear models. General purpose nonlinear models such as the ones used in this work exist but care must be taken to identify a simple structure that ensures good generalization capability. Model complexity can be handled using cross-validation techniques and monitoring over-fitting performance.

4. Appropriate training methods - Even if an appropriate structure is known or identified, the model parameters need to be determined in an efficient manner. Typically the parameters evolve as solutions to an optimization problem. An ill-posed problem or a non-convex formulation typically leads to sub-optimal (or bizarre) behavior. Also, special cases such as the imbalance class learning discussed in chapter 5 call for modifications in the optimization problem and one needs to give attention to the design of the learning algorithm.

5. Design for generalization - An important aspect of machine learning is to develop models that can generalize well to unseen situations. A model is only trained to perform well on a representative data set of the system but expected to perform well in general. Typically, the trained models are evaluated on an unseen test data set. It should be noted that a test set is another representative of the system and there is no guarantee that the model can perform well in general. Hence, several validations need to be done to build maximum confidence in the model validity. Also, validations need to be problem specific. For instance, a model validated for one-step ahead predictions may not perform well for multi-step ahead predictions when developing a simulator. Further, application specific contingency plan is required to monitor and compensate for any undesirable behavior resulting from the model's behavior. Although this is not a focus in the present work, it could be a very good direction for the future.

6. Region of validity - It is very important to know the region of validity for the

developed model. An engine model developed from experiments at 2500 RPM may not be a good simulator for the engine at 3000 RPM. Also, care needs to be taken during model interpolations and more crucially during extrapolations. Typically, nonlinear black box models are not valid outside the region for which it is trained. Also, care must be taken when switching between different models based on operating region.

The above list is not comprehensive and problem specific concerns always arise and precautions need to be observed.

8.3 Future Work

The research presented in this thesis can be considered to be the first application of some of the machine learning algorithms to the HCCI engine modeling and controls problem. Although some interesting problems both in algorithm development and applications have been addressed in this work, there are several other opportunities to extend the work in the future.

The engine simulators developed are validated on several data sets at different operating conditions. However, a robust mechanism to determine a confidence metric for the models can be very useful in controls applications. For example, a bayesian approach or bootstrapping method can be used to develop confidence intervals. Further, application specific contingency plan might be useful to monitor and compensate for any undesirable behavior resulting from the model's behavior.

It is demonstrated in this work that engine operating boundaries can be identified for partially stable combustion systems like the HCCI engine. However, extending the algorithm to solve application related problems including developing an onboard diagnostics unit to detect engine misfires for emission regulations, using the model to design excitation signals for system identification without misfiring the engine,

implementing the model in the MPC framework to identify actuator extremes for extending the operating region of HCCI etc. can serve fruitful in automotive research. Also, more unstable modes of HCCI including different modes of misfires, knocking and ringing can be included.

The L-ELM and SG-ELM algorithms are shown to work well for the HCCI engine and several other benchmark data sets but they have been observed to converge to local excitations. A theoretical study/convergence analysis for use in nonlinear system identification to evaluate local vs global convergence of parameters can answer algorithm related questions.

The MPC solution developed in this work considers reference tracking and control effort as objectives. In future, a more comprehensive controller can be developed with sufficient availability of sensing capability to add emissions, vehicle drivability etc. in the MPC objective function. Further, several experiments need to be conducted to evaluate the MPC control framework on the experimental HCCI engine to prove its potential for adoption in real applications.

APPENDICES

APPENDIX A

Appendix

A.1 Loss functions

A plot showing the different loss functions used for classification can be shown in Fig. A.1. The 0-1 loss is the most efficient loss function for binary classification but is non-differentiable and hence not used in developing algorithms. A logistic loss is given by equation (A.1) is used in logistic regression algorithm while a hinge loss (equation (A.2)) and squared loss (equation (A.3)) are implemented in SVM and ELM algorithms respectively.

$$L_{logistic} = \log(1 + e^{-yf(x)}) \tag{A.1}$$

$$L_{hinge} = max(0, 1 - yf(x)) \tag{A.2}$$

$$L_{squared} = (1 - yf(x))^2 \tag{A.3}$$

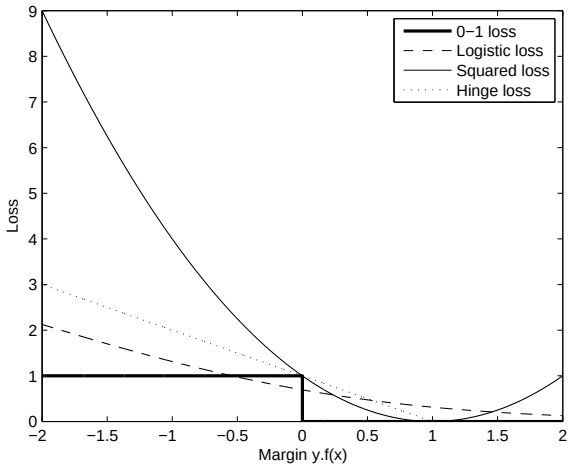


Figure A.1: A comparison of the loss functions of the algorithms used in this paper with the baseline 0-1 error standard.

A.2 Logistic Regression

Logistic regression (LR) is a classical linear classifier that proves to be effective especially for large data set problems owing to its computational efficiency. LR makes use of a logistic function given by equation (5.2) which confines the output of the function to lie between 0 and 1. Unlike linear regression model which solves a least squares problem with a squared loss function, LR solves a nonlinear optimization problem using a logistic loss function (see Figure A.1 in appendix A.1). The logistic loss function is particularly attractive for classification because the algorithm do not penalize the correctly classified points (at large positive margin in Figure A.1) as much as the squared loss.

$$\psi(x) = \frac{1}{1 + e^{-x}} \tag{A.4}$$

The conditional probability of estimating y from x can be expressed in terms of the model parameters $\beta = [\beta_0 \quad \beta_1]^T$ as

$$P(Y = y|X = x) = \frac{1}{1 + e^{-y(\beta_1^T x + \beta_0)}} \tag{A.5}$$

where X and Y represent the input and output random variables. The goal of logistic regression is to determine β such that $P(Y|X,\beta)$ is maximized using the following optimization problem

$$\beta = \arg\max_{\beta} P(Y|X,\beta) = \arg\max_{\beta} \Pi_{i=1}^{N} P(y_i|x_i,\beta) \tag{A.6}$$

Expressing the above in log likelihood form, the optimization problem becomes

$$\begin{aligned}
\beta &= \arg\max_{\beta} \log\{\Pi_{i=1}^{N} P(y_i|x_i,\beta)\} \\
&= \arg\max_{\beta} \sum_{i=1}^{N} \log P(y_i|x_i,\beta) \\
&= \arg\max_{\beta} \sum_{i=1}^{N} \log \frac{1}{1+e^{-y(\beta_1^T x+\beta_0)}} \\
&= \arg\max_{\beta} -\sum_{i=1}^{N} \log 1+e^{-y(\beta_1^T x+\beta_0)} \\
&= \arg\min_{\beta} \sum_{i=1}^{N} \log\left(1+e^{-y(\beta_1^T x+\beta_0)}\right)
\end{aligned} \tag{A.7}$$

The LR decision hypothesis is given by

$$f(x) = sgn(\beta_1^T x + \beta_0) \tag{A.8}$$

A.3 Levenberg-Marquardt back-propagation algorithm

The training data is a set of n examples $\{(u_1,y_1),(u_2,y_2),....,(u_n,y_n)\}$ obtained as time sequence from the HCCI combustion process. The objective of training is to approximate the system $f(.)$ by minimizing the cost function

$$L(w) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{y_d}(y_j^i - \hat{y}_j^i)^2 + \sum_{k=1}^{N} w_k^2 \tag{A.9}$$

where $w \in \mathbb{R}^N$ represents the network parameters (weights and biases). The first term represents the sum squared error (SSE) between system outputs and model outputs while the second term is used for regularization and λ represents regularization constant. The cost function can be expressed in vector form as

$$L(w) = \frac{1}{2}E^T E + \frac{\lambda}{2}W^T W \tag{A.10}$$

In order to derive the regularized LM algorithm, the gradient (G) of $L(w)$ with respect to W and the Jacobian (J) of E with respect to W has to be determined. The regularized LM algorithm update algorithm is given by

$$w_{i+1} = w_i - [J_i^T J_i + \mu diag(J_i^T J_i) + \lambda I]^{-1} G_i \tag{A.11}$$

$$G_i = J_i^T E_i + \lambda I w_i \tag{A.12}$$

where i represents update counter, μ the stability factor, and J is given by

$$J = \begin{pmatrix} \frac{\partial {e_1}^1}{\partial w_1} & \frac{\partial {e_1}^1}{\partial w_2} & \cdots & \frac{\partial {e_1}^1}{\partial w_N} \\ \frac{\partial {e_2}^1}{\partial w_1} & \frac{\partial {e_2}^1}{\partial w_2} & \cdots & \frac{\partial {e_2}^1}{\partial w_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial {e_{y_d}}^1}{\partial w_1} & \frac{\partial {e_{y_d}}^1}{\partial w_2} & \cdots & \frac{\partial {e_{y_d}}^1}{\partial w_N} \\ \frac{\partial {e_1}^2}{\partial w_1} & \frac{\partial {e_1}^2}{\partial w_2} & \cdots & \frac{\partial {e_1}^2}{\partial w_N} \\ \frac{\partial {e_2}^2}{\partial w_1} & \frac{\partial {e_2}^2}{\partial w_2} & \cdots & \frac{\partial {e_2}^2}{\partial w_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial {e_{y_d}}^2}{\partial w_1} & \frac{\partial {e_{y_d}}^2}{\partial w_2} & \cdots & \frac{\partial {e_{y_d}}^2}{\partial w_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial {e_{y_d}}^n}{\partial w_1} & \frac{\partial {e_{y_d}}^n}{\partial w_2} & \cdots & \frac{\partial {e_{y_d}}^n}{\partial w_N} \end{pmatrix} \in \mathbb{R}^{ny_d x N} \tag{A.13}$$

When μ aproaches zero, the Gauss-Newton algorithm is obtained while a large

μ gives the gradient descent method. Hence when the error goes down following an update, it indicates that the quadratic approximation on $L(w)$ is right and μ is reduced to perform Gauss-Newton type update while if the error goes up, μ is increased so that the search is done along reducing gradient with small step size. The regularization constant λ can be tuned using a hold-out training set to avoid over-fitting.

The network trained using the above procedure represents a series-parallel structure [22]. The advantage of using a series-parallel structure for training is that the system's true output is used for training compared to the parallel structure where the model estimate is used during training [22]. This prevents the inaccuracies of the model to be fed back during training making it efficient. However during testing, the series-parallel structure is converted to a parallel structure (Fig. 2.6) by feeding back the model outputs as initial conditions for the next prediction. Now the model can predict the output based on the provided inputs and the initial conditions of delay units. Hence even before the inputs are given to the actual system, the model can be used to check whether the future input trajectory is optimal and/or if the inputs cause instabilities for a system like HCCI.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] R. Thring, "Homogeneous Charge Compression Ignition engines," 1989, SAE paper 892068.

[2] M. Christensen, P. Einewall, and B. Johansson, "Homogeneous charge compression ignition using iso-octane, ethanol and natural gas- a comparison to spark ignition operation," in *International Fuels & Lubricants Meeting & Exposition*, Tulsa, OK, USA, Oct 1997, SAE paper 972874.

[3] T. Aoyama, Y. Hattori, J. Mizuta, and Y. Sato, "An experimental study on premixed-charge compression ignition gasoline engine," in *International Congress & Exposition*, Detroit, MI, USA, Feb 1996, SAE paper 960081.

[4] K. Epping, S. Aceves, R. Bechtold, and J. Dec, "The potential of HCCI combustion for high efficiency and low emissions," in *SAE Powertrain & Fluid Systems Conference & Exhibition*, ser. SAE Technical Paper 2002-01-1923, San Diego, CA, Oct 2002.

[5] G. H. Abd-Alla, "Using exhaust gas recirculation in internal combustion engines: a review," *Energy Conversion and Management*, vol. 43, no. 8, pp. 1027–1042, 2002.

[6] C. Chiang and C. Chen, "Constrained control of homogeneous charge compression ignition (HCCI) engines," in *5th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2010.

[7] J. Bengtsson, P. Strandh, R. Johansson, P. Tunestal, and B. Johansson, "Model predictive control of homogeneous charge compression ignition (HCCI) engine dynamics," in *2006 IEEE International Conference on Control Applications*, 2006.

[8] N. Ravi, M. J. Roelle, H. H. Liao, A. F. Jungkunz, C. F. Chang, S. Park, , and J. C. Gerdes, "Model-based control of HCCI engines using exhaust recompression," in *IEEE Transactions on Control Systems Technology*, 2009.

[9] L. Re, F. Allgöwer, L. Glielmo, C. Guardiola, and I. Kolmanovsky, *Automotive Model Predictive Control: Models, Methods and Applications*, ser. Lecture Notes in Control and Information Sciences. Springer, 2010.

[10] Chak, C. Kwong, G. Feng, and J. Ma, "On the approximation capability of neural networks - dynamic system modeling and control," *Asian Journal of Control*, pp. 122–130, 2001.

[11] M. Beham and D. L. Yu, "Modeling a variable valve timing spark ignition engine using different neural networks," *Journal of Automobile Engineering*, vol. 218, pp. 1159–1171, 2004.

[12] Y. Tan and M. Saif, "Nonlinear dynamic modeling of automotive engines using neural networks," in *IEEE International Conference on Control Applications*, 1997, pp. 408–410.

[13] L. Biao, L. Qing-chun, J. Zhen-hua, and N. Sheng-fang, "System identification of locomotive diesel engines with autoregressive neural network," in *4th IEEE Conference on Industrial Electronics and Applications*, May 2009, pp. 3417–3421.

[14] I. Arsie, C. Pianese, and M. Sorrentino, "Development of recurrent neural networks for virtual sensing of nox emissions in internal combustion engines," *SAE International Journal of Fuels and Lubrication*, vol. 2, no. 2, pp. 354–361, 2010.

[15] I. Arsie, S. D. Iorio, C. Pianese, G. Rizzo, and M. Sorrentino, "Recurrent neural networks for air-fuel ratio estimation and control in spark-ignited engines," in *IFAC World Congress*, Elsevier, Ed., Jul 2008.

[16] D. V. Prokhorov, "2008 special issue: Toyota prius hev neurocontrol and diagnostics," *Neural Netw.*, vol. 21, no. 2-3, pp. 458–465, Mar. 2008.

[17] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[18] D. Prokhorov, "Neural networks in automotive applications," in *Computational Intelligence in Automotive Applications*, ser. Studies in Computational Intelligence, D. Prokhorov, Ed. Springer Berlin Heidelberg, 2008, vol. 132, pp. 101–123.

[19] R. Johri, A. Salvi, and Z. Filipi, "Real-Time Transient Soot and NOx Virtual Sensors for Diesel Engineusing Neuro-Fuzzy Model Tree and Orthogonal Least Squares," *Journal of Engineering for Gas Turbines and Power*, 2012.

[20] V. Vapnik, *The Nature of Statistical Learning Theory.* New York: Springer, 1995.

[21] O. Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models.* Springer, 2001.

[22] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," vol. 1, no. 1, pp. 4–27, Mar. 1990.

[23] S. Haykin, *Neural Networks and Learning Machines*, ser. Neural networks and learning machines. Pearson Prentice Hall, 2009, no. v. 10.

[24] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[25] V. Kůrková, "Kolmogorov's theorem and multilayer neural networks," *Neural Netw.*, vol. 5, no. 3, pp. 501–506, Mar. 1992.

[26] A. Ranganathan, *The Levenberg-Marquardt Algorithm*, 2004.

[27] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, Nov. 1994.

[28] V. M. Janakiraman, X. Nguyen, and D. Assanis, "Nonlinear identification of a gasoline HCCI engine using neural networks coupled with principal component analysis," *Applied Soft Computing*, vol. 13, no. 5, pp. 2375 – 2389, 2013.

[29] R. S. Sexton, R. E. Dorsey, and J. D. Johnson, "Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing," *European Journal of Operational Research*, vol. 114, no. 3, pp. 589 – 601, 1999.

[30] N. Barnes, A. O'Neill, and D. Wood, "Rapid, supervised training of a two-layer, opto-electronic neural network using simulated annealing," *Optics Communications*, vol. 87, no. 56, pp. 203 – 206, 1992.

[31] B. Cohen, D. Saad, and E. Marom, "Efficient training of recurrent neural network with time delays," *Neural Networks*, vol. 10, no. 1, pp. 51 – 59, 1997.

[32] A. Blanco, M. Delgado, and M. Pegalajar, "A real-coded genetic algorithm for training recurrent neural networks," *Neural Networks*, vol. 14, no. 1, pp. 93 – 105, 2001.

[33] R. S. Sexton and J. N. Gupta, "Comparative evaluation of genetic algorithm and backpropagation for training neural networks," *Information Sciences*, vol. 129, no. 14, pp. 45 – 59, 2000.

[34] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Comput.*, vol. 3, no. 2, pp. 246–257, Jun. 1991.

[35] E. Hartman, J. D. Keeler, and J. M. Kowalski, "Layered neural networks with gaussian hidden units as universal approximations," *Neural Comput.*, vol. 2, no. 2, pp. 210–215, Apr. 1990.

[36] T. Chen and R. Chen, "Approximation capability to functions of several variables, nonlinear functionals and operators by radial basis function neural networks," *IEEE Transactions on Neural Networks*, 1995.

[37] A. D. Niros and G. E. Tsekouras, "A novel training algorithm for rbf neural network using a hybrid fuzzy clustering approach," *Fuzzy Sets and Systems*, vol. 193, no. 0, pp. 62 – 84, 2012.

[38] A. Alexandridis, H. Sarimveis, and G. Bafas, "A new algorithm for online structure and parameter adaptation of rbf networks," *Neural Networks*, vol. 16, no. 7, pp. 1003 – 1017, 2003.

[39] A. Staiano, R. Tagliaferri, and W. Pedrycz, "Improving rbf networks performance in regression tasks by means of a supervised fuzzy clustering," *Neurocomputing*, vol. 69, no. 1315, pp. 1570 – 1581, 2006.

[40] J. Gomm and D. Yu, "Selecting radial basis function network centers with recursive orthogonal least squares training," *Neural Networks, IEEE Transactions on*, vol. 11, no. 2, pp. 306 –314, mar 2000.

[41] D.-S. Huang and W.-B. Zhao, "Determining the centers of radial basis probabilistic neural networks by recursive orthogonal least square algorithms," *Applied Mathematics and Computation*, vol. 162, no. 1, pp. 461 – 473, 2005.

[42] V. Vapnik, *The Nature of Statitical Learning Theory*. Springer-Verlag GmbH, 1995.

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

[44] K. Veropoulos, C. Campbell, and N. Cristianini, "Controlling the sensitivity of support vector machines," in *Proceedings of the International Joint Conference on AI*, 1999, pp. 55–60.

[45] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 9*. MIT Press, 1997, pp. 155–161.

[46] B. Schlkopf, P. Bartlett, A. Smola, and R. Williamson, "Support vector regression with automatic accuracy control," in *Proceedings of ICANN'98, Perspectives in Neural Computing*, 1998.

[47] A. J. Smola and B. Schlkopf, "A tutorial on support vector regression," Statistics and Computing, Tech. Rep., 2003.

[48] B. Schlkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," in *Neural Computation*. MIT Press, 2000.

[49] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, 2006.

[50] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification." *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 42, no. 2, pp. 513–529, 2012.

[51] J. Heywood, *Internal combustion engine fundamentals*, ser. McGraw-Hill series in mechanical engineering. McGraw-Hill, 1988.

[52] F. Zhao, T. N. Asmus, D. N. Assanis, J. E. Dec, J. A. Eng, and P. M. Najt, *Homogeneous Charge Compression Ignition (HCCI) Engines*. SAE International, March 2003.

[53] J. Martinez-Frias, S. M. Aceves, D. Flowers, J. R. Smith, and R. Dibble, "HCCI engine control by thermal management." SAE International, 10 2000.

[54] G. Haraldsson, P. Tunestl, B. Johansson, and J. Hyvnen, "HCCI combustion phasing in a multi cylinder engine using variable compression ratio." SAE International, 10 2002.

[55] M. Y. Au, J. W. Girard, R. Dibble, D. Flowers, S. M. Aceves, J. Martinez-Frias, R. Smith, C. Seibel, and U. Maas, "1.9-liter four-cylinder HCCI engine operation with exhaust gas recirculation," 05 2001.

[56] M. Yao, Z. Zheng, and H. Liu, "Progress and recent trends in homogeneous charge compression ignition (HCCI) engines," *Progress in Energy and Combustion Science*, vol. 35, no. 5, pp. 398 – 437, 2009.

[57] X. Lu, W. Chen, and Z. Huang, "A fundamental study on the control of the HCCI combustion and emissions by fuel design concept combined with controllable EGR. part 2. effect of operating conditions and EGR on HCCI combustion," *Fuel*, vol. 84, no. 9, pp. 1084–1092, 2005.

[58] J. E. Dec and M. Sjberg, "Isolating the effects of fuel chemistry on combustion phasing in an HCCI engine and the potential of fuel stratification for ignition control," 03 2004.

[59] G. M. Shaver, J. C. Gerdes, and M. J. Roelle, "Physics-based modeling and control of residual-affected HCCI engines," *Journal of Dynamic Systems, Measurement, and Control*, vol. 131, no. 2, p. 021002, 2009.

[60] C. Chiang and A. Stefanopoulou, "Dynamics of homogeneous charge compression ignition (HCCI) engines with high dilution," in *American Control Conference, 2007. ACC '07*, july 2007, pp. 2979 –2984.

[61] G. M. Shaver, "Stability analysis of residual-affected HCCI using convex optimization," *Control Engineering Practice*, vol. 17, no. 12, pp. 1454 – 1460, 2009.

[62] M. C. Weikl, F. Beyrau, and A. Leipertz, "Simultaneous temperature and exhaust-gas recirculation-measurements in a homogeneous charge-compression ignition engine by use of pure rotational coherent anti-stokes raman spectroscopy," *Appl. Opt.*, vol. 45, no. 15, pp. 3646–3651, May 2006.

[63] Y. Urata, M. Awasaka, J. Takanashi, T. Kakinuma, T. Hakozaki, and A. Umemoto, "A study of gasoline-fuelled HCCI engine equipped with an electromagnetic valve train," 06 2004.

[64] R. Scaringe, C. Wildman, and W. K. Cheng, "On the high load limit of boosted gasoline HCCI engine operating in NVO mode," *SAE Int. J. Engines*, vol. 3, pp. 35–45, 04 2010.

[65] M. M. Andreae, W. K. Cheng, T. Kenney, and J. Yang, "On HCCI engine knock," 07 2007.

[66] T. Johansson, B. Johansson, P. Tunestal, and H. Aulin, "HCCI operating range in a turbo-charged multi cylinder engine with vvt and spray-guided di," *SAE*, vol. 2009-01-04, no. 2009-01-0494, 2009.

[67] J. Hyvnen, G. Haraldsson, and B. Johansson, "Supercharging HCCI to extend the operating range in a multi-cylinder VCR-HCCI engine," 10 2003.

[68] K. Godfrey, *Perturbation signals for system identification*, ser. Prentice Hall international series in acoustics, speech, and signal processing. Prentice Hall, 1993.

[69] V. Janakiraman, J. Sterniak, and D. Assanis, "Support vector machines for identification of HCCI combustion dynamics," in *9th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, July 2012.

[70] V. M. Janakiraman, X. Nguyen, and D. Assanis, "A system identification framework for modeling complex combustion dynamics using support vector machines," *Lecture Notes in Electrical Engineering - Informatics in Control, Automation and Robotics*, 2012.

[71] S. Jade, E. Hellström, L. Jiang, and A. G. Stefanopoulou, "Fuel governor augmented control of recompression HCCI combustion during large load transients," in *Proc. American Control Conference*, 2012, pp. 2072–2077.

[72] N. Ravi, M. J. Roelle, H. H. Liao, A. F. Jungkunz, C. F. Chang, S. Park, and J. C. Gerdes, "Model-based control of HCCI engines using exhaust recompression," in *IEEE Transactions on Control Systems Technology*, 2009.

[73] K. Chang, A. Babajimopoulos, G. A. Lavoie, Z. S. Filipi, and D. N. Assanis, "Analysis of load and speed transitions in an HCCI engine using 1-d cycle simulation and thermal networks." SAE International, 04 2006.

[74] Y. Wang, S. Makkapati, M. Jankovic, M. Zubeck, and D. Lee, "Control oriented model and dynamometer testing for a single-cylinder, heated-air HCCI engine." SAE International, 04 2009.

[75] C. Chiang, A. G. Stefanopoulou, and M. Jankovic, "Nonlinear observer-based control of load transitions in homogeneous charge compression ignition engines," *Control Systems Technology, IEEE Transactions on*, vol. 15, no. 3, pp. 438–448, 2007.

[76] Y. Urata, M. Awasaka, J. Takanashi, T. Kakinuma, T. Hakozaki, and A. Umemoto, "A study of gasoline-fuelled HCCI engine equipped with an electromagnetic valve train," 06 2004.

[77] R. Scaringe, C. Wildman, and W. K. Cheng, "On the high load limit of boosted gasoline HCCI engine operating in NVO mode," *SAE Int. J. Engines*, vol. 3, pp. 35–45, 04 2010.

[78] A. Soliman, G. Rizzoni, and V. Krishnaswami, "The effect of engine misfire on exhaust emission levels in spark ignition engines." SAE International, 02 1995.

[79] S.-C. Kong, "A study of natural gas/DME combustion in HCCI engines using CFD with detailed chemical kinetics," *Fuel*, vol. 86, no. 1011, pp. 1483 – 1489, 2007.

[80] Z. Zheng and M. Yao, "Charge stratification to control HCCI: Experiments and CFD modeling with n-heptane as fuel," *Fuel*, vol. 88, no. 2, pp. 354 – 365, 2009.

[81] Z. Wang, S. Shuai, J. Wang, and G. Tian, "A computational study of direct injection gasoline HCCI engine with secondary injection," *Fuel*, vol. 85, no. 1213, pp. 1831 – 1841, 2006.

[82] I. V. Kolmanovsky and E. G. Gilbert, "Support vector machine-based determination of gasoline direct injected engine admissible operating envelope." SAE International, 03 2002.

[83] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. on Knowl. and Data Eng.*, vol. 21, no. 9, pp. 1263–1284, 2009.

[84] A. Dobson, *An Introduction to Generalized Linear Models, Second Edition*, ser. Texts in Statistical Science Series. Taylor & Francis, 2010.

[85] G. A. Lavoie, J. Martz, M. Wooldridge, and D. Assanis, "A multi-mode combustion diagram for spark assisted compression ignition," *Combustion and Flame*, vol. 157, no. 6, pp. 1106 – 1110, 2010.

[86] G. T. Kalghatgi and R. A. Head, "Combustion limits and e?ciency in a homogeneous charge compression ignition engine," *Int. J. Engine Res*, vol. 7, pp. 215–236, 2006.

[87] M. Shahbakhti and C. R. Koch, "Characterizing the cyclic variability of ignition timing in a homogenous charge compression ignition engine fueled with n-heptane/iso-octane blend fuels," *Int. J. Engine Res*, vol. 9, pp. 361–397, 2008.

[88] K.-A. Toh, "Deterministic neural classification," *Neural Comput.*, vol. 20, no. 6, pp. 1565–1595, Jun. 2008.

[89] Y. LeCun, L. Bottou, G. Orr, and K.-R. Mller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*, ser. Lecture Notes in Computer Science, G. Orr and K.-R. Mller, Eds. Springer Berlin Heidelberg, 1998, vol. 1524, pp. 9–50.

[90] L. Ngia, J. Sjoberg, and M. Viberg, "Adaptive neural nets filter using a recursive levenberg-marquardt search direction," in *Signals, Systems amp; Computers, 1998. Conference Record of the Thirty-Second Asilomar Conference on*, vol. 1, 1998, pp. 697–701 vol.1.

[91] J. Platt, "A resource-allocating network for function interpolation," *Neural Comput.*, vol. 3, no. 2, pp. 213–225, Jun. 1991.

[92] G. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning rbf (gap-rbf) networks," *Trans. Sys. Man Cyber. Part B*, vol. 34, no. 6, pp. 2284–2292, Dec. 2004.

[93] ——, "A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation," *IEEE Transactions on Neural Networks*, vol. 16, pp. 57–67, 2005.

[94] N. Liang, G. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *Neural Networks, IEEE Transactions on*, vol. 17, no. 6, pp. 1411–1423, 2006.

[95] G. Zhao, Z. Shen, C. Miao, and Z. Man, "On improving the conditioning of extreme learning machine: A linear case," in *Information, Communications and Signal Processing, 2009. ICICS 2009. 7th International Conference on*, dec. 2009, pp. 1 –5.

[96] F. Han, H.-F. Yao, and Q.-H. Ling, "An improved extreme learning machine based on particle swarm optimization," in *Bio-Inspired Computing and Applications*, ser. Lecture Notes in Computer Science.

[97] M. T. Hoang, H. Huynh, N. Vo, and Y. Won, "A robust online sequential extreme learning machine," in *Advances in Neural Networks*, ser. Lecture Notes in Computer Science.

[98] H. T. Huynh and Y. Won, "Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks," *Pattern Recognition Letters*, vol. 32, no. 14, pp. 1930 – 1935, 2011.

[99] V. Akpan and G. Hassapis, "Adaptive predictive control using recurrent neural network identification," in *Control and Automation, 2009. MED '09. 17th Mediterranean Conference on*, june 2009, pp. 61 –66.

[100] S. Lyashevskiy and L. Abel, "Nonlinear systems identification using the lyapunov method," *System identification (SYSID'94) : a postprint volume from the IFAC symposium, Copenhagen, Denmark, 4-6 July 1994*, vol. 1, July 1994.

[101] K. Ross, *Elementary Analysis: The Theory of Calculus*, ser. Undergraduate Texts in Mathematics. Springer, 1980.

[102] H. Khalil, *Nonlinear Systems*. Prentice Hall, 2002.

[103] P. Ioannou and J. Sun, *Robust adaptive control*.

[104] J. Spooner, M. Maggiore, R. Ordóñez, and K. Passino, *Stable Adaptive Control and Estimation for Nonlinear Systems: Neural and Fuzzy Approximator Techniques*, ser. Adaptive and Learning Systems for Signal Processing, Communications and Control Series. Wiley, 2004.

[105] R. Nowak and B. Van Veen, "Nonlinear system identification with pseudorandom multilevel excitation sequences," in *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, vol. 4, april 1993, pp. 456 –459 vol.4.

[106] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Y. Lechevallier and G. Saporta, Eds. Physica-Verlag HD, 2010, pp. 177–186.

[107] N. Le Roux, M. Schmidt, and F. Bach, "A Stochastic Gradient Method with an Exponential Convergence Rate for Strongly-Convex Optimization with Finite Training Sets," INRIA, Tech. Rep. arXiv:1202.6258v1, 2012.

[108] S. Shalev-Shwartz and N. Srebro, "Svm optimization: inverse dependence on training set size," in *Proceedings of the 25th international conference on Machine learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 928–935.

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

[110] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2011.

[111] M. M. Andreae, "Effect of ambient conditions and fuel properties on homogeneous charge compression ignition engine operation," in *PhD Dissertation*. Massachusetts Institute of Technology. Dept. of Mechanical Engineering., 2006.

[112] J. MacIejowski, *Predictive Control: With Constraint*, ser. Pearson Education. Prentice Hall, 2002.

[113] M. Lawrynczuk, "Neural networks in model predictive control," in *Intelligent Systems for Knowledge Management*, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2009, vol. 252, pp. 31–63.

[114] S. Effati and M. Ranjbar, "A novel recurrent nonlinear neural network for solving quadratic programming problems," *Applied Mathematical Modelling*, vol. 35, no. 4, pp. 1688 – 1695, 2011.