The Pennsylvania State University
The Graduate School

DATA-DRIVEN MODELING AND PATTERN RECOGNITION OF
DYNAMICAL SYSTEMS

A Dissertation in
Mechanical Engineering
by
Pritthi Chattopadhyay

© 2018 Pritthi Chattopadhyay

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2018
The dissertation of Pritthi Chattopadhyay was reviewed and approved* by the following:

Asok Ray  
Distinguished Professor of Mechanical Engineering and Mathematics  
Dissertation Advisor, Chair of Committee

Christopher D. Rahn  
Associate Dean for Innovation, College of Engineering  
Professor of Mechanical Engineering

Minghui Zhu  
Dorothy Quiggle Assistant Professor of Electrical Engineering

Shashi Phoha  
Professor of Electrical Engineering  
Research Professor of Applied Research Laboratory

Mary Frecker  
Professor of Mechanical Engineering  
Associate Head for Graduate Programs of Mechanical Engineering

*Signatures are on file in the Graduate School.
Human-engineered complex systems need to be monitored consistently to ensure their safety and efficiency, which might be affected due to degradation over time or unanticipated disturbances. For systems that change at a fast time scale, instead of active health monitoring, preventative system design is more feasible and effective. Both active health monitoring and preventative system design can be done using physics-based or data-driven models. In comparison to physics-based models, data-driven models do not require knowledge of the underlying system dynamics; they determine the relation between the relevant input and output variables from a training data set. This is useful when there is lack of understanding of the system dynamics or the developed models are inadequate. One such scenario is combustion, where the difficulties include nonlinear dynamics involving several input parameters; existence of bifurcations in the dynamic behavior and extremely high sensitivity of the combustor behavior to even small changes in some of the design parameters. Similarly, for batteries, sufficient knowledge of the electrochemical characteristics is necessary to develop models for parameter identification at different operating points of the nonlinear battery dynamics. This dissertation develops dynamic data-driven models for combustor design and battery health monitoring, using concepts of machine learning and statistics, which do not require much knowledge of the underlying system dynamics.

But the performance of a data-driven algorithm depends on many factors namely:

1. Availability of training data which covers all events of interest. For applications involving time series data, each individual time series must also be sufficiently long, to encompass the dynamics of the underlying system for each event.

2. The quality of extracted features, i.e. whether they capture all the information about the system.
3. The relation between the relevant input and output variables remaining constant during the time the algorithm is being trained.

Hence, the second part of the dissertation develops an unsupervised algorithm for scenarios where condition (iii) might not hold; quantifies the effect of the non-conformity of condition (i) on the performance of an algorithm and proposes a feature extraction algorithm to ensure conformity of condition (ii).
# Table of Contents

List of Figures xi
List of Tables xi
Acknowledgments xii

Chapter 1
Introduction 1
1.1 Motivation 1
1.2 Data-driven vs Physics-based Modeling 2
   1.2.1 Machine Learning 3
   1.2.2 Dynamic data-driven Method of Combustor Design 4
   1.2.3 Dynamic data-driven Method of Battery State Estimation 6
1.3 Factors affecting performance of a machine learning algorithm 8
   1.3.1 Condition 1 - Quantification of the effect of length of data
   on performance of the algorithm 9
   1.3.2 Condition 2 - Alphabet selection for feature extraction 9
   1.3.3 Condition 3 - Unsupervised Context Extraction 11
1.4 Contributions of the Dissertation 12
1.5 Organization of the Dissertation 14

Chapter 2
Mathematical Preliminaries 15
2.1 Symbolic Time Series Analysis (STSA) 15
   2.1.1 Symbolization of Time Series 15
   2.1.2 Construction of PFSA from Symbol String 16
      2.1.2.1 D-Markov Modelling 17
2.2 Summary .................................................. 19

Chapter 3
Graph Theoretic Unsupervised Context Extraction 21
3.1 Introduction ................................................. 21
3.2 Problem Formulation for Unsupervised Context Extraction .... 22
  3.2.1 Graph Creation ........................................ 22
  3.2.2 Community Detection ................................. 23
    3.2.2.1 Modularity ..................................... 23
    3.2.2.2 Fast Community Detection Using Modularity ... 24
3.3 Experimental Validation .................................... 24
  3.3.1 Data Acquisition ..................................... 24
  3.3.2 Data preprocessing and Feature extraction ............. 25
  3.3.3 Results and Discussion .............................. 25
3.4 Problem Formulation for evaluation of robustness of feature extraction 28
  3.4.1 Approach ............................................. 29
  3.4.2 Feature Extraction ................................... 31
    3.4.2.1 Cepstrum ....................................... 31
    3.4.2.2 Principal Component Analysis (PCA) .......... 31
    3.4.2.3 Symbolic Time Series Analysis ............... 31
  3.4.3 Classification of Extracted features .................. 32
    3.4.3.1 k-nearest neighbor (k-NN) ...................... 32
    3.4.3.2 Support Vector Machine (SVM) ................. 32
    3.4.3.3 Sparse Representation Classification (SRC) .. 32
3.5 Results of Field data Analysis ................................ 33
  3.5.1 Methodology of Data Analysis ......................... 34
  3.5.2 Results and Discussion .............................. 36
3.6 Summary and Conclusions .................................. 37

Chapter 4
Information Theoretic Approach for Alphabet Size Selection
  for Symbolic Time Series Analysis 38
4.1 Introduction ............................................. 38
4.2 Problem Formulation ..................................... 39
  4.2.1 Parzen Window based Mutual Information Estimation ... 43
  4.2.2 Robustness ........................................ 44
4.3 Experimental Validation and Results ........................ 45
4.4 Summary, Conclusions & Future Work ....................... 53
## Chapter 5

**Online Estimation of State of Charge of Battery**

5.1 Introduction ........................................... 54
5.2 Definitions ............................................. 56
5.3 Wavelet-based Time Series Segmentation ............... 57
5.4 Maximum Entropy Partitioning and Symbolization ....... 59
5.5 Online Pattern Classification ............................ 62
5.6 Experimental Validation ................................ 67
  5.6.1 Data Acquisition and Processing .................. 67
  5.6.2 Results and Discussion ............................ 70
  5.6.3 Computational costs ............................... 74
5.7 Summary, Conclusions, & Future Work ................. 76

## Chapter 6

**Data-driven Design of Combustors for Thermoacoustically stable operations**

6.1 Introduction ........................................... 78
6.2 Description of the Experimental Apparatus .............. 79
6.3 Description of the proposed design algorithm ........... 80
  6.3.1 Gaussian Process Regression ...................... 81
  6.3.2 Development of the Design Methodology .......... 84
  6.3.3 Stability Prediction using $P_{rms}$ as a feature ... 85
    6.3.3.1 Results and Discussion ......................... 87
  6.3.4 Stability Prediction using PFSA as feature ........ 88
    6.3.4.1 Results and Discussion ......................... 90
6.4 Summary, Conclusions & Future Work ................... 97

## Chapter 7

**Dynamic Data-Driven Combustor Design**

7.1 Mathematical Preliminaries ........................... 100
  7.1.1 Entropy rate ..................................... 100
  7.1.2 D-Markov Machines .............................. 101
    7.1.2.1 State Splitting in D-Markov Machines ........ 101
    7.1.2.2 State Merging in D-Markov Machines ........... 102
  7.2 Technical Approach ................................ 105
  7.3 Results and Discussion ............................. 107
    7.3.1 Comparison between D=1 and $D \geq 1$ features .. 108
  7.4 Summary, Conclusions, and Future Work ............... 112
Chapter 8  
**Summary & Future Work**  
8.1 Contributions of the Dissertation .................................................. 115  
8.2 Future Work .................................................................................. 118  

Bibliography ......................................................................................... 119
List of Figures

3.1 Soil types ................................................................. 26
3.2 Modularity score .......................................................... 27
3.3 ROC Plots for Combinations of Classifiers and Feature Extractors . 35

4.1 Information-theoretic framework for time-series partitioning and alphabet size selection ................................................. 42
4.2 Mutual information as a function of alphabet size $|\Sigma|$ for two-class Duffing system with (a) $D = 1$, (b) $D = 2$; for (c) four-class Duffing system with $D = 1$ and $\eta_{stop} = 0.01$ ................................................. 46
4.3 Misclassification error for two-class problem ($D = 1$) with variance fraction: (a) 0.67, (b) 0.5, (c) 0.25 .............................................. 47
4.4 Top: Mutual information as a function of alphabet size $|\Sigma|$ for Port 3 and Port 5 levels of premixing with (a) $D = 1$ and (b) $D = 2$
Bottom: Variation of classification error as a function of alphabet size $|\Sigma|$ for Port 3 and Port 5 levels of premixing level with (c) $D = 1$ and (d) $D = 2$ ................................................. 50
4.5 Comparison of classification error with varying alphabet size for (Port 5) premixing: (a) $D = 1$, (b) $D = 2$ .............................................. 51

5.1 Construction of finite state automata (FSA) from time series pair . 61
5.2 The flow chart for the proposed method of feature extraction ...... 62
5.3 Flow chart of the proposed feature extraction algorithm .......... 65
5.4 Flow chart of the proposed method training and testing phases ... 66
5.5 Profile of input current data for the battery experiment .......... 68
5.6 An example of pre-processing for synchronized input current and output voltage data. .............................................. 69
5.7 Normalized input-output pairs at different SOC .................... 71
5.8 Examples of symbol strings for four different types of partitioning; $|\Sigma|_{in} = |\Sigma|_{out} = 4$ and $|\Sigma|_{magnitude} = |\Sigma|_{phase} = 4$ ............... 71
5.9 Profiles of classification accuracy at different values of SOC .... 72
5.10 Cross validation with different output alphabet size $|\Sigma|$ ........ 72
5.11 Cross validation with different types of partitioning ........................................ 72
5.12 Cross validation with different lengths of training data .................................. 73
5.13 Cross validation at different aging stages (i.e., different SOH) ......................... 73
5.14 Cross validation with different number of SOC classes .................................. 73

6.1 Schematic diagram of the combustion apparatus ................................................. 80
6.2 Examples of pressure signals in the combustor .................................................. 81
6.3 ROC curve as a function of the decision threshold ............................................ 86
6.4 $P_{rms}$ predicted by GP regression algorithm: 95% confidence region ............... 87
6.5 True and predicted stability maps ................................................................. 88
6.6 Effects of time series length on $P_{rms}$ profiles ............................................. 89
6.7 Flowchart of the combustor design algorithm using PFSA as a feature ............... 91
6.8 Feature divergence $F_{div}$ predicted by GP regression algorithm for
  inlet velocity=40m/sec and $\phi=0.55$ .......................................................... 94
6.9 mean error in prediction over 20 runs using 8 sec time series ......................... 95
6.10 mean uncertainty in prediction over 20 runs using 8 sec time series ............... 96
6.11 mean error in prediction over 20 runs using 2 sec time series ....................... 97
6.12 mean uncertainty in prediction over 20 runs using 2 sec time series ............... 98
6.13 Sensitivity comparison of $PFSA$ and $P_{rms}$ features .................................. 98

7.1 Mean error in prediction over 20 runs ............................................................ 108
7.2 Mean uncertainty in estimation over 20 runs .................................................. 109
7.3 Standard deviation of mean error over 20 runs ............................................. 110
7.4 Standard deviation of uncertainty over 20 runs ............................................ 110
7.5 Profile of $P_{rms}$ plot for a typical subset ...................................................... 111
7.6 Entropy rate for the same subset as in Fig. 7.5 ............................................. 112
7.7 Correlations of $P_{rms}$ divergence with $D = 1$ and $D \geq 1$ feature
divergences for $|\Sigma| = 3, N_{max} = 20, \eta = 0.02$ ........................................ 113
7.8 Correlations of $P_{rms}$ divergence with $D = 1$ and $D \geq 1$ feature
divergences for $|\Sigma| = 5, N_{max} = 20, \eta = 0.02$ ........................................ 114
## List of Tables

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Table Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Experimental Results: Misclassification % in the three cases</td>
<td>28</td>
</tr>
<tr>
<td>3.2</td>
<td>Classification Performance, Performance Loss, and Robustness</td>
<td>36</td>
</tr>
<tr>
<td>5.1</td>
<td>Experiments at different SOH stage and SOC range</td>
<td>68</td>
</tr>
<tr>
<td>5.2</td>
<td>SOC range for different classes</td>
<td>68</td>
</tr>
<tr>
<td>5.3</td>
<td>Execution time for test data with a length of 1000 under different conditions</td>
<td>75</td>
</tr>
<tr>
<td>5.4</td>
<td>Average SOC resolution for different class assignments</td>
<td>77</td>
</tr>
<tr>
<td>6.1</td>
<td>Confusion Matrix for the decision threshold ( \Theta = 0.66 )</td>
<td>86</td>
</tr>
<tr>
<td>6.2</td>
<td>Classification accuracy for ( D )-Markov Machines (( D = 1 )) using SVM classifiers for different data lengths</td>
<td>90</td>
</tr>
<tr>
<td>6.3</td>
<td>Mean error for a single run on four different window lengths (sampled at 8192 Hz) of time series at inlet velocity=40m/sec and ( \phi=0.55 )</td>
<td>94</td>
</tr>
</tbody>
</table>
Acknowledgments

This work would never have been possible without the guidance of Prof. Asok Ray, my thesis advisor. I would also like to thank Prof Shashi Phoha, Prof. Chris Rahn and Prof Minghui Zhu, for their valuable inputs and for being in my dissertation committee. I would also like to thank Dr. Thyagaraju Damarla, Dr. Achintya Mukhopadhyay and Dr. Brian Smith for guiding me. I would like to thank all the members of my lab- Soumalya Sarkar, Yue Li, Devesh Kr. Jha, Nurali Virani, Sudeepa Mondal, Chandrachur Bhattacharya, Sihan Xiong, Michael Hauser, Najah Jasim and Yiweu Fu, for their enlightening insights. This work would never have been possible without the unwavering support of my parents. I would also like to thank all my teachers in school and college.

My work has been supported in part by the U.S. Air Force Office of Scientific Research (AFOSR) under Grant No. FA9550-15-1-0400 and by U.S. Army Research Office (ARO) under Grant No. W911NF-13-1-0461. The findings and conclusions of this dissertation do not necessarily reflect the view of the funding agencies.
Chapter 1

Introduction

1.1 Motivation

Human-engineered complex systems need to be monitored consistently, to ensure their performance (e.g., safety and efficiency). This is because safety or efficiency of such systems might be affected due to multiple factors such as: (i) degradation of the system components over time, and (ii) occurrence of unanticipated environmental disturbances. For systems, where safety is a key issue, health monitoring involves fault diagnosis & prognosis and their subsequent mitigation; for example, if an anomalous behavior is detected, steps are taken to bring back the system to safe operating zones. For systems, where efficiency is an issue, health monitoring involves detection (e.g., when the system performance falls below a specified threshold), prognosis (e.g., identification of the remaining life of the system component) and determination of subsequent corrective action(s) (replacement/repair of pertinent components) to be taken. An example is maintenance of aircraft turbine engines, where a common cause of degradation is corrosion in the hot sections, which may result in degradation of thermodynamic efficiency. Another example is battery systems. Many applications like plug-in electric vehicles and hybrid locomotives require large battery packs which contain several hundreds of battery cells to meet the large and dynamic power demands of the applications. Knowledge of the battery health enhances preventive maintenance and reduction of life cycle cost through timely recharging and/or replacement of battery cells. However, active health monitoring may not be feasible in systems that change
at a fast time scale. In such a situation, once instability has set in, it may not be possible to bring back the system to a safe operating state. In such systems, the best strategy might be to passively identify operating conditions that might result in instabilities. With this information available, the system may then be designed to avoid these undesirable conditions, which is often called preventative system design; combustor systems design is one such example. Both active health monitoring and preventative system design can be performed using model-based or data-driven methods. The next section reviews the advantages and disadvantages of these methods.

1.2 Data-driven vs Physics-based Modeling

Most real systems are governed by nonlinear, coupled differential equations of a large number of process variables. Physics based modeling methods are based on an actual understanding of the underlying system dynamics. But, since it is not possible to create a model of infinite complexity, most physics based models are a simplified representation of the system dynamics. Therefore, their accuracy depends on the level of simplification and the actual effects of all unincluded variables on the system dynamics. If that is significant, the physics based model will be a poor representation of the system dynamics. In addition, modeling systems is essentially a trial and error process, which takes time. Finding a suitable model and refining it until it produces the desired results is often a long process.

In comparison, data-driven methods do not represent the actual system dynamics [1]. They are only concerned with a user defined objective (task). They try to learn the relation between the relevant input and output variables, for example, the relation between a observed data point and the class it belongs to, or the relation between some input parameter and the resultant system response etc. This could be useful in cases where modeling the system is difficult due to a lack of understanding of the system dynamics or the models created being inadequate. But the accuracy of data-driven algorithms also depends on many factors, which are discussed in detail in section 1.3.

Two such scenarios where physics based modeling can be difficult are combustor design and battery health determination. This dissertation proposes dynamic data-
driven (machine learning based) algorithms for both cases. The next subsection briefly describes the main concepts of machine learning, a field of computer science which involves developing data-driven methods to accomplish different tasks.

1.2.1 Machine Learning

Machine learning (ML) [2], [3] is a field of computer science in which computers learn from data. Computers are trained to discover patterns in the data. If a feedback is provided to them regarding their performance, it is known as “supervised learning” and “unsupervised learning” otherwise. Supervised machine learning algorithms can be further divided into two categories, depending on the label associated with each data point. Classification refers to the scenario, where the data belongs to a finite number of classes (labels), and the objective of the ML algorithm is to determine the mapping from a data point to class label. Hence, in this case, the labels associated with a data point are discrete. If instead, the labels associated with each data point are continuous, the machine learning task is referred to as Regression. In mathematical terms, given a specific domain (set of data points) \((X, P(X))\), and a label space \(Y\), the training data in a supervised machine learning algorithm consists of \((x_i, y_i)\) pairs. The “task” in the ML algorithm refers to learning the function \(f\) (i.e \(p(y|x)\)) from the training data. The function \(f\) is then used to predict the corresponding label, \(y\) of a new instance \(x\). The most common machine learning algorithms used currently are briefly delineated below:

1. **Neural Networks** : These are a class of learning algorithms that are structured in terms of an interconnected set of neurons. They are used for a variety of purposes like modeling relationships between input and output, finding patterns in data etc.

2. **Deep learning** : The current state of the art in machine learning is Deep learning. The term refers to neural networks with multiple hidden layers (at least more than 2). With sufficient training data and high computational power, deep networks are able to model very complex relationships between input and output.

3. **Support Vector Machine** : Support vector machines (SVMs) are super-
vised learning algorithms used for classification and regression. In linearly separable binary classification tasks, the algorithm finds the hyper-plane that has the largest distance to the nearest training-data point of any class, since larger the margin, lower is the generalization error of the classifier. For non separable cases, it minimizes the probability of a data point lying on the wrong side of the margin. For high dimensional data sets, SVMs have been found to give good performance.

4. **Bayesian networks**: A Bayesian network, is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). Multiple algorithms exist in literature for their inference and learning.

### 1.2.2 Dynamic data-driven Method of Combustor Design

Traditionally, the objective of combustor system design has been issues like higher efficiency, power generation and emission [4, 5]. However, with the implementation of low emission technologies like lean premixed combustion, combustors have to operate in regimes where they are prone to thermo-acoustic instabilities. Use of newer grades of fuels like bio fuels and hydrogen-based fuels like synthetic gas further aggravate the problem, because they possess energy content and heat release pattern, widely different from those of conventional hydrocarbon fuels. Consequently, the behavior of the system is significantly different in terms of phenomena like occurrence of instabilities, blowout and flashback. Thermoacoustic instabilities in such combustion chambers result from the coupling between unsteady heat release rate and acoustic pressure fluctuations inside the chamber. [6, 7]. Thus, designing combustors is a challenging problem due to the difficulties in modelling the nonlinear dynamics involved in thermoacoustic instabilities. Physics based models generally are based on some simplifying assumptions, since the created models cannot be of infinite complexity. These difficulties limit the application of model-based design optimization to combustion systems that generally involve several input parameters (e.g., inlet velocity of air, air-fuel ratio, premixing level, and combustor geometry) [4, 5, 8]; these parameters potentially affect the combustion dynamics. Examples are existence of bifurcations in the dynamic behavior of
combustors and extremely high sensitivity of the combustor behavior to even small changes in some of the design parameters. On the other hand, it is economically in-feasible and unrealistic to have sensors for online measurements of all dynamic variables involved in combustion and to conduct experiments at a sufficiently dense set of operating points. The current state-of-the-art of mitigating combustion instabilities at the design stage itself mostly involves introduction of passive devices, such as quarter wave tube arrangement [9] and perforated liners [10]. These devices improve the stability of the system by damping the oscillations in the combustion chamber. However, these passive devices cannot mitigate different types of anomalous behavior, as they are not designed on the basis of actual performance of the combustor. An alternative approach that is used is the implementation of active control devices/mechanisms, where appropriate actions are initiated like injection of secondary fuel [11, 12, 13]. The objective of these control actions are primarily to alter the phase difference between the pressure and heat-release-rate oscillations. Due to the complexity of the physics involved, control algorithms often use data-driven methods. However, since the underlying dynamics is extremely fast (e.g., in the order of kHz for some of the circumferential modes of instability in gas turbine afterburners) real time control is challenging. Another passive method of avoiding combustion instabilities is identification of the stable operating zone at the design stage itself and limiting the parameter space of the design variables to the stable operating zone only. The most commonly used approach is the network model [14], where the combustor is resolved into a network of interconnected simpler elements and the response of each component to specific units is studied. To account for the nonlinear flame dynamics, the linear transfer function approach is generalized for development of flame describing function [15]. The flame describing function, which describes an input-output relation for the flame element, is usually derived from experiments or CFD analysis. However, the network model and the flame describing function, in particular, have limitations in predicting different dynamic regimes of a combustor. In recent times, with the availability of increasing computing power, high fidelity computer simulations have also been used for predicting the dynamic behavior [16]. But these simulations can be too time-consuming to use practically as a design tool. Recently the concept of dynamic data-driven application systems (DDDAS) [17] has found its way into design methodologies due to the
advent of fast sensing and computational technologies as well as due to the inherent flexibility of DDDAS. Both quantitative and qualitative data have been used in various fields of design, where a given software or program analyzes the collected data to produce a decision that would aid in the design of the system under consideration. Especially the dynamic characterization of the system evolution allows for continual optimization of the design space as new data become available, thus enhancing the overall design quality. In particular, the notion of DDDAS has been used in the field of combustion monitoring and control. In a recent work, DDDAS has been used for the prediction of instability and flame lean-blow-out in combustors [18, 19] using symbolic time series analysis (STSA) [20]. More recently, such an imaging-based analysis has also been reported using neural networks, where flame images have been used to detect the onset of combustion instability [21] also using STSA. However, from the viewpoint of developing a design methodology for thermoacoustically stable operations, these technologies are rather uncommon.

This dissertation proposes two dynamic data-driven algorithms for combustor design. One is a fixed structure data-driven method, while the second one is a variable structure data-driven method. The proposed design methodology only needs limited amounts of process data in the form of time series and does not require any detailed knowledge of the underlying combustion dynamics. Given the information in the form of time series data at certain operating conditions, a Bayesian non-parametric statistical method has been adopted to predict the system behavior for operating conditions at which data may not be available. In addition, the algorithm also quantifies the confidence in the estimate of the system response. The design algorithm thus produces a mapping from a set of operating conditions to that of stability regions in the combustion system. Once this map is generated, combustor designers can use it for deciding the best parameters, which would have the least probability of resulting in combustion instability.

1.2.3 Dynamic data-driven Method of Battery State Estimation

The state-of-charge (SOC) and state of health (SOH) are crucial parameters for operation of battery systems. The current state-of-the-art methods of battery param-
eter identification fall in three broad categories: (i) empirical modeling (ii) physics-based modeling and (iii) dynamic data-driven analysis. All the categories require experimental data for validation. Empirical modeling methods (e.g., impedance measurements and open circuit voltage testing [22]) often employ dedicated hardware and/or software and require tested battery cells, and are essentially off-line. Even though in specific cases they might provide good estimates, they are time-consuming and cost-prohibitive for general applications. Physics-based modeling methods have been extensively used for identification of battery parameters; examples are reduced-order system identification, linear switch models [23]), and Kalman filtering [24]. For model-based SOC estimation, equivalent-circuit models make use of extensive empirical data for parameterization [25], while electrochemical models are usually partially observable [26] and their embedded nonlinear and coupled nature requires certain model simplification (e.g., model order reduction) before they can be used for industrial application. On the other hand, Kalman filtering for SOC estimation is prone to the problem of non-convergence of the estimation error. As previously mentioned, the accuracy of these methods depends on how much knowledge of the electrochemical characteristics of the battery cells is available, to arrive at appropriate model structures for parameter identification at different operating points of the nonlinear battery dynamics. In contrast, dynamic data-driven methods do not require much knowledge of the battery dynamics. If comparable and adequate training data are available under different operating conditions, data-driven methods are significantly more efficient than model-based methods in terms of computation execution time and memory requirements. Several data-driven methods have been reported in literature for battery SOC identification, based on different machine learning algorithms, for e.g., neural networks (ANNs) [27], fuzzy logic [28], support vector regression [29], and symbolic dynamics [30]. [31] surveyed different methods to estimate SOC, and listed their benefits and drawbacks. Similarly, several algorithms have been proposed in literature for battery SOH identification. Nuhic et al. [32] used support vector machines (SVM) to estimate SOH under different environmental and load conditions by processing the training and test data based on load collectives. Lin et al. [33] constructed a probability neural network (PNN) and trained it for SOH identification. He et al. [34] proposed a method based on dynamic Bayesian networks (DBN), in which
a DBN was trained for each class of SOH values in the training data. A forward algorithm was then applied to estimate the SOH in real time. Hu et al. [35] used k-nearest neighbor (k-NN) regression [2]. Li et al. [36] have reported a dynamic data-driven method for SOH identification by using time series of the battery voltage at different aging stages. The underlying concept is built upon the theory of symbolic dynamic filtering (SDF) [37][38] that extracts the dynamic information from symbolized time series of signals as probabilistic finite state automata (PFSA). In 2015, Li et al. [39] have extended their earlier work by making use of an ensemble of pairs of synchronized battery input/output (i.e., current/voltage) time series for identification of the state-of-health (SOH) parameter. This dissertation proposes a data driven, computationally efficient method for estimation of state of batteries, which does not require knowledge of the electrochemistry of batteries, as an extension of [39].

1.3 Factors affecting performance of a machine learning algorithm

In the previous section, the relative advantages and disadvantages of physics based and data-driven methods were analyzed. But for any machine learning algorithm to perform well practically, the following conditions are necessary.

- **Condition 1**: Availability of sufficient training data which covers all events or classes of interest. For time series applications, this also extends to the requirement of an individual time series being sufficiently long, to encompass the dynamics of the underlying system.

- **Condition 2**: The extracted features must encapsulate all the relevant information in the data (for example, be capable of discriminating between classes of interest).

- **Condition 3**: The relation between the relevant input and output variables should not change during the time the algorithm is being trained.

This dissertation proposes algorithms for scenarios where conditions 1 or 3 might not hold, and also quantifies the effect of the non-conformity of these con-
ditions, on the overall performance of the algorithm. In addition, this dissertation proposes an algorithm to ensure conformity of condition 2.

1.3.1 Condition 1 - Quantification of the effect of length of data on performance of the algorithm

The performance of any machine learning algorithm is strongly dependent on the quality of training data available. In case of feature extraction from a time series, and specially construction of finite state automata from time series, as is done in Symbolic Time Series Analysis, the accuracy of the estimated PFSA is dependent on the length of the time series. Longer the time series, higher is the probability that the estimated matrix represents the true system behavior. But in real time estimation problems, it is imperative that an algorithm makes decisions after seeing as few samples as possible. This dissertation develops a method for real time parameter estimation in batteries as an extension of [39], where the uncertainties due to finite lengths of both training and testing time series data are modeled, and their effect quantified. The battery SOC identification method is formulated as a pattern classification problem, and the ensemble of time series pairs of synchronized battery inputs (i.e., charging/discharging current) and battery outputs (i.e., voltage) are used for feature extraction. The uncertainties due to finite lengths of both training and testing time series data are modeled as Dirichlet and multinomial distributions respectively, based on the earlier work of Wen et al. [40], and subsequently their effect on the classification performance is quantified. The proposed method has been validated with experimental data of a (commercial-scale) lead-acid battery.

1.3.2 Condition 2 - Alphabet selection for feature extraction

Symbolic Time Series Analysis (STSA) encodes the behavior of (possibly non-linear) dynamical systems from the observed time series by symbolization and construction of probabilistic finite state automata (PFSA) [37]. This is followed by computation of the state emission matrices, that are representative of the evolv-
ing statistical characteristics of the dynamical system. Therefore, in STSA, the emission matrices are the features extracted from time series. The core assumption in the STSA analysis for construction of probabilistic finite state automata (PFSA) from symbol strings is that the symbolic process under both nominal and off-nominal conditions can be approximated as a Markov chain of order $D$, called the $D$-Markov machine, where $D$ is a positive integer. The quality of the extracted features here, is dependent on the alphabet size chosen (size of the symbol alphabet) and also on the partitioning locations (for discretization of the time series). Stauer et al. [41] reported a comparison of maximum entropy partitioning and uniform partitioning, where it is concluded that maximum entropy partitioning is, in general, a better tool for change detection in symbolized time series than uniform partitioning. Buhl and Kennel [42] reported symbolic false nearest neighbor partitioning (SFNNP) to optimize generating partitions by avoiding topological degeneracy. However, SFNNP suffers from high computational complexity and low robustness to noise. Rajagopalan and Ray [43] introduced wavelet space partitioning (WSP), where the wavelet transform largely alleviates the above shortcoming and is particularly effective with noisy data. Subbu and Ray [44] introduced Hilbert-transform-based analytic signal space partitioning (ASSP) as an alternative to WSP. Nevertheless, these techniques emphasize on modeling more than anomaly detection. Jin et al. [45] reported the theory and validation of a wavelet-based feature extraction tool that used maximum entropy partitioning of the space of wavelet coefficients. Even if this partitioning is optimal (e.g., in terms of maximum entropy or some other criteria) under nominal conditions, it may not remain optimal at other conditions. Along this line Sarkar et al. [46] proposed a time-series partitioning procedure to extract low-dimensional features from time series while optimizing the class separability; however, this method is strongly dependent on the choice of the classifier tool. This dissertation proposes a sequential partitioning method to extract low-dimensional features from time series, such that the mutual information [47] between the resulting feature and the pattern classes is maximized (i.e., it finds features which carry the most amount of information about the pattern class the underlying time series belongs to). The proposed algorithm has been validated on simulation data as well as data from a laboratory scale combustor.
1.3.3 Condition 3 - Unsupervised Context Extraction

Condition 3 might not always hold in real life. This might happen, if say the data provided to the ML algorithm are sensor measurements collected over a long period of time. The interpretation of sensor data depends strongly on the operational context in which the data was generated. Some examples include effects such as changes in soil stiffness due to precipitation, for unattended ground seismic sensor responses; changes in lighting conditions, for the picture taken by a camera of the same person; and environmental temperature, for infrared image sensors, for example. Hence, both context and event (class) affect the sensor measurements. But generally only event labels are available for training data, which has been collected over a long period of time, during the course of which intrinsic factors affecting sensor measurements might have changed. With change in context, in a pattern classification problem, the decision boundaries for the classes of interest may also change. Hence, using a single classifier for deciding the class might lead to poor results. Thus, using the ML terminology, data under each context can be considered as a separate task.

Multi task learning (MTL) is a well researched field in machine learning. The aim in MTL is to jointly learn different tasks, which is different from learning each task separately. In essence, the techniques developed in MTL seek to discover relationships between the different tasks, and use it to come up with more accurate classifiers or predictors for each task. The intuition is that common information relevant to prediction can be shared among these tasks and learning them jointly can result in better generalization performance than independently learning each task. Different techniques proposed in the current literature are based on different set of assumptions. These assumptions could be that task parameters lie close to each other in some geometric sense [48], or parameters share a common prior [49]; [50];[51]), or they lie in a low dimensional subspace [52] or on a manifold ([53]). Two different approaches of MTL have been proposed in [54] and [55] where tasks are assumed to be clustered and parameters of tasks within a cluster are either exactly the same or lie close to each other. [56] assumed that the task parameters share a latent subspace, the intrinsic dimensionality of which is not assumed to be known a priori. They used an infinite latent feature model - the Indian Buffet Process - to automatically infer this number. Other approaches of transferring information
between tasks include sharing parameters of Gaussian processes [57]; sharing a common structure on the predictor space ([58]); and structured regularization in kernel methods [59]. [60] reviews some recent work on multi task learning in the field of Deep Learning. But, it is seen that parameter sharing is the most commonly used approach for neural network based multi task learning too. Thus, given a set of “tasks”, the above mentioned techniques seek to discover which tasks are related to each other, and based on some assumption made regarding degree/nature of relatedness, extract the classifier parameters for all the unique tasks discovered. The work proposed in this dissertation, in comparison, performs unsupervised context extraction, to find out the different “tasks” present in the data. Only bulk training data collected over a long time is available to the algorithm, along with the associated event labels. Since the data is a function of both context and event, the work proposed here uses graph theory to isolate the effect of context in the data, and discover the different sub populations within the data, which would be representative of different contexts (tasks). With the knowledge of the identified contexts, separate classifiers are trained for each context, leading to much improved results on real experimental data, in comparison to a single classifier trained on the entire data.

Another related concept is that of robustness. If it is assumed/known a-priori, that even under different contexts, the divergence between two tasks would be within some bound, and it is not possible to collect data from all contexts, then it is desired to extract robust features from the context whose data is available. The features should be capable of discriminating between events/classes of interest on both tasks with reasonable accuracy. This dissertation proposes some analytical measures to quantify robustness in target classification, and using that, it makes a comparative evaluation of performance robustness of different feature extractors and classifiers, on an experimental dataset.

1.4 Contributions of the Dissertation

The contributions of the dissertation are as follows:

1. **Unsupervised Context Extraction and Context Based Event Classification**: This dissertation develops an algorithm for context based event
classification. Graph theoretic concepts are used to determine in an unsupervised manner the number and identity of contexts (machine learning “tasks”) in sensor data. After context extraction, separate classifiers are trained for each context. The proposed approach of context based event classification has been validated on experimental data.

2. **Alphabet Size Selection for Symbolization of Time Series**: A critical issue in modeling a PFSA from time series is the selection of the alphabet size and the partition locations for symbolization. These factors directly affect the quality of the extracted features, and the resultant classification performance. This dissertation develops an information theory based algorithm to determine the sub-optimal alphabet size and the corresponding partitioning locations for symbolization of time series, such that the mutual information between the resultant features and the class labels is maximized (i.e. the extracted features carry most of the information about the underlying class). The proposed approach has been validated on simulated data and experimental data from a laboratory-scale combustor.

3. **Quantification of effect of limited training and testing data on accuracy of real time SOC identification**: This dissertation develops a data-driven, electrochemistry independent and computationally efficient method for battery state identification. In addition, the method also performs real time identification of battery SOC, where the uncertainties due to limited lengths of training and testing time series data are quantified, and their effects on the final performance analyzed. The proposed method has been validated with experimental data of a (commercial-scale) lead-acid battery under varying input-output (e.g., current-voltage) conditions.

4. **Dynamic data-driven Combustor Design**: Two dynamic data-driven algorithms for combustor design have been developed in this dissertation. The algorithms, when provided with limited experimental data at some operating conditions, determine the mapping from operating conditions to system stability. This mapping is then used to predict the system stability for unseen conditions. The proposed algorithms also give their confidence about the stability estimates. This information about the predicted system behavior,
together with confidence in that prediction, at all operating conditions, can be used by combustor designers, to decide system parameters, which would have the least probability of resulting in instability. While the first algorithm assumes a fixed model structure for all operating conditions, the second developed algorithm gets rid of this assumption, and proposes a variable structure algorithm. In this approach, the structure, under user chosen constraints of accuracy and redundancy, best describing the time series of each operating condition is identified, and an information theory based criteria is then used for comparing between models with varying structures and/or parameters. Both algorithms have been validated on experimental data from a laboratory scale combustor.

1.5 Organization of the Dissertation

The dissertation is organized into nine chapters including the present one. The organization of the dissertation is presented below.

1. Chapter 2 presents some of the mathematical concepts used in this dissertation.

2. Chapter 3 presents the developed methodology for unsupervised context extraction and context based event classification.

3. Chapter 4 presents the developed information theory based sequential algorithm for alphabet size selection for symbolization of time series.

4. Chapter 5 presents the developed real time SOC identification algorithm.

5. Chapter 6 presents the developed fixed structure data-driven algorithm for combustor design.

6. Chapter 7 presents the developed variable structure dynamic data driven combustor design algorithm.

7. Chapter 8 summarizes the dissertation and proposes some future research directions.
2.1 Symbolic Time Series Analysis (STSA)

This chapter describes the underlying concept of symbolic time series analysis (STSA), which has been used in some algorithms developed in this dissertation.

STSA encodes the behavior of (possibly nonlinear) dynamical systems from the observed time series by symbolization and construction of state machines (i.e., probabilistic finite state automata (PFSA)) [37]. This is followed by computation of the state emission matrices that are representative of the evolving statistical characteristics of the dynamical system.

The major steps for construction of a PFSA from sensor signal outputs (e.g., time series) of a dynamical system are as follows.

1. Discretization of range space of the time series to convert the scalar or vector-valued data into symbol strings, where the symbols are drawn from a (finite) alphabet [61].

2. Encoding of probabilistic state machines from the symbol strings [37][62].

2.1.1 Symbolization of Time Series

This step involves quantization of the range space of the time series. The signal space is partitioned into a finite number of cells each labeled with a symbol, i.e., the number of cells is equal to the cardinality $|\Sigma|$ of the alphabet $\Sigma$. If the value of
time series at a given instant of time is located in a particular cell, then it is coded with the symbol associated with that cell. As such, a symbol from the alphabet \( \Sigma \) is assigned to each (signal) value corresponding to the cell where it belongs. In this way, a symbol time series also called a symbol string, is generated from the original time series data.

There are multiple ways of partitioning the range space of the time series. Some commonly used methods include maximum entropy partitioning (MEP) and uniform partitioning (UP) \[43\]. In UP, the range space is divided into equal-sized cells. On the other hand, MEP maximizes the entropy of the generated symbols and therefore, the information-rich cells of a data set are partitioned finer and those with sparse information are partitioned coarser, i.e., each cell contains (approximately) equal number of data points under MEP. In both UP and MEP, the choice of alphabet size \(|\Sigma|\) largely depends on the specific data set and the allowable loss of information.

### 2.1.2 Construction of PFSA from Symbol String

The core assumption in the STSA analysis for construction of probabilistic finite state automata (PFSA) from symbol strings is that the symbolic process under both nominal and off-nominal conditions can be approximated as a Markov chain of order \( D \), called the \( D \)-Markov machine, where \( D \) is a positive integer. While the details of the \( D \)-Markov machine construction are reported in \[37\],\[38\], the pertinent definitions and their implications are succinctly presented below.

**Definition 2.1.1.** *(Symbol Block)* A symbol block, also called a word, is a finite-length string of symbols \( s_i \) belonging to the alphabet \( \Sigma \), where the length of a word \( w \triangleq s_1s_2 \cdots s_\ell \) with \( s_i \in \Sigma \) is \(|w| = \ell\), and the length of the empty word \( \epsilon \) is \(|\epsilon| = 0\). The parameters of DFA (Deterministic Finite Automata) are extended as:

1. The set of all words constructed from symbols in \( \Sigma \), including the empty word \( \epsilon \), is denoted as \( \Sigma^* \).

2. The set of all words, whose suffix (respectively, prefix) is the word \( w \), is denoted as \( \Sigma^*w \) (respectively, \( w\Sigma^* \)).

3. The set of all words of (finite) length \( \ell \), where \( \ell > 0 \), is denoted as \( \Sigma^\ell \).
Definition 2.1.2. (DFSA [63]) A deterministic finite-state automaton (DFSA) $G$ is a triple $(\Sigma, Q, \delta)$ where:

1. $\Sigma$ is a (nonempty) finite alphabet with cardinality $|\Sigma|$;
2. $Q$ is a (nonempty) finite set of states with cardinality $|Q|$;
3. $\delta : Q \times \Sigma \rightarrow Q$ is the state transition map.

Definition 2.1.3. (PFSA [37, 38]) A probabilistic finite-state automaton (PFSA) is constructed on the algebraic structure of deterministic finite-state automata (DFA) $G = (\Sigma, Q, \delta)$ as a pair $K = (G, \Pi)$, i.e., the PFSA $K$ is a 4-tuple $K = (\Sigma, Q, \delta, \Pi)$, where:

1. $\Sigma$ is a nonempty finite set, called the symbol alphabet, with cardinality $|\Sigma| < \infty$;
2. $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$ is the state set with cardinality $|Q| \leq |\Sigma|^D$, i.e., the states are represented by equivalence classes of symbol blocks of maximum length $D$ corresponding to a symbol sequence $S$.
3. $\delta : Q \times \Sigma \rightarrow Q$ is the state transition mapping, which generates the symbol sequences.
4. $\Pi : Q \times \Sigma \rightarrow [0, 1]$ is the morph matrix of size $|Q| \times |\Sigma|$; the $ij^{th}$ element $\Pi(i, j)$ of the matrix $\Pi$ denotes the probability of finding the symbol $\sigma_j$ at next time step while making a transition from the state $q_i$.

2.1.2.1 $D$-Markov Modelling

This subsection introduces a special class of PFSA, called D-Markov machine, which has a simple algebraic structure and is computationally efficient for construction and implementation [37, 38].

Definition 2.1.4. (D-Markov) A D-Markov machine [37] is a PFSA in which each state is represented by a (nonempty) finite string of $D$ symbols where,

1. $D$, a positive integer, is the depth of the Markov machine.
2. \( Q \) is the finite set of states with cardinality \(|Q| \leq |\Sigma|^D\). The states are represented by equivalence classes of symbol strings of maximum length \( D \), and each symbol in the string belongs to the alphabet \( \Sigma \).

3. \( \delta : Q \times \Sigma \rightarrow Q \) is the state transition map that satisfies the following condition: if \(|Q| = |\Sigma|^D\): There exist \( \alpha, \beta \in \Sigma \) and \( s \in \Sigma^* \) such that \( \delta(\alpha s, \beta) = s\beta \) and \( \alpha s, s\beta \in Q \).

**Remark 2.1.1.** It follows from Definition 2.1.4 that a \( D \)-Markov chain is treated as a statistically stationary stochastic process \( S = \cdots s_{-1}s_0s_1\cdots \), where the probability of occurrence of a new symbol depends only on the last \( D \) symbols, i.e.,

\[
P[s_n|\cdots s_{n-D}\cdots s_{n-1}] = P[s_n|s_{n-D}\cdots s_{n-1}].
\]

The construction of a \( D \)-Markov machine is based on: (i) state splitting that generates symbol blocks of different lengths according to their relative importance; and (ii) state merging that assimilates histories from symbol blocks leading to the same symbolic behavior. On an alphabet \( \Sigma \), the total number of possible states after state splitting becomes less than or equal to \(|\Sigma|^D\); operations of state merging may significantly reduce that number [38]. However, no state splitting or state merging is required for \( D = 1 \), which is the simplest configuration of a \( D \)-Markov machine.

The PFSA states represent different combinations of blocks of symbols on the symbol string. In the graph of a PFSA, the directional edge (i.e., the emitted event) that interconnects a state (i.e., a node) to another state represents the transition probability between these states. The procedure for estimation of the emission probabilities is presented next.

Given a (finite-length) symbol string \( S \) over a (finite) alphabet \( \Sigma \), there exist several PFSA construction algorithms to discover the underlying irreducible PFSA model \( K \) of \( S \). These algorithms start with identifying the structure of the PFSA \( K \triangleq (Q, \Sigma, \delta, \pi) \). To estimate the state emission matrix, a \(|Q| \times |\Sigma|\) count matrix \( C \) is constructed and each element \( c_{kj} \) of \( C \) is computed as: \( c_{kj} \triangleq 1 + N_{kj} \), where \( N_{kj} \) denotes the number of times that a symbol \( \sigma_j \) is generated from the state \( q_k \) upon observing the symbol string \( S \). The maximum a posteriori probability (MAP) estimates of emission probabilities for PFSA \( K \) are computed by frequency counting as:
\[ \pi (\sigma_j \mid q_k) \triangleq \frac{c_{kj}}{\sum \ell c_{k\ell}} = \frac{1 + N_{kj}}{\mid \Sigma \mid + \sum \ell N_{k\ell}} \quad (2.1) \]

The rationale for initializing each element of the count matrix \( C \) to 1 is that if no event is generated at a state \( q \in Q \), then there should be no preference to any particular symbol and it is logical to have \( \pi(\sigma|q) = \frac{1}{\mid \Sigma \mid} \forall \sigma \in \Sigma \), i.e., the uniform distribution of event generation at the state \( q \). The above procedure guarantees that the PFSA, constructed from a (finite-length) symbol string, will have an (elementwise) strictly positive emissivity map \( \Pi \). Having computed the emission probabilities \( \pi (\sigma_j \mid q_k) \) for \( j \in \{1, 2, \cdots , \mid \Sigma \mid \} \) and \( k \in \{1, 2, \cdots , \mid Q \mid \} \), the estimated \( (\mid Q \mid \times \mid \Sigma \mid) \) emission probability matrix of the PFSA is obtained as:

\[ \Pi \triangleq \begin{bmatrix}
\pi (\sigma_1 \mid q_1) & \cdots & \pi (\sigma_{\mid \Sigma \mid} \mid q_1) \\
\vdots & \ddots & \vdots \\
\pi (\sigma_1 \mid q_{\mid Q \mid}) & \cdots & \pi (\sigma_{\mid \Sigma \mid} \mid q_{\mid Q \mid})
\end{bmatrix}. \quad (2.2) \]

This emission probability matrix is taken to be the feature corresponding to the time series, a representation of the underlying system dynamics.

Bahrampour et al. [64] compared the performance of three feature extractors namely Cepstrum, Principal Component Analysis (PCA) and Symbolic Time Series Analysis (STSA) for target detection and classification. The feature extraction algorithms were executed in conjunction with three different machine learning algorithms, namely, support vector machines (SVM), k-nearest neighbor (k-NN), and sparse representation classifier (SRC). The results show consistently superior performance of STSA-based feature extraction over both Cepstrum-based and PCA-based feature extraction in terms of successful detection, false alarm, and wrong detection and classification decisions. Mallapragada et al. [65] compared the performance of STSA with PCA for robotic applications. Rao et al. [66] reviewed STSA and its performance relative to other commonly used machine learning algorithms, such as Bayesian Filters and Artificial Neural Networks.

### 2.2 Summary

This chapter succinctly presents the underlying theory of Symbolic Time Series Analysis and D-Markov Models.
Algorithm 1 Symbolic Time Series Analysis (STSA) for Feature Extraction

Require: Symbolic strings of length $N$ obtained at $I$ different operating conditions of system under analysis: $S_i = \{s^1_i, s^2_i, s^3_i, \ldots, s^N_i\}, i = 0, 1, \ldots, I - 1$; the alphabet size $|\Sigma| = |\Sigma_1| \times |\Sigma_2|$; the depth $D$ of Markov machine; and number of states $|Q|$, where $|Q| \leq |\Sigma|^D$.

Ensure: Extracted emission matrices $\Pi_i \in \mathbb{R}^{|\Sigma|^D \times |\Sigma|}, i = 0, 1, \ldots, I - 1$ for each symbol string and the feature divergences $\{m_i\}_{i=0}^{I-1}$.

Initialize $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_{|\Sigma|}\}$, and $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$.

for $i = 0$ to $I - 1$ do

    for $k = 1$ to $|Q|$ do

        for $j = 1$ to $|\Sigma|$ do

            Count the number of event that symbol $\sigma_j$ occurs after symbol (combination) of $q_k = \{\sigma_1^k \ldots \sigma_{|D|}^k\}$, denoted as $N(\sigma_j, q_k)$, from the symbol string $S_i$.

        end for

    end for

    for $j = 1$ to $|\Sigma|$ do

        compute the estimated emission probability $\pi(\sigma_j | q_k)$ (see Eq. (2.1)).

    end for

end for

Construct the estimated emission matrix $\Pi_i$ for symbol string $S_i$ (see Eq. (2.2)).

end for
3.1 Introduction

In Chapter 1, section 1.3.3 surveyed different MTL techniques proposed in literature. The proposed approaches had (i) training data with context (“task”) labels (ii) training data for all distinct tasks (which might be labeled or unlabeled).

This chapter considers two scenarios where the above do not hold:

- Training data collected over a long time is available to the user. The event (class) labels have been provided, but the context labels are not known (i.e., whether they all came from the same distribution (“task”) or not).

- Labeled data is available for only one task. If it is assumed that the divergence between two possible tasks would be within some bound, it is desired to extract robust features using the training data of the available task, which would be largely insensitive to changes due to context, while giving acceptable performance on both tasks.

The major contributions of the work presented in this chapter are as follows:

1. An unsupervised method of context extraction developed using a graph theoretic approach. The proposed algorithm finds the number and identity of
contexts in the data. After context extraction, separate classifiers are trained for each context. The proposed methodology of context based event classification is validated on real experimental data.

2. Analytical measures based on area under the ROC curve have been proposed to make a comparative evaluation of performance robustness of different feature extractors for a given rocket mortar launch problem.

3.2 Problem Formulation for Unsupervised Context Extraction

As discussed in Chapter 1, both context and event affect the sensor data. If the event is fixed, all features under the same context are expected to be similar and hence, close by in the feature space. With this intuition, clustering techniques can be used to identify the different clusters. There are many clustering techniques available in literature, which fall under different categories like density based, centroid based, hierarchical, and agglomerative clustering [67]. In this work, a graph theoretic approach, modularity based community detection was used to identify clusters and create the context alphabet. Unlike most other clustering methods, this approach has the advantage of providing both the optimal number of communities (clusters) and the members of each community. The majority of methods developed till now for community detection in graphs can be reviewed in [68].

3.2.1 Graph Creation

In the graph representation of the dataset, each feature vector from the corresponding time series is a node and the weight on the edge connecting two nodes is proportional to the similarity between the two feature vectors, which in this case was the inverse of the Euclidean norm between the extracted feature vectors. Thus a complete weighted graph is created. After this, all edges with similarity $\leq$ some user defined $\epsilon$ are removed. The weighted incomplete graph is then converted into a incomplete unweighted multigraph by replacing each edge with weight $w$ into $\text{floor}(w)$ number of unweighted edges between the same pair of vertices.
3.2.2 Community Detection

A community in a graph is a cluster of nodes with more intra cluster edges than inter cluster edges. Community detection in graphs aims to identify the modules and possibly, their hierarchical organization in graphs, by only using the information encoded in the graph topology. This involves selecting a quality measure, which would quantify the desired properties of the communities and then assigning nodes to different communities such that the given measure was optimized. Here, modularity proposed in [69] was used as the quality measure.

3.2.2.1 Modularity

It is based on the idea that a random graph is not expected to have a cluster structure, so the possible existence of clusters is revealed by the comparison between the actual density of edges in a subgraph and the density one would expect to have in the subgraph, if the vertices of the graph were attached regardless of community structure. This expected edge density depends on the chosen null model, i.e. a copy of the original graph, keeping some of its structural properties, but without community structure. So, if one has the original nodes, and preserves the degree of each vertex, but adds edges randomly, then modularity can be written as follows:

\[
Modularity = \frac{1}{2m} \sum_{vw} \left[ A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v c_w) \tag{3.1}
\]

Where \( k_v \) is the degree of node \( v \). \( c_v \) is the community to which node \( v \) belongs, \( m \) is the total number of edges. \( A_{vw} \) is an element in the adjacency matrix \( A \), which gives us whether or not there is actually an edge between vertex \( v \) and \( w \), in case of a weighted graph, the weight of such an edge. The second term is the probability of there being a edge between vertices \( v \) and \( w \), if the edges were added at random, preserving the degree of each vertex. For good community structure, the term in the square brackets should be positive and large when both vertices are in same community.
3.2.2.2 Fast Community Detection Using Modularity

In this work, fast community detection algorithm ([70]) was used. It is an agglomerative hierarchical clustering algorithm. Starting with a state in which each vertex is the sole member of one of \( n \) communities, where \( n \) is the number of nodes in the graph, at each step two such communities are merged, whose merging results in the greatest increase of modularity with respect to the previous configuration. At any stage, only those pairs of communities need to be considered, which have one or more edges connecting them. This is because modularity can never be increased by joining pairs of communities which don’t share edges. The change in modularity upon joining two communities is given by \( e_{ij} + e_{ji} - 2a_ia_j \) where \( e_{ij} \) is the fraction of edges between community \( i \) and community \( j \)

\[
a_i = \sum_k e_{ik}
\]  

over all communities \( k \). For an extension to a weighted graph or multigraph, the elements of the matrix \( A \), \( A_{ij} \) are just the edge strengths or the number of edges between the pair of vertices \( i \) and \( j \) respectively. Let \( m \) and \( n \) be the number of edges and nodes in the graph. The first step of the algorithm (finding which pair of communities are to be merged) takes \( O(m) \) time, since at most \( m \) pairs would need to be considered. After the pair of communities are merged, the values of \( e_{ij} \) corresponding to these 2 communities needs to be modified. Therefore this step would take \( O(n) \) time. These 2 steps need to be repeated \( n-1 \) times. Hence, the entire algorithm runs in \( O(n(m+n)) \) time. The number of partitions found in this procedure is \( n \), each with a different number of clusters from \( n \) to 1. The largest value of modularity in this subset of partitions corresponds to the optimal partitioning of the dataset.

3.3 Experimental Validation

3.3.1 Data Acquisition

A series of experiments were designed to validate the proposed technique for context-aware event classification. 3-axis geophones were deployed to identify two
different types of walking which are, (i) Normal walking and (ii) Stealthy walking. Seismic response from geophones, used in the analysis, were collected on two different types of test fields characterized by gravel road, moist soil road respectively. As the characteristics of seismic response changes significantly with the change of soil properties, two different ground types are considered to be two physical contexts. 80 experiments (40 for normal walking and 40 for stealthy walking), constituting of two different human subjects, were performed for each context. The seismic sensors (geophones) were buried approximately 15 cm deep underneath the soil surface. Human subjects passed by the sensor sites at a distance of approximately 2 m. The signal from the geophone was acquired at a sampling frequency of 4 kHz for 10 seconds for each experiment. The main task of the context aware event classifier is to discriminate between normal and stealthy walking over different soil types with high accuracy.

### 3.3.2 Data preprocessing and Feature extraction

The feature extraction method used in this work is built upon the concept of symbolic dynamic filtering (SDF)[37], [38]. In the signal preprocessing step of analysis of our experimental data, the DC component of a seismic signal has been first eliminated, resulting in a zero mean signal. Then, the signal has been partitioned using the maximum entropy partitioning approach with a symbol size of 7. The maximum number of allowable states of the D-Markov machine was varied, and the classification performance on a validation test set found in each case. This process has been repeated 3 times to obtain average error. The number of states has then be chosen to be 10, as it resulted in the best performance on the validation set. The features thus obtained from a time-series data set, i.e. the stationary state probability vector of the D-Markov machine thus represented, are then used for classification.

### 3.3.3 Results and Discussion

The classifier used in this analysis is Linear Support Vector Machine [71]. The entire data set is randomly divided into training and testing(60% and 40% respectively) sets ten times and the analysis has been repeated for each combination of
Figure 3.1: Soil types

(a) Gravel

(b) Moist soil

Figure 3.1: Soil types
training and testing set. Three different cases are compared. They are as follows:

1. **No context knowledge:** The training samples only have the event labels. The data has been collected under two different physical contexts (section 3.3.1), but, it is assumed that this knowledge is not available to the user. Hence, instead of a contextual classifier, a single classifier is trained. This classifier is then used to classify the event of the test samples. The mean and std of error % has been found to be 14.38 and 3.82. (table 3.1). Hence, even though no context knowledge was available, the error percent is under 20, which shows the superiority of the chosen feature extraction method ($D$-Markov feature).

2. **Perfect context knowledge:** The training data is labeled with the event and context labels. A separate classifier is trained for each context (corresponding to the ground truth context). KNN classifier is then used to partition the feature space and assign the context label to the unseen test samples (i.e., each test sample is assigned the context of the majority of its training neighbors). For each sample, the trained classifier corresponding
to the assigned context is then used to classify its event. The mean and std of error % is found to be 8 and 2.4 (table 3.1). The best classification performance is obtained in this case, as expected.

3. **With context identification:** The training data has the event labels associated with it. The previously described unsupervised context extraction method is used to determine the number and identity of the contexts. KNN classifier is then used to partition the feature space and assign the context labels to the unseen test samples. For each sample, the trained classifier corresponding to the assigned context label is then used to classify its event. The mean and std of error % is calculated to be 8.1 and 2.81 (table 3.1). So, the average performance is the same as case 2, albeit a larger standard deviation. The reason behind this could be that the context information extracted might not fully coincide with the ground truth context, so, the resulting context-specific classifiers might not be the best hyper-planes separating the events under the true context.

Table 3.1: Experimental Results: Misclassification % in the three cases

<table>
<thead>
<tr>
<th>Case</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14.53</td>
<td>3.82</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>2.4</td>
</tr>
<tr>
<td>3</td>
<td>8.1</td>
<td>2.81</td>
</tr>
</tbody>
</table>

The benefit of context-aware adaptation of classifiers is clearly visible from the above results as the performance improves by choosing the appropriate classifier for the estimated context.

### 3.4 Problem Formulation for evaluation of robustness of feature extraction

In the scenario considered in section 3.2, data from a variety of contexts are available to the user, albeit without the context labels. Now, we consider a scenario where data from only one context is available. But the test data might come from
another context. Hence, the objective here is to extract features which are indicative of the class they belong to, but robust to the variations in the data distribution due to change in context.

### 3.4.1 Approach

Let there be two environments, Env $i$ and Env $j$ with $i \neq j$, where these two environments are moderately different and uncertain. While Env $i$ is available for unlimited experimentation and is used for generating training data sets, access to Env $j$ is limited and is restricted to generating only test data sets, which are not available while training classifiers. Let us consider the following two situations.

1. Situation X(ii): Environmental condition match, i.e., both training and testing are conducted in Env $i$, and this is considered to be the reference situation.

2. Situation X(ij): Environmental condition mismatch, i.e., training is conducted in Env $i$ and testing is conducted in Env $j$ with $j \neq i$, and this is considered to be a perturbed situation.

The objective here is to quantify the classification performance and robustness for a given situation based on the respective ROC curves. The probability of successful detection $P_D$ is expressed as a function of the probability of false alarm $P_{FA}$, where $P_D, P_{FA} \in (0,1)$ and the supremum classification performance occurs if the ROC curve approaches the unit step function, i.e., $P_D$ goes to 1 for the full range of $P_{FA}$.

In the context of target detection & classification, in order to quantify classification performance, performance loss, and performance robustness, some definitions are introduced, using the ROC curve.

**Definition 3.4.1.** (Classification Performance) The classification performance in a situation $X^{(ij)}$ is defined to be the weighted area in the respective ROC curve.

$$\pi[(ij), w] \triangleq \int_0^1 d\theta \ w(\theta) \ P_D^{(ij)}(\theta) \quad (3.3)$$
where $w(\theta)$ is a weighting function with $w(\theta) \geq 0 \ \forall \theta \in (0,1)$ and $\int_0^1 d\theta \ w(\theta) = 1$, whose choice is a design parameter.

**Remark 3.4.1.** The choice of an appropriate weighting function is dependent on the purpose at hand. Generally, users may not be interested in the performance near the endpoints where either $P_{FA} \to 1$ or $P_D \to 0$. Thus, users may assign $w(\theta)$ to be zero near the end points and positive values in an intermediate range. For example, the weight $w(\theta)$ could be concentrated between, say, $P_{FA} = 0.05$ to $P_{FA} = 0.5$, depending on the situation under consideration. A particular choice is to assign $w(\theta) = \delta(\theta - \theta_0)$ to be the Dirac delta distribution. In this case, the performance loss represents the difference in detection probability, under the Neyman-Pearson criterion [72], that could be directly obtained from the confusion matrix at that operating point $\theta_0$.

**Definition 3.4.2.** (Performance Loss) The performance loss under a situation $X^{(ij)}$ relative to the situation $X^{(ii)}$ is defined to be the weighted area in the difference between the respective ROC curves.

$$\ell[(ij),(ii),w] \triangleq \int_0^1 d\theta \ w(\theta) \ (P_{D}^{(ii)}(\theta) - P_{D}^{(ij)}(\theta))$$

(3.4)

where $\ell > 0$ indicates performance loss and and $\ell < 0$ indicates performance gain.

**Remark 3.4.2.** For a mismatch between the training and testing site environments (i.e., $j \neq i$), most likely there would be performance loss implying that $\ell > 0$. However, since the positivity of $\ell$ is not guaranteed, one may encounter performance gain under a mismatch in rare cases. Therefore, Eq. (3.4) does not take an absolute value or a norm of the difference between the $P_D$ functions.

**Definition 3.4.3.** (Performance Robustness) The performance robustness for a situation $X^{(ij)}$ relative to the situation $X^{(ii)}$ is defined in terms of the ratio of the respective $P_D$’s as follows.

$$\rho[(ij),(ii),w] \triangleq \int_0^1 d\theta \ w(\theta) \left( \frac{P_{D}^{(ij)}(\theta)}{P_{D}^{(ii)}(\theta)} \right)$$

(3.5)

**Remark 3.4.3.** Although not guaranteed, it is likely that $0 < \rho < 1$. For a
mismatch between the training and testing sites (i.e., $j \neq i$), a larger value of $\rho$ represents a better robustness of classification.

### 3.4.2 Feature Extraction

Three different methods of feature extraction from time series are considered in this work.

#### 3.4.2.1 Cepstrum

Given a signal time series $f(t)$, $t = 1, \ldots, N$, its Cepstrum [73] is computed in the following form.

$$f_c(t) = \Re \left( \mathcal{F}^{-1} (\log |F(\omega)|) \right)$$

(3.6)

where $F(\omega)$ is the Fourier transform of the signal $f(t)$; the operator $\mathcal{F}^{-1}$ is the inverse Fourier transform; and $\Re(z)$ indicates the real part of a complex scalar $z$. After obtaining the Fourier transform of the signal, frequency components with small values are discarded before taking the inverse Fourier transform to prevent insignificant components from having high Cepstrum values. The algorithm is presented in [74].

#### 3.4.2.2 Principal Component Analysis (PCA)

While the Cepstrum-based feature extraction makes use of only the information embedded in a time series, PCA [2] takes advantage of the information of the ensemble of training data in addition to the local information to extract the features. In the specific application in this work, the number of data points is smaller than the dimensionality of data space and the PCA algorithm is modified accordingly as described in [74].

#### 3.4.2.3 Symbolic Time Series Analysis

Chapter 2 presents the details of STSA.
3.4.3 Classification of Extracted features

Three different existing methods of classification are considered in this work for operation with each of the three methods (i.e., PCA, Cepstrum, and SDF) of feature extraction.

3.4.3.1 k-nearest neighbor (k-NN)

The k-NN method is one of the most frequently used tools of pattern classification [2][75] and has been used in the current work for performance comparison with other classification methods. In the standard k-NN algorithm, an unseen test data point is assigned the class label associated with the majority of its k neighbors. A ROC curve can be generated by adjusting the number of nearest neighbors to consider, when deciding the class label of the test data point. For example, if \( k = 7 \), the k-NN model would declare a test point to be a target detection if any of 4, 5, 6, or 7 of the nearest 7 neighbors were positive examples. A more strict test, with lower probability of detection accompanied by a lower false alarm rate, would be to declare only targets with 5, 6, or 7 positive example neighbors as detections. In this way, \( k + 1 \) different classifiers are generated, and the ROC is the convex hull of their \((P_D, P_{FA})\) points.

3.4.3.2 Support Vector Machine (SVM)

The SVM method [2][75] has been used in the current work for performance comparison with other classification methods. A Gaussian kernel has been used for SVM, similar to what was done in [74] to construct confusion matrices for target classification with unattended ground sensor (UGS) systems. In this work, the bias term is parameterized to produce a family of \((P_D, P_{FA})\) trade-off points to produce an ROC curve for a support vector machine from an empirical testing data set, and the ROC curve is the convex hull of the resulting \((P_D, P_{FA})\) points.

3.4.3.3 Sparse Representation Classification (SRC)

The concept of sparse representation was first used for solving the problem of face recognition [76]. A large matrix \( A \triangleq [A_1 A_2 \cdots A_C] \), consisting of the training data, is constructed where \( A_i, i \in \{1, \cdots, C\} \) is a \((n \times N_i)\) training matrix consisting
of the training samples belonging to the $i^{th}$ class, $n$ is the number of features, and $N_i, i \in \{1, \cdots, C\}$ is the number of training samples in the class $i$, and $C$ is the total number of class labels. It is assumed that a test sample from the $i^{th}$ class lies approximately within the subspace formed by the training data of the $i^{th}$ class. For a given test vector $y$ in the sparse classification algorithm, the following $\ell_1$-optimization problem [77] needs to be solved as:

$$\min \|x\|_{\ell_1} \text{ such that } \|y - Ax\|_{\ell_2} \leq \epsilon$$

(3.7)

In the above optimization problem, the user-selected parameter $\epsilon$ is representative of the upper bound of the noise spectrum in the data [47], where the optimal solution $x$ of the above $\ell_1$ optimization is shown to be a sparse vector. For $x \in \mathbb{R}^M$, where $M$ is the total number of training samples, let $\delta_i(x) \in \mathbb{R}^M$ be a new vector whose only non-zero elements are the entries in $x$ that are associated with class $i$ in $A$. In the noiseless scenario, if the test data $y$ belongs to the $i^{th}$ class, then $\delta_j(x)$ should be a zero vector for all possible $j \neq i$. However, since noisy data are encountered in real-life applications, the residuals $r_i = \|y - A\delta_i(x)\|_2$ needs to be computed for classification of the test sample $y$ and the label of the test vector is predicted as the argument $i^*$ that is the minimizer of the residuals. Let the respective residuals for a unseen data point be $r_1$ and $r_2$. To create a ROC curve for the sparse classifier, the residual $r_1$ is scaled by a weight term $w \geq 0$, so that detection $i^*$ belongs to class 1 if $wr_1 < r_2$, and belongs to class 2 otherwise, where $w$ is varied over a range. One of the main advantages of the SRC algorithm is that a careful selection of the features is not necessary. Nevertheless, the number of features must be sufficiently large so that the sparse representation is correctly computed.

### 3.5 Results of Field data Analysis

This section presents the results of analysis for classification performance, loss of performance, and performance robustness (see Eqs. (3.3), (3.4), and (3.5), respectively) of two different types of mortars, which were generated from two sets of field data. Each data set consists of audio (acoustic) sensor measurements of both
launch and impact events for a mortar launcher [78]. The data have been collected from two types of mortar launcher, which are denoted as type \( A \) and type \( B \), respectively, and at each of two different sites, which are denoted as Site\#1 and Site\#2. Launch and impact events are combined in the test and training sets, to produce a total set of 120 events: 30 of each launcher at each site. In preprocessing, the events are clipped from the acoustic signal, removing periods of silence. The bias is removed and the amplitudes of the time series signals are normalized to make a zero-mean and unit-variance data set.

### 3.5.1 Methodology of Data Analysis

In order to test the robustness of each feature extraction/classification technique combination, only data from Site\#1 are used for training. Because of the limited number of samples in the available data sets, and to remove dependence on any particular testing set, a \( k \)-fold cross validation method is used with \( k = 5 \). That is, for each of the launchers type \( A \) and type \( B \), 30 sample events from Site\#1 are randomly divided into 5 equal-sized subsets. The entire feature extraction & classification procedure of training and testing is then repeated a total of 5 times, each time using 4 of the 5 subsets of data from Site\#1 as training data and the 5\(^{th}\) subset for testing. Therefore, all samples appear as a testing example exactly once. In this setting, each training subset contains 24 examples each for mortar type \( A \) and mortar type \( B \), all taken from Site\#1. Correct classification of a mortar (either type \( A \) or type \( B \)) is denoted as a detection, and misclassification of a mortar of true type \( B \) as a type \( A \) and vice versa is denoted as a false alarm.

For each of launcher type \( A \) and type \( B \), the 30 sample events from Site\#1 are randomly divided (80%, 20%) into two subsets: for training, and testing. Feature extraction is performed as follows:

**PCA:** The components which contain 96% of the total energy of the training set of 48 training examples, having 24 from each launcher type, are retained to create a feature vector of length 16.

**Cepstrum:** The first 60 components are retained to make a feature vector.

**SDF:** The training data set is generated by the concatenation of all 24 time-series for each of type \( A \) and type \( B \) mortars in Site \#1. For symbol generation,
the alphabet size $|\Sigma| = 10$.

In summary, each of the three feature extractors (i.e., PCA, Cepstrum, and SDF) thus produce 24 training feature vectors for each of the two mortar types. Each of the three classifiers (i.e., SVM, k-NN, and SRC) is then trained on these 48 feature vector, at Site#1. For the $k$-nearest neighbor method, $k$ has been chosen via dividing the current training set into a training set and a cross-validation set (80% and 20%, respectively), computing the performance of the classifier, repeating this process 3 times, and choosing the $k$ with the best average performance, from three possible values ($k=3$ or 5 or 7). The same cross-validation procedure has used to choose the parameters $\sigma$ for the Support Vector Machine method, and $\epsilon$ in the Sparse Representation Classifier.

![Figure 3.3: ROC Plots for Combinations of Classifiers and Feature Extractors](image)

For each combination of feature extractor and classifier (a total of $3 \times 3 = 9$ combinations), tests are conducted on the 12 (i.e., 6 of each type) remaining samples at Site #1. All 30 examples at Site #2 (which is a different site from Site #1 where the training has been performed), are used to determine the robustness of each combination of the feature extractor and classifier. As described previously, parameters of these classifiers are tuned to provide the full range of the respective ROC curve.

The ROC curves are obtained by averaging the false alarm probability cor-
responding to each of the 7 different probability of detection levels (0/6, 1/6, ... for Site #1, and 0/30, 1/30, ... for Site#2) over 5 combinations of training and testing sets (as previously explained). These ROC curves are displayed in Fig. 3.3 as convex hulls [79] in the individual plots.

After the ROC curves are generated, the next task is computation of classification performance, performance loss, and performance robustness by following Eq. (3.3), Eq. (3.4), and Eq. (3.5), respectively, with a user-selectable weighting function \( w \). Assuming that the user may not be interested in a classifier for which \( 0.1 < P_{FA} < 0.50 \) for which \( P_D > 0.60 \), the weighting function \( w \) is selected as:

\[
w(\theta) = \begin{cases} 
2.5 & \text{if } 0.1 \leq \theta \leq 0.5 \\
0 & \text{otherwise}.
\end{cases}
\]

The results are summarized in Table 3.2.

**Table 3.2: Classification Performance, Performance Loss, and Robustness**

<table>
<thead>
<tr>
<th>Feature Extractor</th>
<th>k-NN</th>
<th>SVM</th>
<th>SRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>Classification Performance 0.63</td>
<td>0.84</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>Performance Loss 0.11</td>
<td>0.05</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td>Performance Robustness 0.65</td>
<td>0.95</td>
<td>0.88</td>
</tr>
<tr>
<td>Cepstrum</td>
<td>Classification Performance 0.96</td>
<td>0.98</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>Performance Loss 0.05</td>
<td>0.35</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>Performance Robustness 0.95</td>
<td>0.65</td>
<td>0.99</td>
</tr>
<tr>
<td>SDF</td>
<td>Classification Performance 0.96</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>Performance Loss 0.12</td>
<td>0.11</td>
<td>0.13</td>
</tr>
<tr>
<td></td>
<td>Performance Robustness 0.87</td>
<td>0.89</td>
<td>0.87</td>
</tr>
</tbody>
</table>

### 3.5.2 Results and Discussion

For the two types (i.e., type A and type B) of mortar launch data, SDF performs very well as a feature extractor in conjunction with each of the three different types of classifiers (i.e., k-NN, SVM and SRC) for mortar type classification when the testing and the training data were used from the same launch site, namely, Site #1. The Cepstrum also performs quite well in this situation, while PCA appears to be a less effective feature extraction method for acoustic data. In terms of robustness, that is, the relationship between performance when the targets of
interest have been launched from the other site, Site #2 and the training has been
performed at Site #1, the degradation of performance of the SDF features appears
to be quite smooth, with approximately the same robustness performance for all
three classifiers. For the k-nn and SRC classifiers, the Cepstral features provide
quite high robustness performance, but relatively low robustness performance for
the SVM classifier. In this situation, the robust performance of PCA is degraded
quite substantially when using a k-NN classifier.

3.6 Summary and Conclusions

In this chapter a contextual decision adaptation methodology was developed. It
was seen that context aware event classification showed a significant improvement
over generic classification. In addition, a concept of performance robustness for
target detection & classification has been introduced by making use of receiver
operating characteristics (ROC) [79]. This concept was then used to evaluate the
robustness of different feature extractors on an experimental dataset.
Chapter 4

Information Theoretic Approach for Alphabet Size Selection for Symbolic Time Series Analysis

4.1 Introduction

Symbolic time series analysis (STSA), has been explained in Chapter 2. But, even though various methods for construction of PFSA from symbol strings have been reported in literature, similar efforts have not been expended on identifying an appropriate alphabet size for partitioning of time series, so that the symbol strings can be optimally or sub optimally generated in a specified sense. This chapter addresses this issue and proposes an information-theoretic procedure for partitioning of time series to extract low-dimensional features. The proposed algorithm identifies boundary locations of the partitioning segments such that the mutual information between the state probability vector of the resulting PFSA and the members of the pattern classes, is maximized. In addition, robustness of the symbolization process to measurement noise has also been investigated. The proposed alphabet-size selection algorithm has been validated by two examples. The first example addresses parameter identification in a simulated Duffing system with sinusoidal input excitation. The second example is built upon an ensemble of time series of chemiluminescence data to predict lean blowout (LBO) phenomena in a
laboratory-scale swirl-stabilized combustor apparatus.

Major contributions of the work presented in this chapter are stated below.

1. Partitioning of time series in a way that maximizes the mutual information \([47]\) between the symbolic dynamic feature (e.g., the state probability vector of PFSA) and the pattern class to which it belongs.

2. Investigation of the robustness of the proposed algorithm to measurement noise.

3. Comparison of the performance of the proposed partitioning technique with another commonly used method, maximum entropy partitioning (MEP).

4. Validation of the proposed algorithm on simulated data from a sinusoidally excited Duffing system, and experimental data from a laboratory-scale swirl-stabilized combustor for lean blowout (LBO) prediction.

4.2 Problem Formulation

The success of a time-series partitioning methodology depends on how well the calculated PFSA captures the underlying system dynamics (e.g., the state probability vector \(p\)). In a pattern classification problem, it is desired to extract features which are indicative of the pattern class to which they belong; i.e. high inter class distance (features belonging to different pattern classes should be well separated) and low intra class variance (features belonging to same class should be similar to each other). In information theory, this concept is encapsulated by mutual information, which quantifies the amount of information obtained about one random variable, through the other random variable. A lower bound of the probability \(P_E\) of incorrect estimation \(\hat{C}\) of the true class \(C\) (i.e., \(P_E = Pr[\hat{C} \neq C]\)) is obtained from the weak form of Fano’s inequality \([47]\) as:

\[
P_E \geq \frac{H(C|P) - 1}{\log_2 |C|} = \frac{H(C) - I(P; C) - 1}{\log_2 |C|}.
\]

(4.1)

where the random vector \(P\) represents the input feature extracted from a given time series whose pattern class (belonging to \(C = \{c_0, c_1, \ldots, c_{|C|-1}\}\)) may be unknown; \(H(C)\) and \(H(C|P)\) are the entropy and conditional entropy of of pattern
class $C$, respectively; and $I(P; C)$ is the mutual information between the input feature and the pattern class. Since $H(C)$ and $|C|$ are fixed, the lower bound of $P_E$ is minimized when $I(P; C)$ reaches its maximum.

Since the continuity and higher order differentiability of the partitioning function in the range space of mutual information is neither guaranteed nor adequately analyzed, a sequential search-based technique has been adopted in this chapter for alphabet size selection instead of a gradient-based optimization procedure. The process of constructing a search space is started with an initial fine grid size, where each of the grid boundaries denotes a possible cell boundary of trial partitions. These boundaries can be obtained, e.g., via either uniform partitioning (UP) or maximum entropy partitioning (MEP) [43] of the range space of the time series.

At first, the range space of time series is divided into $L$ regions that is marked by $(L-1)$ boundaries $\Upsilon$ (excluding the end points) for search via MEP or UP. The positive integer $L$, where $L > |\Sigma|$, is specified by the user. The cost function to be maximized is the scalar-valued mutual information $I(P; C)$, and the objective is to find the corresponding optimal/sub-optimal partitioning boundary in $P$. The cost is dependent on a specific partitioning $\Lambda$, because the extracted feature $P$ is a function of the chosen $(|\Sigma| - 1)$-dimensional partitioning vector $\Lambda$; the cost is denoted by $I(P(\Lambda); C)$. This sub-optimal partitioning scheme involves sequential estimation of the elements of the partitioning vector $\Lambda$.

The partitioning process is initiated by searching the optimal cell boundary that divides the data range into two cells, i.e., $\Lambda_2 = \{\lambda_1\}$, where $\lambda_1$ is optimized as:

$$\lambda_1^* = \lambda_1 \in \Upsilon \ I(P(\Lambda_2); C)$$  (4.2)

Now, the two-cell optimal partitioning is given by $\Lambda_2^* = \{\lambda_1^*, \lambda_2\}$. The next step is to partition the data range into three cells as $\Lambda_3$ by dividing either of the two cells of $\Lambda_2^*$ by placing a new partition boundary at $\lambda_2$, where $\lambda_2$ is evaluated as:

$$\lambda_2^* = \lambda_2 \in \Upsilon \setminus \Lambda_2^* \ I(P(\Lambda_3); C)$$  (4.3)

where $\Lambda_3 = \{\lambda_1^*, \lambda_2\}$. The optimal 3-cell partitioning is obtained as $\Lambda_3^* = \{\lambda_1^*, \lambda_2^*\}$. In this (local) optimization procedure, the cell that provides the largest increment in $I(P; C)$ upon further segmentation ends up being partitioned. Iteratively, this
procedure is extended to obtain the parameter $|\Sigma|$ of cell partitioning as follows:

$$\lambda_{|\Sigma|-1}^* = \lambda_{|\Sigma|-1} \in \mathcal{R} \cup \Lambda_{|\Sigma|-1}^*$$  \hspace{1cm} (4.4)$$

where $\Lambda_{|\Sigma|} = \Lambda_{|\Sigma|-1}^* \cup \{\lambda_{|\Sigma|-1}\}$ and the optimal $|\Sigma|$ is given by $\Lambda_{|\Sigma|}^* = \Lambda_{|\Sigma|-1}^* \cup \{\lambda_{|\Sigma|-1}^*\}$. In this optimization procedure, the mutual information increases monotonically with additional sequential operation provided that the mutual information is computed correctly. This property of monotonicity allows the formulation of a rule for stopping the sequential optimization algorithm. The process is stopped when the normalized mutual information, relative to $H(\mathcal{C})$ with a uniform class prior, crosses a user specified positive threshold $I_{\text{max}} \in [0, 1]$. The stopping criterion is: $\Lambda_{|\Sigma|}^*$ is the optimal partitioning and $|\Sigma|$ is the optimal alphabet size if

$$\frac{I(\mathcal{P}(\Lambda_{|\Sigma|}^*); \mathcal{C})}{H(\mathcal{C})} > I_{\text{max}}$$  \hspace{1cm} (4.5)$$

An alternative stopping criterion in Eq. (4.5) could be when the normalized mutual information gain from additional partitioning is less than a user specified positive scalar threshold $\eta_{\text{stop}}$, as stated below:

$$I(\mathcal{P}(\Lambda_{|\Sigma|}^*); \mathcal{C}) - I(\mathcal{P}(\Lambda_{|\Sigma|-1}^*); \mathcal{C}) \leq \eta_{\text{stop}}$$  \hspace{1cm} (4.6)$$

In contrast to the exhaustive search of the entire partitioning space, the computational complexity in the proposed algorithm increases linearly with $|\Sigma|$. Thus, the proposed approach would allow a finer grid size for the search with comparatively low computational complexity. Figure 4.1 provides an outline of the alphabet size selection method explained above for $|\mathcal{C}| = 2$.

Labeled time series data from the training set are fed into the “Alphabet Size Selection” block in Figure 4.1. For each possible partitioning, feature vectors are generated for each time series by symbolization and subsequent PFSA construction (represented by the “STSA feature extraction” sub-block on the left side within the “Alphabet Size Selection” block), which together with the provided class labels (in the simplex plane sub-block on the right side within the “Alphabet Size Selection” block) are then used to compute mutual information between the class and the extracted features via the Parzen window method [80]. The computed
Figure 4.1: Information-theoretic framework for time-series partitioning and alphabet size selection

Mutual information is then fed back to the “STSA feature extraction” sub-block on the left side within the “Alphabet Size Selection” block in Figure 4.1. Given the information of the mutual information for each possible partitioning at a given alphabet size, the partitioning is updated to further increase the mutual information between the class and the features by including an additional partition boundary at this step (the one resulting in highest mutual information). The process is repeated until the stopping criterion is satisfied. Then, the optimal partitioning is used to extract features in both training and testing phases. A classifier is then trained and the class labels for the testing data predicted.
4.2.1 Parzen Window based Mutual Information Estimation

This subsection explains how the Parzen window method [80, 81] is used for estimation of mutual information at each step of sequential optimization. In classification problems, the class labels associated with each feature can only have discrete values, while the input features \( \mathcal{P} \) (i.e., state probability vectors \( \mathbf{p} \) of PFSA) are usually continuously varying. Similar to its usage in Eq. (4.1), the mutual information between input feature \( \mathcal{P} \) and class \( \mathcal{C} \) becomes:

\[
I(\mathcal{P}; \mathcal{C}) = H(\mathcal{C}) - H(\mathcal{C}|\mathcal{P}) \tag{4.7}
\]

where \( H(\mathcal{C}) \) is obtained with a uniform class prior. The conditional entropy \( H(\mathcal{C}|\mathcal{P}) \) based on the input feature \( \mathcal{P} \) (that is a \( |Q| \)-dimensional random vector) is obtained as:

\[
H(\mathcal{C}|\mathcal{P}) = -\int_{\mathcal{P}|\mathcal{C}} p_{\mathcal{P}|\mathcal{C}}(\mathbf{p}) \times \sum_{i=0}^{|\mathcal{C}|-1} p_{\mathcal{P}|\mathcal{C}}(c_i|\mathbf{p}) \log_2 p_{\mathcal{P}|\mathcal{C}}(c_i|\mathbf{p}) \, d\mathbf{p} \tag{4.8}
\]

where \( |\mathcal{C}| \) is the number of classes. By applying the Bayesian rule and using the fact that \( \sum_{i=0}^{|\mathcal{C}|-1} p_{\mathcal{P}|\mathcal{C}}(c_i|\mathbf{p}) = 1 \) for any given \( \mathbf{p} \), the probability \( p_{\mathcal{P}|\mathcal{C}}(c|\mathbf{p}) \) becomes:

\[
p_{\mathcal{P}|\mathcal{C}}(c|\mathbf{p}) = \frac{p_{\mathcal{P}|\mathcal{C}}(\mathbf{p}|c) p_{\mathcal{C}}(c)}{\sum_{i=0}^{|\mathcal{C}|-1} p_{\mathcal{P}|\mathcal{C}}(\mathbf{p}|c_i) p_{\mathcal{C}}(c_i)} \tag{4.9}
\]

The Parzen window estimator at each class \( c \in \mathcal{C} \) is obtained as:

\[
\hat{p}_{\mathcal{P}|\mathcal{C}}(\mathbf{p}|c) = \frac{1}{n_c} \sum_{i \in \mathcal{I}_c} \varphi(\mathbf{p} - \mathbf{p}^i, h_c) \tag{4.10}
\]

where \( n_c \) is the number of training samples belonging to the class \( c \in \mathcal{C} \) and \( \mathcal{I}_c \) is the set of the respective indices of training samples (i.e., \( |\mathcal{I}_c| = n_c \)); \( \varphi \) is the Parzen window function; and \( h_c \) is the Parzen window width parameter for the pattern class \( c \in \mathcal{C} \). In this work, a \( d \)-variate Gaussian window (with covariance matrix \( S \)
is chosen as the Parzen density estimator.

$$\varphi(p, h) = \frac{1}{(2\pi)^{d/2} h^d |S|^{1/2}} \exp \left( -\frac{p^T S^{-1} p}{2h^2} \right)$$  \hspace{1cm} (4.11)$$

where the parameter $h$ controls the trade-off between variance and bias of the estimator. An increment in $h$ would reduce the variance at the expense of increased bias and vice-versa for a decrement in $h$. Following [80], the current work uses $h_c = \frac{1}{2 \log_e(n_c)}$ for each $c \in C$. Parzen [81] showed that the estimated density converges to the true density if $\varphi$ and $h$ are selected properly. By combining Eqs. (4.9), (4.10) and (4.11), the Parzen window estimator is constructed [80] as:

$$\hat{p}_{c|\mathcal{P}}(c|p) = \frac{\sum_{i \in C} \exp \left( -\frac{(p-p_i)^T S^{-1} (p-p_i)}{2h^2} \right)}{\sum_{k=0}^{\mid C \mid - 1} \sum_{j \in C_k} \exp \left( -\frac{(p-p_j)^T S^{-1} (p-p_j)}{2h^2} \right)}$$  \hspace{1cm} (4.12)$$

If the integration in Eq. (4.8) is replaced by summation of the sample points with equal sample probability, then the conditional entropy (based on an input feature $\mathcal{P}$) derived from the training data belonging to all classes in $\mathcal{C}$ becomes:

$$\hat{H}(C|\mathcal{P}) = -\frac{1}{n} \sum_{j=1}^{n} \sum_{i=0}^{\mid C \mid - 1} \hat{p}_{c|\mathcal{P}}(c_i|p^j) \log_2 \hat{p}_{c|\mathcal{P}}(c_i|p^j)$$  \hspace{1cm} (4.13)$$

where $n \triangleq \sum_{i=0}^{\mid C \mid - 1} n_{c_i}$ is the total number of training samples under consideration and $p^j$ is the feature vector computed from the $j^{th}$ training data in the ensemble of all classes. Finally, the estimated mutual information is obtained from Eqs. (4.12) and (4.13). For $\mid C \mid \ll n$, the computational complexity of Parzen window estimation [80] in Eq. (4.13) is of the order $n^2 \times d$; this implies that, unlike the histogram-based methods, Parzen window estimation does not require excessive memory.

### 4.2.2 Robustness

This subsection investigates robustness of classification performance to perturbations in partition locations. In this procedure, a zero-mean Gaussian noise is added
to generate samples of random boundary locations. If a large number of samples are drawn, the effect of the perturbations is realized from the statistical characteristics of the set of mutual information values corresponding to each sample. To this end, the $i^{th}$ partition boundary location is drawn from a Gaussian distribution $N(\lambda_i, \sigma_i)$ (i.e., centered at $\lambda_i$ with standard deviation $\sigma_i$). In particular, $\sigma_i$'s are chosen to be fractions of $\min(\lambda_i - \lambda_{i-1}, \lambda_{i+1} - \lambda_i)$. At each step, $M$ samples are drawn from the distribution centered at a partition location $\lambda_i$ that is not yet included in the partition set, i.e., the $j^{th}$ independent and identically distributed (iid) sample $\lambda_i^j \sim N(\lambda_i, \sigma_i)$, $j = 1, \cdots, M$. The mutual information corresponding to the features extracted after incorporating $\lambda_i^j$ into the existing partition set is obtained as $(I(\mathcal{P}(\Lambda_j^i); \mathcal{C}))$, where $\Lambda_j^i = \Lambda_{i-1}^* \cup \{\lambda_i^j\}$. The mutual information of the feature vectors resulting from adding $\lambda_i$ into the existing partition set in this work is taken to be the 95th percentile of the set of mutual information values corresponding to $M$ samples drawn from the distribution centered at $\lambda_i$, i.e.,

$$I(\mathcal{P}(\Lambda_i); \mathcal{C}) = P_{95}\{I(\mathcal{P}(\Lambda_j^i); \mathcal{C}), j = 1, \cdots, M\}$$

(4.14)

Hence, at the $i^{th}$ step, the (suboptimal) partition location $\lambda_i^*$ is obtained as:

$$\lambda_i^* = \arg \max_{\lambda_i} (I(\mathcal{P}(\Lambda_i); \mathcal{C}))$$

(4.15)

**Pattern Classification using STSA Features:** The suboptimal partitioning obtained by the above-mentioned procedure is used to construct a PFSA from each training and testing time series; the state probability vector of the PFSA is the extracted feature for each time series. In this work, two commonly used pattern classifiers, namely, k nearest neighbor (k-NN) and support vector machine (SVM) have been adopted [2, 82].

### 4.3 Experimental Validation and Results

This section presents two examples for validation of the proposed procedure.

**Example #1: Duffing System Simulation:** The exogenously excited Duffing system is nonlinear and exhibits complex behavior with chaotic and bifur-
Figure 4.2: Mutual information as a function of alphabet size $|\Sigma|$ for two-class Duffing system with (a) $D = 1$, (b) $D = 2$; for (c) four-class Duffing system with $D = 1$ and $\eta_{stop} = 0.01$. 
Figure 4.3: Misclassification error for two-class problem ($D = 1$) with variance fraction: (a) 0.67, (b) 0.5, (c) 0.25
cation properties; its governing equation is:

\[
\frac{d^2y}{dt^2} + \beta \frac{dy}{dt} + \alpha y(t) + y^3(t) = A \cos(\omega t)
\]  

(4.16)

where the amplitude \( A = 22.0 \), excitation frequency \( \omega = 5.0 \), and the initial conditions are: \( y(0) = 1.0 \) and \( \frac{dy}{dt}(0) = 0.0 \); however, these initial conditions have no significance because only the steady-state oscillatory responses have been analyzed. At first, only two classes of Duffing system are defined based on the range of \( \beta \), that are: (i) Class 1 \((0.100 \leq \beta \leq 0.147)\) and (ii) Class 2 \((0.147 \leq \beta \leq 0.194)\). Two hundred runs of the Duffing system have been simulated for each class, to generate the data set for analysis, out of which 30 samples are chosen for determining the optimal partitioning, and three-fold cross validation has been performed on the remaining data set to determine the classification performance. Parameters \( \alpha \) and \( \beta \) are chosen randomly from independent uniform distributions within the prescribed ranges. The length of the simulation time window is 80 seconds sampled at 100 Hz, which generates 8,000 data points. Thus, each time series is 8000 samples long. The range of the time-series is divided into 40 grid cells via Uniform Partitioning.

The proposed algorithm is then used to identify the optimal partitioning and alphabet size. Figure 4.2a and Figure 4.2b depict the nature of mutual information between the state probability vector and the class labels for the depth of the \( D \)-Markov machine of the input feature being \( D = 1 \) and \( D = 2 \), respectively. For \( D = 2 \), normalized mutual information converges to 1 much earlier for alphabet size \( |\Sigma| = 5 \) than that for \( D = 1 \) and \( |\Sigma| = 9 \). In each case, stopping criterion follows Eq. (4.6) with the parameter \( \eta_{\text{stop}} = 0.01 \). Figures 4.3a, 4.3b and 4.3c show the classification performance of the k-NN classifier [2] with \( k = 5 \) for three different levels of robustness, i.e., different variances that are fractions, 0.67, 0.5, 0.25, of the inter partition width, respectively. It is observed that the classification errors are smaller with smaller variance which is a consequence of smaller robustness variance fractions. Figure 4.2c shows the nature of mutual information between the state probability vector and the class labels for the Duffing system with 4 classes, corresponding to four different combinations of the ranges: \((0.100 \leq \beta \leq 0.147)\), \((0.147 \leq \beta \leq 0.194)\), \((0.934 \leq \alpha \leq 1.067)\), \((0.8 \leq \alpha \leq 0.934)\), within
which the parameters $\alpha$ and $\beta$ in Eq. (4.16) are located. The convergence rate of
the normalized mutual information is smaller in this case than that for the binary
classification scheme because a larger alphabet is required to be able to capture the
information of four classes (discriminate between the four classes). The stopping
criterion follows Eq. (4.6)) with the parameter $\eta_{stop} = 0.01$.

**Example #2: LBO Prediction in a Combustor:** Ultra-lean combustion is commonly used for NOx reduction and is susceptible to thermo-acoustic instabilities and lean blowout (LBO) [83]. It is well known that occurrence of LBO could be detrimental for operations of both land-based and aircraft gas turbine engines. In essence, a sudden decrease in the equivalence ratio may lead to LBO in gas turbine engines, which could have serious consequences. Hence early detection and accurate prediction of LBO is necessary for adequate control. The proposed procedure of time series partitioning and alphabet size selection has been evaluated under multiple operating conditions (which are airflow rates and premixing levels of fuel and air in the experimental setup analyzed here), a detailed description of which is reported in [83]. A series of experiments have been conducted on this laboratory apparatus with liquefied petroleum gas (LPG) fuel at airflow rates of 150, 175 and 200 liters per minute (lpm) for two different fuel-air premixing lengths (i.e., distance of fuel injection port from the dump plane) of $L_{fuel} = 25$ cm, and 15 cm for Port 3, and Port 5, respectively [83]. For each experiment protocol, chemiluminescence time series data were collected while reducing the fuel-air ratio $\phi$ in steps till the combustor system reached LBO. The main challenge here is to predict quantitatively how far a combustion process is from the onset of LBO in real time. It is easier to predict LBO under high premixing (i.e., port 3) as the precursor events are more dominant [83] than that under lower premixing (i.e., port 5).

A nested classification architecture [83] is proposed in accordance with the range of the non-dimensional equivalence ratio of $\phi/\phi_{LBO}$ for early detection of LBO. In the training phase, the chemiluminescence time series of duration 3 sec (at the sampling rate of 2 kHz) for each premixing length are grouped into two classes as: *Alarm* ($1 \leq \phi/\phi_{LBO} \leq 1.20$) and *Nominal* ($\phi/\phi_{LBO} > 1.20$). The class *Alarm* is subdivided into two finer classes as: *Impending LBO* (ILBO) for $1 \leq \phi/\phi_{LBO} \leq 1.1$, and *Progressive LBO* (PLBO) for $1.1 < \phi/\phi_{LBO} \leq 1.2$. Identification of the
Figure 4.4: **Top:** Mutual information as a function of alphabet size $|\Sigma|$ for Port 3 and Port 5 levels of premixing with (a) $D = 1$ and (b) $D = 2$. **Bottom:** Variation of classification error as a function of alphabet size $|\Sigma|$ for Port 3 and Port 5 levels of premixing level with (c) $D = 1$ and (d) $D = 2$.
Figure 4.5: Comparison of classification error with varying alphabet size for (Port 5) premixing: (a) $D = 1$, (b) $D = 2$

PLBO phase is critical for avoidance of LBO as control actions need to be initiated typically near the PLBO-ILBO boundary. The proposed sequential partitioning optimization scheme is started with 20 grid cells. The robustness is chosen as $\sigma$ fraction of 0.25. Figure 4.4a shows the mutual information between the $D$-Markov feature vectors with $D = 1$ and the class labels, for both premixing levels; Figure 4.4b presents a similar analysis for $D = 2$. It is observed that the normalized
mutual information converges to 1 with a much smaller alphabet size $|\Sigma|$ for $D = 2$ than that for $D = 1$, reflecting the fact that $D$-Markov features with larger memory should be able to capture the same class information with a smaller $|\Sigma|$ (better feature quality); however, the number of PFSA states for $D = 2$ could be larger than that for $D = 1$, leading to high computational complexity. It is observed that normalized mutual information for high premixing (port 3) converges to 1 for a smaller $|\Sigma|$ than that for low premixing (port 5). This phenomenon is more apparent for $D = 1$, where the alphabet size for port 3 and port 5 are chosen as $|\Sigma| = 7$ and $|\Sigma| = 12$, respectively, according to a stopping rule of $I_{max} = 1$ (see Eq. (4.5)). The reason behind this observation is attributed to large class separability for high premixing, due to the presence of dominant precursor events leading to LBO. Support vector machines (SVM) with radial basis functions [2, 82] have been used here, trained on 70% of the data set at each layer of the nested classification. Variance of the radial basis function is optimized for each layer of the nested classification via a grid search method and it is found to be 1 in most of the cases. Figures 4.4c and 4.4d depict the variations of classification error while the proposed partitioning scheme sequentially increases $|\Sigma|$ for both $D = 1$ and $D = 2$. The error bars represent the standard deviations of the classification error over 10-fold cross validation. The classification error is smaller for high premixing (port 3) than that for low premixing (port 5). It is also observed that relatively smaller classification error occurs at $D = 2$ than that at $D = 1$, especially for for small $|\Sigma|$.

**Performance Comparison:** The proposed partitioning has also been compared with that of a benchmark partitioning method, namely, maximum entropy partitioning (MEP). Figure 4.5 shows the profiles of classification error in the proposed approach and in MEP as a function of $|\Sigma|$ for the port 5 scenario. By applying the normalized mutual information based stopping criterion as mentioned earlier, the classification error is seen to be smaller for the proposed scheme at $D = 1$ and $|\Sigma| \geq 6$ in Figure 4.5a and at $D = 2$ and $|\Sigma| = 4$ in Figure 4.5b.
4.4 Summary, Conclusions & Future Work

The proposed technique addresses the issues of alphabet size selection and partitioning of time series data for symbolization of time series, by proposing an information-theoretic approach to find features from sensor data, which encapsulate a lot of information about the underlying class. Thus the objective of the proposed feature extraction algorithm is to maximize the mutual information between the input features and pattern classes in the framework of symbolic time series analysis (STSA) [20]. The proposed technique is validated on two examples: (i) simulation data for a Duffing system [84] and (ii) experimental data of chemiluminescence time series generated from a swirl-stabilized combustor [83] for lean blowout (LBO) prediction. The proposed partitioning technique yields satisfactory performance of pattern classification in several test phases. Adding an explicit term for class separability in the currently proposed objective function is a topic of future research. In addition, the following topics are recommended for future research:

- Using simultaneous optimization techniques instead of sequential ones.
- Investigation of the trade-off between the performance gain and the computational complexity.
- Validation of the proposed algorithm for other applications.
Online Estimation of State of Charge of Battery

5.1 Introduction

The state-of-charge (SOC) is a crucial parameter for operation of battery systems, which depicts the battery system’s current capacity (i.e., the maximum charge that can be drawn from its fully charged condition). Accurate estimates of SOC mitigate the risk of battery cells being over-charged and over-discharged. Many applications like plug-in electric vehicles and hybrid locomotives require large battery packs which contain several hundreds of battery cells to meet the large and dynamic power demands of the applications. From these perspectives, real-time SOC identification would enhance the efficiency of power and energy allocation within the battery packs.

Changes in the battery SOC (e.g., frequent discharge by acceleration and regeneration by deceleration in the operations of electric vehicles and locomotives) may take place over small time-windows. In such a scenario, the estimated values should be updated frequently. Therefore, a crucial evaluation factor for dynamic data-driven applications in battery state estimation, is to generate robust and accurate identification of SOC in real time with limited lengths of test data. Very few data-driven methods, available in current literature, have addressed this particular issue. The reason being that most of the data-driven methods require adequate
length of training and test data to make the extracted features accurate, which usually leads to a delayed SOC identification, a disadvantage in real time estimation problems.

Li et al. [30] have reported a dynamic data-driven method, as a feasible alternative to model-based methods, for SOC identification, by using the time series of the battery voltage. The underlying concept is built upon the theory of symbolic time series analysis (STSA) [37][43][38] that extracts the dynamic information as low-dimensional features by symbolization of sensor time series and subsequent generation of probabilistic finite state automata (PFSA). The performance of SOC identification in this context has been studied under different training and test data lengths. As an extension of [30][39], this work develops a method for SOC identification by formulating it as a pattern classification problem, making use of an ensemble of time series pairs of synchronized battery inputs (i.e., charging/discharging current) and battery outputs (i.e., voltage), for information compression and feature extraction. The uncertainties due to finite lengths of both training and testing data are modeled as Dirichlet and multinomial distributions respectively, based on [40]. The proposed method has been validated with experimental data of a (commercial-scale) lead-acid battery.

The application part of this work focuses on lead-acid batteries that are widely used in automobiles and electric locomotives [85]. Nevertheless, the proposed method can be easily extended to other battery types (e.g., lithium-ion) and identification of additional important operating parameters (e.g., state-of-health) [30]. Significant contributions of the present work are delineated below.

- Characterization of the battery dynamics based on the synchronized input-output time series via symbolic dynamic analysis: The proposed representation of (possibly nonlinear) input-output characteristics is analogous to a transfer function realization and alleviates the need for linearization of the system dynamics.

- Wavelet-based segmentation of time series for separation of the active parts (e.g., charge/discharge pulses) from the inactive parts (i.e., constant charge or discharge): While battery systems may be operated under constant load (i.e., when there are no significant fluctuations) over a time span, the active
dynamic responses of the battery can be localized based on the information derived in the time-frequency domain [86]. The proposed algorithm uses a wavelet based segmentation to separate the active parts from the inactive parts.

- Incorporation of the effects of finite-length symbol strings on pattern classification: The uncertainties due to finite-length symbol strings are quantified by Dirichlet and multinomial distribution models, respectively [40]. Experimental results have provided an insight for selection of the lengths of training and testing data to achieve the desired performance.

5.2 Definitions

This subsection introduces definitions of pertinent battery parameters at a given ambient temperature [85]. In this work, the capacity measurement has been used to calibrate the SOH at different stages of battery life.

Definition 5.2.1. (Battery capacity) The capacity $C(t)$ of a battery at time $t$ is the maximum charge (in units of ampere-hours) that can be discharged from a fully charged condition at a rate $C(t)/30$ (in units of amperes).

Definition 5.2.2. (SOH) Let a new battery be put into service at time $t_0$. The state of health $SOH(t)$ of the (possibly used) battery at the current time $t$, where $t \geq t_0$, is defined to be the ratio of the battery capacities at time epochs $t$ and $t_0$, i.e.,

$$SOH(t) = \frac{C(t)}{C(t_0)} \text{ for all } t \geq t_0 \quad (5.1)$$

Definition 5.2.3. (DOD and SOC) Let a battery be fully charged at time $t$ and let $I(\tau)$ be the applied current (in units of amperes) at time $\tau$. Then, depth of discharge (DOD) and state-of-charge (SOC) at time $t + \Delta t$ are respectively defined as

$$DOD(t + \Delta t) = \frac{1}{C(t)} \int_{t}^{t+\Delta t} I(\tau) \, d\tau \text{ for } \Delta t \geq 0 \quad (5.2)$$

$$SOC(t + \Delta t) = 1 - DOD(t + \Delta t) \text{ for } \Delta t \geq 0 \quad (5.3)$$
Remark 5.2.1. It is noted that \( SOH \in [0,1] \) for all time \( t \geq t_0 \), where \( t_0 \) is the time of putting a new battery into service. In this work, the battery is assumed to fail to meet the operational requirement, i.e., it needs to be replaced, when the SOH parameter is less than 80%.

5.3 Wavelet-based Time Series Segmentation

This section presents a wavelet based segmentation algorithm. The wavelet-based analysis provides pertinent information of the signal simultaneously in the time domain and the frequency domain [87].

For a given wavelet basis function \( \psi(t) \), the scaled and translated child wavelets are defined as [87]:

\[
\psi_{\alpha,\tau}(t) = \sqrt{\frac{1}{\alpha}} \psi\left(\frac{t - \tau}{\alpha}\right)
\]  

(5.4)

where \( \alpha \in (0, \infty) \) is the scale of wavelet transform and \( \tau \in (-\infty, \infty) \) is the time shift, and \( \psi \in L_2(\mathbb{R}) \) is such that \( \int_{-\infty}^{\infty} \psi(t)dt = 0 \) and the norm \( ||\psi|| = 1 \).

The continuous wavelet transform (CWT) of a function \( x(t) \) at a scale \( \alpha \) is represented as

\[
\tilde{x}(\alpha, \tau) = \int_{-\infty}^{\infty} \bar{\psi}_{\alpha,\tau}(t) x(t)dt
\]  

(5.5)

where \( \bar{\psi} \) indicates the complex conjugate of \( \psi \); and the distinction between \( \bar{\psi} \) and \( \psi \) vanishes for real-valued wavelet basis functions. It is also noted that the time series of a continuous-time signal \( x(t) \) is its representation in the discrete-time domain and hence the integral in Eq. (5.5) is replaced by summation over a (finite) time series \( \{x[n]\}, \, n = 1, 2, \cdots, N \) to obtain the discrete wavelet transform (DWT) at a scale \( \alpha_m \) and time shift \( \tau_\ell \) as:

\[
\tilde{x}[\alpha_m, \tau_\ell] = \sum_{n=1}^{N} \bar{\psi}_{\alpha_m,\tau_\ell}[n] x[n]
\]  

(5.6)

Every wavelet basis function can be associated with a purely periodic signal of frequency, called center frequency \( f_c \), which maximizes the Fourier transform of the wavelet modulus [88]. Consequently, the relationship among frequency points \( \varphi_m \), the associated scales \( \alpha_m \) for a given wavelet basis function and a sampling
period $\Delta$ of the time series is given as:

$$\alpha_m = \frac{f_C}{\varphi_m \Delta} \quad (5.7)$$

where $\Delta$ is the sampling period of the analyzed signal. The wavelet basis function for analysis is selected based on the time-frequency characteristics of analyzed signals.

For a time series $\{x[n]\}$, $n = 1, 2, \ldots, N$, the discrete Fourier transform is obtained as:

$$\hat{x}[k] = \sum_{n=1}^{N} e^{-2\pi i kn/N} x[n] \quad \text{for } k = 1, 2, \ldots, N \quad (5.8)$$

Remark 5.3.1. In general, the Fourier transform $\hat{x}[k]$ of a signal is a complex number but its power spectral density $|\hat{x}[k]|^2$ is a non-negative real number.

Following Parseval’s theorem, the energy of the time series $x[n]$ can be expressed in the discrete-time and discrete-frequency domains [89] as:

$$\sum_{n=1}^{N} |x[n]|^2 = \sum_{k=1}^{N} |\hat{x}[k]|^2 \quad (5.9)$$

Since the summand on the right hand side of Eq. (5.9) can be interpreted as a probability histogram describing the energy distribution of the signal at frequency points $k$, the energy spectral density of a signal $x[n]$ is defined as:

$$S_x[k] = |\hat{x}[k]|^2 \quad \forall k \in \{1, 2, \ldots, N\} \quad (5.10)$$

The steps of the wavelet-based segmentation procedure are as follows.

- Step 1: Collection of (finite) time series data $x[n]$, $n = 1, 2, \ldots, N$ for a given sampling interval $\Delta$. It is noted that the length $N$ of the time series is user-selectable.

- Step 2: Computation of the discrete Fourier transform $\hat{x}[k]$, $k = 1, 2, \ldots, N$ and the corresponding PSD $S_x[k]$, $k = 1, 2, \ldots, N$. This step follows Equations. (5.8) and (5.10)).
Step 3: Identification of the frequency points of interest, \( \varphi_m \), \( m = 1, \cdots, M \).
In this step, the set \( \{ S_x[\varphi_m] \} \) is formed in terms of the \( M \) points with highest
values of the PSD \( S_x[k], k = 1, 2, \cdots, N \). [Note: \( M \) is usually significantly
less than \( N \)].

Step 4: Computation of the corresponding wavelet scales \( \alpha_m \), \( m = 1, 2, \cdots, M \).
This step follows
Eq. (5.7) with the corresponding frequency points \( \varphi_m \) and the central fre-
quency \( f_c \) of the chosen wavelet basis function.

Step 5: Identification of the set of segmented time indices \( \Gamma^m \) for each scale
\( \alpha_m \). In this step, the set of time shift indices \( \Gamma^m \subset \{ \tau^m_\ell \}_{\ell=1}^N \) are selected
as the top \( \xi_T \) fraction of the values of the wavelet coefficients \( \tilde{x}_{\alpha_m}(\tau^m_\ell) =
\tilde{x}[\alpha_m, \tau^m_\ell], \ell = 1, 2, \cdots, N \) at a scale \( \alpha_m \).

Step 6: Identification of the set, \( T \), of all segmented time indices. This
step is completed by taking the union of the sets of \( \Gamma^m \)'s among all scales
\( \alpha_m, m = 1, 2, \cdots, M \) as \( T = \bigcup_{m=1}^M \Gamma^m \).

5.4 Maximum Entropy Partitioning and
Symbolization

This section addresses the partitioning in the input-output space of single-input
single-output (SISO) systems for symbolization of 2-dimensional time series data.
Time series of input-output pairs are partitioned into a mutually exclusive and
exhaustive set of finitely many segments, where a symbol string is generated by
assigning a unique symbol to each segment of the input-output space.

Quasi-stationarity of SISO systems is assumed such that the system behavior
is statistically stationary at the fast time scale of the process dynamics, while
there exists observable non-stationary behavior evolving at a slow time scale. The
notion of two time scales and their significance are discussed in the context of
symbolization in \[38\].

In this work, the maximum-entropy partitioning (MEP) \[43\] of the time-series
data has been adopted to construct the symbol alphabet \( \Sigma \) and to generate symbol
Algorithm 2 Maximum Entropy Partitioning for 2-dimensional Time Series

Require: A 2-dimensional string $X[n] = [x_1[n] \ x_2[n]]$ for $n = 1, 2, 3, \ldots, N$ of synchronized and normalized time series data set; Alphabet size $|\Sigma_1|$ for the first coordinate of the 2-dimensional data and alphabet size $|\Sigma_2|$ for the second coordinate of the 2-dimensional data.

Ensure: Partition vector $\varphi_1 \in \mathbb{R}^{|\Sigma_1|+1}$ for first coordinate data; Partition Matrix $\varphi_2 \in \mathbb{R}^{|\Sigma_1| \times (|\Sigma_2|+1)}$ for the 2-dimensional data set.

Assign $\varphi_1(1) = -\infty$, i.e., the minus infinity
Assign $\varphi_2(m, 1) = -\infty$, for $m = 1, 2, \cdots, |\Sigma_1|$
Assign $\varphi_1(|\Sigma_1| + 1) = \infty$, i.e., the positive infinity
Assign $\varphi_2(m, |\Sigma_2| + 1) = \infty$, for $m = 1, 2, \cdots, |\Sigma_1|$

Sort the data string $x_1$ in the ascending order as $x_1^s$
Let $K = \text{length}(x_1^s)$
for $i = 1$ to $|\Sigma_1|$ do
    if $i \neq |\Sigma_1|$ then
        $\varphi_1(i + 1) = x_1^s \left \lceil \frac{i \times K}{|\Sigma_1|} \right \rceil$
    end if
Define $x_1^i \triangleq \{x_2[n] | \varphi_1(i) < x_1[n] < \varphi_1(i + 1)\}$
Sort the data string $x_2$ in the ascending order as $x_2^{is}$
Let $L = \text{length}(x_2^{is})$
for $j = 1$ to $|\Sigma_2| - 1$ do
    $\varphi_2(i, j + 1) = x_2^{is} \left \lceil \frac{j \times L}{|\Sigma_2|} \right \rceil$
end for
end for

strings. In this partitioning, the information-rich regions of the data set are partitioned finer and those with sparse information are partitioned coarser to maximize the Shannon entropy of the generated symbol string from the reference data set.

In this work, a pair of time series that represent input and output data is partitioned in the associated two-dimensional space to construct a symbolic string. Four alternative types of partitions have been used in the input-output space:

- **Partition Type 1** (Cartesian coordinates): First partition in the input axis (e.g., abscissa), and then partition in the output axis (e.g., ordinate) at individual input segments.

- **Partition Type 2** (Cartesian coordinates): First partition in the output axis,
Partitioning of Input-Output
Time series

\[ \cdots \gamma \delta \gamma \beta \beta \alpha \beta \beta \gamma \gamma \delta \cdots \]

Symbol Sequence

Probabilistic Finite State Automata (PFSA)

Figure 5.1: Construction of finite state automata (FSA) from time series pair and then partition in the input axis at individual output segments.

- Partition Type 3 (Polar coordinates): First partition in the magnitude, and then partition in the phase at individual magnitude segments.

- Partition Type 4 (Polar coordinates): First partition in the phase, and then partition in the magnitude at individual phase segments.
Figure 5.2: The flow chart for the proposed method of feature extraction

Figure 5.1 depicts the underlying concept of symbolization of a 2-dimensional time series, where each segment in the top plot is labeled by a unique symbol. This mapping is called symbolic dynamics as it attributes a (physically admissible) symbol string to the dynamical system starting from an initial state.

5.5 Online Pattern Classification

This section presents a general framework for online pattern classification in the symbolic domain; the patterns are constructed from (finite-length) symbol strings as probabilistic finite state automata (PFSA) with (possibly) diverse algebraic parameters (e.g., alphabet size $|\Sigma|$ and state cardinality $|Q|$). The uncertainties due to the finite length of the symbol string in both training and testing phases, which could influence the final classification decision, are quantified using the concepts of (a priori) Dirichlet and (a posteriori) multinomial distributions [90]. While the details are reported by Wen et al. [40] and Sarkar et al. [91] reported an application in fault detection, the essential concepts are succinctly summarized below for completeness of the work.

Let there be $L$ classes of interest, denoted by $C_1, C_2, \ldots, C_L$, and let the same symbol alphabet $\Sigma$ be used for each corresponding symbolic system. During the training phase, a symbol string $S^i \triangleq s_1^i s_2^i \ldots s_{N_i}^i$ of (finite) length $N_i$ is obtained
from each class $C_i$. Then, a PFSA $K^i = (Q^i, \Sigma, \delta^i, \pi^i)$ is obtained for each class, whose structures (i.e., $Q^i$ and $\delta^i$) may not necessarily be the same [40].

In the training phase, the distribution of each row $\Pi^i_m (m = 1, 2, \ldots, |Q|)$ of the emission probability matrix $\Pi^i$ is modeled as a Dirichlet distribution conditioned on a symbol string $S^i$:

$$f_{\Pi^i_m|S^i}(\theta^i_m) \sim \text{Dirichlet}(\mathbf{N}^i_m + 1) \quad (5.11)$$

where $\theta^i_m$ is a realization of the random vector $\Pi^i_m$ as:

$$\Pi^i_m = \begin{bmatrix} \Pi^i_{m1} & \Pi^i_{m2} & \ldots & \Pi^i_{m|\Sigma|} \end{bmatrix}$$

and the vector $\mathbf{N}^i_m$ in Eq. (5.11) is generated as:

$$\mathbf{N}^i_m = \begin{bmatrix} N^i_{m1} & N^i_{m2} & \ldots & N^i_{m|\Sigma|} \end{bmatrix}$$

By the Markov property of the PFSA $K^i$, the $(1 \times |\Sigma|)$ row-vectors, $\{\Pi^i_m\}, m = 1, \ldots, |Q|$, are statistically independent of each other. Therefore, the a priori density $f_{\Pi^i|S^i}$ of the emission probability matrix $\Pi^i$, conditioned on the symbol string $S^i$, is given as

$$f_{\Pi^i|S^i}(\theta^i) = \prod_{m=1}^{|Q|} f_{\Pi^i_m|S^i}(\theta^i_m) = \prod_{m=1}^{|Q|} \left( (\mathbf{N}^i_m + |\Sigma| - 1)! \prod_{n=1}^{|\Sigma|} \frac{\theta^i_m N^i_{mn}}{(N^i_{mn})!} \right) \quad (5.12)$$

where $\theta^i \triangleq \left[ (\theta^i_1)^T \ (\theta^i_2)^T \ \ldots \ (\theta^i_{|Q|})^T \right]^T \in [0, 1]^{Q \times |\Sigma|}$. The details of derivation of Eq. (5.12) can be found in [40].

In the testing phase, let $\tilde{N}^i_{mn}$ be the number of times the symbol $\sigma_n$ is emanated from the state $q^i_m \in Q^i$ in the test symbol string $\tilde{S}$. The probability mass function $\Pr(\tilde{N}^i_m|\Pi^i_m)$ of the random row-vector $\tilde{\mathbf{N}}^i_m$ is modelled as a multinomial distribution conditioned on a given state and $\Pi^i_m$:

$$\tilde{\mathbf{N}}^i_m = \begin{bmatrix} \tilde{N}^i_{m1} & \tilde{N}^i_{m2} & \ldots & \tilde{N}^i_{m|\Sigma|} \end{bmatrix} \sim \text{Multi}(\Pi^i_m) \quad (5.13)$$
and the conditional probability is

\[
P_r(\tilde{N}_m^i|\Pi_m^i) = (\tilde{N}_m^i)! \prod_{n=1}^{[\Sigma]} \frac{(\Pi_m^{in})^{\tilde{N}_{mn}^i}}{\tilde{N}_{mn}^i}!
\]

(5.14)

where \(\tilde{N}_m^i \triangleq \sum_{n=1}^{[\Sigma]} \tilde{N}_{mn}^i\) and \(\Gamma(\bullet)\) is the standard gamma function with \(\Gamma(n) = (n - 1)!\) \(\forall n \in \mathbb{N}_1\). The details of derivation of Eq. (5.14) can be found in [40].

Similar to the argument provided in the training phase, all row vectors in the emission matrix for the testing phase are also statistically independent of each other. Therefore, the probability of observing \(\tilde{S}\) conditioned on the emission probability matrix \(\Pi^i\) is given as:

\[
Pr(\tilde{S}|\Pi^i) \triangleq \prod_{m=1}^{[Q]} Pr(\tilde{N}_m^i|\Pi_m^i)
\]

(5.15)

\[
= \prod_{m=1}^{[Q]} \left( (\tilde{N}_m^i)! \prod_{n=1}^{[\Sigma]} \frac{(\Pi_m^{in})^{\tilde{N}_{mn}^i}}{\tilde{N}_{mn}^i}! \right)
\]

(5.16)

The results, derived in the training phase and the testing phase, are now combined. Given a symbol string \(S^i\) in the training phase, the probability of observing a symbol string \(\tilde{S}\) in the testing phase is

\[
Pr(\tilde{S}|S^i) = \prod_{m=1}^{[Q]} \frac{\tilde{N}_m^i)! (N_m^i + [\Sigma] - 1)!}{\tilde{N}_m^i + N_m^i + [\Sigma] - 1)!} \times \prod_{n=1}^{[\Sigma]} \frac{(\tilde{N}_{mn}^i + N_{mn}^i)!}{(\tilde{N}_{mn}^i)! (N_{mn}^i)!}
\]

(5.17)

The details of derivation of Eq. (5.17) can be found in [40].

The posterior probability of the observed symbol string \(\tilde{S}\) belonging to the class
Figure 5.3: Flow chart of the proposed feature extraction algorithm

\( C_i \), denoted as \( \Pr(C_i|\tilde{S}) \) is given as

\[
\Pr(C_i|\tilde{S}) = \frac{\Pr(\tilde{S}|S^i) \Pr(C_i)}{\sum_{j=1}^{L} \Pr(S|S^j) \Pr(C_j)}, \quad i = 1, 2, \cdots, L
\]  

(5.18)

where \( \Pr(C_i) \) is the known prior distribution of the class \( C_i \). Then, the classification decision is made as follows.

\[
D_{\text{class}} = \arg \max_i \Pr(C_i|\tilde{S})
\]

\[
= \arg \max_i \left( \Pr(\tilde{S}|S^i) \Pr(C_i) \right)
\]  

(5.19)

If no prior information on \( \Pr(C_i) \) is available, it is logical to assume a uniform distribution over the classes. In that case, the rule of classification decision becomes

\[
D_{\text{class}} = \arg \max_i \Pr(\tilde{S}|S^i)
\]  

(5.20)

Remark 5.5.1. If the information of \( N_{mn}^i \)'s and \( \tilde{N}_{mn}^i \)'s are available, no other information is needed to obtain the statistics of the symbol strings \( S^i \)'s and \( \tilde{S} \). Therefore, \( N_{mn}^i \)'s and \( \tilde{N}_{mn}^i \)'s are sufficient statistics of \( S^i \)'s and \( \tilde{S} \), respectively.

The flow chart in Fig. 5.3 summarizes the feature extraction portion of the SOC classification method. First, the raw current and voltage time series data are normalized to make them zero-mean and unit-variance. Second, the segments in the current and voltage time series, that contain dynamic information of the system are extracted from the data, using the concept of wavelet-based segmentation in
Figure 5.4: Flow chart of the proposed method training and testing phases

the time-frequency domain. Third, maximum entropy partitioning (MEP) [43] is adopted to partition the range space of the time series pair. This results in a single symbol string from each time series pair. STSA, as explained in Chapter 2 is then done to extract features from each symbol string.

Fig. 5.4 gives an overview of the online pattern classification scheme. The proposed online classification method consists of an off-line training phase and an online testing phase. In the offline training phase, probabilistic finite state automata (PFSA) \( K^i, i = 1, 2, \ldots, L \), are constructed from the symbol strings \( S^i \) for every class \( i \). Then, the probability density function \( f_{\Pi^i|S^i} \) of the emission probability matrix \( \Pi^i \), conditioned on the symbol string \( S^i \), is fitted as a Dirichlet distribution [92] based on the emission counts \( \{N^i_{mn}\} \). In the testing phase, given an extracted symbol string \( \tilde{S} \) of finite length, (after normalization and wavelet segmentation of the raw current-voltage time series pair), using the PFSA structure \( K^i \) (i.e. the states \( Q^i, \Sigma^i \)), the associated emission counts \( \{\tilde{N}^i_{mn}\} \) are computed for each class \( i \). Then, the probability \( \Pr(\tilde{N}^i|\Pi^i) \) of observing the test symbol string \( \tilde{S} \) conditioned on a given state \( Q^i \) and emission matrix \( \Pi^i \) is modelled as a multinomial distribution [92]. Thus, the probability \( \Pr(\tilde{S}|S^i) \) of observing a symbol string \( \tilde{S} \) in the testing phase given a symbol string \( S^i \) in the training phase is computed using \( \Pr(\tilde{N}^i|\Pi^i) \) and \( f_{\Pi^i|S^i} \) (see Eq. (5.17)). A classification decision is then made by choosing the class, corresponding to the maximum posterior probability \( \Pr(C_i|\tilde{S}) \) (see Eq. (5.18)).
5.6 Experimental Validation

This section validates the algorithm of battery parameter identification with an ensemble of experimental data that have been collected from a commercial-scale lead-acid battery.

5.6.1 Data Acquisition and Processing

A fresh (12V AGM VRLA with 56Ah capacity) lead-acid battery has been used in the experiments. The battery was charged/discharged according to given input (current) profiles at room temperature and an ensemble of synchronized time-series of the input charge/discharge current and output voltage responses has been collected at the sampling frequency of 1 Hz. A typical input current profile for this experiment is shown in Fig. 5.5.

The input profile are repeated “hotel-pulses” cycles. Each individual “hotel-pulses” cycle (i.e., duration of \( \sim 120 \) s) consists of a “hotel” load (i.e., relatively steady discharge due to “hotel” needs like lighting and other electrical equipments) and a discharge pulse followed by a charge (i.e., regeneration) pulse [93], as shown in Fig. 5.5. The amplitude of the “hotel” load and the discharging & charging pulses are numerically fluctuating in the experiment, which made each cycle different from others. This pattern of input cycles largely simulates a real-time working condition for an electric locomotive.

Remark 5.6.1. In many instances of industrial applications, the designer may not have the detailed knowledge of the anticipated load profile to which the battery system will be subjected. This procedure of battery testing under a “hotel-pulses” load is analogous to pseudo-random excitation [94] that is a standard practice for system identification of electromechanical systems in various industrial applications when testing in the actual operating environment may not be feasible. The rationale for this procedure is that a “hotel-pulses” signal could be viewed as a combination of different types of excitation signal that the battery could be subjected to. Since the proposed algorithm is executed on the synchronized pair of input (current) and output (voltage) time series, testing in an industrial setting is expected to yield satisfactory results.
This experiment has been conducted over a wide range of SOC at different battery aging stages (i.e., different values of the battery state-of-health (SOH)). Table 5.1 presents the coverage of battery operating conditions within the operating range of the experiment.

The raw time series of input current and output voltage are individually normalized, followed by wavelet-based segmentation based on the time series of output voltage. While segmentation extracts the relevant segments of normalized data based on the information of their frequency content, normalization involves time

Table 5.2: SOC range for different classes

<table>
<thead>
<tr>
<th>Class Index</th>
<th>SOC Range</th>
<th>Number of Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.53 ~ 0.63</td>
<td>230</td>
</tr>
<tr>
<td>2</td>
<td>0.63 ~ 0.68</td>
<td>232</td>
</tr>
<tr>
<td>3</td>
<td>0.68 ~ 0.72</td>
<td>232</td>
</tr>
<tr>
<td>4</td>
<td>0.72 ~ 0.76</td>
<td>232</td>
</tr>
<tr>
<td>5</td>
<td>0.76 ~ 0.80</td>
<td>231</td>
</tr>
<tr>
<td>6</td>
<td>0.80 ~ 0.84</td>
<td>232</td>
</tr>
<tr>
<td>7</td>
<td>0.84 ~ 0.88</td>
<td>232</td>
</tr>
<tr>
<td>8</td>
<td>0.88 ~ 0.99</td>
<td>233</td>
</tr>
</tbody>
</table>
translation and (down or up)-scaling of the raw data for conversion into zero-mean unit variance time series as:

\[ x[n] = \frac{x_{\text{raw}}[n] - \mu_N[n]}{\sigma_N[n]} \]  

(5.21)

where \( \mu_N[n] \) is the mean value and \( \sigma_N[n] \) is the standard deviation of the time series over a time span of \( N \) data points centered at the time index \( n \). The experimental data of each duty cycle (i.e., both input current and output voltage) are normalized in this moving average fashion. First, the data set is smoothed under moving average with a shift window of 240 s (i.e., 240 consecutive data points). Then, each element of that data set is divided by its own standard deviation. Thus, the time series now is zero-mean and unit-variance.

Figure 5.7 exemplifies typical profiles of normalized and segmented input-output pairs at different charging conditions. It is observed that the dynamic responses under the same input current pattern are significantly different at different SOC levels. Figure 5.8 depicts the symbol strings generated corresponding
to four different types of partitioning schemes, where an identical alphabet size of 4 was chosen for all (i.e., $|\Sigma|_{in} = |\Sigma|_{out} = 4$ and $|\Sigma|_{magnitude} = |\Sigma|_{phase} = 4$, and total alphabet size $|\Sigma| = 4 \times 4 = 16$). It is observed from Fig. 5.8 that the symbol strings generated from different partitioning schemes are different as the symbols represent different regions in the input-output space.

5.6.2 Results and Discussion

This section presents the performance of the proposed SOC identification scheme. The parameters for feature extraction (via STSA analysis) and pattern classification used to generate the results are presented below.

- The partition type I in Cartesian coordinates is applied. The alphabet size for input is $|\Sigma|_{in} = 3$ and alphabet size for output is $|\Sigma|_{out} = 5$, and the total alphabet size is $|\Sigma| = |\Sigma|_{out} \times |\Sigma|_{in} = 15$.

- The depth in the $D$-Markov machine is set at $D = 1$, which implies that $|\Sigma| = |Q|$.

- The training/testing data are collected at $SOH = 1$ (new) and the SOC range of $0.53 \sim 0.99$.

- The number of SOC classes assigned for classification problem is $NC_{SOC} = 8$. The hotel-pulses cycles are equally assigned to each class. The SOC range for each class is demonstrated in Table 5.2.

- The length of training data is $L_{train} = 5000$.

Since the number of “hotel-pulses” cycles are required to be the same for each class, the SOC range for each class may differ from each other (see Table 5.2) due to uneven data collection at different SOC range.

Figure 5.9a exhibits correct classification even for very short data lengths (e.g., $\sim 150$), which implies that the algorithm is able to correctly predict the class after observing test data that are of very short length. Figure 5.9b shows a more general scenario, where modestly longer (e.g., $\sim 200$) test data are needed for correct classification. Figure 5.9c portrays a worst case scenario, which shows the
need of having significantly longer test data to achieve acceptable levels of accurate classification.

The performance of any pattern classification scheme depends strongly on the dataset under consideration. Hence, in order to evaluate the true performance of the proposed scheme, multiple random combinations of training/testing data sets have been created from the ensemble of experimental data, and all the results averaged. For each class at corresponding SOC range, 50 training/testing sets have been created. The average misclassification rate for SOC identification at different lengths of test data is presented below. (Note that, under different situations, same parameters settings and the operating conditions have been used as mentioned
(a) Example 1 at $SOC = 0.86$
(b) Example 2 at $SOC = 0.71$
(c) Example 3 at $SOC = 0.80$

Figure 5.9: Profiles of classification accuracy at different values of SOC

Figure 5.10: Cross validation with different output alphabet size $|\Sigma|$ above unless any changes are specifically mentioned.)

Figure 5.10 presents the results under partition type 1 with varying alphabet size $|\Sigma|_{out}$ for output while the alphabet sizes of the input are held fixed at $|\Sigma|_{in} = 3$. As $|\Sigma|_{out}$ is increased, dynamic characteristics of the output are expected to be captured to a larger extent through the symbolization process, which leads
to a reduction of the misclassification rate as seen in plots of Fig. 5.10. The misclassification rate with $|\Sigma|_{out} = 6$ is approximately half the value with $|\Sigma|_{out} = 3$ when the length of the tested data is of 400 and larger. The classification accuracy is also improved as more testing data is used, which is expected intuitively. The improvement in performance due to increased length of test data is more significant for the first 10 cycles of observed test data.
Figure 5.11 presents the results under different partition types. These results have been generated for alphabet sizes assigned as: $|\Sigma_{in}| = |\Sigma_{out}| = 4$ for partition types 1 and 2; and $|\Sigma_{magnitude}| = |\Sigma_{phase}| = 4$ for partition types 3 and 4. The misclassification rates for all four partition types become increasingly similar to each other as the length of test data is increased. It is noted that partition type 1 presented in Fig. 5.11 results in better performance as compared to those in Fig. 5.10 for $|\Sigma_{out}|$ in the range of 3 to 5. Increasing the size $|\Sigma_{in}|$ of the input alphabet also improves the classification accuracy.

Figure 5.12 presents the results for different lengths of training data. For each training/testing combination in cross validation, training data are randomly selected from all available data for each class. The results present the misclassification rate averaged over 50 training/testing sets. As the training data length is increased, the average misclassification rate is reduced, as expected. It is also observed that the improvement in accuracy by increasing the length of training data beyond 4000 is insignificant.

Figure 5.13 presents the results of cross validation at different battery aging stages (i.e., different SOH values). The results show that the proposed method has consistent performance at different battery aging stages, except for the noticeable difference when the battery is quite aged (i.e., SOH=0.80). This proves the reliability of the predicted performance at different battery operating conditions, which is determined by two major factors: SOC and SOH.

Figure 5.14 presents the results of cross validation for different number of total SOC classes. As the number of classes is increased, the average resolution of the SOC range for each class becomes finer, as shown in Table 5.4. Keeping the length of training data fixed, as the number of SOC classes is increased, it is possible to use shorter test data for each class. However, for a larger number of SOC classes, the misclassification rate tends to increase significantly and the convergence rate of the algorithms becomes slower.

### 5.6.3 Computational costs

This subsection presents the computational costs (i.e., execution time) of the SOC classification algorithm. In this work, all results have been generated on a single
Table 5.3: Execution time for test data with a length of 1000 under different conditions

| $|\Sigma|_{in} \times |\Sigma|_{out}$ | 5  | 6  | 7  | 8  | 9  | 10 |
|--------------------------------|----|----|----|----|----|----|
| $3 \times 3 = 9$               | 10.11 | 12.21 | 14.23 | 15.89 | 17.82 | 20.44 |
| $3 \times 4 = 12$              | 13.73 | 16.35 | 19.06 | 21.49 | 24.06 | 26.58 |
| $3 \times 5 = 15$              | 17.70 | 21.01 | 24.51 | 27.79 | 31.27 | 34.40 |
| $3 \times 6 = 18$              | 22.76 | 26.46 | 30.87 | 35.20 | 39.71 | 43.82 |

Li et.al. [39] addressed the issue of computation cost of a similar feature extraction tool with a much longer data set. It shows that the proposed STSA tool for online feature extraction requires very small execution time and memory. Since the training phase is conducted off-line and the length of training data for SOC identification is much less than that for SOH identification in [39], the computation cost of the proposed SOC identification does not have a major significance in the training phase.

In the testing phase, when a new input-output pair is acquired, the posterior probabilities for all classes need to be updated in real time, in order to decide which class it belongs to. For example, if the sampling frequency is 1 Hz in the experiment, then the execution time of the the online algorithm should be in the order of a fraction of a second. There are two major parameters in the feature extraction process that may affect the execution time: (i) the number of SOC classes $NC_{SOC}$ and (ii) total alphabet size $|\Sigma|$. Table 5.3 presents the required execution time for analyzing test data with a length of 1000 under different combinations of the two parameters, $NC_{SOC}$ and $|\Sigma|$. The proposed method is capable of analyzing test data of length 1000 well within 50 seconds for the combinations shown in Table 5.3, so that the posterior probabilities update can be done easily for a sampling frequency as high as 20 Hz. This observation satisfies the requirement of online execution for a vast majority of practical applications.
5.7 Summary, Conclusions, & Future Work

The proposed method of pattern classification for battery state-of-charge (SOC) identification is built upon the concept of symbolic time series analysis (STSA) [37]. In this setting, time series of synchronized pairs of battery input (i.e., current) and output (i.e., voltage) have been analyzed and D-Markov machine modelling used for feature extraction. The online battery SOC identification is formulated as a pattern classification problem. Dirichlet and multinomial distributions are used to quantify the effect of less training and testing data lengths. The performance of the proposed method has been validated on experimental data of a commercial scale lead-acid battery. The pertinent conclusions drawn from the work reported here are summarized below.

- Symbolic Time Series Analysis (STSA), as a low-complexity feature extraction tool, is capable of real-time execution on in-situ computational platforms (e.g., sensor nodes of battery cells). It provides a computationally efficient and electrochemistry-independent method of identifying the battery SOC parameter as an alternative to physics-based modeling analysis.

- Extracted STSA features capture the information, embedded in the input-output time series, for SOC identification. The underlying software can be implemented on a large scale package of battery cells.

- The proposed identification method provides good and consistent performance with short-length test data under different operating conditions.

Some topics of future research are delineated below.

- Usage of D-Markov machines [38] with $D \geq 1$ to accommodate longer memory of synchronized symbolic input-output time series.

- Extension of the proposed method to achieve robust performance for changing patterns of the input profiles (e.g., for stochastic nature of the charging and discharging current inputs) in actual operating environments.

- Investigation of the impact of temperature changes on battery dynamics for SOC identification.
Table 5.4: Average SOC resolution for different class assignments

<table>
<thead>
<tr>
<th>Number of Classes</th>
<th>Average SOC Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.082</td>
</tr>
<tr>
<td>6</td>
<td>0.068</td>
</tr>
<tr>
<td>7</td>
<td>0.058</td>
</tr>
<tr>
<td>8</td>
<td>0.051</td>
</tr>
<tr>
<td>9</td>
<td>0.045</td>
</tr>
<tr>
<td>10</td>
<td>0.041</td>
</tr>
</tbody>
</table>

- Validation of the proposed method for other types of batteries (e.g., Li-ion and Ni-MH) as well as for different discharge or charge cycle patterns.
Chapter 6

Data-driven Design of Combustors for Thermoacoustically stable operations

6.1 Introduction

This chapter develops a novel method of data driven combustor design. Data generated from limited number of runs under some operating conditions in a laboratory setup have been used to generate the knowledge base about the system dynamics, which has then been used for designing the combustor system. The proposed design method only needs limited amounts of process data in the form of time series and does not require any detailed knowledge of the underlying combustion dynamics. Given the information in the form of time series data at certain operating conditions, a Bayesian nonparametric statistical approach has been taken to predict the system behavior for operating conditions at which data may not be available. In addition, the algorithm also quantifies the confidence in the estimate of the system response. The design algorithm produces a mapping from a set of operating conditions to that of stability regions. Once this map is generated, combustor designers can use it for predicting the system response at operating conditions for which experimental data may not be available and also statistically quantify the uncertainties in the estimate. This information facilitates the iden-
tification of the combustion system parameters, which will allow the design to be statistically significant in terms of satisfying the system specifications. The objective here is prevention of combustion instabilities in order to prevent serious structural damage in the combustor. Hence, the user must design a combustor with a smaller probability of becoming unstable, which translates to the following two requirements:

1. *Extraction of a feature* that is highly sensitive to deviations in the underlying state of the system from the nominal state. Then, even a small deviation in the system behavior from the nominal state, would be manifested as a large feature divergence.

2. *Quantification of the uncertainty* in the estimate of the system stability for unknown parameters (e.g., combustor length). This would help the user to make the parameter choice, which would have a significantly reduced probability of resulting in an unstable system.

From the above perspectives, major contributions and innovations of the work in this chapter are summarized below:

1. Development of a data-driven method for combustor design, based on symbolic time series analysis (STSA) [37, 20], which satisfies the above two requirements.

2. Validation of the above method on experimental data from a swirl-stabilized combustor apparatus [95].

### 6.2 Description of the Experimental Apparatus

A swirl-stabilized, lean-premixed, laboratory-scale combustor has been used for validation of the proposed algorithm with experimental data. Figure 6.1 depicts a schematic diagram of the variable-length combustor apparatus [95], consisting of an inlet section, an injector, a combustion chamber, and an exhaust section. There is an optically-accessible quartz section followed by a variable-length steel section.

High pressure air is delivered to the apparatus from a compressor system after passing through filters to remove any liquid or particles that might be present. The
air supply pressure is set to 180 psig (1.338 MPa) using a dome pressure regulator. The air is pre-heated to a maximum temperature of 250 degree C by an 88 kW electric heater. The fuel used in this experiment is natural gas (approximately 95% methane) which is supplied to the system at a pressure of 200 psig (1.475 MPa). The flow rates of the air and natural gas are measured by thermal mass flow meters. The desired equivalence ratio and mean inlet velocity are set by adjusting the flow rates with needle valves.

Synchronized pressure time series have been collected under different operating conditions, by varying the inlet velocity, equivalence ratio and combustor length. Figure 6.2 displays typical profiles of pressure oscillations over a time window of 30 milliseconds to unambiguously present the distinctive characteristics of stable and unstable signals around their respective mean values.

Figure 6.1: Schematic diagram of the combustion apparatus

### 6.3 Description of the proposed design algorithm

The first step in the proposed data-driven algorithm for combustor design involves feature extraction from the pressure time series collected under different operating conditions. The features encapsulate the temporal dynamics of the combustion system. The next step involves inferring the relationship between the operating conditions and the corresponding system response, represented here, in the form of a function of features extracted from the corresponding pressure time series. Then, using the inferred relation, the distribution of the system response is predicted for each unseen operating condition. Therefore, the user/combustor designer now has at his/her disposal the ensemble of information, representing the estimated
system response at every point in the space of operating conditions. Using this information, the user can then determine the parameters of the combustor to be constructed, which would exhibit the desired response. Chapter 2 explains the concept of Symbolic Time Series Analysis, which was used for feature extraction here. Subsection 6.3.1 describes the underlying theory of Bayesian nonparametric regression (i.e., Gaussian Process(GP) regression) used for determining the relation between operating conditions and system response in detail. Subsection 6.3.2 develops the combustor design methodology.

6.3.1 Gaussian Process Regression

This subsection describes the technique used for determining the relation between operating conditions and the system responses as a function of the extracted features. This relation is then used for predicting the responses for operating conditions for which experiments have not been conducted.
Given a set of operating conditions and the corresponding continuous-valued system responses, there exist several regression algorithms in the machine learning literature (e.g., [2]), which infer the underlying relation between the conditions and response, under different assumptions on the characteristics of the relation. The inferred relation can then be used to predict the response of an unseen system condition (i.e., whose response is unknown). In contrast, Gaussian process (GP) regression [96] is a nonparametric method that can model arbitrary relations between the condition and response without making any specific assumptions on the relation. In addition, most regression algorithms only provide point estimates of the response, but they do not quantify the confidence in that estimate. Being a Bayesian method, GP also quantifies the uncertainties in the predictions resulting from possible measurement noise and errors in the parameter estimation procedure. Hence, GP regression has been adopted in this work to infer the relation between the operating conditions and the corresponding system response. This subsection succinctly presents the underlying theory of Gaussian process (GP) regression. Further details are available in standard literature (e.g., [96]).

A stochastic process is a collection of random variables, \( \{\xi(t) : t \in T\} \), where \( T \) is an index set. A Gaussian process is a stochastic process such that any finite collection of random variables has a multivariate jointly Gaussian distribution. In particular, a collection of random variables \( \{\xi(t) : t \in T\} \) is said to be drawn from a Gaussian process with mean function \( m(\cdot) \) and covariance function \( k(\cdot, \cdot) \) if, for any finite set of elements \( t_1, \ldots, t_l \in T \), the corresponding random variables \( \xi(t_1), \ldots, \xi(t_l) \) have multivariate jointly Gaussian distribution:

\[
\begin{bmatrix}
\xi(t_1) \\
\vdots \\
\xi(t_l)
\end{bmatrix}
\sim N
\left(
\begin{bmatrix}
m(t_1) \\
\vdots \\
m(t_l)
\end{bmatrix},
\begin{bmatrix}
k(t_1, t_1) & \cdots & k(t_1, t_l) \\
\vdots & \ddots & \vdots \\
k(t_l, t_1) & \cdots & k(t_l, t_l)
\end{bmatrix}
\right)
\] (6.1)

where \( m(t) \triangleq E[\xi(t)] \) is the mean function and \( k(t, t') \triangleq E[(\xi(t) - m(t))(\xi(t') - m(t'))] \) is the covariance function. Equation (6.1) is denoted in vector notation as: \( \xi \sim GP(m, K) \). The GP regression algorithm is now described below.

Let \( X = \{x_i\} \) and \( Y = \{y_i\}, i = 1, \ldots, n \), be the training data set, where \( X \) denotes a set of operating conditions and \( Y \) denotes the corresponding set of system responses. The objective here is to determine the relation between \( X \) and
\( Y \) so that, given an unknown operating condition \( x \), the corresponding system response \( y \) can be predicted. In the GP regression algorithm, it is assumed that \( y = \xi(x) + \varepsilon \), where \( \varepsilon \) is independent and identically distributed (iid) (additive) noise and \( \mathcal{N}(0, \sigma^2) \). That is, the response \( y \) is assumed to be a stochastic process that is a function of the operating condition \( x \) with additive noise. Then, a zero-mean Gaussian process prior \( GP(0, K) \) is assumed for the function \( \xi \). By the property of GP in Eq. (6.1)), the marginal distribution over any set of operating conditions as: 

\[
\text{ate jointly Gaussian distribution. Hence, by concatenating the training and testing responses are jointly distributed as:}
\]

\[
\begin{bmatrix}
\xi(X) \\
\xi(X^{test})
\end{bmatrix} \sim \mathcal{N}
\begin{pmatrix}
0, \\
K(X, X) & K(X, X^{test}) \\
K(X^{test}, X) & K(X^{test}, X^{test})
\end{pmatrix}
\]

(6.2)

where \( \xi(X) = [\xi(x_1), \cdots, \xi(x_n)]' \); \( \xi(X^{test}) = [\xi(x_1^{test}), \cdots, \xi(x_m^{test})]' \); \( K(X, X^{test}) \in \mathbb{R}^{n \times m} \) such that \( (K(X, X^{test}))_{ij} = k(x_i, x_j^{test}) \); \( K(X, X) \in \mathbb{R}^{n \times n} \) such that \( (K(X, X))_{ij} = k(x_i, x_j) \); \( K(X^{test}, X^{test}) \in \mathbb{R}^{m \times m} \) such that \( (K(X^{test}, X^{test}))_{ij} = k(x_i^{test}, x_j^{test}) \); \( K(X^{test}, X) \in \mathbb{R}^{m \times n} \) such that \( (K(X^{test}, X))_{ij} = k(x_i^{test}, x_j) \).

Since the system responses \( Y \) and \( Y^{test} \) are contaminated with additive iid Gaussian noise, i.e., 

\[
\begin{bmatrix}
\varepsilon \\
\varepsilon_{test}
\end{bmatrix} \sim \mathcal{N}
\begin{pmatrix}
0, \\
\sigma^2 I & 0 \\
0 & \sigma^2 I
\end{pmatrix}
\]

where \( \varepsilon = [\varepsilon^1, \cdots, \varepsilon^n]' \); and \( \varepsilon_{test} = [\varepsilon_{test}^1, \cdots, \varepsilon_{test}^m]' \), it follows that 

\[
\begin{bmatrix}
Y \\
Y^{test}
\end{bmatrix} = \begin{bmatrix}
\xi \\
\xi_{test}
\end{bmatrix} + \begin{bmatrix}
\varepsilon \\
\varepsilon_{test}
\end{bmatrix} \sim \mathcal{N}
\begin{pmatrix}
0, \\
K(X, X) + \sigma^2 I & K(X, X^{test}) \\
K(X^{test}, X) & K(X^{test}, X^{test}) + \sigma^2 I
\end{pmatrix}
\]

(6.3)

where \( Y^{test} = [y_1^{test}, \cdots, y_m^{test}]' \). The rules for conditioning on Gaussians, \( Y^{test}|Y \sim N(\mu^{test}, \Sigma^{test}) \) yield:

\[
\mu^{test} = K(X^{test}, X)(K(X, X) + \sigma^2 I)^{-1} Y
\]

(6.4)
\[ \Sigma^{test} = K(X^{test}, X^{test}) + \sigma^2 I \]
\[ - K(X^{test}, X) \left( K(X, X) + \sigma^2 I \right)^{-1} K(X, X^{test}) \] (6.5)

Thus, the algorithm predicts the mean and variance of the system response for every test condition. Instead of a zero-mean prior (i.e., \( E[\xi(x)] = 0 \)), a mean function \( m(x) \) could also be incorporated into the prior. In such a case,

\[ Y^{test}|Y, X, X^{test} \sim N(\mu^{test}, \Sigma^{test}) \] (6.6)

\[ \mu^{test} = m(X^{test}) + K(X^{test}, X)(K(X, X) + \sigma^2 I)^{-1}(Y - m(X)) \] (6.7)

instead of Eq. (6.4), and \( \Sigma^{test} \) remains unchanged in Eq. (6.5).

### 6.3.2 Development of the Design Methodology

This subsection develops the combustor design methodology. Synchronized time series data of pressure oscillations have been collected under different operating conditions on the experimental apparatus described in section 6.2, by varying the following parameters:

1. Inlet velocity from 25 to 50 m/sec in steps of 5 m/sec.
2. Equivalence ratio \((\phi)\) as 0.525, 0.550, 0.600 and 0.650.
3. Combustor length from 25 to 59 inches in steps of 1 inch.

Time series data of pressure oscillations have been collected at a sampling rate of 8192 Hz for each of the above \(6 \times 4 \times 35 = 840\) distinct operating conditions. The time span of data collection has been 8 seconds (i.e., 65,536 measurement data per channel) for each time series, which is within the safe limit of operation of the combustor apparatus and which is long enough to provide sufficient information for statistical analysis. The root mean square (rms) value, \( P_{rms} \), of pressure has been calculated for each time series. For this data set, it has been observed that all systems with \( P_{rms} \geq 0.07 \) psi are unstable, while all those with \( P_{rms} < 0.07 \) psi are stable.
6.3.3 Stability Prediction using $P_{rms}$ as a feature

This section uses $P_{rms}$ as a feature, since $P_{rms}$ is one of the most commonly used features for stability detection in literature. The first step involves randomly dividing the complete dataset into training and the testing sets in the proportion of 70% and 30%, respectively. The inputs to the GP algorithm consist of (operating condition, $P_{rms}$) pairs. For each test point (i.e., operating condition), the GP algorithm then predicts the mean $\mu$ and variance $\sigma^2$ of the distribution of the response ($P_{rms}$) at that point using Eqs. (6.5), (6.6) and (6.7) under the Gaussian assumption. Then, for every test point, the probability of sampling values exceeding 0.07 is calculated from the predicted response distribution by computing the Gaussian cumulative distribution function (CDF) (i.e., $P(X > 0.07) = 1 - \text{CDF}(0.07)$). If the computed probability $P(X > 0.07)$ exceeds the user-defined threshold $\Theta$, the combustion system is predicted to become unstable if operated at this condition. The predicted stability labels are then compared with the ground truth of stability labels, and the number of errors are noted. The entire procedure is repeated for 20 random combinations of training and testing sets, and their average performance is computed. The procedure for determination of optimal threshold $\Theta$ is explained below.

The entire procedure explained in the above paragraph has been executed for thresholds $\Theta$ in the range of 0.3 to 0.8 in steps of 0.02. Figure 6.3 shows the receiver operating characteristic (ROC) plot, where each point (marked as x) corresponds to a single value of $\Theta$, the abscissa denotes the average (over 20 runs) false alarm rate (i.e. percent of times stable cases are incorrectly predicted as unstable), and the ordinate denotes the average (over 20 runs) true positive rate (i.e., percent of times unstable cases are correctly predicted as unstable). Higher threshold means that the user needs a higher probability of samples from the predicted response distribution to be more than 0.07psi, in order to predict the system to be unstable, if operated at this condition. In other words, the user predicts system instability for a system condition, only when the probability of $P_{rms}$ being more than 0.07psi under this condition is quite high. Hence, as the threshold value is increased, false alarm rate (stable cases are incorrectly predicted as unstable) falls and misdetection rate increases. The extreme scenarios correspond to the threshold being 0 or 1. If it is 0, every case is predicted to be unstable, which would result
in 100% true detection rate and 100% false alarm rate. On the other hand if the threshold is 1, every case is predicted to be stable, resulting in 0% false alarm rate and 0% true detection rate. In the figure 6.3, one goes from the upper right corner (high false alarm rate and high true detection rate) to the bottom left corner (low false alarm rate and low true detection rate), as the threshold is increased.

The cost function chosen in this work to determine the optimal threshold is the average misclassification rate (over 20 runs). Misclassification rate is the sum of false alarms (number of stable test cases wrongly predicted as unstable) and misdetections (number of unstable test cases wrongly predicted as stable) divided by total number of test cases. Therefore, misclassification rate is the percent of times predicted stability label is not equal to the true stability label. For the decision threshold \( \Theta = 0.66 \), the average misclassification rate is found to be minimum. Hence, \( \Theta = 0.66 \) is taken to be the optimal decision threshold.

![Figure 6.3: ROC curve as a function of the decision threshold](image)

Table 6.1: Confusion Matrix for the decision threshold \( \Theta = 0.66 \)

<table>
<thead>
<tr>
<th></th>
<th>Predicted unstable (%)</th>
<th>Predicted stable (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>True unstable</td>
<td>91.83</td>
<td>8.17</td>
</tr>
<tr>
<td>True stable</td>
<td>22.08</td>
<td>77.92</td>
</tr>
</tbody>
</table>
6.3.3.1 Results and Discussion

Table 6.1 depicts the confusion matrix for the decision threshold $\Theta = 0.66$. The first row shows the percent of true unstable cases predicted correctly as unstable and incorrectly as stable on the average. The second row shows the percent of true stable cases predicted incorrectly as unstable and correctly as stable on average. The cost function is chosen by the end user. In the cost function chosen here, equal weight is given to false alarms and misdetections. The end user can decide the relative importance (weights) of these terms to be according to his requirements; and depending on the cost function, the optimal decision threshold $\Theta$ may differ. It is noted that, instead of predicting distinct class labels (i.e., stable or unstable) using the decision threshold $\Theta$, the output of the algorithm could be retained as a probability distribution over the set of classes - stable and unstable.

![Figure 6.4: $P_{rms}$ predicted by GP regression algorithm: 95% confidence region](image)

Figure 6.4 shows the 95% confidence intervals (i.e., $[\mu - 2\sigma, \mu + 2\sigma]$) of $P_{rms}$, predicted by the GP algorithm for all test cases, and the corresponding true values of $P_{rms}$ superimposed on it, for a single run (i.e., a specific combination of training and testing data sets). For this run, for 3.42% of test cases, the true value of $P_{rms}$ does not lie in the predicted 95% confidence interval. Figure 6.5 shows the true stability map and the predicted stability map for decision threshold $\Theta = 0.66$ for the same run. On an average, it is observed that the true $P_{rms}$ value does not lie in the confidence interval for 5.43% of test cases.

Although $P_{rms}$ appears to serve as a good indicator of stability in this set of
experiments, a natural question arises whether $P_{\text{rms}}$ could be universally adopted as a feature. Since $P_{\text{rms}}$ is the standard deviation of the signal in the statistical sense, it is sensitive to noise. Therefore, $P_{\text{rms}}$ may not be an ideal choice as a feature, because of its lack of robustness to measurement noise. Hence, the next subsection describes the process of identifying a discriminative, robust feature, indicative of system stability. Figure 6.7 presents a flowchart of the proposed combustor design algorithm, using PFSA as a feature.

### 6.3.4 Stability Prediction using PFSA as feature

In the design of a real-life combustor, sufficiently long durations of data acquisition might not be feasible due to various challenges inflicted on the system performance and operability (e.g., high-amplitude pressure and temperature oscillations, and local air-fuel ratio variations leading to flame blowout) specifically during combustion instabilities. Figure 6.6 shows representative plots of $P_{\text{rms}}$ calculated for different durations of time series in increments of 0.1 s for: (a) a stable signal and (b) an unstable signal. It is seen in Fig. 6.6(a) that the threshold $P_{\text{rms}}$ of 0.07 psi is exceeded at time series window lengths ranging from about 2.5 s to 4 s. Although $P_{\text{rms}}$ over the entire 8 s window is less than the threshold of 0.07 psi, an online stability criterion based on solely a $P_{\text{rms}}$ threshold may generate false alarms for those durations, where the threshold is exceeded. Similar conclusions can be drawn from Fig. 6.6(b) where, for an actually unstable signal, $P_{\text{rms}}$ drops below 0.07 psi within several time windows between 1 s to 3 s and close to 4 s. In
both cases, a hard $P_{rms}$ threshold is likely to yield misclassifications.

**Remark 6.3.1.** In general, stability decisions made on $P_{rms}$-based hard thresholding would not be robust relative to measurement noise. In contrast, symbolic dynamics-based decisions are expected to be significantly more robust as established by Biem Garben [97], which is the approach taken for feature extraction from now in the combustor design algorithm.

It is now demonstrated how the features extracted from time series using the approach of STSA, can be used with confidence at relatively shorter time windows provided that they are sufficiently long to capture the system dynamics.
Different durations of time series data (i.e., 2 s, 4 s, 6 s and 8 s) have been considered in this chapter. In addition, different alphabet sizes for construction of the D Markov machine have also been considered. For each time window length, alphabet size pair, approximately 80% of the feature vectors that are extracted from the corresponding pressure time series along with their true stability labels have been used for training a binary classifier in the setting of support vector machines (SVM) [2, 82]. The trained classifier is then used to predict the stability labels for the remaining 20% of the data. The D-Markov machine (alphabet size) which yields the best classification accuracy by using a radial basis function (RBF) kernel-based SVM classifier has been chosen for each duration of data under consideration.

The classification error is defined in terms of the percent of misclassified samples among the test data. Table 6.2 lists the classification accuracy using D-Markov machines, with $D = 1$, for time series data of different window lengths at a fixed sampling rate. It is seen in Table 6.2 that the classification accuracy is very high and they are comparable for all different durations under consideration.

### 6.3.4.1 Results and Discussion

The objective in this analysis is to determine the degree of stability/instability of the combustion system at different combustor lengths for a given inlet velocity $u$ and equivalence ratio $\phi$, which would then be used for designing the combustor. Accordingly, the entire dataset has been divided into subsets of constant inlet velocity and equivalence ratio. Hence, each subset consists of a set of combustor lengths and the corresponding pressure time series. The data in each subset are first randomly divided into training and the testing sets in the proportion of 80% and 20%, respectively. In the training phase, sufficiently long (e.g., 8-sec duration) time-series data have been collected for each operating condition. Therefore, the
Figure 6.7: Flowchart of the combustor design algorithm using PFSA as a feature
most stable condition is determined based on $P_{\text{rms}}$, which is calculated for each
time series. The combustor length (in the training set) with the corresponding
lowest $P_{\text{rms}}$ is taken to be the one representing the nominal state of the combustion
system under the given input of $u$ and $\phi$.

Each time series in a subset is first partitioned by Maximum Entropy Partitioning (MEP) [43]. The resulting symbol string is then compressed as a PFSA
by assigning the states as symbol strings of finite length. The features chosen in
this work are the morph matrices of the PFSA (see Definition 2.1.3). This feature encodes the dynamics of the time series in the form of a morph matrix as
described in Chapter 2, where the feature extraction procedure is represented by
the “STSA” block in the flowchart of Fig. 6.7. The morph matrix feature, corresponding to the nominal combustor length, is taken to be the nominal feature for
this subset. For every case (i.e., each combustor length) in the training set, the divergence of its corresponding feature from the nominal feature is calculated and is denoted as \( F_{\text{div}} \) (here it is the Frobenius matrix norm of the difference between the two morph matrices, which is an indication of how far away the system is from the most stable operating condition); this is represented by the “Computation of feature divergence block” in the flowchart of Fig. 6.7. This metric is introduced in order to have an estimate of the behavior of the predicted state of the system at different lengths of the combustor. A higher value of \( F_{\text{div}} \) indicates that the system is likely to be more unstable, while a lower value of \( F_{\text{div}} \) indicates that the system response is closer to the most stable operating condition. The inputs to the GP regression algorithm (see Subsection 6.3.1) thus consist of (combustor length, feature divergence) pairs. as represented in the inputs to the block “Training GP regression algorithm” in the flowchart of Fig. 6.7). For the GP regression algorithm, a variety of mean and covariance functions can be used. Here, the following mean and covariance functions are considered.

1. Mean function: (i) Constant \( m(x) = c \), (ii) Linear \( m(x) = \sum_{i=1}^{J} a^i x^i \), and (iii) Sum of the constant and linear terms yields: \( m(x) = c + \sum_{i=1}^{J} a^i x^i \), where \( J \) is the dimension of the input space.

2. Covariance function: (i) Linear \( k(x^p, x^q) = x^p \ast (x^q)' \), and (ii) squared exponential automatic relevance determination (SEARD) \( k(x^p, x^q) = sf^2 \ast \exp(-(x^p - x^q)' \ast P^{-1} \ast (x^p - x^q)/2) \), where the \( P \) matrix is diagonal with automatic relevance determination (ARD) parameters \( \ell_1, ..., \ell_J \), and \( sf^2 \) is the signal variance [96].

The log likelihood of the training data for all combinations of mean and covariance functions has been compared. It is observed that the combination of constant mean function and SEARD covariance function resulted in highest likelihood; hence, this pair of mean and covariance functions have been used for all subsequent analysis in the work. In addition, GP regression being a Bayesian algorithm, it is not necessary to know the optimal values of the hyperparameters (i.e., \( c, \{\ell_i^2\} \) and \( sf \)) in the mean and covariance function beforehand. The algorithm identifies the optimal values of these hyperparameters (by identifying which values maximize the log likelihood of training data).
For each subset of constant inlet velocity and equivalence ratio, GP regression algorithm is used to determine the mapping from combustor length to the system response (i.e., divergence $F_{\text{div}}$ from the nominal condition), represented by the block “Trained GP regression algorithm” in the flowchart of Fig. 6.7. Using this mapping, the algorithm then predicts the distribution of $F_{\text{div}}$ for the combustor lengths in the testing set. For each test case (i.e., combustor length), the GP algorithm predicts the mean $\mu$ and variance $\sigma^2$ of the distribution of the system response at that point using Eqs. (6.5) and (6.7) under the Gaussian assumption. In other words, for a given combustor length, the GP regression algorithm predicts how different the system response is expected to be from the nominal state. One of the reasons for using GP for system response prediction is its ability to quantify the uncertainty in the estimate of the response. The proposed algorithm thus estimates the most likely response (i.e., mean) together with the variations about the mean, which may accrue from possible sources of uncertainties in the estimation (e.g., measurement noise and insufficient training data).

The difference between the predicted mean and true response value for each test case is noted. In addition, the number of test cases for which the true test value does not fall in the predicted 95% confidence intervals (i.e., $[\mu - 2\sigma, \mu + 2\sigma]$) is recorded, which is represented by the block “Error metric” in the flowchart of Fig. 6.7. The entire procedure has been repeated for 20 random combinations of training and testing sets, and their average performance is computed. For every combination of training and testing sets of combustor lengths under a given inlet velocity and equivalence ratio, four different sampling durations of the corresponding time series have been considered: 2s, 4s, 6s, and 8s.

Figure 6.8 shows the mean ($\mu$) and the 95% confidence intervals (i.e., $[\mu - 2\sigma, \mu + 2\sigma]$) of $F_{\text{div}}$, predicted by the GP algorithm for all test cases, and the corresponding true values of $F_{\text{div}}$ superimposed on it, for a single run (i.e., a specific combination of training and testing data set) for four different window lengths of time series: 2 s, 4 s, 6 s, 8 s. The inlet velocity and equivalence ratio for this subset are 40 m/sec and 0.55 respectively. For this run, it is observed that, for all test cases, the true value of $F_{\text{div}}$ lies in the predicted 95% confidence interval, for all four window lengths of time series. In addition, for each window length of time series, the difference between predicted mean and true response value has been noted for
Figure 6.8: Feature divergence $F_{div}$ predicted by GP regression algorithm for inlet velocity=40m/sec and $\phi=0.55$

each test case, and the average is taken over all test cases in the testing set. The average errors over the testing set in this run for the four different window lengths of time series data are listed in Table 6.3.

Table 6.3: Mean error for a single run on four different window lengths (sampled at 8192 Hz) of time series at inlet velocity=40m/sec and $\phi=0.55$

<table>
<thead>
<tr>
<th>Window Length</th>
<th>2 sec</th>
<th>4 sec</th>
<th>6 sec</th>
<th>8 sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean error</td>
<td>0.0003</td>
<td>-0.0004</td>
<td>0.0008</td>
<td>-0.0025</td>
</tr>
</tbody>
</table>

Figures 6.9 and 6.10 respectively show the mean error in prediction and mean predicted uncertainty in estimation (i.e. square root of the predicted variance) on a testing set, averaged over 20 runs (i.e., random combinations of training and testing
sets, by randomly dividing each subset in the ratio of 80% and 20% respectively 20
times), for 18 subsets for features extracted from time series with window length
of 8 seconds. The mean error in prediction over 18 subsets is 0.0107. The mean
uncertainty in prediction over 18 subsets is 0.0083.

Similarly, figures 6.11 and 6.12 respectively show the mean error in prediction
and mean predicted uncertainty in estimation, averaged over 20 runs, for 18 subsets
for features extracted from time series with window length of 2 seconds. The mean
error in prediction over 18 subsets is 0.0118. The mean uncertainty in prediction
over 18 subsets is 0.0106.

Thus, the accuracy of the algorithm for a time window length of 2 seconds, is
very close to that for window length of 8 seconds. These results show that the
window length of time series data does not significantly affect the accuracy of the
estimated mapping between combustor length and system response, provided that
the data are sufficiently long to capture the system dynamics. So, the features
extracted using STSA are more robust to window length, than $P_{rms}$.

![Figure 6.9: mean error in prediction over 20 runs using 8 sec time series](image)

The implication of the above results for a design exercise is that these figures
Figure 6.10: mean uncertainty in prediction over 20 runs using 8 sec time series

provide the information on how close the system is to the stability boundary. Considering the uncertainties involved, the designer would try to select a set of parameters, within the constraint of permissible range at which $F_{div}$ is lowest, as represented by the block “User defined criteria for stability” in the flowchart of Fig. 6.7.

The plots in Figure 6.13 compare the profiles of normalized $F_{div}$ and normalized $P_{rmsdiv}$ for the test run corresponding to inlet velocity=40m/s and equivalence ratio=0.55, for 2s and 8s window lengths of time series data (whose prediction results are displayed in Figure 6.8). Normalized $F_{div}$ follows a trend similar to that of normalized $P_{rmsdiv}$, thus showing that the PFSA feature is consistent with $P_{rmsdiv}$ in terms of quantifying the system stability. Since $F_{div}$ is significantly larger in magnitude than $P_{rmsdiv}$ for the majority of cases, the PFSA feature is apparently more sensitive (and hence more discriminative) to changes in system dynamics. In other words, for the same change in system dynamics, the divergence of the PFSA feature from the original feature would be much higher than that corresponding to the $P_{rms}$ feature. For applications to combustor design, this increased sensitivity is preferable because of the capability of correctly detecting smaller anomalies with
Figure 6.11: mean error in prediction over 20 runs using 2 sec time series

the same threshold. In other words, the user would like to design a combustor with a smaller chance of becoming unstable. In such a scenario, if a more sensitive feature is used, even a small deviation in the system behavior from the nominal state, would be manifested as a large feature divergence; this would dissuade the user from choosing the combustor parameters that could potentially lead to instabilities. Under this train of logic, the combustor parameters, chosen corresponding to least feature divergence, would be more conservative and thus have a smaller chance of combustion instability.

6.4 Summary, Conclusions & Future Work

A data-driven method, based on a Bayesian nonparametric technique of Gaussian process (GP) regression, has been proposed in this work as a tool for combustor design. The underlying assumption of GP regression is that, for any finite set of operating conditions, the corresponding system responses are jointly Gaussian; but no assumptions are made on the nature of the relation between operating conditions
Figure 6.12: mean uncertainty in prediction over 20 runs using 2 sec time series

(a) PFSA and $P_{rms}$ feature divergence (b) PFSA and $P_{rms}$ feature divergence
for 2 sec data \hspace{1cm} \text{for 8 sec data}

Figure 6.13: Sensitivity comparison of $PFSA$ and $P_{rms}$ features
and the resulting system response. However, the jointly Gaussian assumption may not strictly hold, because of the introduction of estimated parameters, which is reflected in the uncertainties of the estimated response.

The method has been validated on experimental data of pressure time-series from a lean-premixed swirl-stabilized combustor. Given an ensemble of training data for a set of constant equivalence ratio and inlet velocity, the proposed al-
algorithm determines the mapping between the combustor length and the system response. The algorithm can then predict the distribution of the system response for every other point in the space of combustor lengths, for which experimental data may not be available. This information can then be used for identifying the combustor length that will yield the desired system response. While the proposed design methodology is more discriminative with respect to small deviations (e.g., those resulting from evolving anomalies) than that based on $P_{rms}$, it is also observed to be more robust to measurement noise that is inherent in the time series data, because of symbolization [97].

Although the work presented in this chapter is validated for design of a laboratory scale combustor as a proof of concept study, it is envisioned that this method can be extended to more complex industrial-scale combustors, because the input to this data-driven approach is the pressure time series, which can be easily generated in such combustors. In addition, in the initial stage of design, the actual experiments used to generate data in the present work may also be replaced by a limited number of high-fidelity simulations, involving unsteady Reynolds-averaged Navier-Stokes equation (URANS) or large eddy simulations (LES) and state of the art models for turbulent combustion appropriate to the combustion mode at hand (e.g., premixed, non-premixed, or partially premixed). Another extension could be using existing knowledge for modifications of existing combustors. From these perspectives, topics of future research on the proposed design method are delineated below.

- Research on how the proposed data-driven method can be gainfully integrated with the current state-of-the-art (including the model-based) tools of combustor design.
- Comparing the performance of the proposed design algorithm with state-of-the-art design methodologies not including usage of $P_{rms}$.
- Testing the design algorithm on combustors of different geometries and input parameters.
- Extension of the proposed methodology for designing combustors under active control.
In the previous chapter, an algorithm was developed for combustor design with a fixed-structure data-driven model. In the present chapter, a variable-structure dynamic data-driven model is developed that is capable of effectively capturing the history of process dynamics. The underlying concept is dynamic, because no fixed structure has been assumed for the model that captures the information embedded in the time series. Instead, a structure of the model is identified for every time series, corresponding to a specific operating condition, under user-selected constraints of configuration and redundancy. An information-theoretic measure is then used for comparative evaluation of different $D$-Markov model structures. The work in this chapter yields more representative features as compared to those extracted in the previous chapter. The designer is thus provided with a quantitative tool of statistical estimation of system stability at new parameters without the need for additional (and possibly expensive and time-consuming) experimentation.

7.1 Mathematical Preliminaries

7.1.1 Entropy rate

This subsection introduces the notion of entropy rate that, given the current state, represents the predictability of $PFSA$. Details are reported in [38].
Definition 7.1.1. The entropy rate of a PFSA \((\Sigma, Q, \delta, \pi)\) is defined as follows.

\[
H(\Sigma|Q) \triangleq -\sum_{q\in Q} \sum_{\sigma\in \Sigma} P(q) P(\sigma|q) \log P(\sigma|q)
\] (7.1)

where \(P(q)\) is the (unconditional) probability of a PFSA state \(q \in Q\); and \(P(\sigma|q)\) is the (conditional) probability of a symbol \(\sigma \in \Sigma\) emanating from the PFSA state \(q \in Q\).

Next the notion of a metric is introduced to quantify the distance between two PFSA.

Definition 7.1.2. Let \(K_1 = (\Sigma, Q_1, \delta_1, \pi_1)\) and \(K_2 = (\Sigma, Q_2, \delta_2, \pi_2)\) be two PFSA with a common alphabet \(\Sigma\). Let \(P_1(\Sigma^j)\) and \(P_2(\Sigma^j)\) be the steady state probability vectors of generating words of length \(j\) from the PFSA \(K_1\) and \(K_2\), respectively, i.e., \(P_1(\Sigma^j) \triangleq [P(w)]_{w \in \Sigma^j}\) for \(K_1\) and \(P_2(\Sigma^j) \triangleq [P(w)]_{w \in \Sigma^j}\) for \(K_2\). Then, the metric as the distance between the PFSA \(K_1\) and \(K_2\) is defined as:

\[
\Phi(K_1, K_2) \triangleq \lim_{n \to \infty} \frac{1}{n} \sum_{j=1}^{n} \|P_1(\Sigma^j) - P_2(\Sigma^j)\|_{\ell_1}
\] (7.2)

where the norm \(\|\cdot\|_{\ell_1}\) indicates the sum of absolute values of the elements in the vector \(\cdot\).

7.1.2 D-Markov Machines

This subsection introduces additional concepts, beyond those explained in Chapter 2, which are necessary for developing the variable structure model.

7.1.2.1 State Splitting in D-Markov Machines

The process of splitting a state \(q\) is executed by replacing the symbol block \(q\) by its branches as described by the set \(\{\sigma q : \sigma \in \Sigma\}\) of words. Maximum reduction of the entropy rate is the governing criterion for selecting the state to split. In addition, the generated set of states must satisfy the self-consistency criterion, which only permits a unique transition to emanate from a state for a given symbol. If \(\delta(q, \sigma)\) is not unique for each \(\sigma \in \Sigma\), then the state \(q\) is split further. In the state splitting
Algorithm 3 State splitting

**Input:** Symbol sequence $s_1 s_2 s_3 \ldots$, where each $s_i$ belongs to the symbol alphabet $\Sigma$

**User defined Input:** Maximum number of states $N_{max}$ and threshold $\eta_{mrg}$

**Output:** PFSA $K = \{\Sigma, Q, \delta, \pi\}$

**Initialize:** Create a 1-Markov machine $\tilde{Q} := \Sigma$

repeat
  $Q := \tilde{Q}$;
  $\tilde{Q} = \arg\min_{Q'} H(\Sigma|Q')$
  where $Q' = (Q\{q\}) \cup (\{\sigma q : \sigma \in \Sigma\})$ and $q \in Q$
until $|\tilde{Q}| < N_{max}$ or $H(\Sigma|Q) - H(\Sigma|\tilde{Q}) < \eta_{mrg}$
for all $q \in \tilde{Q}$ and $\sigma \in \Sigma$
do
  if $\delta(q, \sigma)$ is not unique then
    $\tilde{Q} := (\tilde{Q}\{q\}) \cup (\{\sigma q : \sigma \in \Sigma\})$
  end if
end for
return $K = \{\Sigma, Q, \delta, \pi\}$

Algorithm 4 Minimal equivalence relation given $q \sim q'$

**Input:** $\delta$, $q$, $q'$, Initial equivalence relation $\sim$

**Output:** Updated equivalence relation $\sim$

**NOTE:** Recursive function $(\sim) := \text{Merge}(\delta, q, q', \sim)$

Set $q \sim q'$;
for all $\sigma \in \Sigma$ do
  if $\delta(q, \sigma) \not\sim \delta(q', \sigma)$ then
    Set $\sim := \text{Merge}(\delta, \delta(q, \sigma), \delta(q', \sigma), \sim)$;
  end if
end for
return $\sim$

7.1.2.2 State Merging in D-Markov Machines

Once state splitting is performed, the resulting D-Markov machine is a statistical representation of the symbol string under consideration. Depending on the choice
Algorithm 5 Minimal PFSA $K_2$ after merging of two states

**Input:** $K_1 = \{\Sigma, Q_1, \delta_1, \pi_1\}$, $q, q'$

**Output:** Merged PFSA $K_2 = \{\Sigma, Q_2, \delta_2, \pi_2\}$

Compute the equivalence relation $\sim$ using Algorithm 4;
Set $Q_2 := Q_1 / \sim$; \% $Q_2$ is the quotient set of $Q_1$ under $\sim$
Compute the stationary-probability vector $\hat{P}_1$ of the PFSA $K_1$;

**for all** $[q] \in Q_2$ **do**
  **for all** $\sigma \in \Sigma$ **do**
    Set $\delta_2([q], \sigma) := [\delta_1(q, \sigma)]$
    Compute $\pi_2([q], \sigma)$
  **end for**
**end for**

**return** $K_2 = \{\Sigma, Q_2, \delta_2, \pi_2\}$

of alphabet size $|\Sigma|$ and depth $D$, the number of states after splitting may run into hundreds. The motivation behind the state merging is to reduce the number of states, while preserving the $D$-Markov structure of the PFSA. A stopping rule is constructed by specifying an acceptable threshold $\eta$ on the distance $\Phi(\cdot, \cdot)$ between the merged PFSA and the PFSA generated from the original time series (see Eq. (7.2)); details are provided in [38].

The next task is to decide which states have to be merged. States that behave similarly (i.e., have similar morph probabilities) have a higher priority for merging. The similarity of two states, $q, q' \in Q$, is measured in terms of morph functions (i.e., conditional probabilities) of future symbol generation as the distance between the two rows of the estimated morph function $\hat{\pi}$ corresponding to the states $q$ and $q'$. The $\ell_1$-norm (i.e., the sum of absolute values of the vector components) has been adopted to be the distance function as seen below.

$$
\mathcal{M}(q, q') \triangleq \|\hat{\pi}(q, \cdot) - \hat{\pi}(q', \cdot)\|_{\ell_1} = \sum_{\sigma \in \Sigma} |\hat{\pi}(q, \sigma) - \hat{\pi}(q', \sigma)| \tag{7.3}
$$

A small value of $\mathcal{M}(q, q')$ indicates that the two states have close probabilities of generating each symbol. Note that this measure is bounded above as $\mathcal{M}(q, q') \leq 2 \ \forall q, q' \in Q$, because $0 \leq \sum_{\sigma \in \Sigma} \hat{\pi}(q, \cdot) \leq 1$ and $0 \leq \sum_{\sigma \in \Sigma} \hat{\pi}(q', \cdot) \leq 1$.
Algorithm 6 State merging in PFSA

**Input:** PFSA $K = (Σ, Q, δ, π)$, threshold $η_{mrg}$, and symbol sequence $\{s_i\}$

**Output:** Merged PFSA $K_m = (Σ, Q_m, δ_m, π_m)$

Set $K_m := K$;

for all $q, q' \in Q_m$ do
  if $q \neq q'$ then
    Set $\text{List\_States}(q, q') = \mathcal{M}(q, q')$ using Eq. (7.3);
  else
    Set $\text{List\_States}(q, q') = 2$;
  end if
end for

sort($\text{List\_States}$); % Place the pair $(q, q')$ with the smallest $\mathcal{M}(q, q')$ on top of the list
Set $(q, q') := \text{pop}(\text{List\_States})$; % Select the pair $(q, q')$ that is on top of the sorted list
loop
  Compute $K_1$ from $K_m$ by merging the states $q$ and $q'$ via Algorithm 5;
  if $d[K_1, \{s_i\}] < η_{mrg}$ then
    Set $K_m := K_1$;
    Recompute $\text{List\_States}$;
    Set $(q, q') := \text{pop}(\text{List\_States})$;
  else
    Set $(q, q') := \text{pop}(\text{List\_States})$;
    if $q == q'$ then
      Break loop;
    end if
  end if
end loop
return $K_m = (Σ, Q_m, δ_m, π_m)$

First, the two closest states (i.e., the pair of states $q, q' \in Q$ having the smallest value of $\mathcal{M}(q, q')$) are merged [38]. Subsequently, distance $Φ(\cdot, \cdot)$ (see Eq. (7.2)) of the merged PFSA from the initial symbol string is evaluated. If $Φ < η$, where $η$ is a specified threshold, then the machine structure is retained and the states next on the priority list are merged. On the other hand, if $Φ > η$, then the process of merging the given pair of states is aborted and another pair of states with the next smallest value of $\mathcal{M}(q, q')$ is selected for merging. This procedure is terminated if no such pair of states exist, for which $Φ < η$. 
7.2 Technical Approach

This section describes the technical details of the proposed combustor design methodology. It starts with the pertinent steps for construction of the design methodology, which is an extension of the work reported in the previous chapter. The underlying algorithm has been developed in the following two steps:

**Step 1 Feature extraction from pressure time series:** The features characterize the combustion dynamics.

**Step 2 GP regression** to identify the relation between the operating condition and the corresponding system response as a function of the features extracted in Step 1. This inferred relation is then used for predicting response for each unobserved operating condition.

For specified values of inlet air velocity \((u)\) and equivalence ratio \((\phi)\), the problem under consideration is to determine a mapping from combustor length \((l)\) to the degree of stability of the combustion system; similar exercises can be carried out with respect to other design variables as well. To achieve this goal, the entire dataset has been divided into 18 subsets of constant \(u\) and \(\phi\). The data in each subset thus consist of a set of combustor lengths \((l)\) and the corresponding pressure time series (each of length 8 seconds). There is a single time series for each operating condition (i.e., each value of \(l\)). The data in each subset are first randomly divided into training and testing sets in the proportion of 80% and 20%, respectively. The rms value, \(Prms\), of pressure is then calculated for each time series. The combustor length in the training set with the lowest \(Prms\) is taken to be the one representing the nominal state of the combustion system for the given values of \(u\) and \(\phi\).

The next step involves symbolization of each time series to obtain a symbol string. In this work, maximum entropy partitioning (MEP) [43] has been adopted to discretize the range space of the time series. The state-splitting and merging algorithms, are then applied on the symbol string to determine the \(D\)-Markov structure best describing the given time series (i.e., finding the number and identity of states that best models the given time series under the chosen constraints of alphabet size \(|\Sigma|\), upper bound on the distance between state probabilities \(\eta\),
maximum number of states after state splitting $N_{\text{max}}$). Here, $N_{\text{max}}$ reflects the importance given by the user to computational complexity, and $\eta$ reflects the importance of accuracy. After the suboptimal $D$-Markov structure is determined, its morph matrix is calculated. The above procedure yields $D$-Markov models for each time series, which might be different in number and identity of states. Higher the number of states in the PFSA of a $D$-Markov machine, lower is the predictability of the symbol sequence, as expected. Hence, in order to compare between two PFSA$s with different number of states, the unnormalized entropy rate (see Definition 7.1.1) would not be an adequate measure; the rationale is that PFSA with a higher number of states would always have a higher entropy rate. If instead, the entropy rate is divided by the maximum entropy rate for that $D$-Markov model (which is a function of the number of states in the PFSA), the measure would be a reflection of the proportion of the system behavior, which can be explained by the best model under the chosen complexity; the measure would always lie between 0 and 1. Higher values would mean that the best model at that complexity is not able to encode the dynamics of the sequence properly. The difference in normalized entropy rates of the PFSA$s under consideration, has thus been taken to be the divergence $F_{\text{div}}$, which will be used as the system response.

The feature, corresponding to the nominal combustor length, is chosen to be the nominal feature for the particular subset of $u$ and $\phi$. For every combustor length in the training set, the divergence of its corresponding feature from the nominal feature is calculated and is denoted as $F_{\text{div}}$. Hence, the system response in the proposed approach is taken to be the feature divergence. The inputs to the GP regression algorithm thus consist of the pair, namely, combustor length $l$ and feature divergence $F_{\text{div}}$, where the objective is to predict $F_{\text{div}}$ for different values of $l$. The combination of constant mean function and SEARD covariance function is observed to yield highest likelihood for the selected dataset; hence, the analysis has been performed under this combination. For each subset of constant inlet velocity ($u$) and equivalence ratio ($\phi$), a GP regression algorithm is used to determine the mapping from the combustor length $l$ to the system response. Using this mapping, the algorithm thus predicts the mean $\mu$ and variance $\sigma^2$ of the distribution of $F_{\text{div}}$ for each combustor length in the testing set.
7.3 Results and Discussion

Given the 18 subsets of constant inlet velocity \( (u) \) and equivalence ratio \( (\phi) \), this section analyzes the effects of \(|\Sigma|\), \( N_{max} \), and \( \eta \) (see Section 7.1) on the performance of the proposed stability map prediction with the following values:

1. \(|\Sigma| = 3 \) and \( 5 \).
2. \( N_{max} = 10, 20, \) and \( 30 \).
3. \( \eta = 0.02, 0.05, \) and \( 0.1 \).

The underlying algorithm has been executed for different combinations of the following parameters: \(|\Sigma|, \eta, \) and \( N_{max} \). Figures 7.1 and 7.2 respectively show the mean error in prediction and mean predicted uncertainty in estimation (i.e. square root of the predicted variance) on a testing set, averaged over 20 runs (i.e., random combinations of training and testing sets, by randomly dividing each subset in the ratio of 80% and 20% respectively) for different values of \( \eta \) while keeping the other parameters fixed at \(|\Sigma| = 5\) and \( N_{max} = 20 \). The entire procedure of system response prediction has been repeated for all 18 subsets, and the results are reported here. It is observed that, for a majority of the subsets, both mean error in the prediction and uncertainty in the estimate increase as \( \eta \) is increased. Figures 7.3 and 7.4 respectively show the standard deviations of the mean error and predicted uncertainty over 20 runs for different values of \( \eta \)'s while keeping the other parameters fixed, for each of the 18 subsets. Again, as \( \eta \) is increased, the generalization ability of the algorithm worsens. Thus, for a majority of the subsets, the performance of the design methodology improves as \( \eta \) is decreased.

This suggests that larger values of \( D \) are necessary to construct the \( D \)-Markov models that encode the process dynamics, implying the need for more memory in Markov models of the combustion process, which is in agreement with the recent findings of Sarkar et al. [83]. The rationale is that a high value of \( \eta \) might result in dissimilar states being merged, leading to inadequacy of the extracted feature which ultimately results in poor prediction of instabilities. Similar behavior has also been observed for other combinations of \(|\Sigma|\) and \( N_{max} \). For the best case (i.e., \( \eta = 0.02 \)), the average error in prediction over all 18 subsets is 0.0037 and the
average uncertainty in the prediction over all subsets is 0.0013, where the system response (i.e., the difference in normalized entropy rate $F_{\text{div}}$) can vary between 0 and 1.

![Figure 7.1: Mean error in prediction over 20 runs](image)

The system behavior has also been analyzed at stable and unstable conditions of combustion. It is observed that the entropy rate is consistently lower for unstable conditions as compared to stable conditions. Figures 7.5 and 7.6 respectively show the profiles of $P_{\text{rms}}$ and entropy rate for one out of the 18 subsets. A reasonable interpretation for this physical phenomenon, is that the system becomes less chaotic as limit cycle oscillations set in, i.e., at the onset of instability. It has also been observed that, even for a set of operating conditions that yield $D$-Markov models with the same number of states, the entropy rate for the unstable conditions is always lower than that for the stable conditions.

### 7.3.1 Comparison between $D=1$ and $D \geq 1$ features

The profile of $P_{\text{rms}}$, observed on the experimental apparatus (see Figure 6.1) over a sufficiently long time window, has been found to be good indicator of combustion stability, where the system is found to be unstable for $P_{\text{rms}} \geq 0.07$ psi. Based on
this knowledge, the design algorithm for $D \geq 1$ is now compared with that for $D = 1$ as explained below.

Reiterating again, there are 18 subsets of constant inlet velocity and equivalence ratio. In each subset, for each 8-second time series associated with a unique combustor length, the corresponding $P_{rms}$ is computed. The combustor length associated with the lowest $P_{rms}$ value ($P_{stable}^{rms}$) is designated as the most stable operating condition ($CL_{stable}$). For each combustor length, the divergence of $P_{rms}$ from the lowest value (i.e., $P_{rms} - P_{stable}^{rms}$) is computed. Then, from each time series in the subset, features are extracted using Symbolic Time Series Analysis, with $D = 1$ and $|\Sigma| = 8$ (see previous chapter). The feature associated with $CL_{stable}$ is designated as the nominal feature, and the divergence of all other features from the nominal feature is computed. In the second scenario, features are extracted from each time series in the subset by using Symbolic Time Series Analysis with $D \geq 1$ (see Section 7.2) for two combinations: (i) $|\Sigma| = 3$, $N_{max} = 20$, $\eta = 0.02$ and (ii) $|\Sigma| = 5$, $N_{max} = 20$, $\eta = 0.02$. Since $\eta = 0.02$ results in the best performance, this $\eta$ value was chosen for the comparison. The rationale behind choosing $N_{max} = 20$ for this comparison, is that they represent a reasonable computational
Figure 7.3: Standard deviation of mean error over 20 runs

Figure 7.4: Standard deviation of uncertainty over 20 runs
Figure 7.5: Profile of $P_{rms}$ plot for a typical subset

complexity, and sufficient flexibility given to the algorithm to find a good model fitting the data. Again, the feature associated with $CL_{stable}$ is designated as the nominal feature, and the divergence of all other features from the nominal feature is calculated. Thus, for each subset, a set of $P_{rms}$ divergence is calculated, along with the corresponding feature divergences for both $D = 1$ and $D \geq 1$. These divergence sets are individually normalized (i.e., divided by the maximum value in the set and thus making $[0,1]$ the range of the divergence). Next, the correlations are computed between the $P_{rms}$ divergence and the feature divergence for $D = 1$, where the correlation coefficient of two random variables is a measure of their linear dependence. Similarly, the correlations between $P_{rms}$ divergence and the feature divergence for $D \geq 1$ are computed. The entire procedure is repeated for all 18 subsets.

Figure 7.7 shows the correlations of $P_{rms}$ divergence with those of feature divergences for ($D = 1$) and ($D \geq 1$) with $|\Sigma| = 3$, $N_{max} = 20$, $\eta = 0.02$. The correlation of ($D = 1$) feature divergence is consistently lower than that of ($D \geq 1$) for 15 out of the 18 subsets. For the remaining three subsets, the correlation value is almost the same in the two cases; the rationale behind this behavior is that, for these
three cases, the simplest model (i.e., with $D = 1$) is possibly the best fitting model even after state splitting and state merging, and it is also possible that the simplest model is much less prone to data overfitting. To investigate the effects of the alphabet size $|\Sigma|$, which is a critical design constraint, Figure 7.8 shows a similar comparison with $|\Sigma| = 5$, while $N_{\text{max}} = 20$ and $\eta = 0.02$ are unchanged. The results are clearly in favor of $(D \geq 1)$ for 16 out of 18 subsets. In the remaining two subsets, $D = 1$ appears to perform slightly better than $D \geq 1$ possibly due to the effects of data overfitting.

### 7.4 Summary, Conclusions, and Future Work

This chapter illustrates a proposed dynamic data-driven methodology, based on the Bayesian nonparametric theory, as a tool of combustor design for gas turbine engines. Using experimental data from a laboratory-scale combustion apparatus, it is observed that the proposed design algorithm yields, on the average, accurate estimates of the system response with high confidence. The advantages of using higher depth $D$-Markov models (i.e., $D \geq 1$ instead of $D = 1$) have also been
investigated. By making the choice of $D \geq 1$, the proposed algorithm is shown to yield features consistently more correlated with $P_{rms}$, which is used as a good indicator of system stability, as compared to the work in the previous chapter with $D = 1$. Possible areas of future research have been delineated in the previous chapter.
Figure 7.8: Correlations of $P_{rms}$ divergence with $D = 1$ and $D \geq 1$ feature divergences for $|\Sigma| = 5$, $N_{max} = 20$, $\eta = 0.02$
Summary & Future Work

8.1 Contributions of the Dissertation

The contributions of the dissertation are summarized below.

1. **Unsupervised Context Extraction and Context Based Event Classification**: If training data consists of sensor measurements collected over a long time, the underlying context might have changed over time. In such cases, if only class labels are available for the data, the different contexts need to be identified, and classifiers trained separately for each context (since the class conditional distribution may be different for each context). Chapter 3 develops an unsupervised algorithm for context extraction, using concepts of graph theory. The proposed algorithm has been validated on real experimental data. The proposed context aware pattern classification algorithm outperforms the single classifier trained on the entire data. In addition, the average performance is the same as the case where ground truth context labels are available.

A more difficult scenario, is when at testing time the context is different from the one at training time (due to environmental disturbances). Algorithms successful in disturbance-free situations may fail to perform satisfactorily when there are moderate perturbations in the environmental conditions of real-life situations [98]. Hence, robustness to disturbances in environmental conditions is a highly desirable property of feature extraction for target de-
tection & classification. Chapter 3 also proposes some analytical measures using the concept of area under the ROC curve, to quantify robustness of algorithms. It then makes a evaluation of the robustness of some commonly used feature extractors, on an experimental dataset.

2. **Alphabet Size Selection for Symbolization of Time Series** : Chapter 4 proposes an information-theoretic approach to feature extraction from time series data, that maximizes the mutual information between the extracted features and class labels. A sequential partitioning technique for partitioning the range space of a time series is developed, such that at every step, the mutual information between the resultant features and the class labels is increased. This technique yields good performance of pattern classification both on simulation and real data (laboratory-scale combustor for lean blow-out (LBO) prediction). In addition, the robustness of the partitions to noise, and the resultant effect on performance has also been investigated. The performance of the proposed approach has also been compared with a standard partitioning tool - maximum entropy partitioning.

3. **Quantification of effect of limited training and testing data on SOC identification** : Most data-driven methods require adequate length of training and test data in order to extract good features, which is a clear disadvantage in real time estimation problems. The proposed data driven, electrochemistry independent method in Chapter 5, quantifies the effect of training and testing data lengths on the estimation accuracy. In addition, when validated on real experimental data, it is found to provide good and consistent performance, even with short-length test data under different operating conditions, which also demonstrates the robustness of the extracted features.

4. **Dynamic Data Driven Combustor Design** :
Combustor dynamics is highly non linear, and occurs at a fast scale. Hence, traditional health monitoring of combustors is not effective, since once instability sets in, it is not possible to reverse the damage. Thus, passive preventative combustor design is essential. Chapter 7 and 8 develop two data driven algorithms for combustor design. Given limited data at some operating conditions, this dissertation develops algorithms for determining the
mapping from operating condition to system stability. Using this mapping, the possibility of system instability at any unseen operating condition can be predicted. Apart from providing estimates of stability at unseen conditions, the proposed algorithms also provides confidence in the estimates. Thus, given the estimate of system stability and the associated confidence, at all operating conditions, the designer can choose the system parameters, which would have a statistically insignificant probability of resulting in instability.

Furthermore during the course of analysis, the PFSA feature extracted from the time series, in the proposed algorithms, has been shown to be more sensitive to system deviations compared to $P_{rms}$, a commonly used feature for combustor stability detection. This is very useful for combustor stability detection, because even a small deviation from the nominal state, is manifested as a large divergence, leading to conservative estimates of system stability, and consequently lower probability of a potentially unstable operating condition being chosen at the design stage. It has also been observed that features extracted from shorter time series, are still capable of encapsulating the dynamics, indicating lower dependence on data length in comparison to $P_{rms}$.

The second algorithm developed in this dissertation builds upon the first algorithm. Instead of a fixed structure algorithm, the second algorithm is dynamic, because no fixed structure has been assumed for the model that captures the information of the dynamics of the time series. Instead, for every time series corresponding to a distinct operating condition, the best model describing it is found out, under user-selected constraints of accuracy and redundancy. An information-theoretic measure is then used for comparing among these models which might be different in structure and/or parameters. It is observed that the second algorithm yields, on the average, better estimates of the system response with higher confidence, compared to the first algorithm. In addition, it also yields features more strongly correlated with $P_{rms}$, which when calculated on a long window of data, is known to be a good indicator of stability.
8.2 Future Work

Future research directions specific to the individual algorithms presented in the dissertation are mentioned at the end of the corresponding chapters. This section presents the broad areas of possible future research.

- **Extraction of Robust Features**: Chapter 3 proposed measures to evaluate robustness of feature extractors. A possible future work is to use this formulation to actually engineer robust features, i.e. formulate the objective of the feature extraction algorithm as extracting features which are discriminative of classes, while being robust to small perturbations in the environment.

- **Transfer learning**: All supervised machine learning algorithms need sufficient data to learn how to discover patterns. But if an agent comes to a new environment, it might not have access to sufficient training data. New labeled data might be coming in intermittently. A future line of work could be to formulate a sequential algorithm which combines old information (from previously visited environments) with new information coming in intermittently.

- **Fusion of Data Driven and Physics Based Methods**: Integrating dynamic data-driven methods with model based methods is an important direction of future research. Physics based models could be designed based on available domain knowledge, and then data driven methods could be used to fill in the gaps i.e. estimate the remaining unaccounted portion.

- **Active control of combustors**: Another area of research could be active control of combustors. Given data generated from simulation mimicking the actual combustor system, which spans different regimes and transitions, algorithms could be trained to determine the best choice of corrective action to be taken, given current estimated state of the combustor. This algorithm could then be used for active control of combustors.
Bibliography


Vita

Pritthi Chattopadhyay

Pritthi Chattopadhyay received her B. Eng. Degree in Instrumentation and Electronics Engineering in 2012 from Jadavpur University, Kolkata, India. She joined Penn State as a doctoral student in Mechanical Engineering in Fall 2012. She received a concurrent MS degree in Electrical Engineering from Penn State University in December 2015. Her research interests are machine learning, statistical modeling, time series analysis, signal processing.