The Pennsylvania State University
The Graduate School
College of Engineering

LEARNING AND DECISION OPTIMIZATION IN DATA-DRIVEN AUTONOMOUS SYSTEMS

A Dissertation in
Mechanical Engineering
by
Devesh Kumar Jha

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2016

The dissertation of Devesh Kumar Jha was reviewed and approved* by the following:

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

Christopher Rahn
Professor of Mechanical Engineering

Jan Reimann
Assistant Professor of Mathematics

Thomas A. Wettergren
Adjunct Professor of Mechanical Engineering

Minghui Zhu
Assistant Professor of Electrical Engineering

Abhishek Srivastav
Special Member, GE Global Research

Karen Thole
Professor and Department Head, Mechanical Engineering

*Signatures are on file in the Graduate School.

Abstract

Modern human-engineered systems like self-driving cars, smart buildings, power grids etc., are becoming increasingly complex so that they present unparalleled challenges for their design, verification and control. Furthermore, these systems generate data at an unprecedented rate. Even though the data is supposed to be of great help in understanding these systems, it has already presented unique challenges for storage, leave alone their analysis and interpretation (for example, Google is supposed to analyze around 20 Petabytes of data everyday and it is expected that the data generated by systems industry will very soon surpass that). Machine learning provides very powerful statistical techniques for analysis and interpretation of large amounts of data and recently has seen great improvements pushed by large-scale applications in software industries. While there has been huge advancements in the individual fields of machine learning and control theory, unfortunately the modern decision systems for complex systems do not take into account the recent advances of machine intelligence for intelligent control. The role of machine learning in systems engineering is still unexplored; however, it holds tremendous potential for achieving human-like (or better) performance in modern autonomous systems, which is the ultimate goal of Artificial Intelligence. Establishing a synergistic relationship between state-of-the-art control and machine learning techniques holds the key to the synthesis of complex systems with high degrees of autonomy.

This thesis explores various concepts and makes contributions to multiple areas for understanding of complex data-driven systems for their decision and control. Among several challenges of complex data-driven systems, one of the central challenge is understanding and representation of data using tools from mathematics and statistics. In this regard, this thesis presents several results for modeling of sequential data using Markov models and Deep Learning. The proposed ideas are tested on various electro-mechanical systems like combustion in

lean pre-mixed systems, fatigue in polycrystalline alloys, event detection in sensor networks and lead-acid battery systems. This thesis brings together a completely interdisciplinary approach using techniques from mathematical analysis, theoretical computer science, machine learning, game theory and control theory to present a framework to solve various challenges posed by modern autonomous systems with fundamental contributions to robotic motion planning and temporal data analysis. In order to bring together control theory and machine learning for a class of modern autonomous systems (e.g., automatic driving in urban scenarios with traffic rules), the thesis presents research for the conditions under which we can bring together these fields in an optimal fashion while keeping desired properties like safety, stability, etc..

This thesis makes fundamental contributions to analysis and representation of time-series data using symbolic dynamics and information theory. In particular, algorithms for order estimation and compact representation of Markov models of time-series data are proposed which have computational advantages. Furthermore for analysis of high-dimensional temporal data, an algorithm using Deep learning and Gaussian processes is proposed. The thesis also makes fundamental contributions to motion planning of robotic systems where a class of sampling-based feedback motion planning algorithms are proposed. The algorithm is extended for motion planning of multiple robots using game theory. Furthermore, data-driven algorithms are proposed to bring together learning and planning in complex robotic systems.

Table of Contents

List of Figures **ix**

List of Tables **xiii**

Acknowledgments **xiv**

Chapter 1
Introduction and Motivation **1**
1.1 Introduction 1
1.2 Motivation 2
1.3 Contributions of Thesis Research 7
1.4 Organization of Thesis 9

Chapter 2
Symbolic Analysis-based Markov Modeling of Time-Series Data **11**
2.1 Mathematical Preliminaries and Background 16
2.2 Problem Formulation 25
2.3 Partitioning of Phase Space 27
2.3.1 Unsupervised Partition Techniques 29
2.3.2 Supervised Partition Techniques 30
2.4 Order Estimation in Markov Chains 30
2.4.1 Log-likelihood rate 31
2.4.2 Signal reconstruction 31
2.4.3 State splitting using entropy rate 32
2.4.4 Depth estimation using spectral analysis 33
2.5 Parameter Estimation for Markov Chains 34
2.6 Sequential Tests for D-Markov Models 36

2.6.1 Likelihood for D-Markov models . . . 37
2.7 xD-Markov Modeling . . . 39
2.8 Statistical Learning Examples from Engineering Systems . . . 41
2.8.1 Fatigue Damage in Polycrystalline alloy structures . . . 42
2.8.1.1 Experiment Details . . . 42
2.8.2 D-Markov Modeling . . . 43
2.8.3 Results and discussion . . . 44
2.8.3.1 Unsupervised Clustering of Fatigue Data . . . 48
2.8.3.2 Concluding Remarks . . . 49
2.8.4 Input-Output Modeling in Battery Systems . . . 50
2.8.4.1 xD-Markov Modeling . . . 52
2.8.4.2 Results and Interpretation . . . 53
2.9 Extensions to Multi-Sensor Fusion . . . 57
2.9.1 Problem Statement for Local Target Detection in Sensor Networks . . . 58
2.9.2 Proposed Technical Approach . . . 59
2.9.2.1 k-means Clustering . . . 61
2.9.2.2 Agglomerative Hierarchical Clustering . . . 62
2.9.2.3 Spectral Clustering . . . 62
2.9.2.4 Hypothesis Testing with Feature-level Sensor Fusion 64
2.10 Future Research Directions . . . 66

Chapter 3
Reduced-Order Markov Modeling of Time- Series Data 69
3.1 Motivation and Introduction . . . 70
3.2 Background and Mathematical Preliminaries . . . 73
3.3 Proposed Approach . . . 74
3.3.1 Estimation of Reduced-Order Markov Model . . . 75
3.3.2 Parameter Estimation of the Reduced-Order Markov Model 78
3.3.3 Model Selection using information theoretic criteria . . . 80
3.4 Analysis of the Proposed Algorithm . . . 82
3.5 Experiment and Data Details . . . 85
3.5.1 Combustion . . . 85
3.5.2 Bearing Prognostic Data . . . 86
3.6 Markov Modeling . . . 86
3.6.1 Combustion . . . 86
3.6.2 Bearing . . . 90
3.7 Classification and Anomaly Detection Results . . . 91
3.7.1 Anomaly Detection . . . 92
3.7.2 Classification . . . 94

3.8 Conclusions and Future Work 97

Chapter 4
Temporal Learning in Video Data Using Deep Learning and Gaussian Processes **100**
4.1 Introduction 101
4.2 Preliminaries and Problem Formulation 103
4.2.1 Convolutional Neural Networks 104
4.2.2 Gaussian Processes 105
4.2.3 Problem Statement 106
4.3 Combustion Experiment Details 106
4.4 Results and Discussion 109
4.5 Conclusions and Future Work 118

Chapter 5
Anytime Algorithms for Policy-based Motion Planning of Robotic Systems **121**
5.1 Motivation and Introduction 122
5.2 Problem Formulation 124
5.3 Background 126
5.4 The Incremental Feedback Policy Algorithm 127
5.4.1 Primitives 127
5.5 Performance Analysis 131
5.6 Numerical Results and Discussion 133
5.6.1 Extensions 136
5.6.2 Experimental Validation on a Robotic Test Bed 136
5.7 Conclusions and Future Work 137

Chapter 6
Data-Driven Algorithms for Anytime Motion Planning with Safety Guarantees **148**
6.1 Introduction 149
6.2 Problem Formulation 151
6.3 Data-Driven Anytime Robust-Adaptive Algorithm 154
6.3.1 Machine Learning-based Estimator 155
6.3.2 Controller Synthesis using Iterative Incremental Game Algorithm 159
6.4 Performance Analysis 163
6.5 Conclusions and Future Work 165

Chapter 7
Conclusions and Future Work **166**
7.1 Conclusions of Thesis Research 166
7.2 Future Research Directions 168

Appendix A
Instability in Combustion Systems **171**

Appendix B
Fatigue Crack Initiation in Polycrystalline alloys **176**

Bibliography **179**

List of Figures

1.1 Schematic drawing of the paradigm of cyber-physical systems which are used to model modern control systems 3

2.1 Markov modeling of time series data and the associated decisions of partitioning and discrete modeling 22
2.2 Schematic diagram of $\mathbf{x}D$-Markov machines, where the (temporal) dynamics of a symbolic stochastic sequence are captured relative to another such sequence 24
2.3 Tree-representation of state splitting in D-Markov machines 33
2.4 Fatigue test apparatus used for experiment 42
2.5 Comparison of depth estimation using (a) spectral decomposition (b) log-likelihood (c) state splitting via entropy rate and (d) signal reconstruction. The metric ρ used for estimating the depth for each approach has been normalized as $\rho_{\text{norm}} = (\rho - \rho_{\min})/(\rho_{\max} - \rho_{\min})$ in the above plot for easy comparison. Note that this normalization does not affect the depth selection process. 45
2.6 Entropy rate against $N_{\max}$ for state-splitting method. models with same depth are shown in same color and separated by vertical lines; and the minimum entropy rate achieved for a fixed value of depth is marked by a circle 47
2.7 Profile of input current data for the experiment 52
2.8 Cross entropy rate vs number of splits (SOH = 1, 3 symbols, downsample time-lag = 7) 53
2.9 Normalized Hamming distance vs number of states (SOH = 1, 3 symbols, downsample time-lag = 7) 54
2.10 Error in prediction of output with decreasing health) 55
2.11 Anomaly measure vs state of health (SOH) 56
2.12 SOC estimation using the regression-based filter. SOH= 1 56
2.13 Feature-space illustration in a dynamic background environment . . 60
2.14 Examples of binary tree structure formed by multiple sensors 63

3.1 Graphical model showing non-determinism in a PFSA. The symbol 1 emitted from state q_1 leads to different states with fixed probabilities indicating non-deterministic behavior. 74
3.2 The symbol emission probabilities for a Markov chain with 3 symbols are shown on a simplex. Symmetric K-L distance is used to find the structure in the state-set in the information space and the states are clustered based on the revealed structure. 77
3.3 Graphical models representing the dependencies between the random variables 79
3.4 Autocorrelation function of time-series data during unstable phase of the combustion process. The time-series data is down-sampled by the lag marked in red square. It is noted that the individual time-series have their own down-sampling lags. 86
3.5 The first plate in the above Figure shows the change in the empirical density calculated for the pressure time-series data as the process deviates from the stable operating condition to unstable operating condition. The second plate shows the spectral decomposition of the 1-step stochastic matrix for the data under stable and unstable operating conditions. 87
3.6 Hierarchical clustering of states during stable and unstable conditions. 88
3.7 Unsupervised model selection during stable and unstable conditions. 90
3.8 Box plot for Hamming distance between the original and reduced-order models obtained after merging based on the results in section 3.4 91
3.9 The model scores using the BIC and AIC criteria and the final model selected is depicted by the black rectangle. 91
3.10 Box plot of the Hamming distance between the original and reduced-order models along with the analytical bound presented in section 3.4. 92
3.11 Anomalous behavior of data during the combustion process 95
3.12 Variation of discrepancy statistics $H_{\mathcal{M}}$ with increasing pressure fluctuations. This also shows an anomaly around the point 200 and qualitatively agrees to the behavior of $\Delta_{\mathcal{M}}$. 96
3.13 Cluster of stable and unstable phase in information space. Each point is a row of the emission matrix for the reduced Markov model with 2 states. The plot shows the change in the Markov model as the process moves from stable and unstable. Red diamonds represent the unstable phase while green diamonds represent the stable phase. 96
3.14 Schematic drawing of test facility at Penn State center for Propulsion. 97

4.1 This figure shows the deep convolutional neural network used with two convolutional layers and two hidden layers. The number of kernels and number of units in the hidden layer are varied to study the affect of size of parameter set on the results. The convolution layers are denoted by C_i and the pooling layers are denoted by S_i. Each rectangular image is a feature map. 104
4.2 Concept of the proposed modeling scheme using the deep convolution net and Gaussian processes which is used to model the hidden dynamics in the sequential image data. 107
4.3 High-speed image data from the stable and unstable phases of combustion. Spatial and temporal changes in the flame structure during the unstable process are visible. The flame enters on the right end and moves towards the left. 108
4.4 Change in image data as the combustion process moves from stable to unstable. From the approximate empirical density and the limit cycle shown in sequence of the euclidean norm between the images, change from minor fluctuations to a quasi-periodic behavior is visible 110
4.5 Autocorrelation of the image residuals (4.4b) during the unstable phase of combustion indicating the limit cycle and its frequency (~ 26) 110
4.6 Performance of Gaussian Process on the residual of the sequential images. With sufficient memory, we are able to achieve perfect performance ($M = 40$); with reduction in memory, we see a reduction in performance. 111
4.7 Performance of the trained deep learning algorithm on transient data 114
4.8 In this figure, we show typical snapshots of flames just before and after the switching detected by the CNN likelihood ratios for figure 4.7a (similar changes are observed for figure 4.7b). The flame enters on the left side and moves towards the right in every image. If seen closely, one can see the flame breaks near the entrance which is due to the periodic behavior that occurs during unstable phase. 115
4.9 Performance of the trained deep learning algorithm and the Gaussian process on the deep learning features. 116
4.10 Performance of the composite deep learning and Gaussian process algorithm on transient data 117
4.11 (a)Schematic of the experimental setup. 1 - settling chamber, 2 - inlet duct, 3 - IOAM, 4 - test section, 5 - big extension duct, 6 - small extension ducts, 7 - pressure transducers. The figure also shows the swirler plate geometry and the mixing tube of the swirler is also shown in detail. 119

5.1 The estimated cost-to-go function obtained by the proposed incremental algorithm for point mass from different initial conditions. . . 139
5.2 The estimated cost-to-go function obtained by the proposed incremental algorithm for Reeds-Shepp system obtained after 19000 iterations. It shows the asymmetry and also the transverse movements possible due to the Lie Bracket operator. Notice that the levels sets are stretched in the longitudinal direction i.e., parallel to the orientation of the car at the goal. (Arrow shows the orientation of the car at the goal) 140
5.3 The path connecting $(3, 3, \pi/9)$ to $(15, 15, \pi/9)$. 141
5.4 The path connecting $(6, 4, 5.5)$ to $(9, 9, \pi/2)$. 142
5.5 The path connecting $(7, 5, 5.34)$ to $(15, 15, \pi/9)$. 143
5.6 The estimated cost-to-go function obtained by the proposed incremental algorithm for Dubins car system obtained after 19000 iterations. As the Dubins car can't move backwards, the value functions are discontinuous and it results in more complicated reachable sets in the sub-Riemannian manifold. (Arrow shows orientation of the car at the goal.) 144
5.7 Picture of the RC car built in the robotics lab at Penn State 145
5.8 This figure shows the performance of RRT* with PID control for trajectory following. In the presence of disturbance,a replanning is required for collision avoidance 146
5.9 This figure shows the performance of feedback RRT where replanning is avoided for the inherent nature of state-based feedback. 147

6.1 The Control Loop with the Machine Learning-based estimator for Motion Planning 153
6.2 Controller synthesis using the kernel-based regression estimator . . 160

7.1 Schematic drawing of the the idea for synthesis of feedback controllers for motion planning under complex constraints 169

A.1 A simplistic schematic diagram showing the self-excited feedback loop to describe combustion instability. 173

B.1 Evolution of the Surface Profile with Microstructural Damage . . . 177

List of Tables

2.1 Cluster purity for five samples for different depths 49

3.1 Operating conditions 85
3.2 Performance of classifiers with different number of states. Mean Error= Lower is better. 97

4.1 Some examples of covariance functions used for Gaussian Process. The first and second are Squared Exponential (SE) and the third function is a rational quadratic (RQ). 107
4.2 Experiment set 1: Protocols with respective ground truth conditions for data collection. 3 s of greyscale image sequence at 3 kHz. 108
4.3 Experiment set 2: Protocols to produce transition from stable to unstable combustion 108
4.4 Size of convolution filters 117
4.5 Results of classification performance for baseline and **proposed** models. AUC= Area under curve (higher is better); FPR@99.7 is False positive rate at 99.7% detection rate (lower is better) 117

Acknowledgments

First and foremost, I would like to express my sincere gratitude for my adviser Dr. Asok Ray. The freedom to think freely, motivation and critique provided by him has been instrumental for this thesis research. It is my pleasure to share my sincere gratitude for my committee members, Dr. Minghui Zhu, Dr. Jan Reimann, Dr. Abhishek Srivastav and Dr. Tom Wettergren. Most of the work done in this thesis is an outcome of discussion on topics of our common interest. The numerous discussions with them have been invaluable to the progress of this thesis and my understanding of different complex concepts. I am indebted to them for their patience while my ideas were taking shape. I would also like to thank Dr. Chris Rahn for sharing his experience through comments on the thesis.

I feel myself really fortunate to had the opportunity to work with Dr. Kushal Mukherjee and Dr. Soumik Sarkar during my early graduate days and thank them for all valuable suggestions that moved forward my research. Special thanks has to go to Dr. Eric Keller for bearing my stupid ideas with design of different experiments done during my graduate school. I would like to thank my colleagues Dr. Yue Li, Mr. Nurali Virani and Dr. Soumalya Sarkar for sharing vision and knowledge in our academic projects. I would like to thank Mr. Zhenyuan Yuan, Mr. Chuong Nguyen for the excitement and hard work they brought to our robotics lab. Thanks to all other members of Dr. Ray's research group for their professional association.

I must say here that none of this would have been possible without the love and support of my parents and family along the way. I am deeply indebted to them for their dedication, encouragement and inspiration without expecting anything in return. A special thanks to my uncle and his family who were always there for emotional support.

Finally, I would like to take this opportunity to thank all my friends in State College and India for all the fun memories. A special thanks to my roommates in State College, Debasish and Vinod for everything.

Lastly, I would like to gratefully acknowledge the financial support I received throughout these years: the U.S. Air Force Office of Scientific Research under Grant

FA9550-15-1-0400 and the U.S. Office of Naval Research under Grant N00014-14-1-0545.

Dedication

This thesis is dedicated to my parents and family for their unwavering support, encouragement and dedication. Thanks for never giving up on me.

Chapter 1

Introduction and Motivation

1.1 Introduction

Artificial Intelligence (AI) has seen tremendous improvements recently and is most likely supposed to bring the next revolution in human history. It is already being used at commercial scale for improving performance, reliability and efficiency of software as well as physical systems. Seemingly extreme concepts of artificial intelligence systems like self-driving cars, smart buildings and Internet of Things (IoT) are inching closer to reality and have received much interest in the research community. These engineered systems are supposed to have capabilities that could enable a large set of actions to be automatic or, within programmed boundaries, i.e., "self-governing". However, the modern engineered systems are becoming increasingly complex and present unparalleled challenges for their design, verification and control. There has also been an unprecedented increase in the volume and speed of data being generated by human engineered systems due to the rapid advancements in low-cost sensing & storage. Furthermore, with the proliferation of mobile communication devices, system inter-dependence has taken a new meaning. It is expected that the next wave of innovation in harnessing the potential of these complex systems will come from explicitly modeling the non-linearities arising from system dynamics, complex interactions and uncertain external disturbances. However, achieving this goal completely from the fundamental principles of physics is extremely challenging, if not impossible. The wealth of data being generated by these systems, on the other hand, presents new opportunities for analysis and synthesis of intelligent behavior

in the modern complex systems. It is expected that autonomy in the modern engineering systems could only be achieved by a synergistic integration of state-of-the-art control & planning approaches with modern machine intelligence techniques. While machine learning and statistical learning has found large-scale use in software industry, its role in engineering systems has been limited and happens to be largely unexplored [1]. This is mainly due to the unreliable, low-level representations possible by state-of-the-art machine learning and statistical learning algorithms as well as the constraints on flexibility of the decision systems towards using those representations. For example, consider the problem of predictive modeling of the environment for self-driving cars in traffic. Situation awareness for such systems requires detection (low-level representation) and reasoning of inter-relationship between the elements (high-level understanding) present in the environment. On the other hand, the decision system has to adapt its plan based on the new state of the environment with the understanding that there might be uncertainties associated with these representations. While the overall scope of the associated problems is much larger, this thesis makes an attempt to investigate and improvise to solve some of the weak links for learning and decision optimization in data-driven systems for intelligent behavior.

1.2 Motivation

Autonomy means the ability and authority for self-governance [1]. Autonomous systems, however, are not new to humankind. A regular feedback control system is, for example, autonomous with regard to stability goals and to some extent, with respect to certain level of external and internal disturbances. However, these control systems have a low degree of autonomy as they can only tolerate a restricted class of parameter variations and disturbances, automate some very simple goals like stability and regularization. Modern engineered systems are expected to achieve very high degrees of autonomy with the goal to achieve human-like performance and capabilities. However, to achieve a very high degree of autonomy, the controllers must be able to perform a number of functions in addition to the conventional control functions like tracking, regulation and stability. The goals of these systems are more complicated to be expressed by the current controller synthesis algorithms (e.g., the self-driving cars need to follow the traffic rules, need to learn their surroundings and

adapt, etc.). Furthermore, the complexity of the systems and their goals require fundamental changes in the system verification and validation techniques. As such the design and verification of modern engineered autonomous systems, with a tight link between computational and physical elements, have become increasingly complex due to the interleaving between the high-level logic of the goals and the low-level dynamics of these systems. Consider, for example, an autonomous vehicle driving in an urban-like environment cluttered with static and dynamic obstacles while obeying traffic rules. Any such vehicle has to be able to capable of negotiating intersections, handling changes in the environments or operating conditions and re-planning in response to such changes. This requires integration of high-level logic with the low-level controller that regulates the physical hardware incorporating the low-level dynamics. The long-endurance nature of these complex autonomous missions also mandates the use of real-time sensory information for situation awareness for estimation of various parameters of the system during operation. The data sensed by these systems is expected to play a central role in optimizing and improving their performance. Thus, there is a requirement of seamless integration of the disparate fields of statistical learning theory with modern control theory as well as symbolic models of decision-making developed in theoretical computer science.

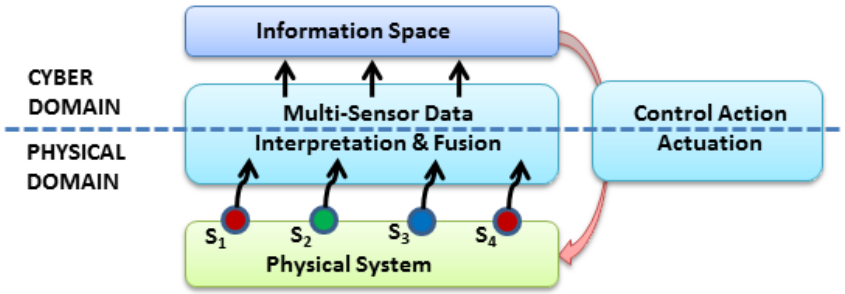


Figure 1.1: Schematic drawing of the paradigm of cyber-physical systems which are used to model modern control systems

Figure A.1 shows the generic idea of data-driven system where the physical system is modeled using the available models of, possibly, a distributed network of system components and the information available from, possibly, a distributed

sensor network is modeled in the cyber layer (thus, they are also known as cyber physical systems). These two layers then interact through a controller. The information (or cyber) space broadly consists of the different algorithms for decision-making and learning using the data observed by the sensing system of the complex physical system. Strictly speaking, this framework is a generalization of the classical feedback control system; however, the controller may be synthesized in a distributed or decentralized fashion and the information may be conditioned using advanced statistical learning algorithms. This provides the flexibility to model a very large class of systems ranging from modern autonomous driving cars to internet of things or control of physical processes like combustion or air-conditioning systems in smart buildings. Thus, it is expected that fundamental contributions to these algorithms will find applications in a multitude of systems.

The current state-of-the-art typically consists of separate approaches to perception and planning, with relatively little work on integrated intelligent autonomous systems. The disparity in the planning and perception module leads to difficulties in the development of a unified, formally verifiable theoretical framework for machine-perception driven planning and control. As a result, despite enormous success of the theoretical frameworks for controller synthesis, the current artificial intelligence techniques are still brittle, where small mistakes in the outcome of the perception module propagate in the planning module leading to inefficient performance [2]. There are several shortcomings in the current understanding of different components of artificial intelligence (e.g., signal processing, machine learning and planning). The most common approaches to AI are data-driven, involving statistical learning, the fundamental problems are associated with understanding data and learning its corresponding representations which can then interact with a decision system for control. Furthermore, the decision systems do not have the flexibility to deal with the low-level, uncertain estimates provided by the statistical intelligence techniques while allowing optimal performance for the system. Some of the challenges, among various others, in achieving a synergistic operation of these data-driven complex systems are enumerated next.

1. **Data to Knowledge:** This is one of the biggest areas of active research in machine learning and statistical learning, and holds the key to advancements in understanding of a large class of data-driven systems. Understanding semantic representation of data has remained one of the biggest unsolved

puzzles in learning theory where, unfortunately, very little progress could be made over the last few decades. Understanding time-series, its mathematical characterization and representation are central to this problem. This could be treated as one of the critical problems in understanding of data-driven systems. However, this problem is challenging for the following reasons.

- Absence of any underlying model for the systems generating data makes defining metrics for performance of models for representation and learning elusive.
- There is no universal metric to decide the free parameters (also called hyperparameters) for the data-driven models and it is generally dependent on the system being analyzed. Most of the widely-used methods are largely empirical (like cross-validation). These models are then selected using an empirical approach like cross-validation.

Thus, these models become a lot dependent on hand-tuning of the free parameters resulting in huge requirements on domain expertise for analysis of complex systems. While there various schools of thought for the inclusion of domain expertise in learning, it is desirable that the learning algorithms can be trained independent of the domain with the same effectiveness. There is a requirement for mathematical formulation of these problems when the current state-of-the-art is mostly driven by empirical analysis.

2. **Statistical Modeling and Analysis of data-driven models:** To realize the full potential provided by the data-driven systems, inference of accurate model structure (e.g., identification of right network architecture hyperparameters when using deep learning or state-space of Markov models) and efficient estimation of model parameters which can be used for embedded application in real-time. This problem is, in general, very hard for the non-convex nature of the parameter space of the models. However, establishing mathematical consistency of the models is important for better reliability of the data-driven, statistical algorithms.

3. **High Performance Decision Systems:** For synthesis of the high performance control and decision-making systems, it is important to be able to achieve optimal performance for these systems. This part of the artificial

intelligence techniques makes use of the available models and receives estimates of the associated system parameters from an observer (model-based or statistical). Establishing optimality, robustness and time-complexity are some important objectives for this task.

4. **Safety in Artificial Intelligence Systems:** Flexible design of the decision-making system so that they can interact with, possibly, a statistical observer is critical to the reliability of the artificial intelligence systems [3–6]. However, this problem is very challenging for it requires guarantees on the rate of convergence of the statistical algorithms and generalization of the guarantees on convergence of the statistical estimators (e.g., most convergence guarantees could be established for i.i.d. data and an important problem is investigation of convergence under non-i.i.d. data). Furthermore, design of robust, adaptive controllers for the complex systems is very challenging and only a very few algorithms are available in open literature for the same.

With this vision and understanding the thesis investigates several shortcomings in understanding & representing time-series data and presents several techniques for representation of time-series data. Specifically, the thesis investigates representation and learning of time-series data using concepts from symbolic dynamics. The proposed technique allows compact representation of the data while preserving essential temporal statistics of the data. The advantage is faster convergence of the model parameters which result in faster, yet reliable, decisions during test. The proposed techniques are tested through numerical experiments as well as experimental data from various physical processes like combustion instability, fatigue crack initiation and health degradation in battery systems. There are various other extensions of the techniques in event detection, target detection, sensor fusion which is also very briefly discussed to show the flexibility of the technique.

One of the weaknesses of the symbolic analysis-based temporal modeling is scalability to higher dimensions. While the work presented in this thesis doesn't consider discretization of high-dimensional data to symbols, symbolization of high-dimensional data is challenging. Thus, a symbolization-based analysis would depend on the efficiency of the discretization process. Deep learning, on the other hand, can effortlessly handle high-dimensional data; however, modeling temporal sequence of high-dimensional data results in increase of the number of parameters to be

estimated. This thesis also presents a technique of learning high-dimensional time-series using deep learning and Gaussian processes with several advantages that discussed later in the thesis. This technique has the advantage that it allows to learn the temporal statistics of high-dimensional data using the best features of deep learning and Gaussian processes while using comparatively lesser number of parameters to learn.

With the vision of being able to design and synthesize controllers which can use statistical learning algorithms as estimators, the thesis presents several algorithms for feedback motion planning of robotic systems which allow state-based control of these systems. Convergence and optimality of these controllers are established and we also present some ideas of integrating statistical learning for control of robotic systems with retaining safety guarantee for the systems.

1.3 Contributions of Thesis Research

The thesis research makes fundamental contributions to representation learning in time series analytics using tools from symbolic dynamics and deep learning. Furthermore, the thesis introduces and analyzes a class of sampling-based feedback controllers for motion planning of robotic systems which have been extended to motion planning planning of multiple robots using game theory. Finally, the thesis also explores the idea of safety in artificial intelligence using game theory and statistical learning theory. The contributions of the thesis research are enumerated next.

1. The thesis presents a technique for compact representation of time-series data for representation learning using a non-deterministic finite state automata. The technique is based on concepts of symbolic dynamics, clustering of directed graphs and minimum description length. The proposed technique has the advantage that it allows to infer a reduced-order model for the time-series data where fewer parameters need to be estimated during test. Furthermore, the thesis presents a sequential test for Markov models of time-series data which can be efficiently used for faster detection and estimation with streaming data. Several extensions of the symbolic dynamics-based model are also discussed with examples to show the flexibility of the technique.

2. The thesis presents a technique for temporal modeling in video data (high-dimensional time-series) using deep learning and Gaussian processes. In the proposed technique the problem is decomposed into feature learning and temporal learning using deep learning and Gaussian processes respectively. The technique is validated on experimental data of combustion instability obtained from a swirl-stabilized combustor in laboratory settings.

3. The thesis presents a sampling-based anytime feedback controller using sampling-based algorithms and dynamic programming. Theoretical guarantees on the convergence and asymptotic optimality of the algorithm are also provided. The algorithm is extended to multiple robots using non-cooperative differential game theory and an incremental algorithm with convergence and optimality guarantees is also provided in the thesis.

4. The thesis presents state-of-the-art results for detection and estimation of combustion instability using Markov models and deep learning. Several results for detection and estimation of instability in combustion systems is provided using Markov modeling and Deep learning techniques. Several benchmarks in terms of detection accuracy and lead time to detection are established based on these results.

5. The thesis also explores the topic of safety in artificial intelligence using game theory and statistical learning theory. In particular, the thesis presents an anytime algorithm for motion planning which retains the guarantees on the safety of the system while it is learning from observations.

The hope is that the algorithms presented in this thesis will help alleviate several problems related to temporal learning and decision-making in autonomous systems. The algorithms presented in this thesis make temporal learning more reliable while keeping computations within bounds. On the other hand, we present state-feedback-based decision systems which can make use of the estimates provided by the machine-learning module for intelligent behavior in these complex systems. Most of the results presented in the thesis have been earlier presented in [7–21].

1.4 Organization of Thesis

The thesis consists of seven chapters including the current one. The contents of different chapters are listed next.

1. Chapter−2 presents Markov modeling time-series data using symbolic analysis. It presents the state-of-the-art understanding of Markov modeling using symbolic analysis, the associated hyper-parameters and some algorithms to estimate the hyper-parameters. The proposed ideas are illustrated and verified using two different data-sets, one for crack detection in polycrystalline alloy structures and the other one for input-output modeling in lea-acid batteries. Furthermore, the chapter presents several associated problems of sequential tests using these models and information fusion in local sensor networks using these models.

2. Chapter−3 presents reduced-order modeling of time-series data using symbolic analysis. The proposed technique allows compact representation of the data by aggregating states of the model using K-L distance and agglomerative clustering. An algorithm for state-merging is presented which results in an non-deterministic finite state automata. Several results using two different data-sets are provided to show that the final model is able to preserve the underlying temporal statistics of the data.

3. Chapter−4 presents temporal modeling of video data (high-dimensional time series) using deep learning and Gaussian processes. The proposed technique makes best use of deep learning and Gaussian processes to model high-dimensional temporal statistics in video data. The proposed ideas of modeling are tested on high-speed grey-scale images obtained from a swirl-stabilized combustor during lean pre-mixed combustion where controlled protocols are used to induce instability in the combustion process.

4. Chapter−5 introduces policy-based algorithms for anytime motion planning of robotic systems. The proposed algorithms are used to synthesize feedback trees for motion planning of robotic systems. Convergence and asymptotic optimality of the proposed algorithms are established and illustrated through numerical simulations. The proposed algorithms are extended to motion

planning of multiple robots using non-cooperative game theory. Guarantees on convergence and optimality of the proposed algorithms are also established and illustrated through numerical simulations.

5. Chapter−6 presents the problem of safety in artificial intelligence. Specifically, the problem of motion planning in the presence of unknown model parameters are presented, where the parameters are estimated online using a statistical learning algorithm. An algorithm using viability theory is proposed to ensure chance-constrained safety of the underlying robotic system while its learning.

6. Chapter−7 presents a summary of the work presented in the thesis and most importantly, presents some important future research directions which can be pursued based on the results presented in the thesis.

Chapter 2

Symbolic Analysis-based Markov Modeling of Time-Series Data

Markov models are used for statistical learning of sequential data when temporal behavior of data is of interest and there is a requirement to relax the assumptions of independence and identical distribution (i.i.d.) on measured data. Various types of models have been proposed and analyzed in literature to address this issue. Some of the common approaches are autoregressive (AR) modeling [22] and hidden Markov modeling (HMM) [23]. In contrast, this chapter focuses on symbolic dynamics-based Markov modeling of time-series data, which has not been so well studied in literature. Symbolic time series analysis (STSA) [24, 25] is a nonlinear technique for representing temporal patterns in sequential data, where the underlying theory is built upon the concepts of symbolic dynamics and chaos theory for analysis of irregular time series data for systems that inherently do not seem stochastic. Essentially STSA consists of two major steps: (i) *discretization*, where the continuous attributes of the sequential data are projected onto a symbolic space which is followed by (ii) *identification* of concise probabilistic patterns that help compress the discretized data. Under this umbrella, *finite-memory* Markov models have been shown to be a reasonably approximate representation of systems with fading memory (e.g., dynamical systems that exhibit stable orbits or mixing) [26,27]. Once the continuous data set is discretized, the memory estimate for the discretized sequence is used for information compression as a finite-memory Markov process, which is represented by a state transition matrix. This procedure is used to find the

causal, dynamical structure intrinsic to the symbolic process under investigation; the objective here is to extract the features that may have any predictive power. The resulting transition matrix could be estimated by frequency counting under the assumption of infinite data and a uniform prior for all the elements of the transition matrix. It is noted that this chapter only considers discrete Markov processes with finite memory.

It is desirable from a machine learning perspective to establish the fundamental limits of information that could be learned from the underlying data when it is compressed as a Markov chain using the symbolic analysis framework. This problem is complex from the following perspectives.

1. The underlying model for data is unknown (or partially unknown), which makes data discretization difficult to analyze.

2. The order of the discrete stochastic process depends on the discretization parameters (i.e., number and locations of partitioning segments) and this coupling is difficult to analyze in the absence of any underlying model.

The above reasons make analysis of this problem very difficult from the perspectives of dynamical systems. As the data model is unknown, there is no metric to characterize the properties of the discrete sequence w.r.t. the original data. Furthermore, the behavior of the discrete sequence is governed by the discretization process. As a result, characterizing the final Markov model w.r.t. the original data is difficult due to this composite process. Dynamical systems theory provides some consistent characterization of partitions but they are, in general, not straight-forward to estimate numerically in the absence of a model. On the other hand, there is no information-theoretic or statistical formalism for the discretization process in open literature. However, in the absence of a model, the use of information-theoretic and statistical measures for data discretization seems natural.

While the discretization step decides the information contents of the symbolic sequence, memory estimation is critical for concise, yet precise, representation of the discretized sequence. There are many techniques in open literature for discretization and memory estimation of Markov processes. For machine learning applications, most of the methods tie a technique with an end objective (i.e., cost function) to find a solution that must satisfy the constraints and minimize the cost. However, most of the existing methods apparently consider the problems of data

discretization and memory estimation as separate entities; apparently no unifying theory exists for signal representation as a Markov model. Moreover, there are no results on the interplay between complexity of the discrete dynamical system and the discretization process (e.g., cardinality of the discrete set and the discretization technique). For example, no Bayesian inference technique combines these steps together to estimate the model parameters for representation of the time series signal. It is noted that the term *symbolization* is often interchangeably used for *discretization* throughout the chapter, and the term *order* for *depth*.

Symbolization is carried out via partitioning of the phase space of the dynamical system within which the system dynamics evolves. In general, there are two main lines of thought behind the symbolization process, one inspired by dynamical systems theory and the other inspired by machine learning. Some of the approaches inspired by the dynamical systems theory could be found in [28–31]. Several partitioning methods have been proposed in literature, such as maximum entropy partitioning [32], symbolic aggregate approximation (SAX) [33], maximally-bijective partitioning [34], and density estimation-based partitioning [35]. In these approaches, the discretization is not tied to any performance objective from the data (e.g., in maximum entropy partitioning, each partition contains equal number of data points and thus it presents an unbiased discrete representation of the system). This is somewhat different from machine learning-inspired techniques where an optimization over the parameters of partitions is performed; however, these techniques are always susceptible to over-fitting of data and may lead to algorithms with poor generalization. Most of the machine learning-inspired discretization techniques are mainly based on some end objectives like classification performance, which use the data labels to improve class separability [36, 37]. Detailed reviews of partitioning methods for time-series symbolization can be found in [38, 39]. A information theory-based approach to select alphabet size was presented recently in [40]. Some empirical results regarding the comparison and performance of different partitioning methods has been reported in [41–43]. Even though many partitioning techniques have been reported in literature, there is apparently no standard partitioning method. The rationale is that effectiveness of discretization depends on several factors such as nature of the dynamical system, topological properties of the system's phase space, and similarity metrics. Moreover, in statistical learning problems, the underlying model for generating the data is often

unknown and thus, it is difficult to impose a consistency criterion based on the system behavior. Consequently, finding a universal rule which can be used for a wide variety of systems with consistent performance is evasive. To the best of authors' knowledge, there are no consistency results reported in literature for signal discretization and representation learning.

The next step in the process is modeling of the symbol sequence which allows further compression of the data as a Markov model. Working in the symbolic domain, the task is to identify concise probabilistic models that relate the past, present and the future states of the system under consideration. For Markov modeling of the symbol sequence, this is achieved by first estimating the depth (or memory size) for the discrete symbolic process and then, estimating the stochastic model from the observed sequence. Various approaches have been reported in open literature for order estimation of Markov chains. Most of these approaches are inspired from information theory and make use of concepts like minimum description length (MDL) [44, 45], complexity theory [46] and/or information criteria like Akaike's information criterion (AIC) [47] or Bayesian information criterion (BIC) [45, 48]. Important results regarding consistency of these techniques have been reported for order estimation. In [49, 50], the authors show that the minimum description length Markov estimator will converge almost surely to the correct order if the alphabet size is bounded a priori. In [51–53], the authors present a more direct way of order estimation by making use of convergence rate of k-block distributions. Some other techniques, based on the use of heuristic information theory-based criterion for order estimation could be found in [54–56]. For machine learning applications, the estimation process follows a*wrapper* approach [23], where a wrapper is a search algorithm which invokes the main modeling module to build several temporal models with varying depths and the search is stopped when the stopping criterion such as information gain or entropy rate shows a marginal improvement with the increment in complexity [57]. This approach in essence creates all models first and then chooses a model based on a given metric and/or a threshold. Making multiple passes through the data, *searching* for correct depth is computationally intensive and might be infeasible for large data sets that are common today.

Recently, a new approach for depth estimation has been proposed based on the spectral properties of the one-step transition matrix [58]. An upper bound on the size of temporal memory has been derived that requires a single-pass

through the time-series data. In [9], the authors presented a rigorous comparative evaluation of spectral property-based approach with three popular techniques – (1) log-likelihood, (2) state splitting using entropy rate, and (3) signal reconstruction error based approach. These three techniques fall under the wrapper approach of depth estimation; therefore computationally, they were found to be in close agreement about the estimated depth.

Once the pertinent parameters (e.g., alphabet size and estimated depth) are estimated, the data set is represented by a stochastic matrix for the inferred Markov chain. The stochastic matrix could be estimated by a Bayesian approach, where the prior and posterior of the individual elements of the matrix can be represented by Dirichlet and categorical distributions, respectively. Under the assumption of finite state-space of the Markov chain and a uniform prior, it can be shown that the posterior estimates asymptotically converge to the maximum likelihood estimate (MLE) for the individual elements. Thus, through this sequential process of discretization and order estimation, a generative Markov model is rendered for representation of the data, which can be used for various learning algorithms such as modeling, classification, and pattern matching.

Even though much work has been performed on discretization and approximation of discrete sequences as finite-order Markov chains, there is apparently no rigorous theory that ties them together for signal representation. Furthermore, since these individual pieces have not been studied together, there is no unifying theory on consistency of signal representation as a Markov model, especially when the underlying model generating the data is unknown. As a result, statistical modeling of data for STSA-based Markov modeling is still largely practiced in an ad-hoc fashion following a *wrapper*-inspired technique where the solution space is searched exhaustively for solutions and thus, still remains hugely dependent on the domain expertise. The challenges here are both algorithmic and computational; the algorithmic challenge is to synthesize a consistent framework with guarantees on performance while the computational challenge is to reduce computations required to arrive at a desired solution mainly inspired by machine learning applications.

It is logical to point out that the presented formalism for statistical learning is different from that of standard hidden Markov models (HMM) in the following ways.

- The state-space of these models is inferred by partitioning the observed phase-

space of the dynamical system while in HMM the state-space is not observed and thus it is difficult to infer the underlying structure.

- Under the assumption that the observed sequence is a finite-order Markov chain, the estimation of parameters is simplified. The sufficient statistics could be obtained by estimating the order of the Markov chain and the symbol emission probabilities conditioned on the memory words of the discrete sequence.

Thus, the current framework of STSA is simplistic and thus, can be easily used for embedded applications for the simplicity of the inference algorithms. The details are provided in the subsequent sections.

This chapter presents the state-of-the-art mathematical formalism for symbolic analysis-based Markov modeling and representation of time-series data. We present several results for Markov modeling under the present context for memory estimation in these models, sequential update of their log-likelihood statistic and extensions to modeling temporal dependence between multiple heterogeneous sequences. Several results using experimental data are provided to show the illustrated concepts and the chapter also delineates some challenging problems for future work. The hope to is to be able to provide a complete overview of the technique and provide numerical results to show its effectiveness.

2.1 Mathematical Preliminaries and Background

This section briefly explains the related ideas and concepts for symbolic time-series analysis-based Markov modeling of data. In particular, the concepts of discretization, order (memory) and Markov modeling are introduced. Symbolic analysis of time-series data is a relatively recent method for anomaly detection and pattern recognition [26], where a time series of sensor data are converted to a symbol sequence via partitioning in the continuous domain [33, 57, 59]. Then, the discrete symbolic dynamic process is compressed as a Markov chain whose states are collection of words over a finite alphabet with finite length. This section also provides the related concepts which are required to explain the subsequent material.

Definition 2.1.1 (Pre-Image) *Given $\phi : M \to L$ (assume ϕ is surjective), the image of $x \in M$ is $\phi(x) \in L$. Then, the pre-image of $y \in L$ is given by the set $\phi^{-1}(y) = \{x \mid \phi(x) = y\}$.*

Definition 2.1.2 (Homeomorphism) *A function $\phi : M \to N$ in a metric space to another is continuous if, whenever a sequence $\{x_n\}$ from M converges to x in M, then the sequence $\{\phi(x_n)\}$ from N converges to $\phi(x)$ in N. If ϕ is continuous, one-to-one, onto and has a continuous inverse, then ϕ is called a homeomorphism.*

Definition 2.1.3 (Dynamical System) *A dynamical system (M, ϕ) consists of a compact metric space M together with a continuous map $\phi : M \to M$. If ϕ is a homeomorphism , the dynamical system (M, ϕ) is called invertible.*

Definition 2.1.4 (Irreducible Dynamical System) *A dynamical system (M, ϕ) is said to be irreducible if, for every pair of open sets U and V, there exists $m \geq 0$ such that $\phi^m U \cap V \neq \emptyset$.*

Definition 2.1.5 (Bilaterally Transitive) *A point $p \in M$ is called to be bilaterally transitive if the forward orbit $\{\phi^n p \mid n > 0\}$ and the backward orbit $\{\phi^n p \mid n < 0\}$ are both dense in M. A homeomorphism ϕ is said to be expansive if there exists a real number $c > 0$ such that $d(\phi^n p, \phi^n q) < c$ for all $n \in \mathbb{Z}$, then $p = q$.*

Definition 2.1.6 (Factor Map) *For two dynamical systems (M, ϕ) and (L, ψ), the second is called a factor of the first and the first an expansion of the second, if there exists a map π of M into L, called a factor map, such that*

- *$\psi\pi = \pi\phi$.*
- *π is continuous.*
- *π is onto.*

Furthermore, π is said to be a finite factor map or that it is bounded one-to-one if

- *there is a bound to the number of pre-images.*
- *every doubly transitive point has a unique pre-image.*

Symbolic representation of dynamical systems (M, ϕ) is considered in this chapter. To understand the idea, consider the case where ϕ is invertible so that all the iterates of ϕ, positive and negative, are used. Then, to describe the orbits $\{\phi^n(y) : n \in \mathbb{Z}\}$ of points $y \in M$, an approximate description can be constructed in the following way. Divide M into a finite number of pieces $E_0, E_1, \ldots, E_{r-1}$ which cover the space M and are mutually disjoint. Then, one can track the orbit of a point $y \in M$ by keeping a record of which of these pieces $\phi^n(y)$ lands in. This yields a corresponding point $x = \ldots x_{-1}x_0x_1 \cdots \in \{0, 1, \ldots r-1\}^{\mathbb{Z}} = X_{[r]}$ (full r-shift space) defined by

$$\phi^n(y) \in E_{x_n} \text{ for } n \in \mathbb{Z}$$

Thus, for every $y \in M$, one obtains a point x in the full r-shift, and the definition shows that the image $\phi(y)$ corresponds to the shifted point $\sigma(x)$.
More formally, the partitions can be defined as follows.

Definition 2.1.7 (Partitions) *A family of sets $\mathcal{R} = \{R_1, R_2, \ldots, R_{N-1}\}$ is called a topological partition for a compact metric space M if the following holds true.*

1. *each R_i is open;*
2. *$R_i \cap R_j = \emptyset$, $i \neq j$;*
3. *$M = \overline{R}_1 \cup \overline{R}_2 \cup \ldots \overline{R}_{N-1}$*

As there are only a finite number of sets in the partition, every set is denoted by a symbol from a finite alphabet and thus the partitioning process for a dynamical system could also be viewed as identification of a many-to-one projection mapping such that the continuous dynamical system is mapped onto a discrete space. More formally, let the continuously varying physical process be modeled as a finite-dimensional dynamical system, $\dot{\mathbf{x}} = f(\mathbf{x}(t))$ where $t \in [0, \infty)$ and $\mathbf{x} \in \mathbb{R}^n$ is the state-vector in the compact phase space Ω of the system. Then, the partitioning process could be defined as a map $\varphi : \Omega \to \mathcal{A}$ such that $\varphi(\mathbf{x}) = a$ if $\mathbf{x} \in R_a$, where $R_a \subseteq \Omega$, $\bigcup_{a \in \mathcal{A}} R_a = \Omega$.

Let $\mathcal{A}$ denote the ordered set of n symbols. The phase space of the symbolic system is the space.

$$X_{[n]} = \mathcal{A}^{\mathbb{Z}} = \{a = (a_k)_{k \in \mathbb{Z}} \mid a_k \in \mathcal{A}\} \tag{2.1}$$

of all bi-infinite sequences of elements from a set of n symbols. The shift transformation σ is defined by shifting each bi-infinite sequence one step to the left. This is expressed as

$$(\sigma a)_k = a_{k+1}$$

The symbolic system $(X_{[n]}, \sigma)$ is called the *full n-shift*. Restricting the shift transformation of a full-shift $X_{[n]}$ to a closed shift-invariant subspace X, one obtains a very general kind of dynamical system (X, σ) called a *subshift*. Given a symbolic phase space X, a k-tuple is called an *allowable k-block* if it equals $a_{[m,m+k-1]}$ for some sequence a. Then, a *shift of finite type*, also called the *topological Markov shift*, is defined as the subshift (i.e., shift-invariant) of a full shift restricted to the set X_G of bi-finite paths in a finite directed graph G derived from a full one by possibly removing some edges. The space could also be denoted by X_V where V is a matrix and v_{ij} denotes the number of edges going out from node i to node j. This gives the resemblance to a Markov chain as normalizing each row of V gives a stochastic matrix which defines a Markov chain over G. Before moving to more details of theory of discrete ergodic processes, which provides a statistical view to the discrete stochastic processes, it is important to clarify that a symbolic sequence in a topological Markov shift is bilaterally transitive if every admissible block appears in both directions and infinitely often.

A (discrete-time, stochastic) process is a sequence $S_1, S_2, \ldots, S_n, \ldots$ of random variables defined on a probability space $(S, \mathcal{E}, P)$. The process has *alphabet* $\mathcal{A}$ if the range of each S_i is contained in $\mathcal{A}$. This chapter focuses on finite-alphabet processes, so, unless otherwise specified, "process" means a discrete-time finite-alphabet process. Also, unless it is specified otherwise, "measure" will mean "probability measure" and "function" will mean "measurable function" w.r.t. some appropriate σ-algebra on a probability space. The cardinality of a finite set A is denoted by $|A|$. The sequence $a_m, a_{m+1}, \ldots, a_n$, where each $a_i \in \mathcal{A}$ is denoted by a_m^n; the corresponding set of all such a_m^n is denoted by $\mathcal{A}_m^n$ except for $m = 1$, when $\mathcal{A}^n$ is used. The k^{th} order joint distribution of the process $\{S_k\}$ is the measure P_k on $\mathcal{A}^k$ defined by the formula

$$P_k(a_1^k) = \Pr(S_1^k = a_1^k), a_1^k \in \mathcal{A}^k \tag{2.2}$$

Definition 2.1.8 (Stationary Process) *A process is stationary, if the joint dis-*

tribution do not depend on the choice of the time origin, i.e.,

$$Pr(S_i = a_i, m \leq i \leq n) = Pr(S_{i+1} = a_i, m \leq i \leq n), \tag{2.3}$$

for all $m, \quad n$ *and* a_m^n.

These stationary finite-alphabet processes serve as models in many settings of interest (e.g., physics, data transmission, and statistics). The main interest here is in statistical and machine learning applications like density estimation, pattern matching, data clustering, and estimation.

The simplest example of a stationary finite-alphabet process is an i.i.d. process. A sequence of random variables, $\{S_n\}$ is independent if

$$\Pr(S_n = a_n | S_1^{n-1} = a_1^{n-1}) = \Pr(S_n = a_n) \tag{2.4}$$

holds for all $n \geq 1$ and all $a_1^n \in \mathcal{A}^n$. It is identically distributed if $\Pr(S_n = a) = \Pr(S_1 = a)$ holds for all $n \geq 1$ and for all $a \in \mathcal{A}$. An independent process is stationary if and only if it is identically distributed. The simplest example of dependent finite-alphabet processes are the Markov processes.

Definition 2.1.9 (Markov Chain) *A sequence of random variables,* $\{S_n\}$*, is a Markov chain if*

$$Pr(S_n = a_n | S_1^{n-1} = a_1^{n-1}) = Pr(S_n = a_n | S_{n-1} = a_{n-1}) \tag{2.5}$$

holds for all $n \geq 2$ *and all* $a_1^n \in \mathcal{A}^n$*. A Markov chain is known as homogeneous or to have stationary transitions if* $Pr(S_n = b | S_{n-1} = a)$ *does not depend on* n*, in which case the* $|A| \times |A|$ *matrix* M *defined by*

$$M_{b|a} = Pr(S_n = b | S_{n-1} = a) \tag{2.6}$$

is called the transition matrix for the chain.

A generalization of the Markov property allows dependence on k steps in the past. A process is called $k-$step (or $k-$order) Markov if

$$\Pr(S_n = a_n | S_1^{n-1} = a_1^{n-1}) = \Pr(S_n = a_n | S_{n-k}^{n-1} = a_{n-k}^{n-1}) \tag{2.7}$$

holds for all $n > k$ and for all a_1^n.

The symbolic time-series analysis is initialized by partitioning the phase space of a dynamical system which is a non-linear mapping from a continuous space to a discrete space. The data from a dynamical system is partitioned based on the choice of a partitioning technique and then, the dynamics of the discrete process is studied. The dynamics of the symbol sequences are compressed as a probabilistic finite state automata (PFSA), which is defined as follows:

Definition 2.1.10 (PFSA) *A probabilistic finite state automata (PFSA) is a tuple* $G = \{\mathcal{Q}, \mathcal{A}, \delta, \boldsymbol{M}\}$ *where*

- $\mathcal{Q}$ *is a finite set of states of the automata;*
- $\mathcal{A}$ *is a finite alphabet set of symbols* $a \in \mathcal{A}$*;*
- $\delta : \mathcal{Q} \times \mathcal{A} \to \mathcal{Q}$ *is the state transition function;*
- $\boldsymbol{M} : \mathcal{Q} \times \mathcal{A} \to [0, 1]$ *is the* $|\mathcal{Q}| \times |\mathcal{A}|$ *emission matrix. The matrix* $\boldsymbol{M} = [m_{ij}]$ *is row stochastic such that* m_{ij} *is the probability of generating symbol* a_j *from state* q_i*.*

Remark 2.1.1 *The* $|\mathcal{Q}| \times |\mathcal{Q}|$ *state-transition matrix* $\boldsymbol{\Pi} : \mathcal{Q} \times \mathcal{Q} \to [0, 1]$ *can be constructed from the state transition function* δ *and the emission matrix* $\boldsymbol{M}$*. The state transition matrix* $\boldsymbol{\Pi}$ *can be defined as follows.*

$$\boldsymbol{\Pi}_{jk} \triangleq \sum_{a \in \mathcal{A}: \delta(q_j, a) = q_k} \boldsymbol{M}(q_j, a), \forall q_j, q_k \in \mathcal{Q}$$

The matrix $\boldsymbol{\Pi} = [\pi_{ij}]$ *is row stochastic and* π_{ij} *is the probability* $Pr(q_j|q_i)$ *of visiting state* q_j *given state* q_i *in the past.*

A PFSA can be seen as a generative machine - probabilistically generating a symbol string through state transitions. Ranging from pattern recognition, machine learning to computational linguistics, PFSA has found many uses. A comprehensive review of PFSA can be found in [60, 61]. From the perspective of machine learning, the regular patterns are inferred in the symbol sequence, which help explain the temporal dependence in the discrete sequence; these patterns words of finite length over the alphabet of the discrete process. Once these patterns are discovered, then

the discrete data is compressed as a PFSA whose states are the words of finite length; this step leads to some information loss. The parameters of the PFSA are the symbol emission probabilities from each of the finite length words (i.e., states of PFSA). Under the assumption of an ergodic Markov chain, the requirement is relaxed for initial states and thus the only *sufficient statistics* [62, 63] for the inferred Markov model are the corresponding symbol emission probabilities. The following information is presented for clarification.

Definition 2.1.11 (Sufficient Statistic) *Let us assume* $S_1, S_2, \ldots, S_n \sim p(s; \alpha)$, *then, any function* $\mathcal{T} = \mathcal{T}(S_1, S_2, \ldots, S_n)$ *is itself a random variable which is called a statistic. The statistic* $\mathcal{T}$ *is sufficient for* α *if the conditional distribution of* $S_1, S_2, \ldots, S_n \mid \mathcal{T}$ *does not depend on* α. *Thus, it follows that* $p(s_1, s_2, \ldots, s_n \mid t; \alpha) = p(s_1, s_2, \ldots, s_n \mid t)$.

For the case of finite-order, finite-state Markov chains it means that the conditional symbol emission probabilities and the initial state summarizes all the relevant information supplied by any sample [64]. Under stationarity assumptions, the initial state becomes unnecessary and thus, the sufficient statistics is provided by the conditional symbol emission probabilities. The details on estimation of the parameters are provided later in the chapter.

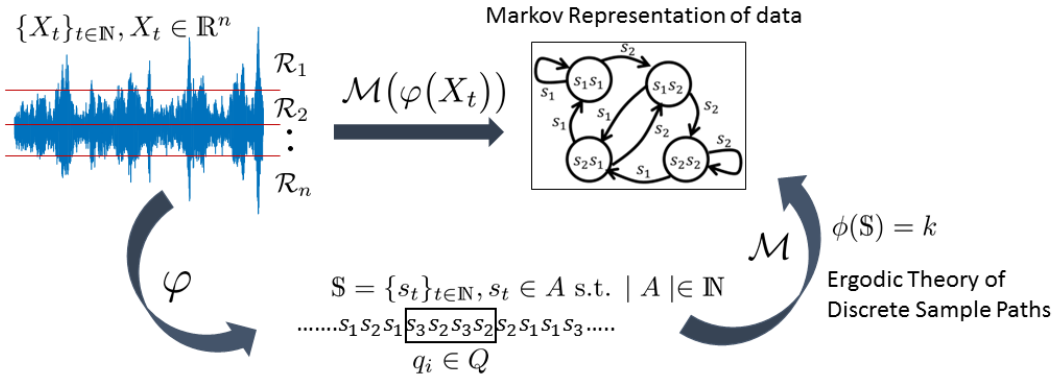


Figure 2.1: Markov modeling of time series data and the associated decisions of partitioning and discrete modeling

For symbolic analysis of time-series data, a class of PFSAs called the D-Markov machine have been proposed [26] as a sub-optimal but computationally efficient approach to encode the dynamics of symbol sequences as a finite state machine. The

main assumption (and reason for sub-optimality) is that the symbolic process can be approximated as a D^{th} order Markov process. For most stable and controlled engineering systems that tend to forget their initial conditions, a finite length memory assumption is reasonable. The states of this PFSA are words over $\mathcal{A}$ of length D (or less); and state transitions are described by a sliding block code of memory D and anticipation length of one [65]. The dynamics of this PFSA can both be described by the $|\mathcal{Q}| \times |\mathcal{Q}|$ state transition matrix $\mathbf{\Pi}$ or the $|\mathcal{Q}| \times 1$ state visit probability vector $\boldsymbol{p}$. Next, definitions of D-Markov machines are presented, which has been recently introduced in literature as finite-memory approximate models for inference and learning.

Definition 2.1.12 *(D-**Markov Machine** [26, 27]) A D-Markov machine is a statistically stationary stochastic process $S = \cdots a_{-1}a_0a_1 \cdots$ (modeled by a PFSA in which each state is represented by a finite history of D symbols), where the probability of occurrence of a new symbol depends only on the last D symbols, i.e.,*

$$P[a_n \mid \cdots a_{n-D} \cdots a_{n-1}] = P[a_n \mid a_{n-D} \cdots a_{n-1}] \tag{2.8}$$

where D is called the depth of the Markov machine.

A D-Markov machine is thus a D^{th}-order Markov approximation of the discrete symbolic process. The PFSA model for the time-series data could be inferred as a probabilistic graph whose nodes are words (or symbol blocks) over $\mathcal{A}$ of length equal to D. This probabilistic model induces a Markov model over the states of the PFSA and the parameters of the Markov model can then be estimated from the data using a maximum-likelihood approach. This completes the process of model inference using STSA.

There is a natural extension of the current modeling framework to modeling dependence between two different symbol sequences. Under this framework, the Markov assumptions are now applied from one symbolic signal to another. The inference task is to figure out the state-space of the first process which results in best predictability of the other process. This modeling technique is now introduced in a more formal way and the underlying concept is illustrated in Figure 2.2.

Definition 2.1.13 *(xD-Markov Machine) [8] Let $\mathbb{S}_1 = \{\ldots s_1s_2s_3\ldots\}$ and $\mathbb{S}_2 = \{\ldots \sigma_1\sigma_2\sigma_2\ldots\}$ be two discrete symbol sequences. Then, a xD-Markov machine*

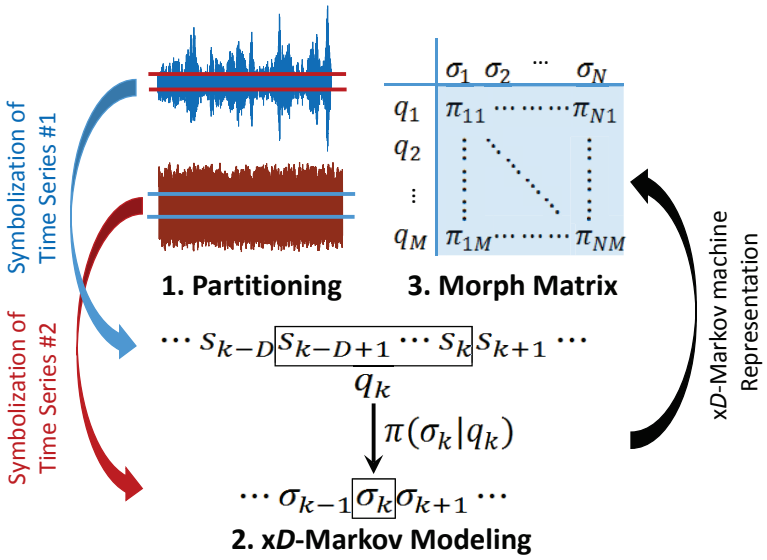


Figure 2.2: Schematic diagram of $\mathbf{x}D$-Markov machines, where the (temporal) dynamics of a symbolic stochastic sequence are captured relative to another such sequence

(where the Markov assumption holds for $\mathbb{S}_2$ w.r.t. the observations of $\mathbb{S}_1$) is defined as a 5-tuple $G_{1\to 2} \triangleq (\mathcal{Q}_1, \mathcal{A}_1, \mathcal{A}_2, \delta_1, \boldsymbol{M}_{12})$ such that:

1. $\mathcal{Q}_1 = \{q_1, q_2, \ldots, q_{|\mathcal{Q}_1|}\}$ *is the state set corresponding to symbol sequence* $\mathbb{S}_1$
2. $\mathcal{A}_1 = \{s_1, ..., s_{|\mathcal{A}_1|}\}$ *is the alphabet set of symbol sequence* $\mathbb{S}_1$
3. $\mathcal{A}_2 = \{\sigma_1, ..., \sigma_{|\mathcal{A}_2|}\}$ *is the alphabet set of symbol sequence* $\mathbb{S}_2$
4. $\delta_1 : \mathcal{Q}_1 \times \mathcal{A}_1 \to \mathcal{Q}_1$ *is the state transition mapping. It is noted that the PFSA structure is built on* $\mathbb{S}_1$ *and thus, the transition map explains the same symbol sequence; however the Markov assumption holds for* $\mathbb{S}_2$ *on the states inferred in* $\mathbb{S}_1$.
5. $\boldsymbol{M}_{12} : Q_1 \times \mathcal{A}_2 \to [0, 1]$ *is the cross emission matrix of size* $|\mathcal{Q}_1| \times |\mathcal{A}_2|$*; the* ij^{th} *element* $(\boldsymbol{M}_{12}(q_i, \sigma_j))$ *of* $\boldsymbol{M}_{12}$ *denotes the probability of finding the symbol* σ_j *in the symbol string* $\mathbb{S}_2$ *at next time step while making a transition from the state* q_i *of the PFSA constructed from the symbol sequence* $\mathbb{S}_2$.

Similarly, a 5-tuple $G_{2\to 1} \triangleq (\mathcal{Q}_2, \mathcal{A}_2, \mathcal{A}_1, \delta_2, \boldsymbol{M}_{21})$ is defined for $\mathbb{S}_1$ w.r.t. the observations of $\mathbb{S}_2$.

Next, an information-theoretic distance is introduced, which is used in the subsequent sections to quantify changes in the Markov models during a physical process.

Definition 2.1.14 (Kullback-Leibler Divergence) *The Kullback-Leibler (K-L) divergence of a discrete probability distribution P from another distribution $\tilde{P}$ is defined as follows.*

$$D_{KL}(P\|\tilde{P}) = \sum_{x\in X} p_X(x)\log\left(\frac{p_X(x)}{\tilde{p}_X(x)}\right)$$

where the distribution $\tilde{P}$ is assumed to be absolutely continuous w.r.t. P (denoted as $\tilde{P} << P$). It is noted that K-L divergence is not a proper distance as it is not symmetric. However, to treat it as a distance it is generally converted into symmetric divergence as follows, $d(P,\tilde{P}) = D_{KL}(P\|\tilde{P}) + D_{KL}(\tilde{P}\|P)$. This is defined as the K-L distance between the distributions P and $\tilde{P}$, where $\tilde{P} << P$ and $P << \tilde{P}$.

2.2 Problem Formulation

As explained in Section 2.1, there are two critical steps for Markov modeling of data using symbolic dynamics. Once the model structure is fixed, the parameters of the model are estimated from the data and thus, the following steps are followed during model inference.

- Symbolization or partitioning
- Order estimation of the discrete stochastic process
- Parameter estimation of the Markov model inferred with the composite method of discretization and order estimation.

The overall idea of modeling the signal as a Markov chain is represented in Figure 2.1, where the various factors to be considered during Markov modeling are emphasized. For example, issues are: how the original time-series data is related to the discrete symbol sequence and then, how to define a metric that will relate the representation in the original space as well as in the discrete space. Another important question is to analyze the relation of the original data with the discrete Markov model inferred using the parameters estimated during the modeling process.

In general, for most of the machine learning applications, no description of the underlying system model is available. This in turn makes evaluation of different

models for signal representation very difficult as the topological structure of the underlying dynamical system is unknown. In the presence of some end objective (e.g., class separability, anomaly detection, etc.), a Bayesian inference rule could be used for selection of the optimal model [37]. However, the problem is ill-posed in the absence of a well-defined end objective when the aim is to just infer the precise model for signal representation which can be later used for various different operations such as clustering or classification. The problem of partitioning the data is particularly ill-posed in the absence of a model, as it is difficult to specify a metric to define a consistent criterion as the process terminates with a Markov model of the data when the true statistical nature of data is unknown. In general, the order of the discrete data depends on the parameters of the partitioning process i.e., the cardinality of the discrete set and the location of the partitions. Thus, although, the precision of the Markov model depends on the partitioning process but in the absence of any model for the data, it is hard to evaluate performance or find a criterion for consistency.

This chapter presents techniques for partitioning of continuous time-series data, and order estimation of discrete stationary Markov processes. In particular, the following issues are discussed.

- **Problem 1.** Identification of partition locations on the phase space of the system for discrete representation.
- **Problem 2.** Inference of the state-space of the system by estimating the size of temporal memory of the discrete sequence
- **Problem 3.** Estimation of parameters for the inferred Markov models, i.e., the estimation of symbol emission parameters.

The following sections attempt to answer the above questions and then, demonstrate the proposed methods of data-sets from some engineering systems. The following issue will also be addressed: How the two steps of discretization and order estimation can be tied together for inferring a Markov model from the time series, where the interplay between discretization and order estimation is explored. Thus, an overall picture of Markov modeling for time-series data is provided and the different algorithms for signal representation are presented by using this technique.

2.3 Partitioning of Phase Space

This section presents some concepts and results based on some special kind of well-behaved partitioning. Most the results in this section are based on earlier work presented in [30, 65], focusing on a special class of partitions called *Markov partitions*. Loosely speaking, by using a Markov partition, a dynamical system can be made to resemble a discrete-time Markov process, with the long-term dynamical characteristics of the system represented by a Markov shift. Before providing a formal definition for Markov partitions, it is necessary to visit some properties of a partitioning from dynamical systems literature. For brevity proofs of some simple properties are skipped. Interested readers are referred to [30] for more details. It is noted that in the subsequent material interior of an arbitrary set $\mathcal{X}$ is denoted by $\mathcal{X}^o$.

Definition 2.3.1 *Given two topological partitions* $\mathcal{R} = \{R_0, R_1, \ldots, R_{n-1}\}$ *and* $\mathcal{P} = \{P_0, P_1, \ldots, P_{n-1}\}$*, their topological refinement* $\mathcal{R} \vee \mathcal{P}$ *is defined as*

$$\mathcal{R} \vee \mathcal{P} = \{R_i \cap P_j : R_i \in \mathcal{R}, P_j \in \mathcal{P}\}$$

Based on the definition 2.3.1, the following are true.

- The common topological refinement of two topological partitions is a topological partition.
- For a dynamical system (M, ϕ) with topological partition $\mathcal{R}$ of M, the set defined by $\phi^n\mathcal{R} = \{\phi^n R_0, \phi^n R_1, \ldots,$
  $\phi^n R_{n-1}\}$ is also a topological partition.

It follows from the above two results that for $m \leq n$, $\bigvee_m^n \phi^k\mathcal{R} = \phi^m\mathcal{R} \vee \phi^{m-1}\mathcal{R} \vee \cdots \vee \phi^n\mathcal{R}$ is again a topological partition. The following notations are used in the sequel.

$$\mathcal{R}^n \equiv \bigvee_{k=0}^{n-1} \phi^{-k}\mathcal{R}$$

The diameter of a partition is defined as $d(\mathcal{R}) = \max_{R_i \in \mathcal{R}} d(R_i)$, where $d(R_i) \equiv \sup_{x,y \in R_i} d(x, y)$. Next, a cylinder set partition is defined for symbol sequences, which will be useful to characterize sufficient conditions for existence of Markov partitions

for a dynamical system. Consider a topological Markov shift (X_V, σ) where the vertex set is labeled by the alphabet $\mathcal{A}$. The next step is forming the partition $\mathcal{C} = \{C_a : a \in \mathcal{A}\}$ of the elementary cylinder sets that are determined by fixing the $0th-$coordinate (i.e., $\mathcal{C}_a = \{x \in X_V : x_0 = a\}$). This partition has the following properties.

1. $\bigcap_{m=0}^{\infty} \bigcap_{-m}^{m} \phi^{-k} C_{a_k} = \{x\}$.
2. If $x \in C_a \cap \sigma^{-1} C_b$, then $x \in C_a$ and $\sigma x \in C_b$, i.e., $x_0 = a, x_1 = b$ (or it means an edge from a to b in terms of a graph.)
3. If $C_a \cap \sigma^{-1} C_b \neq \emptyset$ and $C_b \cap \sigma^{-1} C_c \neq \emptyset$, then $C_a \cap \sigma^{-1} C_b \cap \sigma^{-2} C_c \neq \emptyset$. This can be extended to length n for arbitrary n. Such a countable set of conditions for $n = 2, 3, \ldots$ is called the Markov property: this turns out to be one of the key properties in the desired symbolic representation from a partition.

Definition 2.3.2 (Generator) *A topological partition is called a generator for a dynamical system (M, ϕ) if $\lim_{n\to\infty} d(\bigvee_{-n}^{n} \phi^k \mathcal{R}) = 0$.*

For an expansive dynamical system (M, ϕ) let $\mathcal{R}$ be a topological partition such that $d(\mathcal{R}) < c$ where c is the expansive constant. Then, $\mathcal{R}$ is a generator.

For a topological partition $\mathcal{R}$, let $X_{\mathcal{R},\phi}$ be called the symbolic dynamical representation of (M, ϕ). Then, for each $x \in X_{\mathcal{R},\phi}$, one defines $E_n(x) \triangleq \bigcap_{k=-n}^{n} \phi^{-k} R_{x_k}$. Markov partitions are defined more formally as follows.

Definition 2.3.3 (Markov Partitions) *Let (M, ϕ) define be an invertible dynamical system. A topological partition $\mathcal{R} = \{R_1, R_2, \ldots, R_{n-1}\}$ of M gives a symbolic representation of (M, ϕ) if for every $x \in X_{\mathcal{R},\phi}$ the intersection of $\cap_{n=0}^{\infty} \bar{E}_n(x)$ contains exactly one point. Then, $\mathcal{R}$ is defined to be a Markov partition for (M, ϕ) if $\mathcal{R}$ gives a symbolic representation of (M, ϕ). Furthermore, $X_{\mathcal{R},\phi}$ is a shift of finite type.*

There are natural extensions of Markov partitions for the cases where the dynamical system is not invertible, where one-sided symbolic representation of (M, ϕ) is denoted as $X^{+}_{\mathcal{R},\phi}$.

Even though Markov partitions provide a consistent way of studying symbolic dynamic systems, it is, in general, very hard to find one such partition even if it exists. Hence, for most of the learning applications, some heuristic-based approach is used to arrive at a discretization. Some of the common techniques are listed below. The partitioning techniques are divided in two broad categories of: supervised and unsupervised partitioning, and several techniques are provided under each category. Usage of any particular type of partition depends on the end objectives and thus it is often a user's choice.

2.3.1 Unsupervised Partition Techniques

An unsupervised partition technique creates a partition over the phase space without the use of any output performance objective, i.e., the only end objective is a discrete representation of the data. Some of the unsupervised partitioning techniques are listed below:

- **Uniform partition:** This is probably the simplest type of partitions. In this technique, the range space of time-series data is partitioned uniformly so that all partitions are of equal width.

- **Maximum entropy partition:** Maximum entropy partition is another simple technique for partitioning of data and it presents an unbiased way of partitioning data. Detailed descriptions of this technique could be found in [32], where the key idea is to uniformly distribute the occurrence probability of each symbol in the generated symbol sequence such that the information-rich regions of the time series are partitioned finer and those with sparse information are partitioned coarser. In general, Shannon entropy of the symbols is expressed as:

$$H(\mathcal{A}) = -\sum_{k=1}^{|\mathcal{A}|} P_k \log P_k \tag{2.9}$$

  where P_k is the probability of occurrence of the symbol s_k in the set $\mathcal{A}$. This technique guarantees an unique solution for single dimensional time-series, thus it is the most widely used partitioning technique.

- **Density estimation-based partition:** In this approach, a non-parametric density estimation problem is formulated as a convex program [66] or as a linear program [35] to obtain the probability distribution of the multi-dimensional observations. The solution of the optimization (using interior-point methods) gives a kernel-based mixture model, where the number of components and their locations are automatically determined. Each component of the mixture has a weight (or prior probability) and it is considered to represent the probability of emission of an observation given a symbol, where the symbol is a latent variable in the mixture model. Using maximum likelihood (ML) or maximum a posteriori probability (MAP), the measurement space can be partitioned into disjoint regions for each symbol. This alphabet-size selection and partitioning technique has kernel parameters, kernel shape, and one optimization parameter, which can be chosen to obtain a suitable discretization of any multi-dimensional measurement space. More details of this approach is available in [35].

2.3.2 Supervised Partition Techniques

A supervised partition technique creates a partition over the phase space using an objective function for learning, such as, classification performance or regression error. In these techniques the idea is to search for the best partition which maximizes the gain in the objective function over a given set of training data. Some examples of supervised partitioning techniques are covered in [15, 40, 67]. Most of the approaches in these papers use a sequential optimization technique which is a greedy and locally-optimal solution. This problem requires solution to a combinatorial optimization problem and thus is very computationally intensive. Furthermore, geometric constraints on the topology of the discretized signal makes this technique hard.

2.4 Order Estimation in Markov Chains

This section addresses the problem of order (or memory) estimation for discrete stochastic processes under assumptions of finite order. Order estimation of discrete ergodic sequences is a well studied problem in literature [49]. Various approaches

have been proposed in literature for order estimation of discrete stochastic processes and some approaches have been proven to be consistent in the sense that they will estimate the correct order with probability one [49]. Several numerically efficient algorithms are presented in this section, which provide an alternative to the order estimation process. Popular techniques of depth estimation in current literature that are evaluated and compared in this chapter are described in the next subsections.

2.4.1 Log-likelihood rate

Log-likelihood rate of a symbol sequence $\vec{s}$ given a D-Markov machine G is computed as follows:

$$\mathcal{L}(\vec{s}|G) \cong \frac{1}{N-D} \sum_{k=D+1}^{N} \log \Pr\left(s_k | s_{k-1}, \ldots, s_{k-D}\right) \tag{2.10}$$

where the effects of the initial state are ignored because they become negligible for long statistically stationary symbol sequences. Using a training set of symbol sequences a PFSA model G is learned for a depth D. The log-likelihood of a set of test symbol sequences are used to measure how well the model G is able to capture the dynamics of symbol sequences generated from a physical process.

Since the log-posterior probability $\log \Pr(G|\vec{s})$ is proportional to the log-likelihood $\mathcal{L}(\vec{s}|G)$, the depth D which maximizes the log-likelihood of the symbol sequence $\vec{s}$ is treated as the maximum likelihood estimate (MLE) of depth for the PFSA model. However, in the the absence of a competing objective to minimize the complexity of the model G in the optimization problem, MLE estimate of depth D is chosen to be the smallest value such that increasing D does not significantly improve the log-likelihood of test symbol sequences.

2.4.2 Signal reconstruction

Symbol sequences are typically generated by partitioning the continuous domain phase-space which maps a region of the phase space to one symbol. Under this coarse graining of the phase space, the mean observation of each partition or cell can be used to invert the map from symbols to the continuous domain. Using this approach, symbols predicted for the next time-step using a learned PFSA G can now be used to generate a predicted value in the continuous domain. There can be

several ways to estimate the mean value of each partition – mean of the observed data that falls in that partition or some centrality measure of the partition, such as centroid. The former approach takes the statistics of the observed data in the partition into account, also it can be used to map semi-infinite partitions, e.g., $[c, \infty)$, where constant c is a scalar, to a central value for that partition. The latter approach might be more generalizable to unseen data.

Similar to before, train and test data set from the same physical process are used to evaluate different models G with increasing values of depth D using continuous domain one-step prediction error (L_2 norm) as the metric. It must be noted that coarse graining of the phase-space will lead to some non-recoverable quantization errors. However by evaluating relative errors between different models, one can get an estimate of the depth that leads to best prediction in the continuous domain.

While the log-likelihood approach tries to estimate the best model using the prediction capability in the symbolic domain, the signal reconstruction approach does the same working the continuous domain. The signal reconstruction approach has the advantage that it can be used to compare models that are may not have been constructed using the same modeling process; for instance, models may not even share the same symbolization. The likelihood approach cleanly decouples the effect of depth on model predictability from other parameters.

2.4.3 State splitting using entropy rate

D-Markov models can be divided into two categories based on modeling approach (1) where all states $q_j \in \mathcal{Q}$ of the PFSA have a fixed length D and (2) where states of the PFSA have a maximum length D. The approach 1 typically utilizes a wrapper approach to construct D-Markov models using increasing values is depth D; using metrics such as entropy rate and a stopping rule on the depth and a model structure is selected. The approach 2, however, starts off with the simplest set of states (i.e., $\mathcal{Q} = \mathcal{A}$ for $D = 1$) and subsequently splitting the state that results in the largest decrease of the entropy rate.

The process of splitting a state $q \in \mathcal{Q}$ is executed by replacing the symbol block q by its *branches* as described by the set $\{aq : a \in \mathcal{A}\}$ of words as shown in Figure 2.3. Maximum reduction of the entropy rate is the governing criterion for selecting the state to split. The entropy rate is computed in terms of the elements

of estimated state probability vector and estimated emission matrix ($\boldsymbol{M}$) as:

$$H(\mathcal{A}|\mathcal{Q}) = -\sum_{q\in\mathcal{Q}}\sum_{a\in\mathcal{A}} \Pr(q)\Pr(a|q)\log\Pr(a|q) \tag{2.11}$$

In addition, the generated set of states must satisfy the self-consistency criterion, which only permits a unique transition (see [27]). In the state-splitting algorithm, a stopping rule is required to stop the state-splitting process and it can be a (1) threshold η_{spl} on the rate of decrease of conditional entropy, (2) $N_{\max}$ the maximum number of allowed states, or (3) the maximum allowed depth.

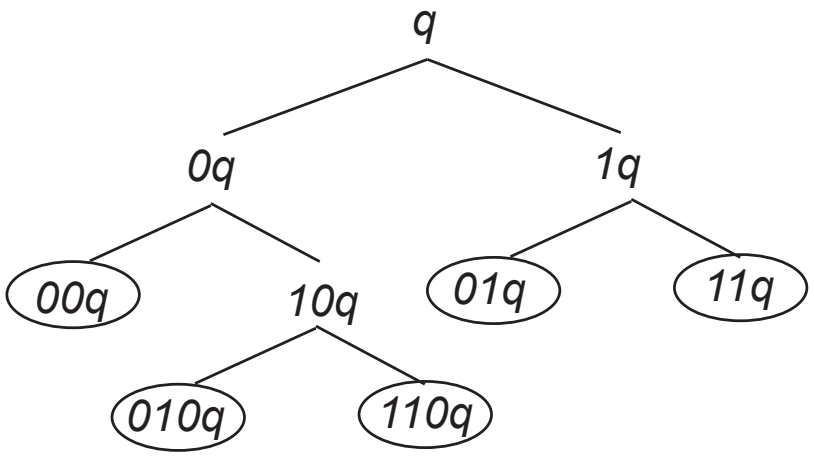


Figure 2.3: Tree-representation of state splitting in D-Markov machines

While the state-splitting approach is effective for obtaining symbol sequences indicating relevant histories that map to states, this greedy approach still requires several passes through the symbol sequence as one goes down the state-splitting tree.

2.4.4 Depth estimation using spectral analysis

The authors in [58] propose a spectral analysis-based method to estimate the upper bound of the length of the symbol sequence constituting a state. Depth D of a symbol sequence has been redefined in [58] as the number of time steps after which probability of current symbol is independent of any past symbol i.e.:

$$\Pr(s_k|s_{k-n}) = \Pr(s_k)\ \forall n > D \tag{2.12}$$

Note that dependence in the proposed definition (eq. 3.1) is evaluated on individual past symbols using $\Pr(s_k|s_{k-n})$ as opposed to the assessing dependence on words of length D using $\Pr(s_k|s_{k-1},\ldots,s_{k-D})$. It is shown that if the observed process is *forward causal* then observing any additional intermediate symbols $s_{k-1},\ldots,s_{k-n+1}$ cannot induce a dependence between s_k and s_{k-n} if it did not exist on individual level.

Let $\mathbf{\Pi} = [\pi_{ij}^{(1)}]$ be the one-step transition probability matrix of the PFSA G constructed from this symbol sequence i.e.

$$\mathbf{\Pi} = \Pr(s_k|s_{k-1}) \tag{2.13}$$

Then using the distance of the transition matrix after steps from the stationary point, depth can be defined as a length D such that

$$|\text{trace}\,(\mathbf{\Pi}^n) - \text{trace}\,(\mathbf{\Pi}^\infty)| \leq \sum_{j=2}^{J} |\lambda_j|^n < \epsilon \;\forall n > D \tag{2.14}$$

J is number of non-zero eigenvalues of $\mathbf{\Pi}$. This approach requires only one pass through the data-set for estimating $\mathbf{\Pi}$

It has been shown earlier in [9] that the above methods for memory estimation agree closely in their estimates. However, the choice of a particular method would depend on the problem at hand.

2.5 Parameter Estimation for Markov Chains

This section presents some results on estimation of parameters of the Markov chain once the structure of the chain is inferred after discretization and order estimation. Given a finite-length symbol string over a (finite) alphabet $\mathcal{A}$, there exists several PFSA construction algorithms to discover the underlying irreducible PFSA model, such as the D-Markov machines [26] [32]. Once the order of the discrete data is estimated, the states of a D-Markov machines are memory words of length equal to the order over $\mathcal{A}$. The sufficient statistics for the D-Markov machine are the symbol emission probabilities conditioned on the memory words. These statistics could be estimated using a Bayesian approach. In general, the symbol emission probabilities conditioned on the individual memory words can be modeled as a categorical

random variable and it is known that Dirichlet distribution is a conjugate prior to the categorical distribution [23, 68]. However, those details are not being skipped here and a simple estimation process is presented, where every random variable has a uniform prior.

Let $N(s_j|q_k)$ denote the number of times that a symbol s_j is generated from the state q_k as the symbol string evolves. The maximum a posteriori probability (MAP) estimate of the emission probability (see Definition 3.2.1) for the PFSA G is estimated by frequency counting as

$$\hat{\pi}(s_j|q_k) \triangleq \frac{C(s_j|q_k)}{\sum_\ell C(s_\ell|q_k)} = \frac{1 + N(s_j|q_k)}{|\mathcal{A}| + \sum_\ell N(s_\ell|q_k)} \tag{2.15}$$

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 $\hat{\pi}_{MAP}(q, s) = \frac{1}{|\mathcal{A}|}\ \forall s \in \mathcal{A}$ (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, must have an (elementwise) strictly positive emission map π.

Having computed the probabilities $\hat{\pi}\left(s_j|q_k\right)$ for all $s_j \in \mathcal{A}$ and $q_k \in Q$, the estimated emission probability matrix of the PFSA is obtained as

$$\widehat{\mathbf{\Pi}} \triangleq \begin{bmatrix} \hat{\pi}\left(s_1 \mid q_1\right) & \dots & \hat{\pi}\left(s_{|\mathcal{A}|} \mid q_1\right) \\ \vdots & \ddots & \vdots \\ \hat{\pi}\left(s_1 \mid q_{|Q|}\right) & \cdots & \hat{\pi}\left(s_{|\mathcal{A}|} \mid q_{|Q|}\right) \end{bmatrix}. \tag{2.16}$$

The stochastic matrix $\widehat{\mathbf{\Pi}}$, estimated from a symbol string, is then treated as a feature of the times series data that represents the behavior of the dynamical system. These features can then be used for different learning applications like pattern matching, classification, and clustering. The estimated stochastic matrices $\widehat{\mathbf{\Pi}}$ are converted into row vectors for feature divergence measurement by sequentially concatenating the $|Q|$ rows (i.e., the $|Q| \times |\mathcal{A}|$ matrix $\widehat{\mathbf{\Pi}}$ is vectorized as a $(1 \times |Q||\mathcal{A}|)$ vector). The convergence of the parameters of the Markov model could be established by Glivenko-Cantelli theorem [69, 70] which guarantees the convergence of empirical distribution of random variables under independent observations.

Remark 2.5.1 *As compared to the hidden Markov model-based approaches, this*

approach presents a much simpler modeling technique where the structure of the model is inferred based on the observations. As the structure of the Markov model is also fixed by the observations, this makes inference of the parameters of the Markov chain much more easier than the standard hidden Markov model approaches like the forward-backward algorithm [23] where dynamic programming is used to estimate the parameters that maximize the likelihood of the observations.

The sufficient statistics for the inferred Markov process serve as a compact stochastic representation of the signal and can be used for different machine learning applications like pattern matching, classification, clustering, anomaly detection. As explained earlier, the symbol emission probabilities conditioned on the memory words for the discrete process could be modeled as random variables. A metric to measure the information gain over the marginal symbol emission probabilities could be defined as follows:

$$\mathfrak{d}_\theta = \sum_{q \in Q} \Pr(q) D_{KL}(P(\mathcal{A} \mid q) \| \tilde{P}(\mathcal{A})) \tag{2.17}$$

where, θ represents the parameters of the Markov model inference, i.e., the partitioning map and the estimated order of the memory words. The term in equation (2.17) measures the measures the discrepancy in the statistics of the symbol emission probabilities when they are conditioned on the memory words as compared to the unconditional statistics for the symbol emission probabilities.

2.6 Sequential Tests for D-Markov Models

A sequential detector is a statistical decision function which uses a random number of samples depending on the observation sequence to detect the underlying hypothesis. A sequential detector, on the average, would need much smaller sequence length than FSS tests [71]. Sequential probability ratio test (SPRT) [72,73] has been traditionally used for binary hypothesis testing and is known to be an optimal detector when the observation sequences are independent and identically distributed (IID) [74]. However, sequential data from physical systems are typically not independent as causal relations, due to physics, leads to statistical dependencies. Hidden Markov models (HMM) are temporal evolution models that are learned from data using

iterative techniques to capture some of those dependencies [75]. In [76–78], the SPRT was extended for data generated from a HMM. Synthesizing sequential tests for $D-$Markov models will help making quicker decisions during test for time-critical processes.

The probability density of the Markov model is represented as a product of categorical distributions and the parameters of this distribution come from a Dirichlet distribution. Then using that the Dirichlet distribution is the conjugate prior of the categorical distribution, we develop a sequential update rule for likelihood ratios of Markov models to test a Markov model against an alternate Markov model for detection problems. Expected increment of the log-likelihood ratios are explicitly provided under each hypothesis and it is shown that the sequential tests for the Markov models will terminate in finite time with probability one. For more detailed discussion, interested readers are referred to [68].

2.6.1 Likelihood for D-Markov models

The D-Markov model for each hypothesis is represented by a morph matrix, as discussed in Section 2.1. Each row of the morph matrix of a D-Markov model is a discrete probability mass function denoting probability of emission of a symbol from a given state. Since the occurrence of states are statistically independent events, the rows represent a set of independent categorical distribution [79]. The Dirichlet distribution is known to be the conjugate prior of the categorical distribution [75]. In other words, if the prior density over parameters of the categorical distribution is represented as a Dirichlet distribution, then posterior density after a sequential update is also given by a Dirichlet distribution. Thus, for a given state, the density over parameters of a particular row of morph matrix is given by Dirichlet distribution. Let S_t for $t \in \mathbb{N}$ denote the symbol sequence $(s_1, s_2, \ldots, s_t)$ of length t. The likelihood of this sequence conditioned on a given model G_k is computed [80] as:

$$P(S_t \mid G_k) = \prod_{i=1}^{|\mathcal{Q}|} \left(\frac{\prod_{j=1}^{|\mathcal{A}|} (m_{ij}^k)^{\alpha_{ij}^t - 1}}{\mathrm{B}(\bar{\alpha}_i^t)} \right) \tag{2.18}$$

where the hyper-parameter α_{ij}^t is initialized at α_{ij}^0, the value of $\alpha_{ij}^t - \alpha_{ij}^0$ is the count of occurrence of symbol s_j after state q_i in a sequence S_t, $\bar{\alpha}_i^t$ denotes the vector $[\alpha_{i1}^t, \alpha_{i2}^t, \ldots, \alpha_{i|\mathcal{A}|}^t]$ for all $i \in \{1, 2, \ldots, |\mathcal{Q}|\}$, $\mathrm{B}(\bar{\alpha}_i^t) = \frac{\prod_{j=1}^{|\mathcal{A}|} \Gamma(\alpha_{ij}^t)}{\Gamma(\sum_{j=1}^{|\mathcal{A}|} \alpha_{ij}^t)}$, and $\Gamma(\cdot)$ is

the Gamma function.

In binary hypothesis testing, such as Bayes' or Neyman-Pearson hypothesis testing, the log-likelihood ratio is computed and compared with a predetermined threshold [71]. In the proposed approach, we use (2.18) to compute the log-likelihood ratio as follows:

$$\Lambda_t = \log\left(\frac{P(S_t \mid G_1)}{P(S_t \mid G_0)}\right) = \log\left(\frac{\prod_{i=1}^{|\mathcal{Q}|}\prod_{j=1}^{|\mathcal{A}|}(m_{ij}^1)^{\alpha_{ij}^t-1}}{\prod_{i=1}^{|\mathcal{Q}|}\prod_{j=1}^{|\mathcal{A}|}(m_{ij}^0)^{\alpha_{ij}^t-1}}\right) \tag{2.19}$$

$$= \log\left(\prod_{i=1}^{|\mathcal{Q}|}\prod_{j=1}^{|\mathcal{A}|}\left(\frac{m_{ij}^1}{m_{ij}^0}\right)^{\alpha_{ij}^t-1}\right) \tag{2.20}$$

$$= \sum_{i=1}^{|\mathcal{Q}|}\sum_{j=1}^{|\mathcal{A}|}(\alpha_{ij}^t - 1)\log\left(\frac{m_{ij}^1}{m_{ij}^0}\right), \tag{2.21}$$

$$\Lambda_t = \sum_{i=1}^{|\mathcal{Q}|}\sum_{j=1}^{|\mathcal{A}|} w_{ij}(\alpha_{ij}^t - 1), \tag{2.22}$$

where the constant w_{ij} denotes $\log(\frac{m_{ij}^1}{m_{ij}^0})$. We obtained a simple relation to compute the log-likelihood ratio in (2.22) and next we will find the sequential update rule for the log-likelihood ratio.

Proposition 2.6.1 (Sequential Update) *Given that the log-likelihood ratio at time t is Λ_t and the state at time $t+1$ is q_{t+1}, if s_{t+1} is the emitted symbol, then the updated log-likelihood ratio Λ_{t+1} is given by :*

$$\Lambda_{t+1} = \Lambda_t + w_{q_{t+1}s_{t+1}}, \tag{2.23}$$

where $w_{q_{t+1}s_{t+1}} = \{w_{ij} : q_{t+1} = i \text{ and } s_{t+1} = j\}$.

Proof 2.6.1 *Recall that the hyper-parameter $\alpha_{ij}^{t+1} - \alpha_{ij}^0$ is the count of occurrence of symbol s_j after the state q_i in the observed symbol sequence S_{t+1}, then it follows that:*

$$\alpha_{ij}^{t+1} = \alpha_{ij}^t + \mathbb{1}_{\{i\}}(q_{t+1})\mathbb{1}_{\{j\}}(s_{t+1}), \tag{2.24}$$

where $\mathbb{1}_A(\cdot)$ is the indicator function with set A, that is, $\mathbb{1}_A(x) = 1$, if x belongs to A; otherwise, $\mathbb{1}_A(x) = 0$. Then, by using the log-likelihood ratio given in (2.22) for

time $t+1$*, and* (2.24)*, we obtain:*

$$\begin{aligned}\Lambda_{t+1} &= \sum_{i=1}^{|\mathcal{Q}|}\sum_{j=1}^{|\mathcal{A}|} w_{ij}(\alpha_{ij}^{t} + \mathbb{1}_{\{i\}}(q_{t+1})\mathbb{1}_{\{j\}}(s_{t+1}) - 1) \\ &= \sum_{i=1}^{|\mathcal{Q}|}\sum_{j=1}^{|\mathcal{A}|} w_{ij}(\alpha_{ij}^{t} - 1) + w_{ij}\mathbb{1}_{\{i\}}(q_{t+1})\mathbb{1}_{\{j\}}(s_{t+1}) \\ &= \Lambda_t + w_{q_{t+1}s_{t+1}}.\end{aligned}$$

The simplicity of sequential update of the log-likelihood ratio for D-Markov models enables online update of the statistic $\Lambda_t(S_t)$. This factor would aid in real-time monitoring of physical systems.

These log-likelihood ratio could be used to design a probability ratio test given the probability of detection, p_d and the probability of false alarms, p_{fa}. The probability of detection and false-alarms could be used to decide the thresholds for stopping the sequential tests. The guarantee on termination of the sequential test in finite time could be established using the expected change in the log-likelihood of the Markov models. More results on the same could be found in [68].

2.7 xD-Markov Modeling

As described earlier in Section 2.1, there is a natural extension to the current Markov modeling approach to modeling of temporal dependence between two different symbol sequences. This section presents the details of inferring these models based on the results in earlier sections.

The xD-Markov modeling problem could be stated as follows:

> Given two symbolic stochastic processes $\mathbb{S}_1$ and $\mathbb{S}_2$, create a generative model, $G_{1\to 2}$, for the process $\mathbb{S}_2$ based on the observations of the stochastic process $\mathbb{S}_1$.

This requires inference of the generative model of causal dependence between the two processes. The temporal dependence is modeled by assuming a Markov structure between the observed variables. Such a cross-dependence is represented as crossed automata which induces a Markov chain.

The states of the crossed PFSA are inferred by using the state-splitting algorithm which under the D-Markov assumption leads to states of variable depth for the crossed PFSA. The states of the crossed PFSA are split using entropy rate as the metric, where the maximum decrease in entropy rate is used to select the state to split. Analogous to the entropy rate of the PFSA defined earlier, the entropy of a crossed PFSA $G_{1\to 2} = (\mathcal{Q}_1, \mathcal{A}_1, \mathcal{A}_2, \delta, \boldsymbol{M})$ is defined next.

Definition 2.7.1 *(Entropy rate of* $\mathbf{x}$*D-Markov machines [8]) The entropy rate of a crossed PFSA* $G_{1\to 2} = (\mathcal{Q}, \mathcal{A}_1, \mathcal{A}_2, \delta, \boldsymbol{M})$ *representing the causal dependence of the process* $\mathbb{S}_2 = \{\sigma_t \in \mathcal{A}_2 : t \in \mathbb{N}\}$ *based up on observations of the state-space of the stochastic process* $\mathbb{S}_1 = \{s_t \in \mathcal{A}_1 : t \in \mathbb{N}\}$ *is defined as*

$$\begin{aligned} H(\mathcal{A}_2|\mathcal{Q}) &\triangleq \sum_{q_i \in \mathcal{Q}} P(q_i) H(\mathcal{A}_2|q_i) \\ &= -\sum_{q_i \in \mathcal{Q}} \sum_{\sigma_j \in \mathcal{A}_2} P(q_i) P(\sigma_j|q_i) \log P(\sigma_j|q_i) \\ &= -\sum_{q_i \in \mathcal{Q}} \sum_{\sigma_j \in \mathcal{A}_2} P(q_i) \boldsymbol{M}_{ij} \log \boldsymbol{M}_{ij} \end{aligned} \tag{2.25}$$

where $P(q_i)$ *is the probability of the PFSA state* $q_i \in \mathcal{Q}$ *and* $P(\sigma_j|q_i)$ *is the conditional probability of the symbol* $\sigma_j \in \mathcal{A}_2$ *given that a PFSA state* $q_i \in \mathcal{Q}$ *has been observed.*

The process of splitting a state $q \in Q$ is executed by replacing the symbol block for q by its branches given by the set $\{sq : s \in \mathcal{A}_1\}$. Then, the maximum reduction of the entropy rate of the crossed PFSA $G_{1\to 2}$ is the governing criterion for selecting the state to split. As a result, not all the states are split and thus, this creates a variable depth (or memeory) crossed-PFSA structure. It is noted that this approach is also a greedy approach which creates a model with suboptimal predictive accuracy; however, this is helpful in restricting the state-space of the constructed automaton. Another point to be noted is that the entropy rate is a submodular function of the size of state-space; thus adding more states will lead to a slower rate of decrese in the entropy rate. Thus, a stopping rule is used to terminate the splitting algorithm based on either the rate of change of entropy rate or the maximum number of states allowed in the final automaton. The state-splitting algorithm for inferring the crossed-PFSA is delineated as an algorithm

in Algorithm 1. The parameters of the crossed Markov model are estimated using maximum a posteriori (MAP) rule with uniform prior. Let $N(\sigma_j|q_i)$ denote the number of times that a symbol σ_j is generated in $\mathbb{S}_2$ when the state q_i as the symbol string is observed in the crossed PFSA $G_{1\to2}$. The maximum a posteriori probability (MAP) estimate of the emission probability of the symbols $\sigma \in \mathcal{A}_2$ conditioned on $q_i \in \mathcal{Q}$ is estimated by frequency counting [27] as follows.

$$\hat{p}_{ij}(\sigma_j|q_i) = \frac{1 + N(\sigma_j|q_i)}{|\mathcal{A}_2| + \sum_j N(\sigma_j|q_i)} \tag{2.26}$$

Algorithm 1: State splitting for variable-depth **x**D-Markov machine

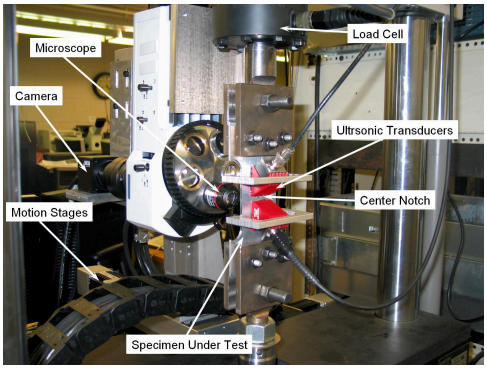
Input: Symbol sequences $\mathbb{S}_1 = \{\dots s_1 s_2 s_3 \dots | s_i \in \mathcal{A}_1\}$ and
$\mathbb{S}_2 = \{\dots \sigma_1 \sigma_2 \sigma_2 \dots | \sigma_i \in \mathcal{A}_2\}$
User Input: Maximum number of states N_{max} and ϵ
Output: PFSA $G_{1\to2} = (\mathcal{Q}, \mathcal{A}_1, \mathcal{A}_2, \delta, \boldsymbol{M})$

```
Create a 1-Markov machine Q* := A_1;
Estimate the size of temporal memory, D(ε) for s⃗ using equation (3.3);
while |Q*| < N_max or H(A_2|Q) − H(A_2|Q*) < ε do
    Q := Q*;
    Q* = arg min_{Q'} H(A_2|Q') where Q' = Q\q ∪ {sq : s ∈ A_1} and q ∈ Q;
for q ∈ Q* and s ∈ Σ_1 do
    if δ(q, s) is not unique then
        Q* := Q*\q ∪ {sq : s ∈ A_1}
                ▷ Ensures consistent algebraic structure for M;
return G_{1→2} = (Q, A_1, A_2, δ, M)
```

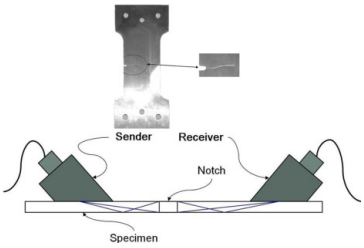
If no event is generated at a combination of symbol $\sigma_j \in \mathcal{A}_2$ and state $q_i \in Q$, then there should be no preference to any particular symbol and it is logical to have $p(\sigma_j|q_i) = \frac{1}{|\mathcal{A}_2|}$. The above procedure guarantees that the **x**D-Markov machines, constructed from two (finite-length) symbol strings, must have an (elementwise) strictly positive cross emission matrix.

2.8 Statistical Learning Examples from Engineering Systems

In this section, we present two examples of learning Markov models for time-series data using data from two different electro-mechanical systems. First we present



(a) Computer-instrumented test apparatus



(b) Cracked 7075-T6 Aluminum alloy specimen and the set-up of ultrasonic sensors on the specimen

Figure 2.4: Fatigue test apparatus used for experiment

results on order estimation using data collected during fatigue of a polycrystalline alloy structure in laboratory conditions. Next, an example of input-output modeling using the x$D-$ Markov model is presented with some examples of anomaly detection and state-of-charge estimation.

2.8.1 Fatigue Damage in Polycrystalline alloy structures

This section describes the experiments used for collection of time-series data of fatigue damage evolution in polycrystalline alloys. A detailed description of crack-initiation in polycrystalline alloys during fatigue is provided in Appendix B. This is followed by a description of D-Markov modeling of the time-series data and unsupervised clustering of fatigue data for identification of different behaviors during fatigue.

2.8.1.1 Experiment Details

This subsection describes the experimental apparatus used for collection of time-series data of fatigue damage evolution in polycrystalline alloy specimens. Figure 2.4 shows a picture of the experimental apparatus that is built upon a computer-instrumented and computer-controlled uniaxial fatigue testing machine. The apparatus is instrumented with ultrasonic flaw detectors and an optical traveling microscope; the details of the operating procedure of the fatigue test apparatus and its instrumentation & control system are reported in [81]. Tests have been conducted using side-notched 7075-T6 aluminum specimens. The test specimens have been subjected to sinusoidal loading under tension-tension mode (i.e., a con-

stant positive offset) at a frequency of 75 Hz. A constant bias offset is provided in the load cycling to ensure that the specimen is always under tension. The tests have been conducted at a constant amplitude sinusoidal load for low-to-medium cycle fatigue, where the minimum and maximum loads are kept constant at 4.85 and 87 MPa, respectively. Each specimen is 3 mm thick and 50 mm wide, and has a slot of 1.58 mm $\times$ 4.5 mm on one side. The side notch increases the stress concentration factor that ensures crack initiation and propagation at the notch end [81]. The ultrasonic sensing device is triggered at a frequency of 5 MHz at each peak of the sinusoidal load. Data are collected after every 2000 load cycles when the frequency of loading is reduced to 0.05 Hz and data for four sinusoidal load cycles are collected. Tests are repeated for five similar samples under the same loading conditions. Time series from one of the specimens is used for comparative study of different techniques for D-Markov modeling while all five specimens are used to study the clustering behavior. Each data set consists of 204 sets of time series data from one sample as it goes from healthy state to fully cracked. The length of each time-series is $20,000$.

2.8.2 D-Markov Modeling

Time-series data is first normalized by subtracting the mean and dividing by the standard deviation of its elements; this step corresponds to bias removal and variance normalization. Data from engineering systems is typically oversampled to ensure that the underlying dynamics can be captured, in this case 5 MHz. Due to coarse-graining from the symbolization process, an over-sampled time-series may mask the true nature of the system dynamics in the symbolic domain (e.g., occurrence of self loops and irrelevant spurious transitions in the PFSA). Time-series is first down-sampled to find the next crucial observation. The first minimum of auto-correlation function generated from the observed time-series is obtained to find the uncorrelated samples in time. The data sets are then down-sampled by this lag, to avoid discarding significant amount of data due to downsampling, down-sampled data using different initial conditions is concatenated, further details of this preprocessing can be found in [58].

The continuous time-series data set is then partitioned using maximum entropy partitioning (MEP) [32], where the information rich regions of the data set are

partitioned finer and those with sparse information are partitioned coarser. In essence, each cell in the partitioned data set contains (approximately) equal number of data points under MEP. A binary alphabet with $\mathcal{A} = \{0, 1\}$ has been used to symbolize the continuous fatigue damage data.

Clearly, the choice of the partitioning method for symbolization will affect the ability of the D-Markov machine G to resolve the dynamics of the underlying system. The choice of the partitioning method is dependent on the goal of the modeling process, readers are referred to survey papers [38, 39] for more details on the choice of symbolization. Work presented here deals only with the task of depth estimation using various approaches and their comparison, given a symbolic sequence.

2.8.3 Results and discussion

Given a symbolic sequence depth is estimated using (1) log-likelihood, (2) signal reconstruction, (3) state splitting via entropy rate and (4) spectral decomposition method. Each approach essentially employs a metric ρ to evaluate different values of depths for modeling the given symbolic sequence. A value D_{opt} is chosen when the gain in performance for $D \geq D_{\text{opt}}$ is marginal. A normalized metric $\rho_{\text{norm}} \in [0, 1]$ is defined as follows for easy comparison of all approaches.

$$\rho_{\text{norm}} = \frac{\rho - \rho_{\text{min}}}{\rho_{\text{max}} - \rho_{\text{min}}} \tag{2.27}$$

Clearly, this normalization does not affect D_{opt} as the optimal depth estimate only depends on the relative improvement of the model G with increasing values of D.

Likelihood Rate

Data from a healthy specimen i.e. without any crack incubation and crack initiation is used for learning the PFSA model structure. Several passes of the data are made to learn PFSA models corresponding to different values of the depth parameter, D. Based on the inferred models, the log-likelihood of the remaining symbol sequences in the data set (test set) is calculated using Eq. (3.15) This process is repeated for multiple training sequences and tested on the remaining sequences. The Maximum Likelihood estimate (MLE) of depth D is chosen to be the smallest value of D such

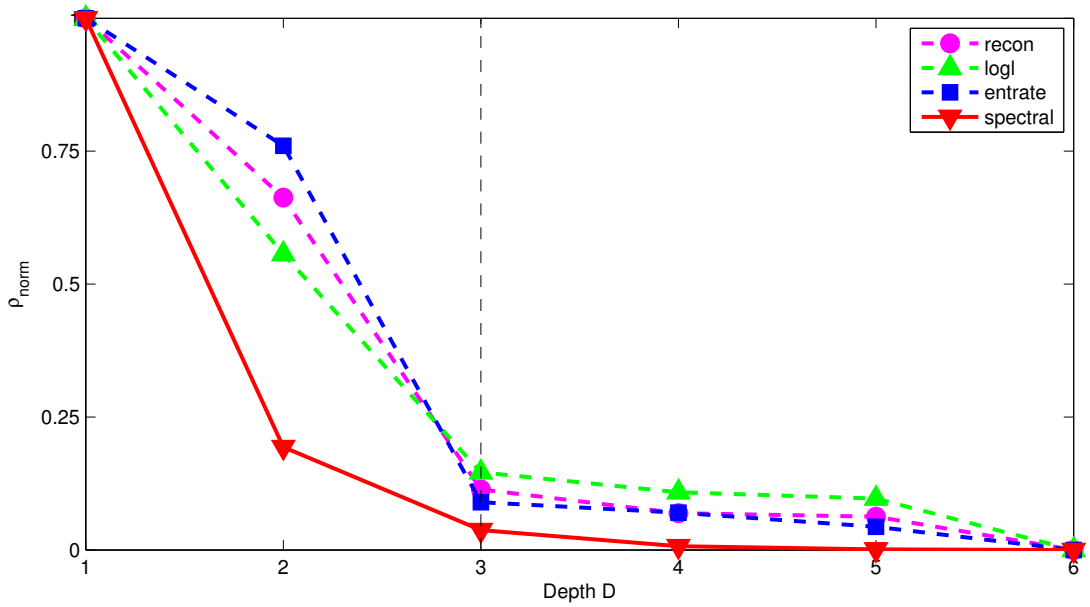


Figure 2.5: Comparison of depth estimation using (a) spectral decomposition (b) log-likelihood (c) state splitting via entropy rate and (d) signal reconstruction. The metric ρ used for estimating the depth for each approach has been normalized as $\rho_{\text{norm}} = (\rho - \rho_{\text{min}})/(\rho_{\text{max}} - \rho_{\text{min}})$ in the above plot for easy comparison. Note that this normalization does not affect the depth selection process.

that increasing D does not improve the likelihood of test sequences significantly. Negative of the log-likelihood is used as the metric ρ for easy comparison with other approaches. Therefore lower values of the $\rho = -\mathcal{L}(\vec{s}|G)$ indicate better predictability of the test symbol sequences given the learned model G

Figure 2.5 shows the a plot of normalized metric ρ_{norm} of test symbol sequences computed using various values of D. It can be seen that after $D = 3$, gain in model predictability tapers off and using this approach $D_{\text{opt}} = 3$ can be chosen as a suitable value of depth. For this case, $\rho_{\text{min}} = 0.454$ and $\rho_{\text{max}} = 0.671$ were used to compute the ρ_{norm}.

Signal Reconstruction

For this approach, each symbol is mapped back to the continuous domain by computing the average of the continuous values that are observed for that bin or partition. For example, the signal range $[-3.8760, -0.0072]$ which is represented by symbol 0 in the symbolic domain, is mapped back to the continuous domain using the average of value -0.5903 of training data in that range.

Similar to previous procedure, the training set here corresponds to the time series generated from a healthy specimen. Using training data, a PFSA model G is learned for depth D. Given the learned model and past D symbols of a test symbol at time-step k, a one-step symbol prediction $\hat{s}_k$ is made. The symbol $\hat{s}_k$ is predicted by sampling from the conditional distribution $\Pr(s_k|s_{k-1}, \ldots, s_{k-D})$ which is parametrized by the matrix $\boldsymbol{M}$

The predicted symbol sequence is mapped back to the continuous domain using the inverse map using partition means described earlier. Different PFSA models with increasing values of depths D are learned, and the metric ρ used for depth selection in this case is defined as the mean signal reconstruction error. For every value of D, the process of estimating the reconstruction error is repeated 100 times to obtain a statistically meaningful estimate. Figure 2.5 shows how the signal reconstruction error based approach compares with other methods of depth estimation. Similar to previous approach $D_{\text{opt}} = 3$. For this case, $\rho_{\min} = 0.740$ and $\rho_{\max} = 0.864$ were used to compute ρ_{norm}.

State Splitting using Entropy Rate

As explained in Section 2.4.3, this approach uses a top-down approach of state splitting to infer the structure of the PFSA. We have used a stopping rule based on the maximum allowed depth for a state. Given D, state-splitting is performed using entropy rate as the criteria until a maximum number of states $N_{\max}$ is reached. Figure 2.6 shows a plot of entropy rate against $N_{\max}$. Clearly, as the maximum number of allowed states $N_{\max}$ is increased, we can get a better model indicated by a lower value of the entropy rate. It must be noted that in this approach different states of the PFSA are allowed to have different depths, as all states may not get split to reduce entropy rate, see section 2.4.3 and figure 2.3. Given a model constructed using the state-splitting method, we can use the maximum depth used for any state in that model as the depth of the underlying symbolic process. In figure 2.6 models that have the same maximum depth over its states are shown in same color and separated by vertical lines.Therefore, as we go from $N_{\max} = 4$ to $N_{\max} = 7$, the maximum depth of any state in the corresponding PFSAs remains fixed at 3.

For this approach, the metric ρ used for selection of D_{opt} is the minimum entropy-rate achieved for a given value of depth. The intuition is that minimum

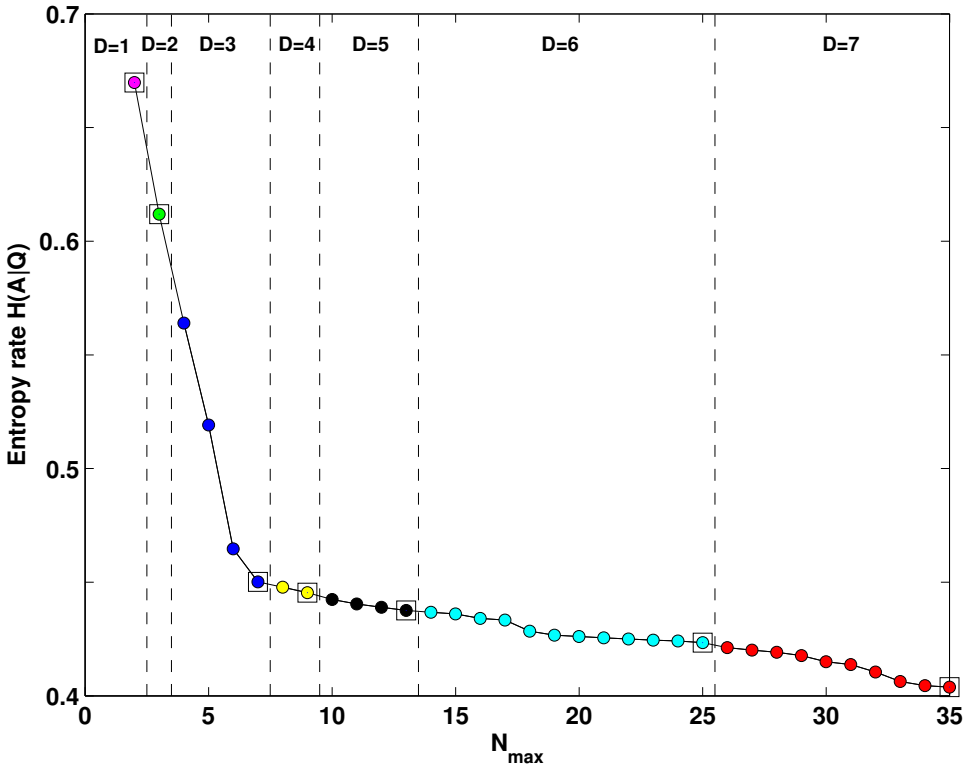


Figure 2.6: Entropy rate against $N_{\max}$ for state-splitting method. models with same depth are shown in same color and separated by vertical lines; and the minimum entropy rate achieved for a fixed value of depth is marked by a circle

value of entropy rate for a given value of depth corresponds to the best model that can be fit to the data using this approach. These points are marked by a square in figure 2.6

Figure 2.5 shows a plot of normalized metric ρ_{norm} with various values of D, similar to previous cases $D_{\text{opt}} = 3$. For this case, $\rho_{\min} = 0.428$ and $\rho_{\max} = 0.671$ were used to compute ρ_{norm}.

Spectral decomposition

For this approach, the one-step transition matrix $\mathbf{\Pi}$ is computed by going over the symbol sequence once. Then using spectral decomposition the Eigen values of $\mathbf{\Pi}$ are computed and equation 3.3 is used to estimate the depth of the symbolic process. For comparison with other methods and computing D_{opt}, the metric ρ for spectral decomposition approach is taken to be $\sum_{j=2}^{J} |\lambda_j|^D$ for a given value of

D. Figure 2.5 shows a plot of ρ_{norm} against different values of D. Similar to other methods, spectral decomposition also estimates $D_{\text{opt}} = 3$ for $\epsilon = 0.03$. For this case, $\rho_{\min} = 0$ and $\rho_{\max} = 0.194$ were used to compute ρ_{norm}.

2.8.3.1 Unsupervised Clustering of Fatigue Data

Unsupervised clustering of fatigue data for different test specimens has been performed to demonstrate the efficacy of depth prediction by the spectral decomposition method. Based on the behavior of the fatigue damage process, the data-sets have been divided into three different classes: (i) crack incubation, (ii) crack initiation, and (iii) crack propagation. The data are symbolized with an alphabet size $|\mathcal{A}| = 3$. Depth corresponding to the new alphabet size is estimated using the one-step transition matrix and $\epsilon = 0.03$. The estimated depth for a specimen is chosen to be the depth that is optimal for $\approx 75\%$ of data sets from that specimen. Then PFSAs are learned for data sets corresponding to instances from each phase of damage propagation. Using unsupervised clustering, PFSA parameters $\mathbf{\Pi}$, learned in the previous step, are clustered using k-means clustering. Cosine similarity is used as the distance measure and the number of replicates is 50 for k-means clustering.

Cluster purity is calculated to measure the efficacy of a chosen value of D to resolve the underlying system dynamics. To compute purity, each cluster is assigned to the class which is most frequent in the cluster; then, the accuracy of assignment is measured by counting the number of correctly assigned samples and dividing it by the total number of samples. Formally, purity is defined as follows:

Definition 2.8.1 (Purity) *Purity* $(\Omega, \mathbb{C}) = \frac{1}{N} \sum_k \max |\omega_k \cap c_j|$
where, $\Omega = \{\omega_1, \omega_2, \ldots, \omega_k\}$ *is the set of clusters and* $\mathbb{C} = \{c_1, c_2, \ldots, c_j\}$ *is the set of classes.*

Table 2.1 shows the purity value computed for data from five specimens for different values of depths. The **bold** boxed values of purity correspond to the estimated depth for that specimen. It can be seen that the purity values saturate at the depth predicted by spectral decomposition for all specimens except specimen 2, for which we still see slight improvement beyond depth of 4. It was also found that the intra-cluster similarity and inter-cluster dissimilarity improves and then saturates as the depth predicted by spectral decomposition is reached. It shows that spectral decomposition method can estimate the correct depth of the symbolic

Table 2.1: Cluster purity for five samples for different depths

Sample	$D = 1$	$D = 2$	$D = 3$	$D = 4$	$D = 5$	$D = 6$
1	0.764	0.764	**0.850**	0.850	0.850	0.850
2	0.835	0.835	0.870	**0.900**	0.910	0.960
3	0.788	0.788	**0.858**	0.852	0.852	0.852
4	0.750	0.737	**0.816**	0.816	0.816	0.816
5	0.899	0.899	0.943	**0.943**	0.943	0.943

process. Also, by using the correct depth a PFSA model can accurately model different phases of damage propagation and distinguish between them. This is very desirable for health monitoring and fault detection tasks as one wants to detect faults early during the initiation phase before it manifests itself as a full crack.

Remark 2.8.1 *A binary alphabet $|\mathcal{A}| = 2$ as used earlier, was found to be too coarse a partition for resolving all three regimes of damage propagation. Analysis with the binary alphabet showed that only two classes viz., the incubation and propagation are visible at this level of symbolic granularity. Also, the internal cluster structure monotonically improves such that the similarity of the same class improves while simultaneously the separation between distinct classes increases. Finally, the cluster structures converge as the depth predicted by the spectral decomposition for $|\mathcal{A}| = 2$ is reached.*

2.8.3.2 Concluding Remarks

For systems with fading memory Markov models are a low complexity approximation that can model the dynamics of the underlying system. Estimation of the length of the temporal memory is a critical step and it helps determine the structure of the Markov models. Three different popular approaches for estimating depth were compared with a recently proposed method based on the spectral decomposition of the one step transition matrix. The spectral decomposition approach is a low complexity method that does not require several passes over the symbolic sequence.

Each of the method compared relied on a metric to find the optimal depth beyond which higher depths yielded only marginal gains with respect to the chosen metric. While the log-likelihood method relied on symbol-wise prediction, the signal reconstruction error used prediction in the continuous domain. The method of state-splitting relies on entropy rate and attempts to find the smallest model that

fits the data. The spectral decomposition approach relies on the distance of the n-step transition matrix from the stationary one as n tends to infinity.

The four approaches were compared using damage propagation data sets from fatigue life monitoring experiments done on polycrystalline alloys. It was found that all the methods agreed on the estimate of depth. It was also shown using unsupervised clustering that beyond the estimated depth, the clustering performance did not improve much. Moreover, using the estimated depth, all three phases of damage propagation could be resolved. This is a very desirable outcome as we want to capture incipient fault signatures in the initiation phase.

The information lost during partitioning for symbolization can affect the dynamics that is observable in the symbolic domain. Clearly, poor symbolization, e.g. only symbol, can potentially obscure critical dynamical behavior and the models constructed with estimated depths may not be able to accurately represent the underlying system dynamics. Since the spectral decomposition approach works with symbolic sequences, depth estimated with this approach should be taken to represent the longest temporal influence observed in the symbolic domain, not in the continuous domain. This issue was highlighted when clustering using a binary alphabet and estimated depth was not able to resolve the initiation phase of fatigue crack propagation.

Overall, the spectral decomposition method for depth estimation was found to be consistent with other popular methods, while being significantly fast. Computational gains results from the fact that the spectral decomposition method is not a wrapper approach that builds many models and then selects the best one. Only one pass through the data set is required for depth estimation.

The interplay of symbolization and depth estimation for task of modeling the dynamics of a fading memory system needs to be studied further. For instance it might be possible to guide the symbolization process based on the predictability of the models constructed using the estimated depth. Performance of the spectral decomposition approach on new data sets for applications such as anomaly detection, pattern classification needs to be evaluated. These topics are suggested for future research.

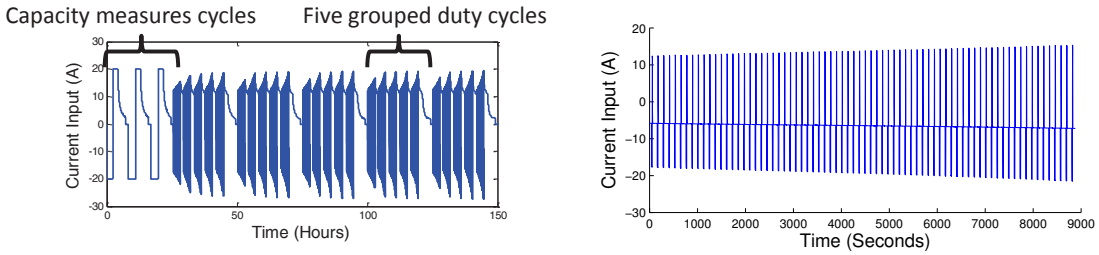
2.8.4 Input-Output Modeling in Battery Systems

A brand new ($12V$ AGM VRLA with $56Ah$ capacity) lead-acid battery has been used in the experiments. As the battery is charged/discharged according to given input (current) profiles at the room temperature and an ensemble of synchronized time-series of the input charge/discharge current and output voltage responses is collected at the sampling frequency of $1Hz$. A typical input current profile for this experiment is depicted in Fig. 2.7. The duration of a input profile is ~ 150 hours, which consists of three capacity measure cycles that are followed by 25 duty cycles.

A capacity measurement cycle is a slow, full discharge/discharge cycle, where the battery is fully discharged followed by a full charge. In this constant-current constant-voltage (CCCV) operation of the experiments, the battery is discharged by a constant current of $-20A$ for ~ 2 hours at first. Then it is charged first by constant current of $20A$ until its output reaches a voltage of $13.8V$; this voltage is kept constant for the next 3 hours with gradually decreasing charging current. The maximum capacity at that time is measured by integrating the current during the charge period. Three computed battery maximum capacities are obtained from these capacity measurement cycles, the mean value of them is considered as the nominal maximum capacity for that particular time. Since there are five similar (i.e., same pattern) input profiles in total are applied to the battery during the whole experiment. The degradation of battery SOH (i.e., the ratio of maximum capacity between "now" and when it is brand new) is obtained.

Total 25 duty cycles are divided into groups of five. The transition time between two consecutive groups is ~ 6 hours with charging to full capacity, while the transition time between two consecutive duty cycles in one group is ~ 1.2 hours with inadequate recharging. Each duty cycle last ~ 2.7 hours, which is composed of ~ 75 "Hotel-Pulse" cycles, as depicted in Fig. 2.7b. Each individual "Hotel-Pulses" cycle (i.e., duration of $120s$) 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, as shown in Fig. 2.7a. The amplitude of the "hotel" load and the discharging & charging pulses are not usually monotonically increasing in a duty cycle, which makes each duty cycle slightly different from others. This pattern of input cycles largely simulates a real-time working condition for an electric locomotive. Further details of the

time-series data characteristics could be found in [82].



(a) Input Current Profile (b) Input Current for one duty cycle

Figure 2.7: Profile of input current data for the experiment

2.8.4.1 xD-Markov Modeling

Time-series data for both the input and output are first normalized individually by subtracting the mean and dividing by the standard deviation of their elements; this step corresponds to bias removal and variance normalization. The input and output data are normalized in a moving window fashion to get rid of any trend or drooling behavior in the data. Then, a wavelet-based segmentation [82] is done to extract relevant segments of the data based on their frequency content.

Data from engineering systems is typically oversampled to ensure that the underlying dynamics can be captured. Due to coarse-graining from the symbolization process, an over-sampled time-series may mask the true nature of the system dynamics in the symbolic domain (e.g., occurrence of self loops and irrelevant spurious transitions in the xD-Markov machine). Time-series is first down-sampled to find the next crucial observation. The first minimum of the absolute auto-correlation function generated from the observed time-series is obtained to find the uncorrelated samples in time. The data sets are then down-sampled by this lag. The time lag for the output data is used to downsample both the input and output data. The rationale is to keep the data synchronous and not miss the relevant dynamics of the observed output. To avoid discarding significant amount of data due to downsampling, down-sampled data using different initial conditions is concatenated. Further details of this pre-processing can be found in [9, 58].

The continuous time-series data set is then partitioned using maximum entropy partitioning (MEP) [32], where the information rich regions of the data set are partitioned finer and those with sparse information are partitioned coarser. In

essence, each cell in the partitioned data set contains (approximately) equal number of data points under MEP. A ternary alphabet $\mathcal{A}$ (i.e., $|\mathcal{A}| = 3$) has been used to symbolize both the input and output data individually.

Clearly, the choice of the partitioning method for symbolization will affect the ability of the xD-Markov machine $G_{1\to 2}$ to resolve the dynamics of the underlying cross-dependence. The choice of the partitioning method is dependent on the goal of the modeling process, readers are referred to survey papers [38, 39] for more details on the choice of symbolization. Work presented here deals only with the task of modeling the cross-dependence between the symbol sequences given that some technique has already been used for discretization of the observed time-series. Once the data are symbolized, the state splitting algorithm presented earlier in section 2.7 is applied to infer the variable depth (D) of the cross model from input current to output voltage for different values of SOH.

2.8.4.2 Results and Interpretation

This section presents the results to demonstrate the efficacy of the proposed method for health monitoring of a lead-acid battery.

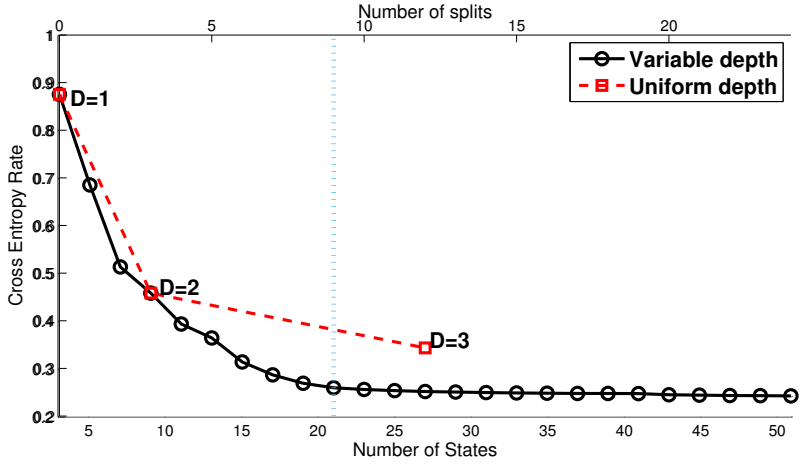


Figure 2.8: Cross entropy rate vs number of splits (SOH = 1, 3 symbols, downsample time-lag = 7)

Figure 2.8 shows the behavior of the change in cross entropy rate (see Eq. (2.25)), that is defined from input current to output voltage, with an increasing number of state splitting in the PFSA constructed from input current with alphabet $\mathcal{A}$. The output voltage symbol sequence has also the same cardinality. The analysis is

carried out at the perfect health of the battery i.e., SOH = 1. Figure 2.8 shows a monotonic decrease in the cross-entropy rate with negligible improvement after the splitting tree contains a certain number of states (implying convergence). In the absence of a competing objective to minimize the complexity of the model obtained by splitting, the state-splitting is terminated when addition of new states does not significantly improve the cross-entropy rate. After the 9th split at the state space of cardinality 21 (denoted by vertical dotted line), the entropy rate improvement becomes negligible as shown in Figure 2.8. This particular state space is chosen subsequently for modeling the cross dependence at different levels of SOH. Figure 2.8 also shows that the cross entropy rate for uniform depth scenario decreases with a lower rate than the variable depth scenario proposed here.

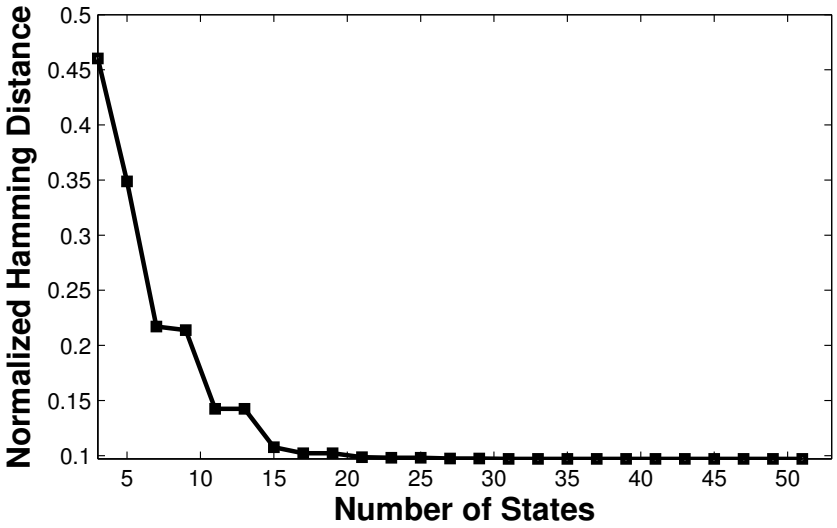


Figure 2.9: Normalized Hamming distance vs number of states (SOH = 1, 3 symbols, downsample time-lag = 7)

The accuracy of the model obtained by state-splitting could be analyzed by using it for making predictions on the output (i.e., voltage) symbol sequence based on the observations on the input (i.e., current) states. The xD-Markov model is first learned from 50% of the time-series of current and voltage with the state splitting algorithm to construct crossed PFSAs with different states (obtained as trees of different lengths). All the models are then used to predict the remaining output symbol sequence (i.e., from the remaining 50% voltage time-series) using the remaining input symbol sequence (i.e., from the remaining 50% current time-series).

Upon observation of an input state, a maximum likely prediction on the output symbol is made using the estimated morph matrix. The error in prediction is then measured as the normalized Hamming distance between the actual and the predicted test results for the voltage symbol sequence which is shown in Figure 2.9. The prediction error monotonically decreases and finally saturates (i.e., no further reduction with extra states) after the model with 21 states (after 9 splits) is reached, which was earlier inferred using cross entropy rate.

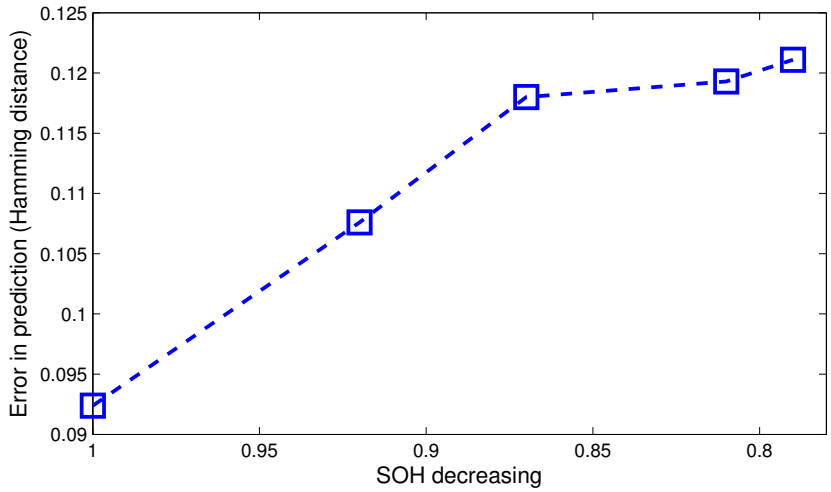


Figure 2.10: Error in prediction of output with decreasing health)

A change in battery health (i.e., SOH) should cause a departure in the input-output behavior from the maximum-likely behavior of the input-output model inferred at perfect health, i.e., for $SOH = 1$. To see this, the model learned at $SOH = 1$ is used to predict the maximum-likely voltage symbol sequence based on observations on current state-space for different health conditions. Figure 2.10 shows variations in the prediction error with deteriorating heath of the battery measured as normalized Hamming distance. The prediction error, based on the state space from the xD-Markov model at $SOH = 1$, increases monotonically as the battery health deteriorates (i.e., SOH drops). Hence, this prediction error can be used as a possible indicator of SOH degradation.

An anomaly measure for reflecting SOH degradation is formulated based on the xD-Markov model at the corresponding health condition of the battery. The xD-markov machine at $SOH = 1$ is considered the reference condition. The measure at an SOH level is defined as the difference in Frobenius norms of the estimated morph

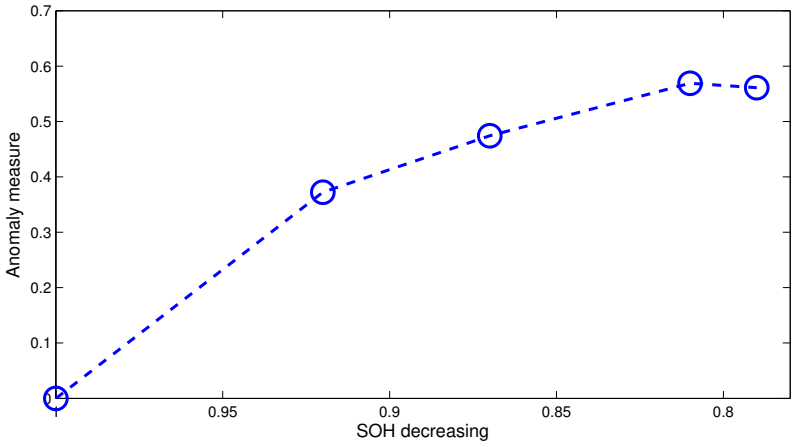


Figure 2.11: Anomaly measure vs state of health (SOH)

matrix at that SOH level and that of the estimated morph matrix at $SOH = 1$. The morph matrices are estimated by the approach that is shown in section 3. Based on the analysis, presented in Figure 2.8, the stopping criteria for state splitting is chosen to be 21. Figure 2.11 presents the variation of the proposed anomaly measure with respect to decreasing SOH. The proposed measure monotonically increases with decreasing SOH. It is noted that the model used for estimating anomaly was inferred without tying the objective function with any performance measure (like anomaly detection or class separability). It is possible to supervise the model inference depending on the tasks like anomaly detection, classification etc. and get a better behavior than that shown in Figure 2.11.

Finally, a regression-based filter is designed to estimate the SOC for the battery. During a training phase, the expected change in the SOC is learned using a k-Nearest Neighbor-based (kNN) [23] regression using the input-output data. For feature extraction, the input-output morph matrices are first estimated using time-series data for 20 minutes in a sliding window fashion. For each calculation, a new "Hotel-Pulse" (see section 2.8.4. Pulse duration is 2 minutes) is added to the window while the last pulse is pushed out. It is noted that the predictions are made every two minutes (which is the original frequency of SOC measurement during experiments). 50% of the data is used for training while the remaining half is used for test. For computational efficiency and limited data (the probabilities need significant amount of data for convergence), the state-splitting for xD-Markov machine construction is terminated after 7 states with $|\mathcal{A}| = 3$ (i.e., the morph matrix having 21 elements). To further reduce the dimensionality of the feature space, the top 8 (out of the 21,

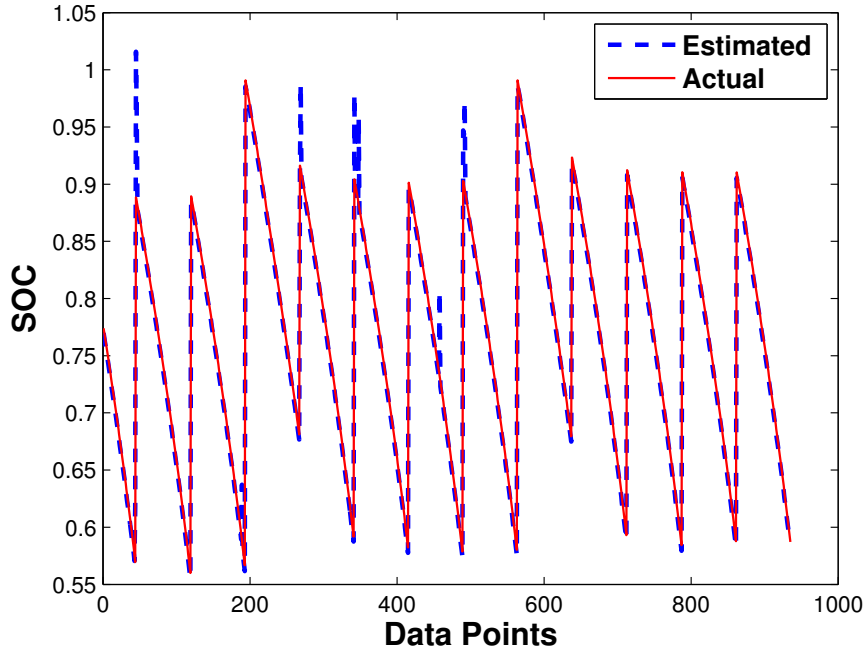


Figure 2.12: SOC estimation using the regression-based filter. SOH= 1 $\sim$ 40 %) features are selected by using Principal Component Analysis (PCA) [23]. These features are then used to learn the kNN-based regression with $k = 4$. During test, the regression is used to make an estimate on the expected change in the battery SOC. With the previous SOC known, this is used to predict the current SOC for the battery. The estimation results are presented in Figure 2.12 that shows a near-perfect estimation of SOC using the proposed filter. The average absolute error in the prediction of SOC is $\sim 0.6\%$.

2.9 Extensions to Multi-Sensor Fusion

There are various extensions of the present framework of Markov modeling for information fusion from local group of sensors in a sensor network. In the following we discuss, one possible extension for target detection in an environment with a dynamic background environment for detection of mobile targets. A brief description of problem statement and proposed approach is presented next. The material in this section is based on the work presented earlier in [83, 84].

2.9.1 Problem Statement for Local Target Detection in Sensor Networks

The presence of a target is determined by analyzing the ensemble of time series data generated from a number of sensors on a network. To this end, a set of sensors is deployed in a local region of the sensor network. The observation ensemble $\mathbb{Y} = \{Y_1, Y_2, \ldots, Y_n\}$ represents the set of time-series data generated by the ensemble of sensors $\mathcal{L} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n\}$.

Target detection in this section is based on a binary hypothesis test of H_0 versus H_1 that represent the hypotheses corresponding to "environmental disturbances only" (i.e., absence of a target) and "presence of a target," respectively. That is, H_0 is referred to as the *null hypothesis* and H_1 as the *alternate hypothesis*. In this setting, the problem of target detection in a dynamic environment is to find a decision rule for H_0 versus H_1 by a synergistic combination of the data collected by the sensor set $\mathcal{L}$ under the following assumptions.

1. The sensors are homogeneous, i.e., all sensors are of the same modality.

2. Even though the sensors have the same modality, their deployment is heterogeneous in the sense that they are not co-located and that their orientations are different; therefore, different sensors may yield different outputs in the presence of a single target.

3. Within a given time span of observation, there is at most one (moving) target. In other words, the observed time series outputs of the sensors correspond to the same event (i.e., the same target if it is present).

4. The time scale of environmental changes is largely similar to that of event occurrence (e.g., target motion). In other words, the environment is dynamically changing during the observation of a local event.

The assumptions 1 to 4, stated above, are often encountered in distributed sensing applications. The usage of homogeneous sensors is cost-effective and placing them non-co-located with different orientations provides heterogeneity that allows a variety of views of the target and environment. Furthermore, the sensors are usually looking for hard-to-find targets, which generally means only one will appear

in a field-of-view within a short time span, or else the field of view would be too large to detect a faint target. Finally, the time scale of encountering hard-to-find targets could be similar to that of environmental changes which is dynamically varying. Thus, the above assumptions are consequences of phenomena that may naturally occur at a node in a sensor network node in practical applications.

2.9.2 Proposed Technical Approach

For collaborative decision-making, it is necessary to construct a framework for information fusion at appropriate levels of granularity or details. From this perspective, the decisions are mostly characterized in terms of information levels at which the fusion takes place. Based on the level of information details, sensor fusion models are classified as the well known triplet: data-level, feature-level and decision-level [85].

Even though data-level fusion entails maximum details, communication of raw data (i.e., symbols in the proposed approach) may require tight clock synchronization in individual network nodes and high communication overhead, which are expensive to implement. On the other hand, decision-level fusion requires relatively less communication overhead at the expense of loss of vital information, which may lead to incorrect interpretations in a dynamic operating environment. Therefore, the focus of this section is to formulate an algorithm for feature-level information fusion for target detection.

Each time series Y_i of the observation ensemble $\mathbb{Y}$ is first individually processed (at the sensor site) and then compressed into a low-dimensional feature), as explained in Section 2.1. The symbolic dynamics-based feature extraction method is used to create generative models of the sensed data and also to serve as compact representations of the events that generate the data. Hence, each time-series Y_i for the sensor $\mathcal{L}_i$ is now statistically represented by a unique PFSA.

The set of PFSA, corresponding to the set of sensors $\mathcal{L}$, is denoted as $\mathbb{G} \equiv \{G_1, G_2, \ldots, G_n\}$. The cardinality of the set of states in each PFSA G_i in $\mathbb{G}$ is constrained to be identical (so that they can be compared in the same space). This is further facilitated by having an identical alphabet size $|\Sigma|$ for all PFSA $G_i \in \mathbb{G}$.

Figure 2.13 presents a qualitative illustration of the feature evolution in a dynamic environment for a small sensor network. Sensors with local environment-

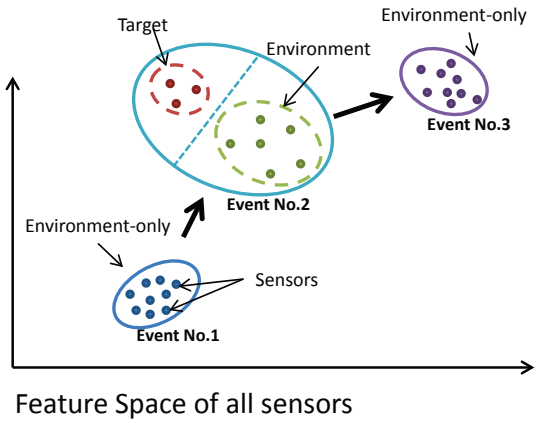


Figure 2.13: Feature-space illustration in a dynamic background environment

only information (i.e., in the absence of a target) should have similar features at any particular instant of time; however, this information at a given location might be different at two different time instants as well as at different locations at the same time instant. Hence, when compared in the feature space, the sensors with environment-only information should be lumped together as a single cluster. If a target is present, response of sensors and thus, the features extracted from the same sensors with target information should be significantly different from the features of sensors with environment-only information. From this persp-ctive, two definitions are introduced to quantify the divergence between the extracted features and clusters.

Definition 2.9.1 *(Feature Divergence) Let* $\widehat{\mathbf{\Pi}}_i$ *be the feature extracted for the PFSA* G_i *corresponding to the sensor* $\mathcal{L}_i$*, and let* $\widehat{\mathbf{\Pi}}_j$ *be the feature extracted for the PFSA* G_j *corresponding to the sensor* $\mathcal{L}_j$*. Then, the (scalar) feature divergence between these two sensors is expressed as:*

$$m(\mathcal{L}_i, \mathcal{L}_j) \triangleq d\big(\widehat{\mathbf{\Pi}}_i, \widehat{\mathbf{\Pi}}_j\big) \tag{2.28}$$

where d *is a suitable metric and popular choices of a metric (e.g., [26] and [86]) are*

- *Euclidean distance,* $d_2(a,b) \triangleq (a-b)(a-b)'$
- *City block distance,* $d_1(a,b) \triangleq \sum_{j=1}^{n} |a_j - b_j|$
- *Cosine distance,* $d_c(a,b) \triangleq 1 - \frac{ab'}{\sqrt{(aa')(bb')}}$

where a, b are two row vectors having the same dimension and the column vector a' is the transpose of a.

Definition 2.9.2 *(Proximity Matrix) Let C_r and C_s be two nonempty clusters, i.e., each of them contains at least one sensor. Let $\mathcal{L}_{ri}$ be the i^{th} sensor in C_r. Let n_r be the number of sensors in C_r. Then, the proximity matrix $A^{\Delta} = [a_{rs}^{\Delta}]$ between any two clusters is obtained elementwise as*

$$a_{rs}^{\Delta} \triangleq \Delta(C_r, C_s) \tag{2.29}$$

where the cluster distance is expressed as

$$\Delta(C_r, C_s) \triangleq \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} m(\mathcal{L}_{ri}, \mathcal{L}_{sj}) \tag{2.30}$$

In the following, we present details of three different clustering methods, namely, k-means, agglomerative hierarchical, and spectral, that have been used to make collaborative target detection. Each of these three clustering methods has been used to analyze the topological similarity of the PFSA set $\mathbb{G}$ corresponding to the ensemble of observation $\mathbb{Y}$. In all these three methods, we assess the structure of the set $\mathbb{G}$ under the two possible hypotheses, H_0 and H_1. A decision rule is reached by finding the discriminative features in the PFSA set $\mathbb{G}$ for maximum separability of the null and the alternate hypothesis. The idea is to use the gap statistics [87] of the clusters to learn whether the attributes of the set $\mathbb{G}$ belong to either the environment-only or the target-present scenario.

2.9.2.1 k-means Clustering

The k-means clustering [88] [89] [90] is used to group the elements of $\mathbb{G}$ into $k = 2$ mutually exclusive clusters. After initiating the positions of cluster centroids, each G_i is assigned to the closest cluster according to a pre-defined distance measure (e.g., cosine similarity). The cluster centroids are then updated by minimizing the sum of the distance measures between every G_i and its corresponding cluster centroid. The cluster assignment and distance minimization processes are iteratively repeated until the algorithm converges and there is no further change in assignment of the elements of $\mathbb{G}$ to the clusters. The positions of cluster centroids are initialized by

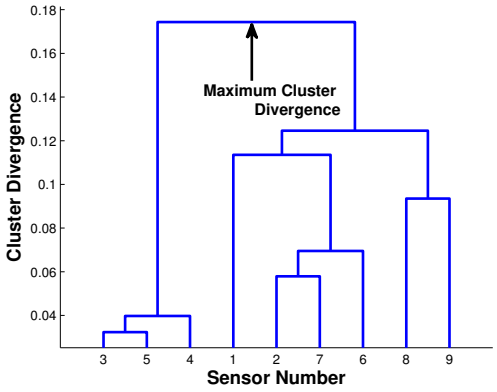
randomizing from the set $\mathbb{G}$. To avoid local minimum due to a bad initialization, 10 replicates with individual random initial starts have been used in parallel to ensure that the algorithm returns the lowest sum of distance measures. The outputs of k-means clustering are locations of centroids of the two clusters and element indices of each cluster. The gap (i.e., cluster centroid distance) statistics can be used to design a threshold for declaration of target.

2.9.2.2 Agglomerative Hierarchical Clustering

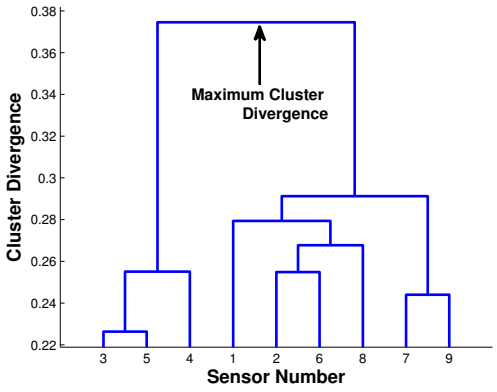
Agglomerative hierarchical clustering [89] is a bottom-up method that generates a sparse network (e.g., a binary tree) of the PGSA set $\mathbb{G}$ by successive addition of edges between the elements of $\mathbb{G}$. Initially, each of the PFSA $G_1, G_2, \ldots, G_n$ is in its own cluster $C_1, C_2, \ldots, C_n$, where $C_i \in \mathcal{C}$, which is the set of all clusters for the hierarchical cluster tree. In terms of the distance measured by Eqs. (2.28) through (2.30), the the pair of clusters that are nearest to each other are merged and this step is repeated till only one cluster is left. Figure 2.14 presents two examples of binary trees obtained from environment-only and target-present scenarios as an example to make the visualization easier (details of the corresponding experiments are presented in the next section). In both examples, the sensor number 3, 4, and 5 are grouped together as a single cluster (i.e., they are subjected to high ambient illumination level compared to others). However, the distance between the cluster, consisting of sensor 3, 4, and 5, and others is significantly larger than that of the environment-only case. This tree structure displays the order of splits in the network, which can be used to find the expected behavior of the network for environment-only and target-present scenarios.

2.9.2.3 Spectral Clustering

Spectral clustering, in general, performs a dimensionality reduction by using spectral properties (i.e., eigenvalues and eigenvectors) of the similarity matrix generated from the feature set (extracted from the sensed data). The PFSA set $\mathbb{G}$ is modeled as a fully connected weighted graph, i.e., every G_i is a node of the graph and the weight for a edge between i and j is calculated from the attributes (or features) of the PFSAs G_i and G_j. In particular, the graph properties are analyzed using two commonly used similarity functions: cosine similarity function $s^c(\cdot, \cdot)$ and the



(a) Binary tree of environment-only scenario



(b) Binary tree of target-present scenario

Figure 2.14: Examples of binary tree structure formed by multiple sensors Gaussian kernel similarity function $s^g(\cdot,\cdot)$ both expressed below in Eq. (2.31).

$$s^c(a_k, a_\ell) \triangleq \frac{a_k a_\ell'}{\|a_k\|\|a_\ell\|} \quad \text{and} \quad s^g(a_k, a_\ell) \triangleq e^{-\frac{\|a_k - a_\ell\|^2}{2\sigma^2}} \tag{2.31}$$

where the $(1 \times |Q||\Sigma|)$ vectors a_k and a_ℓ are vectorized forms of the matrices $\widehat{\mathbf{\Pi}}_k$ and $\widehat{\mathbf{\Pi}}_\ell$, respectively; the column vector a_ℓ' is the transpose of a_ℓ; the Euclidean norm of the vector a_k is denoted as $\|a_k\|$; and σ is a non-zero real parameter.

Once an $n \times n$ similarity matrix A (i.e., $A^c \triangleq [s^c(a_k, a_\ell)]$ or $A^g \triangleq [s^g(a_k, a_\ell)]$) is constructed, which is symmetric-positive-definite, it is normalized to obtain the graph Laplacian as

$$L = I - M^{-1/2} A M^{-1/2} \tag{2.32}$$

where M is a positive-definite diagonal matrix and is computed as

$$M_{ii} = \sum_{j} A_{ij} \tag{2.33}$$

By the Rayleigh-Ritz theorem [91], the clustering results are obtained in terms of the $|Q|$-dimensional eigenvector f corresponding to the second smallest eigenvalue of the (positive semi-definite) Laplacian matrix L. The graph partition in clusters is constructed by using the sign of the elements of the eigenvector f as an indicator function, i.e.,

$$\begin{cases} \mathcal{L}_i \in \mathcal{C}_1, & \text{if } f_i \geq 0 \\ \mathcal{L}_i \in \mathcal{C}_0, & \text{if } f_i < 0 \end{cases} \tag{2.34}$$

where C_1, C_0 represent the two clusters in the set $\mathbb{G}$.

2.9.2.4 Hypothesis Testing with Feature-level Sensor Fusion

The two most dissimilar clusters in the PFSA set $\mathbb{G}$ are obtained for each clustering algorithm; for example, see the two clusters at the top of the hierarchical trees in Figure 2.14. A decision on the presence of target is made based on the divergence (or gap) between these two clusters. In the presence of a target, the PFSA set $\mathbb{G}$ should be more heterogeneously partitioned into two clusters, one carrying largely the information on the environment-only scenario and the other carrying largely the information on the target-present scenario. It is noted that sensors with environment-only information are always present because the presence of a target is spatially sparse. For the environment-only scenario, even the two maximally separated clusters (e.g., the ones with maximum cluster divergence in the hierarchical cluster tree) should be close to each other in comparison to the target-present scenario. Hence, the maximum cluster divergence for the cluster set $\mathbf{C}_1$ would be higher as compared to that for the set $\mathbf{C}_0$, where the subscripts 0 and 1 represent the null hypothesis H_0 of environment-only (i.e., no target) scenario and the alternate hypothesis H_1 of target-present scenario, respectively. It is concluded from these observations that the maximum cluster divergence, $\Delta(\cdot\,,\cdot\,)$, which is achieved for the top two clusters in the hierarchical clustering tree, should be strongly influenced by the presence or absence of a target. Spatial sparsity of the

event implies that the cluster divergence should be larger in the presence of a target, i.e.,

$$\max_{C_r,\ C_s \in \mathbf{C}_1} \Delta(C_r, C_s) > \max_{C_r,\ C_s \in \mathbf{C}_0} \Delta(C_r, C_s) \tag{2.35}$$

A decision for target presence could be made with appropriate boundaries to distinguish the cluster gap statistics in the absence and presence of targets. To this end, the distribution of cluster divergence $\Delta(\cdot\,,\cdot\,)$ (e.g., divergence between the two clusters as seen at the top of the hierarchical tree in Figure 2.14) for each hypothesis could be obtained during a training process. Then, the target detection problem is formulated as a binary hypothesis test in terms of the hypothesis pair as

$$\begin{cases} H_0 : X \sim \mathcal{P}_0 \\ H_1 : X \sim \mathcal{P}_1 \end{cases} \tag{2.36}$$

where X is the range space of the random variable representing the divergence between the two clusters in the PFSA set $\mathbb{G}$; and the probability measures $\mathcal{P}_0$ and $\mathcal{P}_1$ represent the cluster divergence under the null hypothesis H_0 (i.e., environment-only scenario) and the alternate hypotheses H_1 (i.e., target-present scenario), respectively. Decision boundaries can then be obtained by choosing a threshold η on the likelihood ratio Λ, which is obtained for Eq. (2.36) as follows.

$$\Lambda(x) = \frac{d\mathcal{P}_1(x)}{d\mathcal{P}_0(x)} \ \text{ for } x \in X \tag{2.37}$$

where it is assumed that the probability measure $\mathcal{P}_1$ is absolutely continuous [92] with respect to the probability measure $\mathcal{P}_0$, i.e., for any any event E belonging to the σ-algebra of X, if $\mathcal{P}_0(E) = 0$, then it is implied that $\mathcal{P}_1(E) = 0$. Under this condition, the likelihood ratio Λ in Eq. (2.37) is the Radon-Nikodym derivative of $\mathcal{P}_1$ with respect to $\mathcal{P}_0$ and is denoted as $\frac{d\mathcal{P}_1}{d\mathcal{P}_0}$.

Sensor networks are designed to satisfy certain performance requirements (e.g., not exceeding a maximum allowable false alarm probability). Under such constraints, the Neyman-Pearson hypothesis testing procedure [92] maximizes the probability of correct detection P_D while the false alarm probabilities P_F are not allowed to exceed a specified bound α, where $\alpha \in (0, 1)$ is called the significance level. From this perspective, an appropriate choice of the decision rule δ is made to maximize

the detection probability as

$$\max_{\delta} P_D(\delta) \text{ subject to } P_F(\delta) \leq \alpha \tag{2.38}$$

A threshold η for rejecting the null hypothesis H_0 in the favor of the alternate hypothesis H_1 could be idenfied by using the following equation.

$$\mathcal{P}_0\big(\{x \in X : \ \Lambda(x) > \eta\}\big) = \alpha \tag{2.39}$$

It follows from Eq. (2.38), Eq. (2.39) and Neyman-Pearson lemma [92] that there exists a unique optimal decision rule $\delta^\star$ with an associated threshold parameter $\eta > 0$, which is dependent on the significance level α, such that

$$\delta^\star(x) = \begin{cases} \text{Select } H_1 & \text{if } \Lambda(x) > \eta(\alpha) \\ \text{Select } H_0 & \text{if } \Lambda(x) \leq \eta(\alpha) \end{cases} \quad \text{for almost all } x \in X \tag{2.40}$$

2.10 Future Research Directions

One possible direction for research is to find a discretization technique which results in order one discrete Markov process. Proving such a discretization always exists might be possible under stationarity conditions for systems with fading memory. For example, the metric introduced in equation (2.17) provides a measure to describe the discrepancy between the independent statistics of the symbols versus the statistics when conditioned on the states of a probable Markov chain. Maximizing such a discrepancy provides a way to maximize the information gain obtained by 1^{st}-order Markov model for the underlying data. However, further investigation is required to answer various related questions in this regard. We state the problem more formally next.

Problem 1. Let the time-series data be denoted as the sequence $\{X_t\}_{t\in\mathbb{N}}$ where $X_t \in \Omega \subseteq \mathbb{R}^d$ and let φ represent the partitioning function such that $\varphi : X_t \mapsto a_t$ where $a_t \in \mathcal{A}$ for all $t \in \mathbb{N}$ and $\mid \mathcal{A} \mid \in \mathbb{N}$ is known and fixed. Then the partitioning function is completely determined by the set $\mathcal{R}_\varphi = \{R_0, R_1, \ldots, R_{|\mathcal{A}|}\}$ such that $\Omega = \overline{R_0} \cup \overline{R_1} \cup \cdots \cup \overline{R_{|\mathcal{A}|}}$. Let us assume that the possible family of sets lies in a set denoted by Ω_φ. An information gain for the Markov model could be measured

by the following equation.

$$d_{\mathcal{R}_\varphi} = \sum_{q \in Q} \Pr(q) D_{KL}(P(\mathcal{A} \mid q) \| \tilde{P}(\mathcal{A})) \tag{2.41}$$

where the measure is parameterized by the set $\mathcal{R}_\varphi$ which depends on the partitioning function φ and the set $\mathcal{Q}$ represents the finite memory-words of the discrete symbol sequence. The above equation measures the information gain by creating a Markov model, where we measure the symbol emission probabilities conditioned on the memory words (or states) of the Markov model, over the independent or marginal symbol emission probabilities. Another interpretation is that equation (2.41) describes the discrepancy between the statistics of $\{s_t\}_{t \in \mathbb{N}}$ when modeled as independent sequence versus when modeled as a stationary Markov process. Then, a partitioning to optimize this measure may capture the true temporal behavior of $\{X_t\}_{t \in \mathbb{N}}$. The problem is to obtain the parameters of the related optimization problem.

$$\mathcal{R}_\varphi^\star = \arg \max_{\mathcal{R}_\varphi \in \Omega_\varphi} \sum_{q \in Q} \Pr(q) D_{KL}(P(\mathcal{A} \mid q) \| \tilde{P}(\mathcal{A}))$$

The following questions need to be answered to characterize $\mathcal{R}_\varphi^\star$.

- Is $\mathcal{R}_\varphi^\star$ unique? Under what conditions of the underlying process $\{X\}_{t \in \mathbb{N}}$, can we get an unique solution?
- What is the order of the corresponding discrete time-series obtained for $\mathcal{R}_\varphi^\star$?
- Let us assume that the pre-image of a symbol is represented by the centroid of the set in $\mathcal{R}_\varphi^\star$ in the original phase-space. Then, how can we characterize the metric $\zeta_\varphi = \sum_{t \in \mathbb{N}} d(\varphi^{-1}(s_t), X_t)$ for signal representation? Is $\mathcal{R}_\varphi^\star$ able to minimize this metric over Ω_φ?
- Now imagine that the size of partitioning set is allowed to vary. Then, how to assign a MDL score to the individual models for different sizes of the partitioning set and how do we select a final model for signal representation?

The above problem would, thus, try to formulate and characterize the properties of a partition for a data set for Markov representations. In the next question, we will try to study the composite problem of signal representation by partitioning

followed by order estimation.

Problem 2. Let the time-series data be denoted as the sequence $\{X_t\}_{t\in\mathbb{N}}$ where $X_t \in \Omega \subseteq \mathbb{R}^d$ and let φ represent the partitioning function such that $\varphi : X_t \mapsto a_t$ where $a_t \in \mathcal{A}$ for all $t \in \mathbb{N}$ and $\mid \mathcal{A} \mid \in \mathbb{N}$ is unknown. A desirable way to characterize a partitioning process is by predicting its effect on the size of temporal memory of the system. Is it possible to synthesize a partitioning function φ_I which preserves the memory of the discrete system under the transformation φ_I, i.e., if $\Pr(X_t \mid X_{t-D}, \dots, X_{t-1}) = \Pr(X_t \mid X_{t-1}, \dots)$ then we have $\Pr(\varphi_I(X_t) \mid \varphi_I(X_{t-D}), \dots, \varphi_I(X_{t-1})) = \Pr(\varphi_I(X_t) \mid \varphi_I(X_{t-1}), \dots)$. Then, how can we characterize a system for which such a discretization is guaranteed to exist? It is noted that the size of partitioning set is unknown and not fixed.

These two problems could be treated as fundamental problems that need to be studied for mathematical characterization of data-driven modeling of systems from a symbolic analysis perspective. They are mainly concerned about inference of model structure for statistical learning. There are some problems which are of interest from applications perspective and related to estimation of various parameters during modeling and inference (e.g., truncated sequential probability ratio-tests for the presented models). However, they are not being presented here.

Chapter 3

Reduced-Order Markov Modeling of Time- Series Data

In the last chapter, we reviewed some essential concepts and presented some state-of-the-art techniques for order estimation (a free-parameter in the discussed approach) and parameter estimation for Markov models. In this chapter, we discuss the problem of compact model selection for representation and learning of time-series data under the umbrella of symbolic time-series analysis. This chapter presents a technique for reduced-order Markov modeling for compact representation of time-series data. Instead of hidden Markov model (HMM)-inspired techniques, symbolic dynamics-based tools have been used to infer an approximate generative Markov model. The time series data is first symbolized by partitioning the continuous measurement space of the signal and then, the discrete sequential data is modeled using symbolic dynamics. In the proposed approach, the size of temporal memory of the symbol sequence is estimated from spectral properties of the resulting stochastic matrix corresponding to a first-order Markov model of the symbol sequence. Then, hierarchical clustering is used to cluster the states of the corresponding full-state Markov model to construct a reduced-order (or size) Markov model with a non-deterministic algebraic structure. Subsequently, the parameters of the reduced-order Markov model are identified from the original model by making use of a Bayesian inference rule. The final model is selected using information theoretic criteria. The proposed concepts are illustrated and validated using two different datasets. We use the proposed method to analyze pressure data obtained from a swirl stabilized

combustor where some controlled protocols are used to induce flame instabilities. Variations in the complexity of the derived Markov model represent how the system operating condition changes from a stable to an unstable regime. The other data set is taken from NASA's data repository for prognostics of bearings on rotating shafts. We show that even with a very small state-space, the reduced models are able to achieve comparable performance and that the proposed approach provides flexibility in the selection of final model for representation and learning.

3.1 Motivation and Introduction

Hidden Markov model (HMM) is a widely used statistical learning tool for modeling uncertain dynamical systems [23], where the associated temporal data are used to infer a Markov chain with unobserved states. In this setting, the learning task is to infer the states and the corresponding parameters of the Markov chain. In addition to HMM, several other nonlinear techniques have been proposed for Markov modeling of time-series data. Symbolic time-series analysis-based Markov modeling is a recently proposed technique [26] where the states of a Markov chain are represented as a collection of words (i.e., symbol blocks, also referred to as memory words) of different lengths, which can be identified from the time-series data on a discrete space with finite cardinality [26, 27, 93, 94]. The symbols are created from the continuously varying time-series data by projecting the data to a set with finite cardinality. A common ground among all these tools of Markov modeling as discrete sequences, is that the Markov chain is induced by probabilistic representation of a deterministic finite state auotmaton (DFSA), often called probabilistic finite state automata (PFSA) [95]. While the PFSA-based inference provides a consistent, deterministic graph structure for learning, the deterministic algebraic structure is generally not a very compact representation and may often lead to large number of states in the induced Markov model. To circumvent this problem attempts have been made to reduce the state-space by merging statistically similar states of the model [27]. The problem is, however, that as these models are constructed by partitioning of phase space of the dynamical system, merging states that are statistically similar leads to algebraic inconsistency. On the other hand, if the states are merged to preserve the algebraic consistency, it leads to statistical impurity in the final models (i.e., states which have different statistics could be merged

together). Other approaches for state aggregation in Markov chains could be found in [96–98]. However, these papers do not consider inference of the Markov model from the data which may not be suitable for analysis of data-driven systems.

The state space for Markov models, created by using symbolic analysis, increases exponentially with increase in memory or order of the symbolic sequence. Estimating the right memory is critical for temporal modeling of patterns observed in the sequential data. However, some of the states may be statistically similar and thus merging them can reduce the size of state-space. This chapter presents reduced-order Markov modeling of time-series data to capture temporal patterns, where we estimate the size of temporal memory of the symbolic data using the spectral properties of a PFSA whose states are words of length one [9, 58]. The constraint of deterministic algebraic structure is not imposed by the end objective, but due to the choice of the data representation model. Thus we propose to merge the states and remove the constraint of deterministic algebraic properties associated with PFSA, where the states of the Markov chain are now collection of words from its alphabet of length estimated in the last step. This state aggregation induces a non-determinism in the finite state model. The parameters of the reduced-order Markov model are estimated by a Bayesian inference technique from the parameters associated with the higher-order Markov model. The final model for data representation is selected using information-theoretic criteria, and thus, we get a unique stopping point to terminate the state-merging procedure. We also present a bound on the distortion of the predictive capability of the models up on reduction in the size of the state-space. The final model obtained is a generative model for the data; however, some predictive capability is lost as we remove the deterministic algebraic structure of a DFSA.

The proposed technique of state merging is inspired by time-critical applications where it is imperative to arrive at a reliable decision quickly as the dynamics of the process being monitored is really fast. In such applications, there are strict constraints on accuracy as well as the time needed to come to a decision. In this chapter, we illustrate the concepts using two different datasets. We discuss in detail the example of combustion instability which is a highly nonlinear and complex phenomena and results in severe structural degradation in jet turbine engines. Some good surveys on the current understanding of the mechanisms for the combustion instability phenomena could be found in [99–103]. Active combustion instability

control (ACIC) with fuel modulation has proven to be an effective approach for reducing pressure oscillations in combustors [104, 105]. Based on the work available in literature, one can conclude that the performance of ACIC is primarily limited by the large delay in the feedback loop and the limited actuator bandwidth [104, 105]. Early detection of combustion instability can potentially alleviate the problems with delay in the ACIC feedback loop and thus possibly improve the performance. Some recent work for detection and prediction of combustion instabilities could be found in [68, 106–109]. While the results in these papers are encouraging, there is no interpretation of the expected changes in the data-driven model that could be observed during changes in the operating regime of the underlying process. In contrast to the work reported in literature, we have presented an overall idea of changes in the underlying stochastic model structure and parameters during the complex instability phenomenon.

Contributions. This chapter presents a technique for Markov modeling of time series data using a PFSA with nondeterministic algebraic structure. Nondeterminism is induced by merging states of a PFSA with deterministic algebraic structure inferred from discrete sequential data, which in turn allows very compact representation of temporal data. In contrast to the approach in [27], we present a method to use information-theoretic criteria to arrive at a consistent stopping criterion for model selection. The resulting reduced-order model has fewer parameters to estimate; this is turn leads to faster convergence rates and thus faster decisions during test (or operation). We also present a bound on the distortion in the predictive capability of the models due to state-space reduction using Hamming distance between the sequences generated by the original and final model. The algorithms presented in the chapter are validated on two different datasets– pressure data obtained from a swirl-stabilized combustor to monitor thermo-acoustic instability and a public data set for bearing prognostics. We show changes in the complexity of the pressure data as the process moves from stable to unstable through the transient phase which is then used to arrive at a criterion that provides perfect class separability. Apart from the results on Markov modeling, the results on combustion instability could be of independent interest in combustion community.

3.2 Background and Mathematical Preliminaries

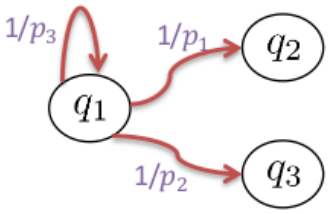
AS explained in the last chapter, symbolic analysis of time-series data is a recent approach where continuous sensor data are converted to symbol sequences via partitioning of the continuous domain [26, 33]. In this chapter, we introduce the non-deterministic PFSA which have a non-deterministic algebraic structure. A non-deterministic PFSA is defined next.

Definition 3.2.1 (PFSA) *A non-deterministic Probabilistic Finite State Automata (PFSA) is a tuple* $G = (\mathcal{Q}, \mathcal{A}, \delta, \boldsymbol{M})$ *where*

- $\mathcal{Q}$ *is a finite set of states of the automata;*
- $\mathcal{A}$ *is a finite alphabet set of symbols* $a \in \mathcal{A}$*;*
- $\delta : \mathcal{Q} \times \mathcal{A} \to 2^{\mathcal{Q}}$ *is the state transition function;*
- $\boldsymbol{M} : \mathcal{Q} \times \mathcal{A} \to [0, 1]$ *is the* $|\mathcal{Q}| \times |\mathcal{A}|$ *emission matrix. The matrix* $\boldsymbol{M} = [m_{ij}]$ *is row stochastic such that* m_{ij} *is the probability of generating symbol* a_j *from state* q_i*.*

Remark 3.2.1 *The probabilistic finite state automaton (PFSA) defined in chapter 2 had a deterministic algebraic structure where a symbol emission from a particular state will lead to a fixed state. The symbol emissions are however, probabilistic. On the other hand, the transition function for a non-deterministic finite state automaton is given by a map,* $\delta : \mathcal{Q} \times \mathcal{A} \to 2^{\mathcal{Q}}$ *which leads to non-determinism. The idea is also presented in Figure 3.1 where we show that the same symbol can lead to multiple states, however in a probabilistic fashion. This allows more flexibility in modeling.*

It is noted that the non-determinism leads to loss of some information in the modeling of discrete time-series; however, for stationary time-sequences, most of the statistical information could be retained even after considerable reduction of states of the Markov model. This concept is elaborated and established in this chapter.



$$q_1 = \{11, 22, 33\} \text{ (for example)}$$
$$p_1 + p_2 + p_3 = 1$$

Figure 3.1: Graphical model showing non-determinism in a PFSA. The symbol 1 emitted from state q_1 leads to different states with fixed probabilities indicating non-deterministic behavior.

Next we very briefly explain the idea of two-stage Minimum description length (MDL) [110] which is used in this work for model selection. The MDL principle is a method for inductive inference which is based on the following insight: any regularity in the data could be used to compress the data, i.e., to describe it using fewer symbols than the number of data needed to describe it literally. Suppose that the class of candidate parametric models is denoted by $\mathfrak{M}$ indexed by a parameter $\theta \in \mathbb{R}^k$. The model class consists of the following candidate models for the data string $\vec{s}$.

$$\mathfrak{M} = \{g(\vec{s} \mid \theta) : \theta \in \Theta \subset \mathbb{R}^k\}$$

Then, under this principle we chose a member of this class and use it to encode a data string $\vec{s}$. This selection is made via an estimator $\hat{\theta}_n$, after which the prefix code is built from $g_{\hat{\theta}_n}$. Ultimately, the code length associated with this model takes the form of penalized likelihood, the penalty being the cost to encode the estimated parameter values $\hat{\theta}_n$. Simply speaking, the two-stage MDL consists of two terms: the first term represents the number of bits required to encode the data sequence $\vec{s}$ given an estimate $\hat{\theta}_n$, whereas the second term represents the number of bits required to encode the components of $\hat{\theta}_n$ [111].

3.3 Proposed Approach

In this section, we present the details of the proposed approach for inferring a Markov model from the time series data. As discussed earlier, the first step is the discretization of the time-series data to generate a discrete symbol sequence. While it is possible to optimize the symbolization of time-series using some optimization

criterion, we do not discuss such a technique here. The data is discretized using the unbiased principle of entropy maximization of the discrete sequence using maximum entropy partitioning (MEP) [26]. The proposed approach for Markov modeling then consists of the following four critical steps

- Estimate the approximate size of temporal memory (or order) of the symbol sequence.
- Cluster the states of the high-order Markov model.
- Estimate the parameters of the reduced-order Markov model (i.e., the transition matrix).
- Select the final model using information theoretic scores (described below, Section 3.3.3).

Memory of the discrete sequence is estimated using a recently introduced method based on the spectral analysis of the Markov model with depth 1. induced by a PFSA [9, 58]. It is noted that these steps are followed during training to estimate the approximate model for data and during test, the parameters are estimated for the reduced model. The key ideas behind these steps are explained next.

3.3.1 Estimation of Reduced-Order Markov Model

Depth D of a symbol sequence has been redefined in [58] as the number of time steps after which probability of current symbol is independent of any past symbol i.e.:

$$\Pr(s_k|s_{k-n}) = \Pr(s_k) \ \forall n > D \tag{3.1}$$

Note that dependence in the proposed definition (eq. 3.1) is evaluated on individual past symbols using $\Pr(s_k|s_{k-n})$ as opposed to the assessing dependence on words of length D using $\Pr(s_k|s_{k-1}, \ldots, s_{k-D})$. It is shown that if the observed process is *forward causal* then observing any additional intermediate symbols $s_{k-1}, \ldots, s_{k-n+1}$ cannot induce a dependence between s_k and s_{k-n} if it did not exist on individual level [58].

Let $\mathbf{\Pi} = [\pi_{ij}^{(1)}]$ be the one-step transition probability matrix of the PFSA G

constructed from this symbol sequence i.e.

$$\mathbf{\Pi} = \Pr(s_k|s_{k-1}) \tag{3.2}$$

Then using the distance of the transition matrix after steps from the stationary point, depth can be defined as a length D such that

$$|\text{trace}\,(\mathbf{\Pi}^n) - \text{trace}\,(\mathbf{\Pi}^\infty)| \leq \sum_{j=2}^{J} |\lambda_j|^n < \epsilon \;\forall n > D \tag{3.3}$$

J is number of non-zero eigenvalues of $\mathbf{\Pi}$. Thus, the depth D of the symbol sequence is estimated for a choice of ϵ by estimating the stochastic matrix for the one-step PFSA. Next, another pass of data is done to estimate the PFSA parameters whose states are words over $\mathcal{A}$ of length D, i.e., $\mathbf{\Pi} = \Pr(s_k|s_{k-1}, \ldots, s_{k-D})$. It is noted that this step is critical for modeling accuracy.

The states of the reduced-order Markov model are then estimated by partitioning the set of words over $\mathcal{A}$ of length D estimated in the last step. This is done by using an agglomerative hierarchical clustering approach. The advantage of using the hierarchical clustering approach is that it helps visualize the structure of the set of the original states using an appropriate metric. Agglomerative hierarchical clustering is a bottom-up clustering approach [112] that generates a sparse network (e.g., a binary tree) of the state set $\mathcal{Q}$ (where $|Q| = |\mathcal{A}|^D$) by successive addition of edges between the elements of $\mathcal{Q}$. Initially, each of the states $q_1, q_2, \ldots, q_n$ is in its own cluster $C_1, C_2, \ldots, C_n$ where $C_i \in \mathcal{C}$, which is the set of all clusters for the hierarchical cluster tree. The distance between any two states in $\mathcal{Q}$ is measured using the K-L distance between the symbol emission probabilities conditioned on them, i.e.,

$$\begin{aligned} d(q_i, q_j) = {} & D_{\text{KL}}(\Pr(\mathcal{A}|q_i)\|\Pr(\mathcal{A}|q_j)) \\ & + D_{\text{KL}}(\Pr(\mathcal{A}|q_j)\|\Pr(\mathcal{A}|q_i)) \end{aligned} \tag{3.4}$$

where the terms on the right have the following meaning.

$$D_{\text{KL}}(\Pr(\mathcal{A}|q_i)\|\Pr(\mathcal{A}|q_j))$$

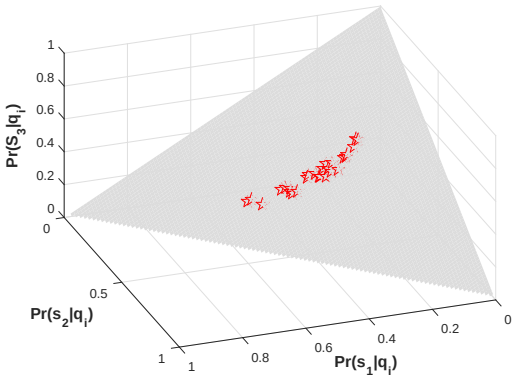


Figure 3.2: The symbol emission probabilities for a Markov chain with 3 symbols are shown on a simplex. Symmetric K-L distance is used to find the structure in the state-set in the information space and the states are clustered based on the revealed structure.

$$= \sum_{s \in \mathcal{A}} \Pr(s|q_i) \log \left(\frac{\Pr(s|q_i)}{\Pr(s|q_j)} \right)$$

In terms of the distance measured by eq. (3.4), the pair of clusters that are nearest to each other are merged and this step is repeated till only one cluster is left. The tree structure displays the order of splits in the state set of the higher-order Markov model and is used to aggregate the states close to each other. For clarification of presentation, we show an example of a Markov chain with 27 states and 3 symbols on a simplex plane in Figure 3.2, where each red pentagon on the simplex represents one row of the symbol emission matrix. The hierarchical clustering is used to find the structure of the state set on the simplex place using the K-L distance. The set of states clustered together could be obtained based on the number of final states required in the final Markov model.

The overall algorithm is presented as a pseudo-code in Algorithm 2. This algorithms is used to find the parameters of the models during training. The parameters during test are estimated using the clustering map $f_{N_{\max}}$ and is further discussed in next section. In the later sections we show how an information theoretic criterion could be used to select the appropriate model to terminate the state merging algorithm or select a final model from the set of reduced-order models.

Algorithm 2: Reduced Order Markov Modeling

Input: The observed symbol sequence $\vec{s} = \{\ldots s_1 s_2 s_3 \ldots | s_i \in \mathcal{A}\}$ and the desired number of states in final model $N_{\max} \leq \mid \mathcal{Q} \mid$

Output: The final Markov model, $\mathcal{M} = (\tilde{\mathcal{Q}}, \tilde{\boldsymbol{M}}, \tilde{\boldsymbol{\Pi}})$

1 Estimate the $\boldsymbol{\Pi}$ matrix for 1-step Markov model using frequency counting with an uniform prior;

2 Estimate the size of temporal memory, $D(\epsilon)$ for $\vec{s}$ using equation (3.3);

3 Estimate $\boldsymbol{M}$ and $\boldsymbol{\Pi}$ for the $D(\epsilon)$-Markov model using frequency counting with an uniform prior;

4 $\mathcal{C}_{|\mathcal{Q}|} = \{q_i \mid q_i \in \mathcal{Q}\}$;

5 **for** $i = \mid \mathcal{Q} \mid -1, \ldots, 1$ **do**

6 find distinct clusters $A, B \in \mathcal{C}_{i+1}$ minimizing $d(A \cup B)$;

7 $\mathcal{C}_i := (\mathcal{C}_{i+1} \setminus \{A, B\}) \cup \{A \cup B\}$

8 **return** $\mathcal{C}_1, \ldots, \mathcal{C}_{|\mathcal{Q}|}$ *and* $f_i : \mathcal{Q} \to \mathcal{C}_i \ \forall i \in \{1, \ldots, \mid \mathcal{Q} \mid\}$

9 Calculate the parameters of reduced model using $\tilde{\mathcal{Q}} = \mathcal{C}_{N_{\max}}$, $f_{N_{\max}}$ and equations (3.6) through (3.12);

3.3.2 Parameter Estimation of the Reduced-Order Markov Model

The parameters of the Markov model obtained after clustering the states of the original PFSA with $|\mathcal{A}|^D$ states is obtained using a Bayesian inference technique using the parameters estimated for the PFSA. In this proposed approach, the state transition matrix $\boldsymbol{\Pi}$, the emission matrix $\boldsymbol{M}$, and the state probability vector $\boldsymbol{p}$ of the original PFSA model G are available, along with the deterministic assignment map $f : \mathcal{Q} \to \widetilde{\mathcal{Q}}$ of the state in $\mathcal{Q}$ (i.e., state set of original model) to one of the state in $\widetilde{\mathcal{Q}}$ (i.e., state set of the reduced order model). Since the reduced order model can represented by the tuple $\widetilde{G} = (\widetilde{\mathcal{Q}}, \widetilde{\boldsymbol{\Pi}})$, where $\widetilde{\boldsymbol{\Pi}} = [\tilde{\pi}_{ij}]$ is the state transition matrix, we employ a Bayesian inference technique to infer the individual values of transition probabilities $\tilde{\pi}_{ij} = \Pr(\tilde{q}_{k+1} = j \mid \tilde{q}_k = i)$ for all $i, j \in \widetilde{\mathcal{Q}}$.

Let Q_k be the random variable denoting the state of PFSA model at some time step $k \in \mathbb{N}$ and S_k denotes the symbol emitted from that state, this probabilistic emission process is governed by the emission matrix $\boldsymbol{M}$. The state of the reduced order model is obtained from a deterministic mapping of the state of the PFSA model, thus the state of this model is also a random variable, which is denoted by $\widetilde{Q}_k = f(Q_k)$. The Bayesian network representing the dependencies between these variables is shown in the recursive as well as unrolled form in the Figure 3.3. The conditional density $\Pr(\widetilde{Q}_k = \tilde{q} \mid Q_k = q)$ can be evaluated by checking if state q

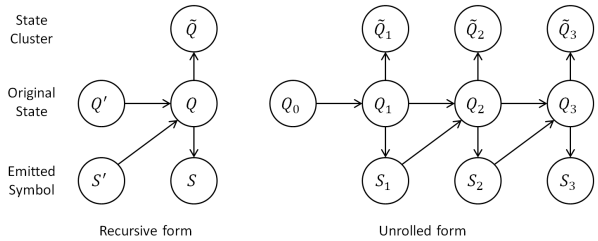


Figure 3.3: Graphical models representing the dependencies between the random variables

belongs to the state cluster $\tilde{q}$ and assigning the value of 1 if true, else assign it the value of 0. Since we know that $\widetilde{\mathcal{Q}}$ partitions the set $\mathcal{Q}$, the conditional density is well-defined. Thus, it can be written as

$$\Pr(\widetilde{Q}_k = \tilde{q} \mid Q_k = q) = \mathrm{I}_{\tilde{q}}(q), \tag{3.5}$$

where I is the indicator function with $\mathrm{I}_{\tilde{q}}(q) = 1$, if element q belongs to the set $\tilde{q}$, else it is 0. The derivation of the Markov model $\Pr(\widetilde{Q}_{k+1} \mid \widetilde{Q}_k)$ using $\Pr(Q_{k+1} \mid Q_k)$, stationary probability vector $\boldsymbol{p}$, and assignment map f is shown ahead.

$$\Pr(\widetilde{Q}_{k+1} \mid \widetilde{Q}_k) = \sum_{q \in \mathcal{Q}} \Pr(\widetilde{Q}_{k+1}, Q_{k+1} = q \mid \widetilde{Q}_k) \tag{3.6}$$

(Marginalization)

$$= \sum_{q \in \mathcal{Q}} \Pr(Q_{k+1} = q \mid \widetilde{Q}_k) \Pr(\widetilde{Q}_{k+1} \mid Q_{k+1} = q) \tag{3.7}$$

(Factorization using Figure 3.3)

$$= \sum_{q \in \mathcal{Q}} \Pr(Q_{k+1} = q \mid \widetilde{Q}_k)\, \mathrm{I}_{\widetilde{Q}_{k+1}}(q) \tag{3.8}$$

(using (3.5))

$$= \sum_{q \in \widetilde{Q}_{k+1}} \Pr(Q_{k+1} = q \mid \widetilde{Q}_k). \tag{3.9}$$

We can obtain $\Pr(Q_{k+1} \mid \widetilde{Q}_k)$ from Bayes' rule as

$$\Pr(Q_{k+1} \mid \widetilde{Q}_k) = \frac{\Pr(\widetilde{Q}_k \mid Q_{k+1}) \Pr(Q_{k+1})}{\sum_{q \in \mathcal{Q}} \Pr(\widetilde{Q}_k \mid Q_{k+1} = q) \Pr(Q_{k+1} = q)}. \tag{3.10}$$

Following the steps to obtain (3.9), we also derive

$$\Pr(\widetilde{Q}_k \mid Q_{k+1}) = \sum_{q \in \widetilde{Q}_k} \Pr(Q_k = q \mid Q_{k+1}). \tag{3.11}$$

We can obtain $\Pr(Q_k \mid Q_{k+1})$ from Bayes' rule as

$$\Pr(Q_k \mid Q_{k+1}) = \frac{\Pr(Q_{k+1} \mid Q_k)\Pr(Q_k)}{\sum_{q \in \mathcal{Q}} \Pr(Q_{k+1} \mid Q_k = q)\Pr(Q_k = q)}. \tag{3.12}$$

Note that, for the distribution $\Pr(Q_k)$ and $\Pr(Q_{k+1})$, we use the stationary probability $\boldsymbol{p}$. Using the equations (3.9), (3.10), (3.11), and (3.12) together, one can easily obtain the desired state transition matrix $\widetilde{\mathbf{\Pi}}$ of the reduced order model. Once the state cluster set $\widetilde{\mathcal{Q}}$ and state transition matrix $\widetilde{\mathbf{\Pi}}$ are available, the reduced order model is completely defined.

3.3.3 Model Selection using information theoretic criteria

In this section, we describe the model selection process during the underlying state merging process for model inference. We compute "penalized" likelihood estimates for different models. Then, the model with the lowest score is selection as the optimal model.

The (unpenalized) log-likelihood of a symbol sequence $\vec{s}$ given a Markov model G is computed as follows:

$$\mathcal{L}(\vec{s}|G) \cong \sum_{k=1}^{N} \log \Pr\left(s_k|q_k\right) \tag{3.13}$$

where the effects of the initial state are ignored because they become negligible for long statistically stationary symbol sequences. It is noted that with a finite symbol sequence, the log-likelihood is always finite. Furthermore, with the Markov models considered in this chapter, the sum is simplified to the following form.

$$\mathcal{L}(\vec{s}|G) \cong \sum_{k=D+1}^{N} \log \Pr\left(s_k|s_{k-1}, \ldots, s_{k-D}\right) \tag{3.14}$$

As discussed earlier, the states are merged using hierarchical clustering and

thus, for every desired number of final states we get the deterministic map $f_{N_{\max}}$ which determines how the original states are partitioned using the hierarchical clustering. This map is known for every terminal number of states and thus, we can find the log-likelihood of the symbol sequence using the following relationship.

$$\mathcal{L}(\vec{s}|\tilde{G}) \cong \sum_{k=D+1}^{N} \log \Pr\left(s_k|\tilde{q}_k = f_{N_{\max}}(q_k)\right) \tag{3.15}$$

where, $\tilde{q}_k$ is the state of the reduced model and q_k is the state of the original full-order model.

In the next step of the model selection process, a "complexity penalty" is added to the log-likelihood estimates, thereby balancing goodness of fit against the complexity of the model (and hence trying to prevent overfitting). We apply two widely-used such model selection functions, namely the Akaike information criterion (AIC) [113] and the Bayesian information criterion (BIC) [114]:

1. $\mathcal{M}_{\text{BIC}} = -2\mathcal{L}(\vec{s}|\tilde{G}) + K\log(N)$, where K is the number of free parameters and N is the number of observations.

2. $\mathcal{M}_{\text{AIC}} = -2\mathcal{L}(\vec{s}|\tilde{G}) + 2K$, where K is the number of free parameters.

The number of free parameters to be estimated from the data is the parameters of the symbol emission parameters, i.e., $K =\mid \mathcal{A} \mid\mid \tilde{\mathcal{Q}} \mid$. It is noted that this allows model selection for individual symbol sequences. The criterion here allows a terminal condition for state merging; however, different symbol sequences can have different models. The model with the minimum score is selected as the best model. Through the results presented in next sections we illustrate the fact that most of the temporal and predictive capabilities can be preserved for the models with a very small number of states when compared to the original model.

Remark 3.3.1 *The final Markov model is a finite depth approximation of the original time-series data. However, compared to the PFSA-based D-Markov machines in [26, 27], the current aggregated model has a non-deterministic algebraic structure, i.e., the same symbol emissions from a state can lead to different states. While this leads to some loss in predictive capability as compared to the models in [26, 27], this allows us to compress the size of the model as per the requirement at hand. This allows faster convergence rates for the symbol emission probabilities as we only*

require fewer parameters to estimate from data, which might lead to faster decisions during testing.

In the rest of the chapter, we will present a Hamming distance-based bound for distortion in the predictive capabilities of reduced models and demonstrate the utility of these models in practical problems of fault/anomaly detection from time-series data.

3.4 Analysis of the Proposed Algorithm

In this section, we will present a bound on the distortion of the model due to the reduction of state-space of the Markov model using Hamming distance between two symbol sequences. We first present the Pinsker's inequality [115] which relates the information divergence with the variational distance between probability measures defined on arbitrary spaces. This is followed by another theorem which can be used to derive Hamming distance bounds using the informational divergence.

Theorem 3.4.1 (Pinsker's inequality) *[115] Let P and Q be two probability distributions on a measurable space $(\mathbb{X}, \Sigma)$. Then, the following is true*

$$d_{TV}(P,Q) \leq \sqrt{\frac{1}{2} D_{KL}(P\|Q)} \tag{3.16}$$

where $d_{TV}(P,Q) = \sup_{A\in\Sigma}\{|P(A) - Q(A)|\}$ is the total variation distance.

Theorem 3.4.2 *[116] Let $\mathbb{X}$ be a countable set and let us denote by x^n the sequence $(x_1, x_2, \ldots, x_n) \in \mathbb{X}^n$. Let q^n be a Markov measure on $\mathbb{X}^n$, that is, $q(x^n) = q(x_1)\prod_{i=2}^{n} q_i(x_i|x_{i-1})$. Then for any probability measure p^n on $\mathbb{X}^n$, the following is true*

$$\bar{d}(p^n, q^n) \leq \left[\frac{1}{2n} D_{KL}(p^n\|q^n)\right]^{1/2} \tag{3.17}$$

where, $\bar{d}$ denotes the normed Hamming distance on $\mathbb{X}^n \times \mathbb{X}^n$:

$$\bar{d}(x^n, y^n) = n^{-1}\sum_{n=1}^{n} d(x_i, y_i), \tag{3.18}$$

where $d(x_i, y_i) = 1$ *if* $x_i \neq y_i$ *and* 0 *otherwise. The* $\bar{d}$*-distance between* p^n *and* q^n *is*

$$\bar{d}(p^n, q^n) = \min E\bar{d}(\hat{X}^n, X^n), \tag{3.19}$$

where min *is taken over all joint distributions with marginals* $p^n = \text{dist}\hat{X}^n$ *and* $q^n = \text{dist}X^n$ *and* E *denotes the expectation operator.*

The above theorem provides us a way to bound Hamming distance between sequences generated by two different distributions. Thus, using the above theorem, we find a bound on the Hamming distance between the symbol sequences generated by the reduced-order Markov model and the original model by estimating the K-L distance between the measure on symbol sequences induced by these models. An approximate estimate of the K-L distance between the original and a reduced model could be expressed and estimated as shown in the following.

Let the original D-Markov model be denoted by $\mathcal{M}$ and the reduced-order model by $\hat{\mathcal{M}}$. The Markov measure on the probability space $(S^n, \mathcal{E}, P)$ where the set S^n consists of sequences of length n from an alphabet $\mathcal{A}$ could be estimated using the symbol emission probabilities. More explicitly, the Markov measure of a sequence S_n on S^n induced by $\mathcal{M}$ is given by $P_{\mathcal{M}}(S_n) = \Pr(q_1) \prod_{i=D+1}^{n} \Pr(s_i \mid q_i)$ (where D is the depth of the model). Then, the K-L divergence between $\mathcal{M}$ and $\hat{\mathcal{M}}$ is given by the following expression.

$$D_{\text{KL}}(P^n_{\mathcal{M}} \| P^n_{\hat{\mathcal{M}}}) = \sum_{S_n \in S^n} P_{\mathcal{M}}(S_n) \log \left(\frac{P_{\mathcal{M}}(S_n)}{P_{\hat{\mathcal{M}}}(S_n)} \right) \tag{3.20}$$

Then, the above expression can be simplified as follows.

$$\log \left(\frac{P_{\mathcal{M}}(S_n)}{P_{\hat{\mathcal{M}}}(S_n)} \right) = \sum_{i=D+1}^{n} \log(\Pr(s_i \mid q_i)) - \log(\Pr(s_i \mid \hat{q}_i)),$$

where, $\hat{q}$ is the merged state and q is the original state. Then the expression on the right could be further bounded using the Lipschitz constant for the logarithm function and under the assumption that $\log(\Pr(s_j \mid q_i)) \neq 0 \; \forall q_i \in \mathcal{Q}$ and all $s_j \in \mathcal{A}$.

$$\sum_{i=D+1}^{n} \log(\Pr(s_i \mid q_i)) - \log(\Pr(s_i \mid \hat{q}_i)) \tag{3.21}$$

$$\leq \sum_{i=D+1}^{n} \left(\frac{\Pr(s_i \mid q_i) - \Pr(s_i \mid \hat{q}_i)}{\Pr(s_i \mid q_i)}\right) \tag{3.22}$$

$$\leq (n - D - 1)\kappa \tag{3.23}$$

where, $\kappa = \max_{q \in Q, s \in \mathcal{A}} \frac{\Pr(s|q) - \Pr(s|\hat{q})}{\Pr(s|q)}$. In the above inequalities, equation (3.22) is obtained from equation (3.21) by using the observation that $\Pr(s_i \mid \hat{q}_i) = \Pr(s_i \mid q_i) + \eta$, where η is the perturbation in the symbol emission probability from q_i when it is clustered into a new state $\hat{q}_i$. Hence, the K-L distance in equation (3.20) could be bounded by the following term.

$$\begin{aligned} D_{\mathrm{KL}}(P^n_{\mathcal{M}} \| P^n_{\hat{\mathcal{M}}}) &\leq \sum_{S_n \in S^n} P_{\mathcal{M}}(S_n)(n - D - 1)\kappa \\ &= (n - D - 1)\kappa \sum_{S_n \in S^n} P_{\mathcal{M}}(S_n) \\ &= (n - D - 1)\kappa \end{aligned} \tag{3.24}$$

Thus, a uniform bound on the Hamming distance between the original and the final model could then be obtained as follows,

$$\bar{d}(P_{\mathcal{M}}(S_n), P_{\hat{\mathcal{M}}}(S_n)) \leq \sqrt{\frac{(n - D - 1)\kappa}{2n}} \tag{3.25}$$

The above inequality thus, allows us to compare models with different state-space based on the predictive accuracy of a reduced model when compared to the original model. As compared to the earlier information theoretic criteria, which were based on the efficiency of data compression by different models, the inequality in (3.25) allows to compare them based on their symbol emission statistics and thus, is computationally efficient. It is possible to find a rather tighter bound in an expected sense by using the stationary distribution of the two Markov chains to find an expected bound on Hamming distance. However, finding the same is left as an exercise for future work. Using the above bound for selection of models could be more efficient than the information theoretic metrics (as it can estimated by using the symbol emission probabilities instead of the penalized likelihoods); however, finding a penalized version of the bound for model selection is also left as a future exercise.

3.5 Experiment and Data Details

In this section, we very briefly describe the two different data-sets which have been used in this chapter to illustrate and validate the proposed concepts. Specifically, we will describe the experiments done at Penn State to investigate instability in lean-premixed combustion and another benchmark data-set for anomaly detection in bearings.

3.5.1 Combustion

A swirl-stabilized, lean-premixed, laboratory-scale combustor was used to perform the experimental study. Tests were conducted at a nominal combustor pressure of 1 atm over a range of operating conditions, as listed in Table 3.1.

Table 3.1: Operating conditions

Parameters	**Value**
Equivalence Ratio	0.525, 0.55, 0.60, 0.65
Inlet Velocity	25-50 m/s in 5 m/s increments
Combustor Length	25-59 inch in 1 inch increments

In each test, the combustion chamber dynamic pressure and the global OH and CH chemiluminescence intensity were measured to study the mechanisms of combustion instability. The measurements were made simultaneously at a sampling rate of 8192 Hz (per channel), and data were collected for 8 seconds, for a total of 65536 measurements (per channel). A total of 780 samples of data were collected from all the tests where in every test the combustion process was driven from stable to unstable by changing either the equivalence ratio, ϕ. However, as the accurate model of the process is not available, an accurate label of transition of the process to unstable phase is not available. It is noted that the data consists the behavior of the process over a large number of operating condition and thus provides a rich set of data to test the efficacy of the algorithm in detecting classes irrespective of the underlying operating conditions.

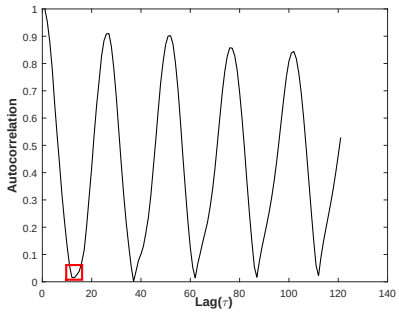


Figure 3.4: Autocorrelation function of time-series data during unstable phase of the combustion process. The time-series data is down-sampled by the lag marked in red square. It is noted that the individual time-series have their own down-sampling lags.

3.5.2 Bearing Prognostic Data

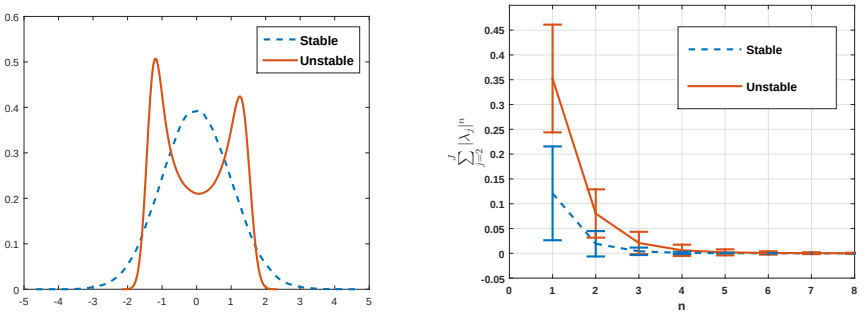
This test data has been picked from NASA's prognostics data repository [117, 118]. A detailed description of the experiments could be found in [119]. The bearing test rig hosts four test bearings on one shaft which is driven by an AC motor at a constant speed. A constant force is applied on each of the bearings and accelerometer data is collected at every bearing at a sampling rate of 20 kHz for about 1 s. The tests are carried for 35 days until a significant amount of debris was found in the magnetic plug of the test bearing. A defect in at least one of the bearings is found at the end of every test. In this chapter, we will use the data from a bearing which shows anomalous behavior in the later parts of test. In particular, out of the three data sets, we use set one where an inner race fault occurred on Bearing 3. In the analysis, we use data from Bearing 3.

3.6 Markov Modeling

In this section, we present results for modeling and analysis of the time-series data which are presented in this chapter.

3.6.1 Combustion

Time-series data is first normalized by subtracting the mean and dividing by the standard deviation of its elements; this step corresponds to bias removal and variance normalization. Data from engineering systems is typically oversampled to



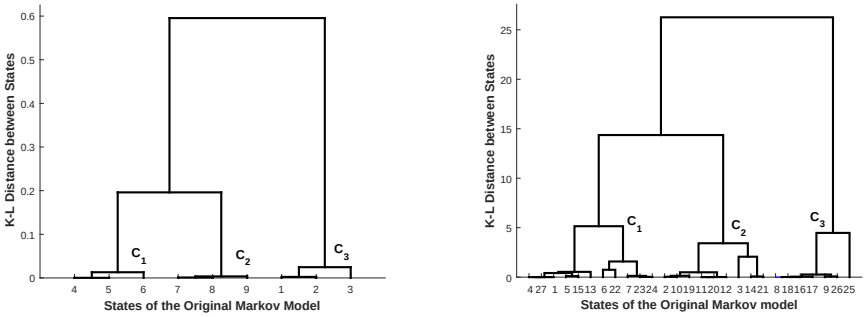
(a) Probability density function for the pressure time series data

(b) Spectral decomposition of the stochastic matrix for 1-step Markov model

Figure 3.5: The first plate in the above Figure shows the change in the empirical density calculated for the pressure time-series data as the process deviates from the stable operating condition to unstable operating condition. The second plate shows the spectral decomposition of the 1-step stochastic matrix for the data under stable and unstable operating conditions.

ensure that the underlying dynamics can be captured (in the current experiments, it was 8192 Hz). Due to coarse-graining from the symbolization process, an over-sampled time-series may mask the true nature of the system dynamics in the symbolic domain (e.g., occurrence of self loops and irrelevant spurious transitions in the Markov chain). Time-series is first down-sampled to find the next crucial observation. The first minimum of auto-correlation function generated from the observed time-series is obtained to find the uncorrelated samples in time. The data sets are then down-sampled by this lag. The autocorrelation function for the time-series data during unstable case is shown in Figure 4.5 where the data are downsampled by the lag marked in red rectangle in Figure 4.5. To avoid discarding significant amount of data due to downsampling, down-sampled data using different initial conditions is concatenated. Further details of this preprocessing can be found in [58].

The continuous time-series data set is then partitioned using maximum entropy partitioning (MEP), where the information rich regions of the data set are partitioned finer and those with sparse information are partitioned coarser. In essence, each cell in the partitioned data set contains (approximately) equal number of data points under MEP. A ternary alphabet with $\mathcal{A} = \{0, 1, 2\}$ has been used to symbolize the continuous combustion instability data. As discussed in section 3.5, we analyze data sets from different phases, as the process goes from stable through the



(a) Hierarchical cluster tree of the states during stable behavior

(b) Hierarchical cluster tree of the states during unstable behavior

Figure 3.6: Hierarchical clustering of states during stable and unstable conditions.

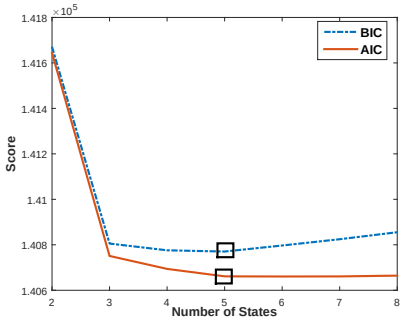
transient to the unstable region (the ground truth is decided using the RMS-values of pressure).

In Figure 3.5a, we show the observed changes in the behavior of the data as the combustion operating condition changes from stable to unstable. A change in the empirical distribution of data from unimodal to bi-modal is observed as the system moves from stable to unstable. We selected 150 samples of pressure data from the stable and unstable phases each to analyze and compare. First, we compare the expected size of temporal memory during the two stages of operation. There are changes in the eigenvalue decomposition rate for the 1-step stochastic matrix calculated from the data during the stable and unstable behavior, irrespective of the combustor length and inlet velocity. During stable conditions, the eigenvalues very quickly go to zero as compared to the unstable operating condition (see Figure 3.5b). This suggests that the size of temporal memory of the discretized data increases as we move to the unstable operating condition. This indicates that under the stable operating condition, the discretized data behaves as symbolic noise as the predictive power of Markov models remain unaffected even if we increase the order of the Markov model. On the other hand, the predictive power of the Markov models can be increased by increasing the order of the Markov model during unstable operating condition, indicating more deterministic behavior. An $\epsilon = 0.05$ is chosen to estimate the depth of the Markov models for both the stable and unstable phases. Correspondingly, the depth was calculated as 2 and 3 for the stable and unstable conditions (see Figure 3.5). The corresponding $D(\epsilon)$ is used to construct the Markov models next. First a PFSA whose states are words over $\mathcal{A}$ of length $D(\epsilon)$ is created and the corresponding maximum-likely parameters ($\boldsymbol{M}$ and $\boldsymbol{\Pi}$)

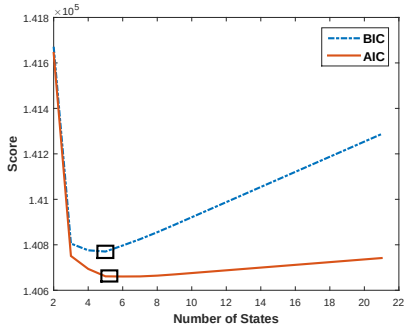
are estimated. Then, the hierarchical clustering algorithm using K-L distance is used to cluster and aggregate the states. It is noted that we create individual models for every sample, i.e., every sample is partitioned individually so that the symbols will have different meaning (i.e., they represent different regions in the measurement space of the signals) for every sample. Consequently, each sample will have a different state-space when viewed in the continuous domain. Thus, we do not show the mean behavior of the samples during any operating regime as the state-space would be inconsistent (even though the cardinality could be the same).

In Figure 3.6, we show the hierarchical cluster tree which details the structure of the state-space for the PFSA with depth $D(\epsilon)$ for a typical sample during stable and unstable behavior. The cluster tree also suggests the symbolic noise behavior of the data during the stable regime (the states are very close to each other based on the K-L distance). However, clearly a coarse clustering of states in the model during the unstable behavior would lead to significant information loss (as the states are statistically different). However, to compare the two Markov models, we keep the cardinality of the final models the same. For example, the algorithm is terminated with 3 states in the final Markov model during the stable as well as the unstable regime. and the final aggregated states are the three clusters depicted in the Figure 3.6. Once the final aggregated states are obtained, we estimate the parameters of the model using the Bayesian inference discussed in section 3.3.2.

Next, we present some results for model selection using the information-theoretic criteria discussed earlier in section 3.3.3. BIC and AIC are used to select the model which achieves the minimum score. As seen in the Figures 3.7a through 3.7b, the model with 5 states is selected for stable as well as for the unstable case (note that the original model for the stable class had 9 states for depth 2 and the unstable model had 27 states for a depth of 3). In contrast to cross-validation, the two criteria provide an unsupervised way for model selection. Thus we see that we need much smaller state-space to preserve the temporal statistics of the data and AIC and BIC provide us with a technique to select the compact model. In Figure 3.8, we show the Hamming distance between the sequences generated by the original model and the reduced models for a typical sample each from stable and unstable combustion. The box-plots are generated by simulating the original model and the reduced-order model to generate symbol sequences of length 1000 from 100 different initial states (i.e., a total of 100 strings are generated) and the Hamming



(a) Model scores using the BIC and AIC criterion during a typical stable condition



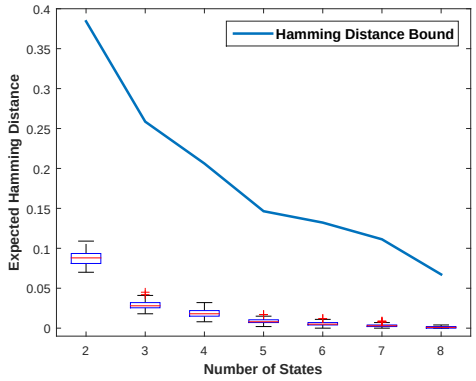
(b) Model scores using the BIC and AIC criterion during a typical unstable condition

Figure 3.7: Unsupervised model selection during stable and unstable conditions.

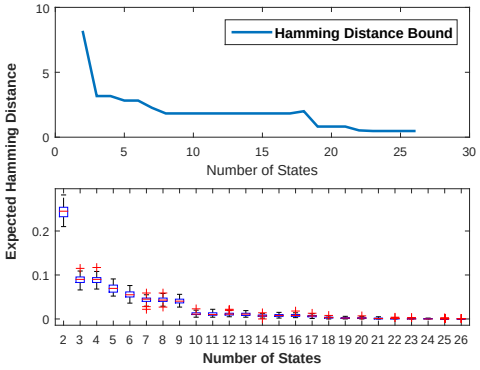
distance between them is calculated. A bound on the Hamming distance between the sequences generated by the original model and final model is also calculated using the inequality (3.25). The results are shown in Figure 3.8. It is possible to use the proposed Hamming distance metric to select a final model; however, this measures the distance between the distributions induced by the Markov models, and model selection using it is left as a future work. It is noted that the bounds on Hamming distance can provide a computationally convenient way to select model scores as it can be found from the symbol emission probabilities of the model instead of explicitly looking at the predictive capability by looking at the likelihoods of the symbol sequences.

3.6.2 Bearing

The same procedure of downsampling and depth estimation is followed for analysis of bearing data as was described in the previous section for combustion. A ternary alphabet is again chosen to discretize the continuous data after downsampling and the maximum entropy partitioning is used to find the partitions. Using the spectral method, a depth of 2 (i.e., a total of 9 states) is estimated for an $\epsilon = 0.02$ (we skip the plot of spectral decomposition plot for brevity). The BIC and AIC score for the different models is shown in Figure 3.9 and the model with five states is selected using the obtained scores (marked in black rectangle). In Figure 3.10, we show the Hamming distance between the sequences generated by the original model (with 9 states) and the reduced models and the corresponding bounds obtained by



(a) Hamming distance between the original and final models for a typical stable combustion sample



(b) Hamming distance between the original and final models for a typical unstable combustion sample

Figure 3.8: Box plot for Hamming distance between the original and reduced-order models obtained after merging based on the results in section 3.4

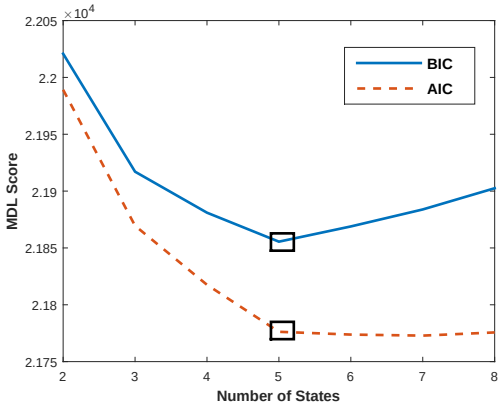


Figure 3.9: The model scores using the BIC and AIC criteria and the final model selected is depicted by the black rectangle.

inequality (3.25).

3.7 Classification and Anomaly Detection Results

In this section, we present some results for anomaly detection and classification using the pressure time-series data to infer the underlying reduced-order Markov model. As we discussed earlier in section 3.5.1, the exact transition point of the system from stable to unstable is unknown, we first present results on anomaly detection and clustering of the data into different clusters which can be then

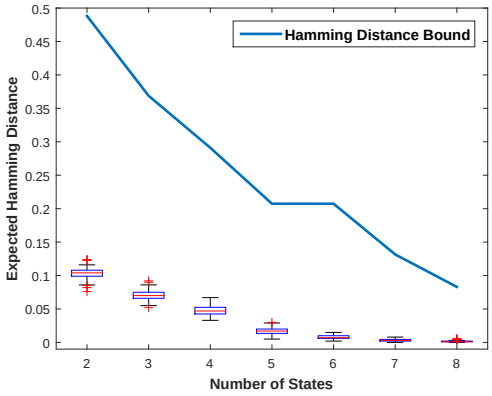


Figure 3.10: Box plot of the Hamming distance between the original and reduced-order models along with the analytical bound presented in section 3.4.

associated with the stable and unstable class. We will present two different metrics for anomaly detection that allows models of different state-space and structure to be compared. It is noted that the word metric is used here in a loose sense; it is meant to be a distance that could be used to compare two different Markov models.

3.7.1 Anomaly Detection

As individual time-series have different state-space, we define some metrics to compare them. These metrics reflect changes in the information complexity of Markov models and reveal different behavior of combustion process based on the changes in the inferred data model. In particular, the following two metrics are defined.

1. Cluster Divergence: This measure is defined for individual Markov models based on the cluster structure of the state-space of the model. Physically, it represents the maximum statistical difference between the states of the Markov model measures using K-L distance. It is calculated for a particular model $\mathcal{M}$ as follows

$$\Delta_{\mathcal{M}} = \max_{q_i, q_j \in \mathcal{Q}} d(q_i, q_j) \tag{3.26}$$

   where d is defined by equation (3.4).

2. Discrepancy Statistics: We measure the discrepancy between the i.i.d. statistics and the Markov statistics for the discretized data. This could be also interpreted as the information gain for Markov models. This measure also

represents the information complexity of the data. If the i.i.d. statistics and the Markov statistics are very close, then the data has no temporal statistics; however, an increase in this measure would indicate the information gain by creating a temporal Markov model for the data. This is measured by the following equation.

$$H_{\mathcal{M}} = \sum_{q \in \mathcal{Q}} \Pr(q) D_{KL}(\Pr(\mathcal{A} \mid q) \| \Pr(\mathcal{A})) \tag{3.27}$$

where $\Pr(\mathcal{A} \mid q)$ represents the symbol emission probability conditioned on a state q of the Markov model and $\Pr(\mathcal{A})$ represents the marginal symbol emission probability. The term D_{KL} represents the symmetric K-L distance between the two distributions.

In Figure 3.11, we present some results to show the behavior of $\Delta_{\mathcal{M}}$ with increasing pressure fluctuations. It is noted that every model has been created in an unsupervised fashion by first discretizing and then, estimating the memory of the discrete sequence. As seen in Figure 3.11a, there are three distinct behaviors that can be associated with $\Delta_{\mathcal{M}}$. With low pressure fluctuations, the metric is very close to 0, indicating that the states of the model are very similar statistically. This is seen until data number 200 with corresponding $P_{rms} \sim 0.065$ psig, which leads to a gradual change to a point where the measure saturates with $P_{rms} \sim 0.12$ psig (when the process becomes unstable). Thus, with this gradual trend with increasing pressure fluctuations, we associate different behaviors with the process. However, as is seen in the Figure 3.11a, the transition from stable to unstable behavior is not clearly defined and is very difficult to label during the experiments as the process is very fast. We show the pressure signals from the three different clusters in Figure 3.11b where it could be seen that the sample number 250 could be seen to approach an approximate limit cyclic behavior (and thus, could be loosely classified as transient stage). An important point to note at this point is that this measure is independent of any operating conditions and only depends on stability (or instability) of the process. This metric is thus used for anomaly detection. In Figure 3.11c, we show the statistics of $\Delta_{\mathcal{M}}$ with four states. We see that there is some loss of information up on merging states in the unstable class; the stable cluster remains unchanged implying that the states are statistically similar and the model distortion up on merging of states is insignificant. Thus, $\Delta_{\mathcal{M}}$ can be reliably

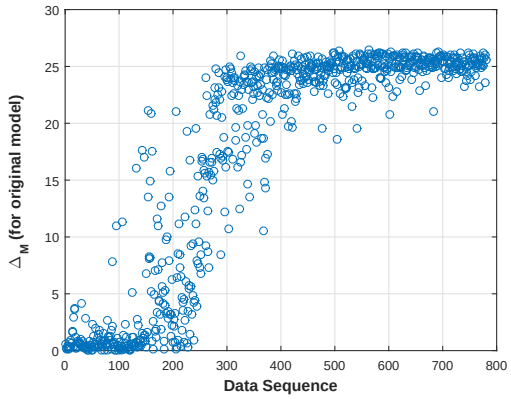
used to detect departure from stable behavior.

The statistics for the discrepancy measure for the full state models is shown in Figure 3.12. The plot in Figure 3.12 also agrees qualitatively with the earlier results on $\Delta_{\mathcal{M}}$. From these plots, we can infer that the Markov statistics for the stable cluster is very similar to the i.i.d. statistics and thus the data is very much independently distributed and conditioning on the inferred states of the Markov models doesn't improve predictability (or information complexity) of the temporal model. Thus, these two measures help infer the changes in the behavior of the data during the combustion process and are useful for anomaly detection.

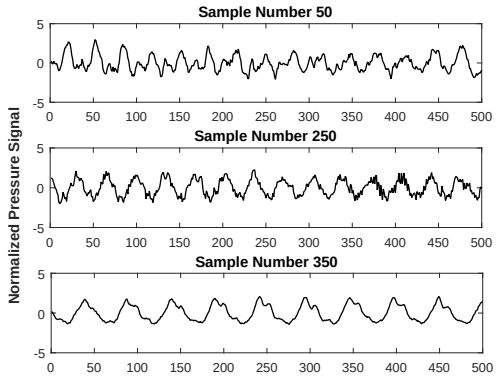
To see more explicitly the changes in the underlying models, the models during stable and unstable phases are visualized in the information space. To do this, we reduce the state space of the models to just 2 states and estimate the corresponding emission parameters. As the models have three symbols, the emission matrix has 2 rows and each row corresponds to the symbol emission probabilities conditioned on the two states. Each of these rows for 100 cases from stable and 100 cases from unstable are plotted on a single simplex plane which is shown in Figure 3.13. The Figure shows the clusters of stable and unstable cases in the information space and that the model with even 2 states are clustered separately. This shows that there is a structured change in the temporal dynamics of the data at the two phases and that the inferred Markov models are able to capture this change. Furthermore, the distinctive features of the models are sufficiently retained even after significant reduction in the state-space of the models.

3.7.2 Classification

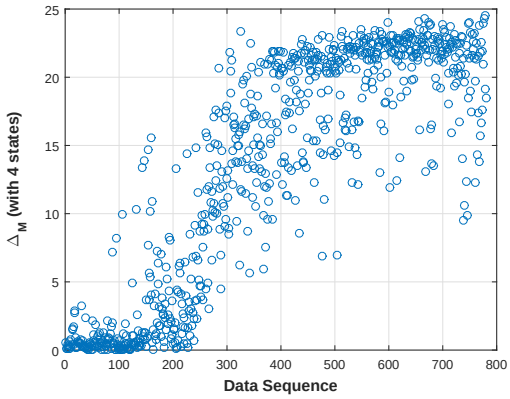
These models are then used to train classifiers using support vector machines (SVM) and decision trees (DT) [23]. The rationale behind using multiple classifier is to show that the performance of the Markov models is independent of the classification technique (i.e., it works equally well with maximum margin classifiers or decision tree classifiers). The SVM classifier is trained using a radial basis function kernel while the decision tree is trained using the standard euclidean distance. The classifiers are trained with 100 data points from each class and are tested on the remaining data (around 80 and 380 for stable and unstable respectively). The tests are repeated for 100 different train and test data sets from the total data. The



(a) $\Delta_{\mathcal{M}}$ for the full state model for the time-series data with increasing pressure root mean square



(b) Typical pressure signals from the three clusters seen in Figure 3.11a



(c) $\Delta_{\mathcal{M}}$ for models with 4 states for the time-series data with increasing pressure root mean square

Figure 3.11: Anomalous behavior of data during the combustion process

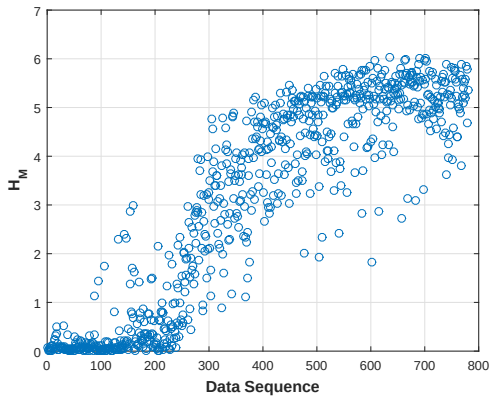


Figure 3.12: Variation of discrepancy statistics $H_{\mathcal{M}}$ with increasing pressure fluctuations. This also shows an anomaly around the point 200 and qualitatively agrees to the behavior of $\Delta_{\mathcal{M}}$.

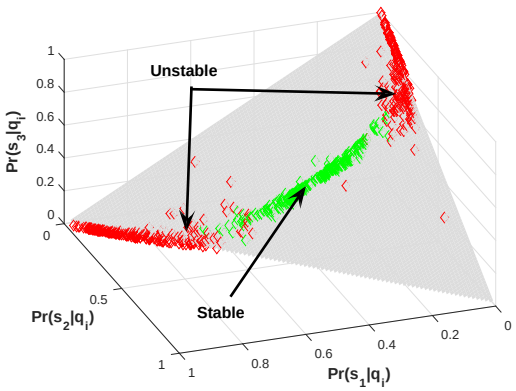


Figure 3.13: Cluster of stable and unstable phase in information space. Each point is a row of the emission matrix for the reduced Markov model with 2 states. The plot shows the change in the Markov model as the process moves from stable and unstable. Red diamonds represent the unstable phase while green diamonds represent the stable phase.

results of classification accuracy are listed in Table 3.2. The SVM classifier is able to achieve around 1.67% error using models with 2 states while the decision tree classifier is able to achieve around 4.70% error using models with 4 states. This provides another way of selecting the final model for state merging in a supervised learning setting. It is noted that the original models contain 9 states for stable and 27 for unstable class.

Table 3.2: Performance of classifiers with different number of states. Mean Error= Lower is better.

Number of States	Classifier	Classification Error (%)
9	SVM	3.48 ± 0.74
	DT	9.83 ± 3.24
8	SVM	3.62 ± 0.71
	DT	9.38 ± 3.11
7	SVM	2.87 ± 0.68
	DT	7.70 ± 2.61
6	SVM	2.48 ± 0.61
	DT	7.00 ± 2.55
5	SVM	2.05 ± 0.54
	DT	6.10 ± 2.17
4	SVM	1.86 ± 0.43
	DT	4.72 ± 2.29
3	SVM	1.69 ± 0.45
	DT	5.56 ± 1.90
2	SVM	1.67 ± 0.43
	DT	4.83 ± 1.80

3.8 Conclusions and Future Work

In recent times the idea of representation learning has become very popular in machine learning literature as it allows to decouple the problem of model learning for data from the end-objectives like classification or clustering. In this chapter, we presented a technique for Markov modeling of time-series data using concepts of symbolic dynamics which allows inference of model structure as well as parameters

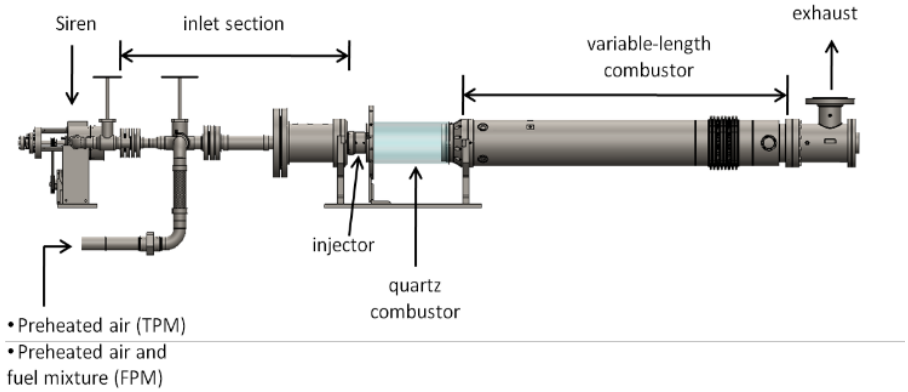


Figure 3.14: Schematic drawing of test facility at Penn State center for Propulsion.

for compact data representation. In the proposed technique we first estimate the size of memory of the discretized time-series data. The size of memory is estimated using spectral decomposition properties of the one-step Markov model created from the symbol sequence. Then, a second pass of data is made to infer the model with the right memory and the corresponding symbol emission matrix is estimated. Then the equivalence class of states based on K-L distance between the states are estimated using a hierarchical clustering of the corresponding states of the Markov model. The proposed concepts were validated using two different datasets– combustion instability and bearing. Modeling of combustion instability still remains a puzzle in the combustion community. The Markov modeling technique was used to analyze the problem of combustion instability. The proposed ideas were tested on experimental data from a swirl-stabilized combustor used to study unstable thermo-acoustic phenomenon during combustion process. The proposed approach allows us to infer the complexity of the time-series data based on the inferred Markov model. Two different metrics were proposed for anomaly detection and classification of the stable and unstable classes. The results presented in this chapter are encouraging as the inferred models are able to identify the stable and unstable phases independent of any other operating condition.

Simultaneous optimization of discretization and memory estimation is also a topic of future research. While the results obtained with Markov modeling for the combustion instability problem are inspiring, further investigation with transient data is required for better characterization of the process.

Appendix

Description of Experimental Apparatus

Figure 3.14 shows a schematic drawing of the variable-length combustor. The combustor consists of an inlet section, an injector, a combustion chamber, and an exhaust section. The combustor chamber consists of an optically-accessible quartz section followed by a variable length steel section (see [120] for more details).

High pressure air is delivered to the experiment 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 using a dome pressure regulator. The air is

pre-heated to a maximum temperature of 250 °C by an 88kW electric heater. The fuel for this study is natural gas (approximately 95% methane). It is supplied to the system at a pressure of 200 psig. The flow rates of the air and natural gas are measured by thermal mass flow meters. The desired equivalence ratio and mean inlet velocity is set by adjusting these flow rates with needle valves. For fully pre-mixed experiments (FPM), the fuel is injected far upstream of a choke plate to prevent equivalence ratio fluctuations. For technically pre-mixed experiments (TPM), fuel is injected in the injector section near the swirler. It mixes with air over a short distance between the swirler and the injector exit.

Chapter 4

Temporal Learning in Video Data Using Deep Learning and Gaussian Processes

One of the biggest weaknesses of symbolic analysis-based Markov modeling is that the symbolization process is not well understood for high dimensional systems. Consequently, the effectiveness of these concepts is very limited for high-dimensional temporal data. This chapter presents an approach for data-driven modeling of hidden, stationary temporal dynamics in sequential images or vidoes (high-dimensional data) using deep learning and Bayesian non-parametric techniques. In particular, a Deep Convolutional Neural Network (CNN) is used to extract spatial features in an unsupervised fashion from individual images and then, a Gaussian process is used to model the temporal dynamics of the spatial features extracted by the Deep CNN. By decomposing the spatial and temporal components and utilizing the strengths of deep learning and Gaussian processes for the respective sub-problems, we are able to construct a model that is able to capture complex spatio-temporal phenomenon while using relatively small number of free parameters (or hyperparameters). The proposed approach is tested on high-speed grey-scale video data obtained of combustion flames in a swirl-stabilized combustor, where certain protocols are used to induce instability in combustion process. The proposed approach is then used to detect and predict the transition of the combustion process from stable to unstable regime. It is demonstrated that the proposed approach is able to detect unstable

flame conditions using very few frames from high-speed video. This is useful as early detection of unstable combustion can lead to better control strategies to mitigate instability. Results from the proposed approach are compared and contrasted with several baselines and recent work in this area, the performance of the proposed approach is found to be significantly better in terms of detection accuracy, model complexity and lead-time to detection. Most of the results in this chapter are based on the results earlier published in [7].

4.1 Introduction

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstractions [121–123]. Deep learning methods are representation-learning methods obtained by composition of non-linear modules or layers that each transform the representation at the previous level into a higher and slightly more abstract level in a hierarchical manner. The main idea is that by cascading a large number of such transformations, very complex functions can be learned in a data-driven manner. Convolutional deep neural nets [124, 125] are designed to process data that come in the form of multiple arrays, for example images. In this technique, a discrete spatial convolution filter is used for detecting highly correlated distinctive motifs across an image. The same process can be extended to find temporal feature by using a convolution filter over time, and using a stack of a number of images as input. However, this will lead to an increase in the number of the hyperparameters used to train such a spatio-temporal convolutional network. Instead, we propose a framework, where a convolutional network is used to extract features from images and a Bayesian nonparametric model like the Gaussian process [126] is used to model the temporal dynamics of the features; the hope is to reduce the number of hyperparameters used to train such a network while modeling the spatio-temporal dynamics in an image sequence or videos. The proposed algorithm is used to detect stable and unstable combustion flame dynamics in a swirl-stabilized combustor. Furthermore, adding a Gaussian process-based filter also allows to reduce over-fitting by the deep CNN to some extent as it allows to relate to the causal dynamics present in the data in a much lower dimensional space.

Combustion instability is a highly nonlinear coupled thermoacoustic phenomena

that results in self-sustained oscillations in a combustor. These oscillations can result in severe structural degradation in jet turbine engines. Some good surveys on the current understanding of the mechanisms for the combustion instability phenomena can be found in [99–103]. The current state-of-the-art techniques rely heavily on model-based approaches for analysis of the process. The current literature lacks rigorous statistical analysis of the combustion instability phenomenon so that the predictive power of the data has been largely overlooked. Active combustion instability control (ACIC) with fuel modulation has proven to be an effective approach for reducing pressure oscillations in combustors [104, 105]. Based on the work available in literature, one can conclude that the performance of ACIC is primarily limited by the large delay in the feedback loop and the limited actuator bandwidth [104, 105]. Model-based approaches for active control are infeasible as complexity of the models and uncertain measurements make real-time estimates difficult. On the other hand, use of machine learning techniques for information extraction remain unexplored for this problem. From the perspective of active control of the unstable phenomena, it is necessary to accurately detect and, desirably, predict the future states of the combustion process. The goal of this chapter is to present a statistical model for the instability phenomenon during combustion which could be used to design a statistical filter to accurately predict with high confidence the system states. This can potentially alleviate the problems with delay in the ACIC feedback loop and thus possibly improve the performance.

Some recent work on statistical analysis of combustion instability using pressure time-series data could be found in [68, 127]. The work presented in [127] shows the change in the underlying Markov model for pressure data as the system approaches the unstable phase resulting in self-sustained oscillations of the flame. Some other popular methods for detection of coherent structures include proper orthogonal decomposition (POD) [128] and dynamic mode decomposition (DMD) [129], which use tools of spectral theory to derive spatial coherent structure modes. Specifically, DMD has been used to estimate the growth rates and frequencies from experimental data and also for stability analysis of the experimental data. Recently, some analyses were also presented using deep learning for detection of combustion instabilities [130, 131]. More recently the work done in [132] presents a neuro-symbolic approach where the output a deep convolutional network are analyzed by a Markov modeling module to construct an anomaly measure. However, as

we demonstrate later, the advantages of using a deep neural network is unclear. Another analysis is presented in [133] using various image analysis techniques like histogram of oriented gradients (HOG) and Wavelets; however, the final decision is made by making a Markov model out of the features and thus requires long sequences to come to a decision. Moreover, perfect separability between stable and unstable classes is not achieved. This chapter presents further insights into the combustion instability phenomena from a data-driven perspective and presents a framework which allows temporal learning in video data in a lower dimensional subspace of features produced by a deep learning module. We show that flames have different spatial structures at unstable behavior when compared to stable behavior and with an appropriate architecture of neural network, we can perfectly capture the associated distinguishing features.

Contributions: This chapter presents a framework for modeling of spatio-temporal dynamics in sequential images using deep learning and Gaussian processes. We use the proposed algorithm to present a statistical analysis of the complex combustion process and show that we are able to capture the change in the model as the system moves from stable to unstable. We show that the deep CNN is able to achieve fairly good classification performance; however the use of a Gaussian process filter to model temporality of extracted features results in perfect detection with very low false alarm rates and much short lead time to detection. The advantage of the proposed method is that it can be used for making early predictions of the transition from stable to a quasi-periodic unstable oscillation.

4.2 Preliminaries and Problem Formulation

In this section, we will very briefly describe the convolutional neural nets and the Gaussian processes. Interested readers are referred to [122, 125, 126] for an in-depth discussion on these topics. After the brief review, we will state the problem considered in this chapter. The idea to use Gaussian processes to model the features obtained from deep CNN is to be able to add memory to the estimation filter without going to the architecture of a recurrent neural network.

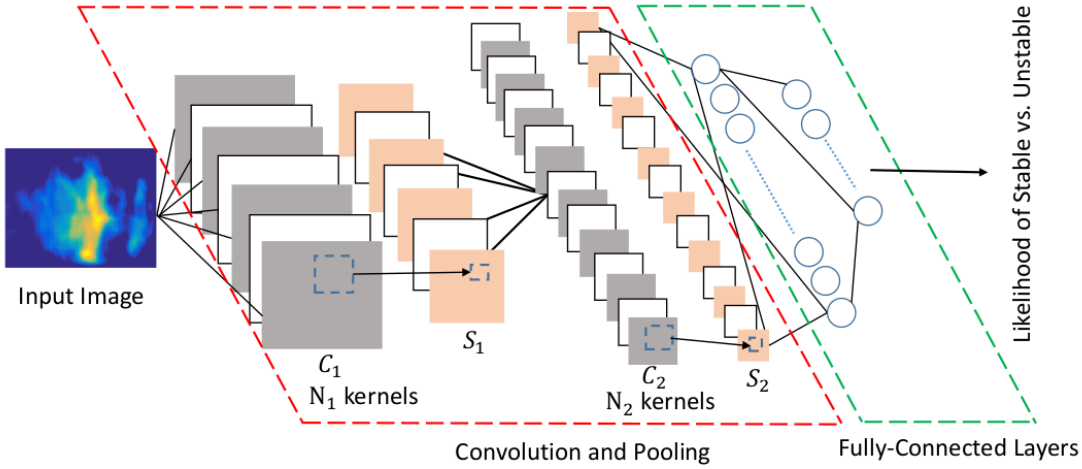


Figure 4.1: This figure shows the deep convolutional neural network used with two convolutional layers and two hidden layers. The number of kernels and number of units in the hidden layer are varied to study the affect of size of parameter set on the results. The convolution layers are denoted by C_i and the pooling layers are denoted by S_i. Each rectangular image is a feature map.

4.2.1 Convolutional Neural Networks

In this section, we briefly introduce the deep convolutional neural networks (CNN) for the completeness of the chapter and motivation for use in this work. Deep convolutional neural networks have a long history [134] but they caught the attention of vision community when they surpassed the state-of-the-art algorithms for image recognition problem by large margins [124]. Since then, it has been very widely used for almost all software applications for recognition in images and videos. These networks are widely used to process data that come in form of multiple arrays, for example a color image composed of $2D$ arrays containing pixel intensities. In practice, the deep convolutional networks have shown to outperform most of the other hand-tuned feature extraction algorithms and thus, have become very popular for learning with image data. This is also the motivation for the use of CNN in this work. The general architecture of a deep convolutional neural network is shown in Figure 4.1 and is structured as a series of stages. In general, the first few stages (left) contain a lot of local information which is aggregated as we go deeper (right) in the network. The first few stages are composed of two types of layers: convolutional and pooling layers (shown as C_i and S_i respectively in the

figure). Units in a convolutional layer are organized in feature maps, within which each unit is connected to local patches in the feature maps of the layer through a set of weights called filter banks (or kernels). The result of this local weighted sum is then passed through a non-linearity. Mathematically, the filtering operation performed by a feature map (or kernel) is equivalent to discrete convolution (and hence the name). The motivation for use of discrete convolution is to be able to find local correlated, distinctive patches (or motifs) in the images.

Contrary to the convolution layer, the role of the pooling layer is to merge semantically similar features into one. A typical pooling unit computes the maximum of a local patch of units in one feature map (or in a few feature maps). Neighboring pooling units take input from patches that are shifted by more than one row or column, thereby reducing the dimension of the representation and creating invariance to small shifts and distortions. This also helps with the overfitting problem by reducing the dimension of the representation. Two or three layers of convolution, non-linearity and max-pooling are stacked, followed by the fully connected layers. The network is then trained using backpropagation using stochastic gradient descent and it allows to train the weights of the filter banks (kernels).

4.2.2 Gaussian Processes

In this section, we very briefly explain the Gaussian process model which is used to model the features extracted by CNN.

Definition 4.2.1 (Gaussian Processes) *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

A Gaussian process (GP) is completely specified by its mean function and covariance function. We define the mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ of a real process $f(\mathbf{x})$ as follows.

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \end{aligned}$$

and the Gaussian process is written as follows.

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{4.1}$$

A Gaussian process is defined as a collection of random variables such that the if the GP specifies $(x_1, x_2) \sim \mathcal{N}(\mu, \Sigma)$, then it must also specify $x_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$ where Σ_{11} is the relevant submatrix of Σ. Some of the common choice for GP covariance function are rational quadratic (RQ), squared exponential (SE) and Matérn covariance functions. Some of the functional forms of the covariance function and the associated free parameters are listed in Table 4.1. The inference problem associated with GP is to infer the related hyperparameters (or the free parameters). The associated hyperparameters with a GP are the parameters corresponding to the mean and covariance function for the GP. The inference methods compute an approximate posterior, an approximate negative log marginal likelihood and its partial derivatives w.r.t. the hyperparameters, given the choice of mean, covariance and likelihood functions. Some of the common inference methods are expectation propagation, Laplace's approximation and Kullback-Leibler divergence minimisation.

4.2.3 Problem Statement

Consider a sequence of images, $\{\boldsymbol{X}_t\}_{t\in\mathbb{N}}$, $\boldsymbol{X}_t \in \mathbb{R}^{d\times d}$, $d \in \mathbb{N}$. $\boldsymbol{X}_t$ represents observations from a dynamical system with finite memory s.t. $\boldsymbol{X}_t = f(\boldsymbol{X}_{t-1}, \ldots, \boldsymbol{X}_{t-m})$. Then, the statistical learning task is to find a representation of the dynamical system f in a lower dimensional subspace such that $\tilde{\boldsymbol{X}}_t = g(\tilde{\boldsymbol{X}}_{t-1}, \ldots, \tilde{\boldsymbol{X}}_{t-p})$, where $\tilde{\boldsymbol{X}}_t = \varphi(\boldsymbol{X}_t)$, $\tilde{\boldsymbol{X}}_t \in \mathbb{R}^{k\times k}$, $k << d$. It should be noted that the feature extraction transformation (φ) may not retain the original system memory (i.e., $m \neq p$). The idea of the underlying problem is also shown as a schematic in figure 4.2. In the proposed problem, the deep neural network could be just used as a feature extractor and we allow the decisions to be made by modeling the features using a Gaussian process. The motivation for using such a composite approach is twofold. Firstly, it can help relate to the causal dynamics of the underlying physical system, and second, it can help with the overfitting of the neural network.

4.3 Combustion Experiment Details

The experiment consists of a swirl combustion chamber where the inlet conditions can be varied to achieve combustion in stable and unstable modes. In these

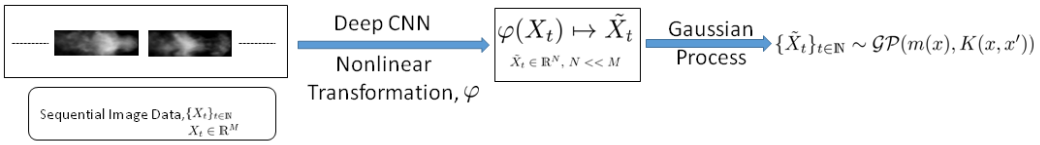


Figure 4.2: Concept of the proposed modeling scheme using the deep convolution net and Gaussian processes which is used to model the hidden dynamics in the sequential image data.

Table 4.1: Some examples of covariance functions used for Gaussian Process. The first and second are Squared Exponential (SE) and the third function is a rational quadratic (RQ).

Mathematical form	Free Parameters
$\sigma^2 \exp(-(x-x')^T \Lambda^{-2}(x-x')/2)$	$\{\lambda_1, \ldots, \lambda_D, \sigma\}$
$\sigma^2 \exp(-(x-x')^T (x-x')/2\ell^2)$	$\{\ell, \sigma\}$
$\sigma^2(1+\frac{1}{2\alpha}(x-x')^T \Lambda^{-2}(x-x'))^{-\alpha}$	$\{\lambda_1, \ldots, \lambda_D, \sigma, \alpha\}$

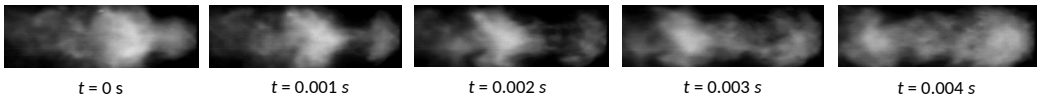
experiments the inlet conditions are controlled via a combination of premixing condition of air and fuel, fuel-flow rate (FFR) and inlet Reynolds number (Re), each combination defining a test protocol. Reynolds number is a dimensionless quantity and is defined as the ratio of inertial and viscous forces in a flow; typically high Reynold number flows are turbulent and vice-versa.

In one test protocol, two inlet Reynolds numbers (Re) were chosen – $= 7,971$ and $= 15,942$ for a fixed fuel flow rate of 0.495 g/s, where the lower Re lead to stable combustion behavior and higher Re exhibited unstable behavior. In another other protocol, the inlet Re is held constant at $10,628$ for two different fuel flow rates (FFR). The details of the protocols along with their ground truths (e.g., stable, relatively stable and unstable) are presented in table 4.2.

Apart from this, an intermediate stage is also tested to capture the transition between stable and unstable combustion. This is done by (1) increasing the air flow rate (AFR) while keeping the FFR constant, and, (2) by decreasing the fuel flow rate while keeping AFR constant so that the system state is varied from stable to unstable. The transition protocols are as follows (It is noted that we only consider the protocol (2) in this chapter, i.e., instability is induced by decreasing the fuel flow rate). All units in liters per minute

The swirl combustor has an Inlet Optical Access Module (IOAM) which is

(a) Image of flames during stable operation



(b) Image of flames during unstable operation shows the periodic blowing out of flame during the unstable oscillations.

Figure 4.3: High-speed image data from the stable and unstable phases of combustion. Spatial and temporal changes in the flame structure during the unstable process are visible. The flame enters on the right end and moves towards the left.

Table 4.2: Experiment set 1: Protocols with respective ground truth conditions for data collection. 3 s of greyscale image sequence at 3 kHz.

Premixing	**FFR (g/s)**	**Re**	**Ground truth**
Partial	0.495	7,971	Stable
	0.495	15,942	Unstable
	0.308	10,628	Unstable
	0.660	10,628	Stable
Full	0.495	7,971	Stable
	0.495	15,942	Unstable
	0.308	10,628	Unstable
	0.660	10,628	Stable

used to collect High-speed images of the combustion process at a resolution of 1024×1024 and a frequency of 3kHz. Data was collected for a period of 3 seconds yielding a sequence of 9000 images for every operating condition. In total 72000 images are generated from the experiment with $36,000$ images each for the stable and unstable class that are analysed using the baselines and the proposed method. Since the flame is limited mainly along the horizontal axis, the images are cropped

Table 4.3: Experiment set 2: Protocols to produce transition from stable to unstable combustion

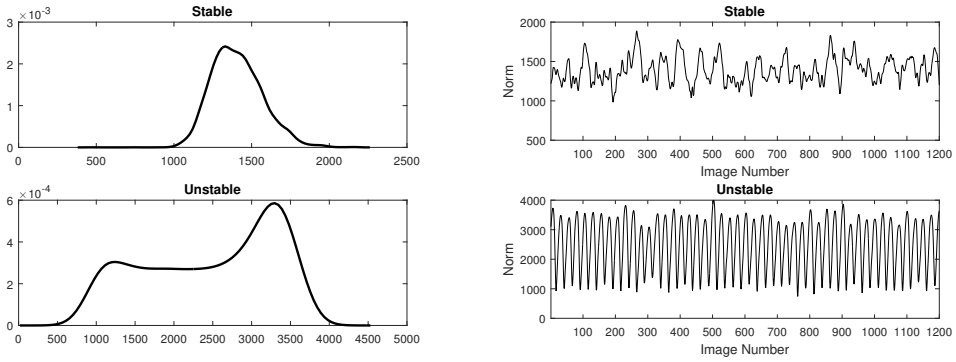
AFR		**FFR (g/s)**		**Ground truth**
start	end	start	end	
500	500	40	30	Stable→Unstable
600	600	50	35	Stable→Unstable

along the vertical dimension to remove background. The resolution is then scaled down to get a final image size of 51×128, that is analyzed in this chapter.

In figure 4.3a, we show sequence of images at different time instants to show the changes in the spatial structure of flames at stable condition. Figure 4.3b presents sequences of images of dimension 392×1000 pixels for unstable ($\mathrm{Re} = 15,942$, FFR$= 0.495\,\mathrm{g\,s^{-1}}$ and full premixing) state. The flame inlet is on the right side of each image and the flame flows downstream to the left. As the combustion is unstable, coherent structure formation along the flow is observed. The figure 4.3 shows formation of mushroom-shaped vortex at $t = 0, 0.001s$ and the shedding of that towards downstream from $t = 0.002\,\mathrm{s}$ to $t = 0.004\,\mathrm{s}$. If looked carefully, one can observe the temporal changes in the flames during unstable combustion; it shows the periodic blowing off of images at times (e.g., $t = 0.001\,\mathrm{s}$). This is sometimes studied as appearance of *coherent structures* which can be loosely defined as organized spatial features which repeatedly appear and undergo a characteristic temporal life cycle. The temporal life cycle can be associated to the limit cycle that the system gets locked on to during an unstable condition which can occur at a lot of operating conditions.

4.4 Results and Discussion

We will first show that the unstable pressure fluctuations occur when the system gets locked onto a quasi-periodic, limit cyclic behavior due to the synergy between heat release rate fluctuations and the acoustics properties of the combustor. In classical combustion literature, these limit cycles are associated with phase lag between heat release rate frequency and velocity (also known as Rayleigh's criterion). It is in general difficult to quantify this phase lag from image data, we are able to capture this cyclic behavior by finding the approximate change in the flame images. In Figure 4.4, we show the lumped changes in the video data as the combustion moves from stable to unstable. The euclidean distance between the images are calculated from an initial reference image at time point k by computing the 2-norm of the image residual with respect to the reference. In Figure 4.4a, we show the approximate empirical density of the euclidean norm for the sequential image data during the two phases. As we can see, there is a change in the empirical density from approximately a uni-modal Gaussian to multi-modal. In Figure 4.4b, we show



(a) Change in the approximate empirical density of the euclidean distance between the images

(b) Change in the euclidean distance between images with time shows the limit cycle during the unstable phase

Figure 4.4: Change in image data as the combustion process moves from stable to unstable. From the approximate empirical density and the limit cycle shown in sequence of the euclidean norm between the images, change from minor fluctuations to a quasi-periodic behavior is visible

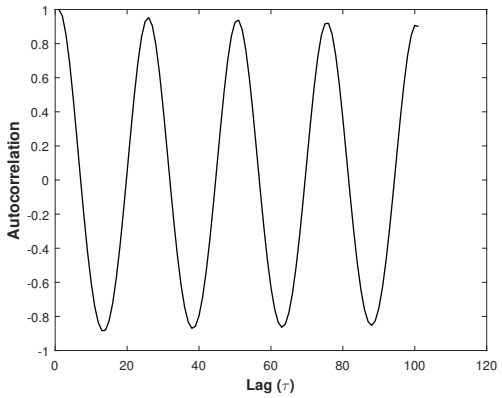


Figure 4.5: Autocorrelation of the image residuals (4.4b) during the unstable phase of combustion indicating the limit cycle and its frequency (~ 26)

the time-series of the residual norm which shows a near-periodic behavior (limit cycle) during the unstable phase. The autocorrelation of the image residuals during unstable phase is also shown in Figure 4.5 which shows a periodic behavior with a frequency ~ 26 showing the limit-cycle behavior during unstable combustion. For the results presented here in figures 4.4 and 4.5, the initial reference image is $k = 1$, but it can be seen from the time-series and the autocorrelation plots that the cyclic pattern is independent of k upto a phase shift. The reference image can also be a mean image calculated using a number of initial images without changing

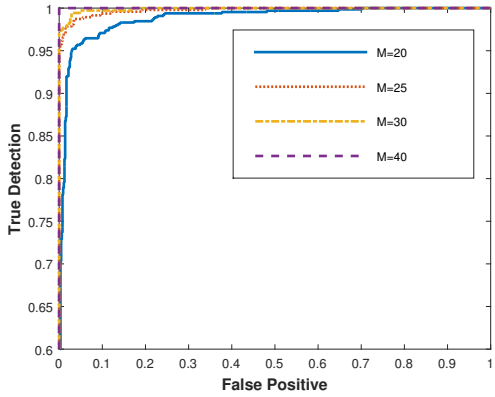


Figure 4.6: Performance of Gaussian Process on the residual of the sequential images. With sufficient memory, we are able to achieve perfect performance ($M = 40$); with reduction in memory, we see a reduction in performance.

the results. From figure 4.4, it is clear that these patterns can be used to classify or detect the two phases; however, as we have lumped a lot of information from images on to a single dimension – the 2-norm of the residual from reference image, we would need a longer sequence of data to detect and classify.

At this point, we would like to point out that the analysis presented in [130, 132, 133] using symbolic analysis of deep learning features and HOG features could potentially be done using the euclidean norm between the images. Using this intuition, our first baseline consists of learning a Gaussian Process on the norm of residual of images from a reference image (we choose $k = 1$ for reference image). The residual norm is first normalized by removing the bias and dividing by the variance (so that the GP is not sensitive to flame luminosity but rather the temporal nature of data). An isotropic squared exponential function is used as the covariance function for the GP which is given by the following equation.

$$k(x, x') = \sigma^2 \exp\left(\frac{-1}{2\ell^2}(x - x')^T(x - x')\right) \tag{4.2}$$

The corresponding free parameters to be inferred for the GP are just two – σ and ℓ associated with the covariance function and they are inferred using the expectation propagation [126]. The likelihood function is modeled as the error function and thus, there are no free parameters associated with the likelihood function. We train two GP models for stable and unstable cases. These two trained GP models are used to calculate the likelihoods $P(x|y)$ for the unstable ($y = 1$) and stable ($y = 0$) classes and use a Naive Bayes classifier to learn the optimal threshold

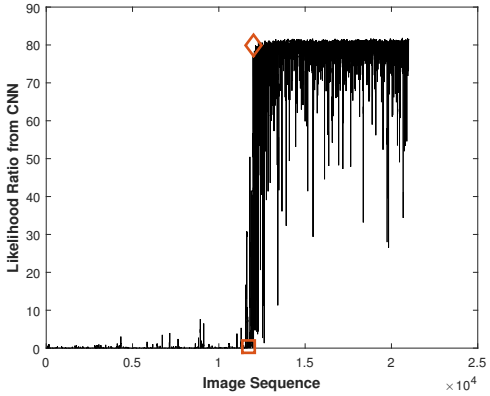
using the training data set. The results of the binary classification (on the test set) are shown as receiver operating curve (ROC) in figure 4.6. We see that the GP can model the limit cycle perfectly with enough memory; with reduction in memory, the performance degrades. If looked closely, the performance degrades when the memory is lower than the limit cyclic behavior of the image residuals (the frequency of the limit cycle could be seen from the the image residuals and their autocorrelation in Figures 4.4 and 4.5). This shows that a Gaussian process efficiently models the temporal structure in the image residuals. This performance is better than other published results in literature in [130, 132, 133]; the work in these papers ignores the fact that the combustion process locks on to a limit cycle and that it is this intrinsic dynamics driving the patterns in the time-series data. For example, in [132] the authors use windows of length 1500 to estimate a measure for anomaly detection. Moreover, in [133], the algorithm can't achieve perfect performance.

Even though we need much shorter memory than recently reported results, the disadvantage of this approach is that we need a relatively longer sequence of images to accumulate enough information to come to a conclusion. This happens as we are only looking at lumped changes in the images and any spatial difference between the images at stable and unstable phenomena is ignored. This motivates the use of a deep CNN, as with a deep enough network and large data-set we might be able to extract relevant spatial features by treating the data as iid and learning a deep CNN to get separability between the classes based on just single frames. Therefore, for extracting spatial features, even though the governing physical dynamics has a finite memory, we assume a memory-less system.

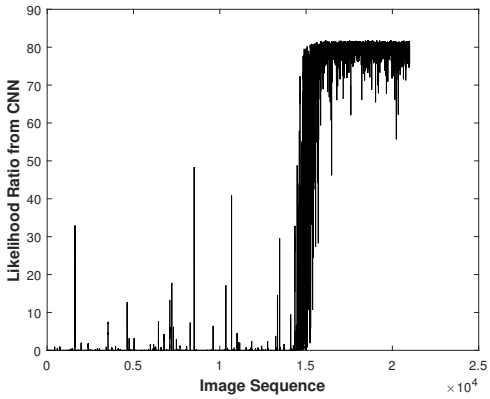
The labeled data is divided into train, validation and test data such that out of the total images, half or $36,000$ are used as training set, $14,400$ are used as validation set and the remaining $21,600$ are used as test data. An equal proportion of stable and unstable classes is kept in all the three sets. These images are then used to train the deep CNN. The architecture of CNN is the similar to that shown in Figure 4.1 with two convolution layers and two pooling layers. The number of kernels (or feature maps) are varied to study the effect of size of parameter set on the performance of the network. Also the number of units are varied to do a study of the effect of the width of the network for the particular problem. The CNN is terminated using early stopping to allow some regularization in the network. The

first convolutional layer has 10 kernels ($N_1 = 10$) and the second convolutional layer has 30 ($N_2 = 30$) kernels. The pooling layers perform a 2×2 max-pooling. The second convolutional layer is followed by a fully-connected layer with 80 units and another fully connected layer with 10 units. This is followed by an output layer which uses logistic regression for classification. A learning rate of 0.02 is used to train the CNN.

We are able to achieve fairly good classification performance with the deep CNN without considering the temporal dynamics in the image sequences. We get a test error rate of 4.68% and it took about 11.45 min to train this network (on an Nvidia geforce Titan X Graphics processing unit). This suggests that a sufficient spatial differences exist between single frames of combustion flames at the stable and unstable condition and the proposed deep network was able to extract those discriminating features from the images. We use the trained models to detect the onset of instability in transient data sets from experiment set 2 that was collected as the system gradually moved from stable to unstable (Table 4.3). This is done by making a Naive Bayes classifier where the likelihood of stable and unstable class are computed using the deep CNN and are used to calculate the likelihood ratio. As we can see in 4.7, the trained model detects the change as the combustion system gradually moves from stable to unstable behavior. It is noted that in both cases, the experiments were done where the system was stable initially and gradually becomes completely unstable;however the exact ground truth for this transition is not known (as the process is very fast). However, the flame images before and after the transition detected by the CNN model suggests that the onset of unstable behavior. To see this more clearly, we show some sequential images for figure 4.7a before and after the transition in the likelihood ratio for the CNN model; the changes in the flame images can be seen in figure 4.8. The images under the column *Before* in figure 4.8 correspond to the point denoted by a red square in figure 4.7a and the images under the column *After* correspond to the point shown by an orange diamond in figure 4.7a. Similar changes are also observed for figure 4.7b and thus they are not shown here. The images under the *After* column can be seen to break near the flame entrance (on left) indicting the onset of periodic blow-off (unstable behavior). Thus, we conclude based on the results of these two experiments that the CNN model is useful for early detection of the phase transition from stable to unstable. Another point to be noted is that the experiments were done at different



(a) Time Series of Likelihood ratio for Transient Data Case-1 in Table 4.3



(b) Time Series of Likelihood ratio for Transient Data Case-2 in Table 4.3

Figure 4.7: Performance of the trained deep learning algorithm on transient data conditions than the experiments done to train the CNN model; this indicates that CNN model is quite general and that are common features in spatial structure of flames, even at different operating conditions.

We propose a spatio-temporal filter where a Gaussian process model is used to model the temporal dynamics of the features extracted using the proposed deep CNN. We demonstrated earlier that the process has a definite temporal behavior with a finite memory. Therefore, modeling the temporal behavior is very desirable and the hope is to improve both detection performance and lead time to detection by explicitly modeling the temporal dynamics. To do this, we train a multi-dimensional Gaussian process using the output of the 2^{nd} fully connected layer of the deep CNN with 10 units. A 10-dimensional squared exponential covariance function with

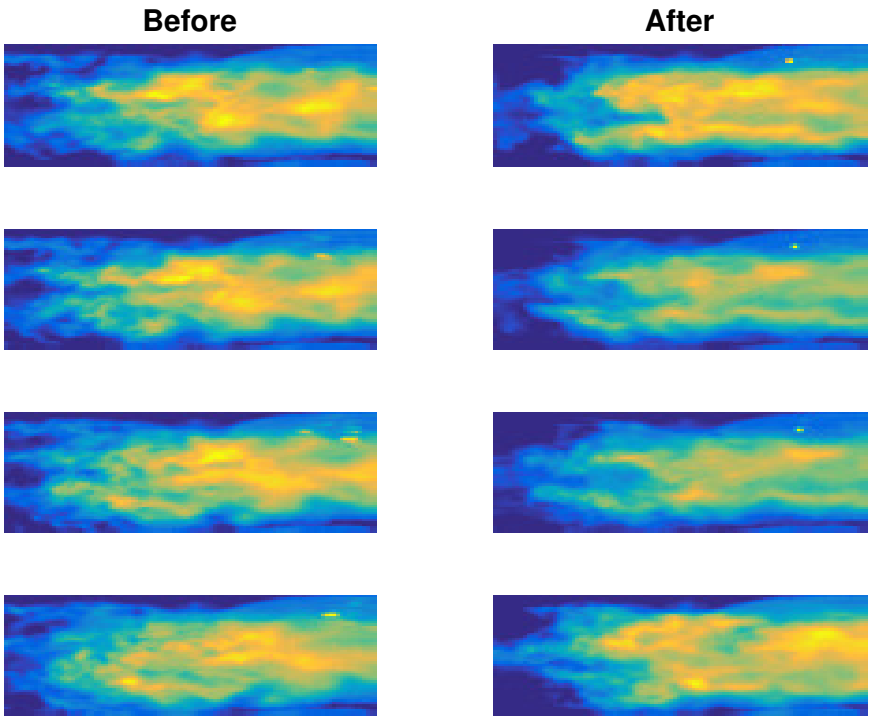


Figure 4.8: In this figure, we show typical snapshots of flames just before and after the switching detected by the CNN likelihood ratios for figure 4.7a (similar changes are observed for figure 4.7b). The flame enters on the left side and moves towards the right in every image. If seen closely, one can see the flame breaks near the entrance which is due to the periodic behavior that occurs during unstable phase.

automatic relevance detection determination (ARD) is chosen as the covariance function of the GP. The covariance function is parameterized as follows.

$$k(x, x') = \sigma^2 \times \exp(-(x - x')^T \Lambda^{-2}(x - x')/2) \tag{4.3}$$

where the matrix Λ is a diagonal matrix of dimension equal to the dimension of the input space and σ^2 is the signal variance. The likelihood function has the shape of a error-function (or cumulative Gaussian), which doesn't have any hyperparameters. Thus the total number of free parameters that need to be inferred from data are the hyperparemeters of the covariance function (i.e., the matrix Λ of size equal to the dimansion of the input space and the signal variance) which are inferred using the Laplace approximation or the expectation propagation [126]. In practice, it

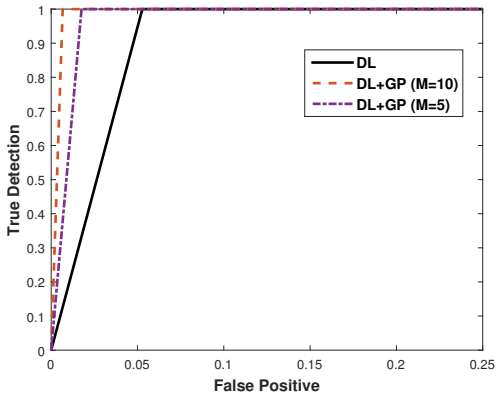


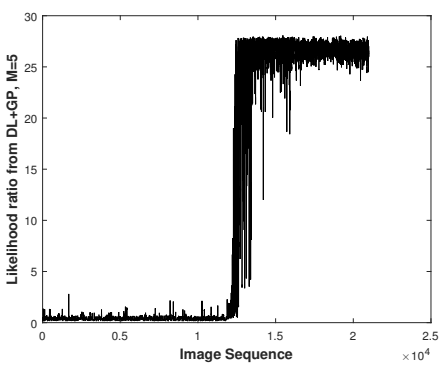
Figure 4.9: Performance of the trained deep learning algorithm and the Gaussian process on the deep learning features.

was found that the Laplace approximation was much faster as compared to the expectation approximation and thus, the results are presented using the Laplace approximation.

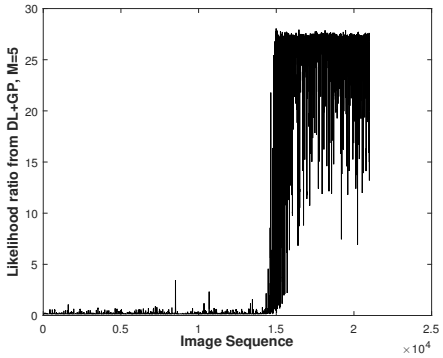
Similar to previous models, a Naive Bayes classifier is trained for the CNN-GP model using the training data set the ROC curve is calculated for the test set. For comparitive evaluation, al results are shown in figure 4.9. As can be seen, for M= 10, the GP trained on outputs of CNN gets a very small False Positive Rate of 0.7% when the true positive rate is 99.7%. Thus, we achieve near perfect classification with a temporal history of 10 frames as compared to 40 frames using GP alone on the image residuals or 300 frames in [132]. With the data being collected at 3 kHz, this leads to a perfect detection time of 3.3 ms. This is a big improvement over the earlier result in [132], where the a detection was made every 100 ms. Using only 5 frames or in 1.7 ms, the proposed approach is able detect the unstable phase with an accuracy of 99.7% and a false positive rate of 1.18%

Furthermore, the trained models are used to test the performance on the transient data using a memory of 5 frames. During test on the transient data, a window of length 5 is used for decision and every time a new frame is obtained, it replaces the last frame in the window (thus, we have 80% overlap. The likelihood ratios for the trained models are calculated using this sliding window. The results are shown in Figure 4.10 for the two transient conditions. The results are similar to those obtained by CNN model; however, it is noticed that it is more stable for case-1.

The current results are comparable to the time scale of the combustion process



(a) Time Series of Likelihood ratio for Transient Data Case-1 in Table 4.3



(b) Time Series of Likelihood ratio for Transient Data Case-2 in Table 4.3

Figure 4.10: Performance of the composite deep learning and Gaussian process algorithm on transient data

Table 4.4: Size of convolution filters

Image Size	Filter 1	Filter 2	Final Image Size
51×128	$15, 17$	$12, 13$	3×22

(which is on the order of ms). The results of all the numerical experiments and the different models trained are listed in Table 4.5. It is thus concluded that the proposed technique achieves very reliable and fast detection for combustion instability.

Table 4.5: Results of classification performance for baseline and **proposed** models. AUC= Area under curve (higher is better); FPR@99.7 is False positive rate at 99.7% detection rate (lower is better)

Model	Memory (M)	AUC	FPR@99.7
GP	20	0.9830	48.15%
GP	25	0.9918	19.15%
GP	30	0.9966	4.85%
GP	40	1.0000	0.00%
CNN	0	0.9750	5.28%
CNN-GP	5	0.9912	1.18%
CNN-GP	10	0.9987	0.67%

4.5 Conclusions and Future Work

Combustion instability still remains a puzzle for combustion researchers and the current state-of-the-art techniques heavily rely on physics-based models. The current analysis presented a data-driven spatio-temporal analysis of combustion flames using deep neural networks and Gaussian processes using high-speed images of flames during lean pre-mixed combustion. The present analysis presented several results on modeling of combustion process during stable and unstable phenomena.

In this chapter, we presented a framework for learning hidden, stationary dynamics in video data using deep convolutional networks and Gaussian processes. The main idea was to reduce the size of the parameter set of a spatio-temporal convolutional network by using a Gaussian process to capture the temporal sequence of the features extracted by the deep learning module. This study presented a rigorous machine learning-based approach to model the combustion phenomenon and suggests that statistical learning techniques could help understand and model the complex physical phenomenon better. The proposed framework was used to model the behavior of lean-premixed flames in a swirl stabilized combustor as the combustion process moves from stable to unstable through a sharp transient phase. The CNN alone is able to achieve fairly good classification performance; however, based on the current results it is concluded that adding Gaussian process allows the filter to be more generalizable as compared to the CNN alone. Based on the numerical experiments done in this chapter, it is also concluded that making a filter with appropriate depth (in the neural network) and memory (in the GP) allows to have generalizable data-driven models for the process which possibly can make correct predictions even with change of lots of associated variables in the process (e.g. mixing-length, fuel-air ratio, combustor geometry).

Using the proposed method with other sensor modalities like pressure to make a data-driven, multi-sensor, hybrid model for combustion instability is a possible topic for future research. Some problems like making predictions on changes in system state and relating to the criterion like Rayleigh's is also suggested as a topic of future research. Also, it seems that for systems with structured dynamics (e.g., engineering systems), using a Bayesian model-based filter will help add reasoning for better understanding of the process; however, a more through analysis is required to understand the benefits of such a hierarchical reasoning. While the results in this

chapter are encouraging, further investigations using data from multiple operating conditions are required to make a model which also makes predictions on statistical margins of stability for the process.

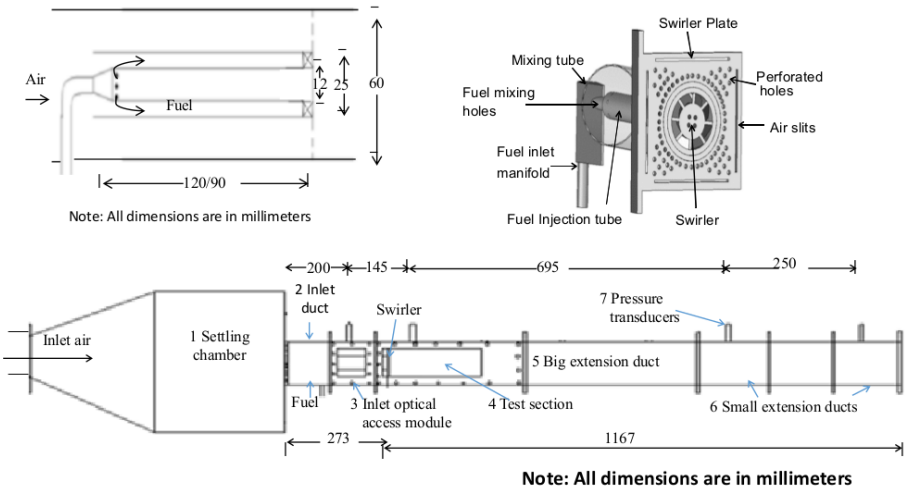


Figure 4.11: (a)Schematic of the experimental setup. 1 - settling chamber, 2 - inlet duct, 3 - IOAM, 4 - test section, 5 - big extension duct, 6 - small extension ducts, 7 - pressure transducers. The figure also shows the swirler plate geometry and the mixing tube of the swirler is also shown in detail.

Appendix

Experimental apparatus

In this section, we very briefly describe the key experimental procedure and apparatus for the completeness of the chapter. The experimental test bed has a swirl combustor with a swirler of diameter 30 mm with 60° vane angles (i.e., geometric swirl number of 1.28). Air is fed into the combustor through a settling chamber of diameter 280 mm with a sudden contraction leading to a square cross section of side 60 mm, providing an acoustically open condition with area ratio of 17. A mesh and honeycomb structure at the immediate downstream of the contraction assures uniform flow to the swirler. The combustor, shown in figure 4.11 consists of a 200 mm long inlet section, an inlet optical access module (IOAM) of length 100 mm, a primary combustion chamber of length 370 mm, and secondary duct of

the same length. The overall length of the constant area ducts was chosen to be 1340 mm. The fuel injection tube is coaxial to a mixing tube which has the same diameter as that of the swirler. The bypass air that does not enter the mixing tube passes through slots on the swirl plate. The slots on the fuel injection tube are drilled at designated distance upstream of the swirler, which dictates the extent of premixing between fuel and air. The larger this distance, more homogeneous the air-fuel mixture is. Two upstream distances of 90 mm and 120 mm were chosen for fuel injections during the experiments, where the former of the two denotes partial premixing and the later provides full premixing. The hi-speed images were collected through IOAM at 3 kHz using Photron High speed star with a spatial resolution of 1024×1024 pixels. Synchronized pressure data was acquired using piezoelectric transducers (PCB make) with resolution $225\,\mathrm{mV\,kPa^{-1}}$ at a location downstream of the IOAM. The data acquisition was triggered simultaneously using NI card and taken for a duration of 3 s yielding in a sequence of $9,000$ images for every operating condition. More details of the combustor could be found in [132].

Chapter 5

Anytime Algorithms for Policy-based Motion Planning of Robotic Systems

In earlier chapters we presented and analyzed some algorithms which can be used to estimate system states in engineering systems using statistical learning, which possibly can be used for information feedback in autonomous systems. Next, for intelligent decision-making we need decision systems which allow feedback-based control. Current state-of-the-art techniques for autonomous robotic systems rely heavily on open-loop planning where state-based feedback control is difficult to integrate. This problem is formulated and solved in this chapter. In this chapter we present policy-based motion planning of robotic systems with differential constraints. Traditionally, the motion planning literature in robotics community has been limited to open-loop trajectory based control which is followed by trajectory tracking online. Trajectory tracking-based control might be difficult for nonholonomic systems due to the differential constraints imposed by the system dynamics. We try to solve the problem of route finding and controller synthesis simultaneously by solving the related feedback problem. We present policy-based motion planning algorithm which is synthesized incrementally using sampling-based algorithms. In particular, we use the rapidly-exploring random graph (RRG) algorithm for nonholonomic systems with differential constraints to construct the state-space of the system. Asynchronous value iterations are used on the sampled state-space of the robot for synthesis of the incremental feedback controller. We show the convergence of the estimates to the optimal value function in continuous state-space to guarantee

asymptotic optimality. Some simulation results for feedback motion planning of Reeds-Shepp and Dubins car are provided using the proposed algorithms.

5.1 Motivation and Introduction

Informally speaking, given a robot with a description of its dynamics, a description of its environment, an initial set of states, and a set of goal states, the motion planning problem is to find a sequence of control inputs so as to guide the robot from the initial state to one of the goal states while avoiding collision in a cluttered environment. It is well-known that robotic motion planning is at least as difficult as the generalized piano mover's problem, which has been proven to be PSPACE-hard [135]. Many planning algorithms have been proposed, including, to name a few, discretized approaches such as A* [136]; continuous approaches involving navigation function and potential fields [136], and more recently, sampling-based geometric planning algorithms such as the rapidly-exploring random trees (RRT) [137], probabilistic roadmaps (PRM) [138] and their optimal variants RRT* and PRM* [139].

PRMs and RRTs (and their optimal variants PRM* and RRT* [139]) are arguably the most influential and widely-used sampling-based motion planning algorithms since the last two decades. They have been shown to work well in practice and possess theoretical guarantees such as probabilistic completeness. The robots, in most of the practical applications, have stringent differential constraint on their motion which in most of the cases need to be properly taken into account. Motion planning for nonholonomic systems with differential constraints has a rich history and a lot of work has been done in the last two decades [136, 140, 141]. However, the problem of motion planning for nonholonomic systems is still open in many aspects [142]. There are two main approaches which are considered for nonholonomic systems in literature [143]: (i) a decoupling approach, in which the problem is decomposed in steps of computing a geometric collision-free path (neglecting the differential constraint), smoothing the trajectory to satisfy the motion constraint and finally reparameterizing the path so that the robot can follow it, or (ii) a direct approach where the differentially constrained motion planning problem is solved in one shot. While the first method has its computational advantages, it suffers from inaccuracies and infeasibility of the solution [143]. The second approach is computationally difficult and is known to be PSPACE-hard (which

implies NP-hard) [142]. Sampling-based algorithms have received a large attention for planning of nonholonomic systems. In the light of these recent developments of the sampling -based algorithms for motion planning, there are two main bodies of research to improve these algorithms for planning under differential constraints: one trying to focus on developing steering functions for various classes of dynamical system using concepts from feedback control theory (see e.g., [144, 145]) and then, another focusing on improving the computational effectiveness of the algorithm using geometric planning (see e.g., [142, 146]).

In practical implementation, open-loop trajectories generated by RRTs are followed through a feedback controller. However, the feedback tracking of planned trajectory may be infeasible or expensive as we need an auxillary controller which can track the planned trajectory online. It necessitates feedback motion planning which explicitly takes into account the feedback tracking during the planning process. With a feedback controller the robot can lookup a table (which is created during controller synthesis) and assuming perfect state information, it can find the corresponding control input. This approach can simplify the control of nonholonomic robots. However, the feedback motion planning problem is challenging in various aspects and most of the computational issues involving feedback planning are still unexplored and open. Interested readers are referred to [143] for a comprehensive survey of the challenges in feedback motion planning.

Inspired by the success of sampling-based algorithms, there has been a lot of work on using these to develop feedback planners by using different metrics to select the best path. Some methods to solve for feedback planners are presented in [144, 147–150]. However, very little work has been done on using these algorithms to incrementally build the state-space of systems to solve for continuous time and space optimal and stochastic control problems for nonholonomic systems. Some examples to solve for POMDPs, stochastic control and approach evasion games could be found in [150–152]. A recent approach to feedback motion planning could be found in [153]. A feedback non-cooperative differential game formulation in multi-agent motion planning setting is presented in [18]. Our approach is very closely related to the value iterations-based approach in [18]; however, we use asynchronous value iterations, provide guarantees of convergence and also provide rigorous numerical simulation results. Our results on optimal performance are based on existence of steering functions between any two states of the system in

the collision-free space. One can choose approximate steering solutions to reduce complexity if optimality is not a big concern. As such, we expect our results to cater to rich set of robotic motion planning problems.

Contributions. This chapter presents an anytime algorithm for feedback motion planning of nonholonomic systems. In this work, we first leverage the existing geometric approaches for nonholonomic motion planning and asynchronous value iterations from Dynamic programming literature to synthesize the feedback motion planners for nonholonomic systems in an incremental fashion. In particular, the sampling-based algorithms are used to create a better refinement of the original continuous state optimal control problem with every iteration and the policy on the discretized space provides an incrementally refined estimate of the value function. Through analysis of the proposed algorithm, we show the convergence of the limiting value functions to the optimal value function for the continuous state-space motion planning problem of the robot. Some numerical simulations are provided for feedback motion planning of Dubins and Reeds-Shepp car using the proposed motion planning algorithm.

5.2 Problem Formulation

Consider a robot which is associated with a time-invariant dynamic system governed by the following differential equation:

$$\dot{x}(t) = f(x(t), u(t)), \tag{5.1}$$

where $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state of robot, and $u(t) \in \mathcal{U}$ is the control of robot. For system (5.1), the set of admissible control strategies for the robot is given by:

$$\mathcal{U} \triangleq \{u(\cdot) \ : \ [0, +\infty) \to U, \text{ measurable}\},$$

where $U \subseteq \mathbb{R}^m$. In particular, the focus of the current work is on the driftless control-affine systems of the form given by the following equation

$$\dot{x}(t) = \sum_{i=1}^{m} f_i(x(t))u_i(t), \tag{5.2}$$

where the available motions of the trajectory are linear combinations given by input control functions $u_i(t)$ and their corresponding actions at each point in the space $f_i(x)$. We make the following assumptions throughout the chapter.

Assumption 5.2.1 *1. The sets $\mathcal{X}$ and U are smooth manifolds. (We define manifolds later in the chapter)*

2. The function f is continuous in (x, u) and Lipschitz continuous in x for any u.

Let the open sets $\mathcal{X}_{\text{obs}}$ and $\mathcal{X}_{\text{goal}}$ denote the obstacle region and the goal region, respectively, for the robot. Define the obstacle free space as $\mathcal{X}_{\text{free}} := \mathcal{X} \setminus \mathcal{X}_{\text{obs}}$. Let $x_{\text{init}} \in \mathcal{X}_{\text{free}}$ be the initial state of the robot. The feasible motion planning problem is then to find a measurable control input $u : [0, \mathbb{T}] \to U$ such that the resulting trajectory $x(t)$ is collision free, i.e., $x(t) \in \mathcal{X}_{\text{free}}$ for all $t \in [0, \mathbb{T}]$ and the robot finally reaches the goal, i.e., $x(\mathbb{T}) \in \mathcal{X}_{\text{goal}}$. The optimal motion planning problem on the other hand is to find a measurable control input u such that the resulting trajectory x solves the feasible motion planning problem with minimum trajectory cost, where cost is defined using an appropriate metric.

Let us denote the trajectory of the system described in (5.2) as $\phi(\cdot; x, u) \triangleq \{\phi(t; x, u)\}_{t \geq 0}$ given the initial state x and the controls of u. Then, $\mathbb{T}(x, u)$ is defined as the first time when the trajectory $\phi(\cdot; x, u)$ hits the set $\mathcal{X}_{\text{goal}}$ while staying in $\mathcal{X}_{\text{free}}$ before $\mathbb{T}(x, u)$. More formally, we define

$$\begin{aligned} \mathbb{T}(x, u) \triangleq \inf\{\mathbb{T} \geq 0 | \phi(\mathbb{T}; x, u) \in \mathcal{X}_{\text{goal}}, \\ \phi(t; x, u) \in \mathcal{X}_{\text{free}} \forall t \in [0, \mathbb{T}]\} \end{aligned} \tag{5.3}$$

Then the problem that we are, interested, in this chapter is finding the optimal cost-to-go function if attainable. The value function for the optimal control problem is defined as follows.

$$v^\star(z) = \inf_{u \in U} \Big\{ \int_0^{\mathbb{T}} g(x(t), u(t)) dt + h(x(\mathbb{T})) | x(0) = z \Big\} \tag{5.4}$$

where, g and h are bounded and measurable functions, which are called the cost rate function and the terminal cost function respectively. We are interested in recovering the optimal cost-to-go function for a robot with nonholonomic dynamics

(see Equation (5.2)) in an anytime fashion using sampling-based algorithms. The goal is to be able to show asymptotic convergence of the optimal value function calculated on the sampled state-space to the optimal policy on the continuous state-space of the robot.

5.3 Background

In this section, we very briefly explain the essential concepts and results from differential geometry which are important to present the later contents clearly and unambiguously. Most of the things we present here are based on the detailed discussion available in [141], [146] and [142]. Interested readers are referred to the same for an in-depth discussion.

A set $M \subseteq \mathbb{R}^n$ is a smooth k-dimensional manifold, if, for all $x \in M$, there exists an open set $V \subset \mathbb{R}^n$ such that $V \cap M$ is diffeomorphic to an open set $V' \subset \mathbb{R}^k$. An example of a manifold is the configuration space of a Dubins car involving its planar position and orientation, i.e., $\mathbb{R}^2 \times \mathbb{S}^1$. A smooth curve on a manifold M is a function $\gamma : I \to M$ for some interval I that include the origin. Given a point $x \in M$, a vector $v \in \mathbb{R}^n$ is said to be a tangent vector of M at x, if there exits a smooth curve $\gamma : \mathbb{R} \to M$ such that $\gamma(0) = x$ and $\dot{\gamma}(0) = v$. The tangent space of M at x is defined as $T_x M := \{\dot{\gamma}(0) | \gamma \text{ is a smooth curve on } M \text{ and } \gamma(0) = x\}$. The arc length of a path $x(t)$ is defined as

$$\ell(x) := \int_0^T \|\dot{x}(t)\| dt$$

where $\|\dot{x}(t)\| = \|\dot{x}(t)\|_2$ is computed using the standard Euclidean inner product on the tangent space of M. The arc length function induces a sub-Riemannian distance d on M, defined for $x_1, x_2 \in M$ as $d(x_1, x_2) := \inf_x \ell(x)$, where the infimum is taken over dynamically feasible trajectories $x(t)$ connecting x_1 and x_2. As such, $d(x_1, x_2)$ is infinite if there is no curve connecting the two points. Now, we can define the sub-Riemannian ball analogous to the standard Euclidean ball as $B(x, \epsilon) = \{\tilde{x} \in M | d(x, \tilde{x}) \leq \epsilon)\}$. This metric is used for the analysis of nonholonomic systems.

The ball-box theorem is used to approximate the sub-Riemannian balls using Euclidean balls. The concept of priviledged or linearly adapted coordinates (denotes

as z_i) is used to define a psuedonorm at any point p in the manifold and this allows us to approximate the sub-Reimannian balls (we skip the details; please see [154]). Given the priviledged coordinates z_i, we can now define the psuedonorm at p as $\|z\|_p := \max\{|z_1|^{1/w_1}, \ldots |z_n|^{1/w_n}\}$. Now, using this norm, we can define the sub-Riemannian w-weighted box at any point p as $\text{Box}^w(p, \epsilon) = \{z \in \mathbb{R}^n : \|z\|_p \leq \epsilon\}$. Then, the ball-box theorem [155] states that the set of states that can be reached by a horizontal path starting from p contains a weighted box of radius $c(p)\epsilon$ and is contained in a weighted ball of radius $C(p)\epsilon$. More formally, for a system of priviledged coordinates $z_1, \ldots, z_n$ at p, $\exists$ constants $c(p)$, $C(p) > 0$ such that $\text{Box}^w(p, c(p)\epsilon) \subset \text{B}(p, \epsilon) \subset \text{Box}^w(p, C(p)\epsilon)$ for all $\epsilon < \epsilon_0$ (where $\text{B}(p, \epsilon)$ is the standard Euclidean ball). This theorem gives the structure as how the distances behave in sub-Riemmanian manifold which has been earlier used for steering of nonholonomic systems [154]. This sub-Riemmanian ball is used in the proposed algorithms for finding the nearest neighbors during graph construction for the nonholonomic system as it reduces the number of possible connections and thus improves the computational efficiency. We also assume that the nonholonomic system is regular so that there exist bounds $0 < c_{\min} < c(p) < C(p) < C_{\max} < \infty$ which is again used to approximate the size of balls for finding nearest neighbors (see section 5.4) [142].

5.4 The Incremental Feedback Policy Algorithm

In this section, we describe the algorithms we use for the incremental policy synthesis. We leverage the algorithm presented in [139, 142, 146] along with value iterations to incrementally estimate the cost-to-go function from every point to the goal set. In the following, the graph after k iterations of the algorithm is denoted by $G_k = (V_k, E_k)$ where V_k is the set of vertices and E_k is the set of edges.

5.4.1 Primitives

- `Sampling`: The `Sample` procedure returns uniformly random samples from set the maximally integrable manifold of the dynamics (which in the case when the system is controllable is the manifold X).

- `Steer:` Given two states x, y, the `Steer` procedure returns a terminal time T and a dynamically feasible trajectory connecting x to y. We assume that this procedure satisfies the topological property required for feasible steering solution for nonholonomic systems. According to this condition, for any $\epsilon > 0$, there exists some real number $\eta_\epsilon > 0$ such that for any two states $x_1, x_2 \in X$ with $\|x_1 - x_2\| \leq \eta_\epsilon$, we have $\|x_1 - \texttt{Steer}(x_1, x_2)(t)\| \leq \epsilon$ for all $t \in [0, T]$. It is noted that this is different than the usual steering presented in [139] as in our case it is non-trivial to move the system to a state with partial optimal cost. Thus, we attempt a full connection to the sampled state and thus add the sampled state to the graph.[1]

- `Nearest neighbor:` Given a state x and a finite set V of states, the `Nearest` procedure returns the state in V that is closest to x; i.e., $\texttt{Nearest}(S, x) \triangleq \text{argmin}_{y \in V} \|y - x\|$. For the problems discussed in this chapter, we use the metric defined in section 5.3.

- `Near vertices:` Given a state x, a finite set V , the `NearVertices` procedure returns the states in V where each of them is r_k-close to x at the kth iteration of the algorithm where $r_k = \beta\left(\frac{\log(|V_k|)}{|V_k|}\right)^{1/D}$ where $|V| = k$. In particular, this procedure returns the subset of V with states close to the state x in the following sense[2]

$$\texttt{Near}(V, x) := V \bigcap \text{Box}^w\left(x, \beta\left(\frac{\log(|V|)}{|V|}\right)^{1/D}\right)$$

  This is different from planning in Euclidean space as nearest neighbors for a candidate point are computed by searching inside the weighted Euclidean box which is used to approximate the sub-Riemannian ball. We also add the corresponding inputs found by the `Steer` procedure to the input sets of the corresponding nodes (this reduces the number of possible neighbors as opposed to the naive Euclidean balls).

[1] The optimality guarantees remain unaffected [139].

[2] $\beta = \frac{C_{\max}}{c_{\min}}(1+\eta)^{1/D}(\frac{\mu(X_{\text{free}})}{D})^{1/D}$, where $\eta > 0$ is a constant, $\mu(X_{\text{free}})$ represents the volume of the free manifold and $D = \sum_i w_i$ and coincides with the Hausdorff dimension of $\mathcal{H}$

Algorithm 3: Incremental Feedback Policy Algorithm

$V(0) \leftarrow x_{\text{init}}$;
$E(0) \leftarrow \emptyset$; $k \leftarrow 1$;
if $d(\texttt{Nearest}(V_k, x_{\text{goal}}), x_{\text{goal}}) > \eta$ **then**
  $x_{\text{rand}} \leftarrow \texttt{Sample}(\mathcal{X})$;
  $x_{\text{nearest}} \leftarrow \texttt{Nearest}(V, x_{\text{rand}})$;
  $(x_{\text{new}}, u_{\text{new}}, T_{\text{new}}) \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$;
  if $\texttt{ObstacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $V_k \leftarrow V_{k-1} \cup \{x_{\text{new}}\}$;
    $E_k \leftarrow E_{k-1} \cup (x_{\text{nearest}}, x_{\text{new}})$;
    $S_k \leftarrow V_k$;
  else
    while $k < K$ **do**
      $x_{\text{rand}} \leftarrow \texttt{Sample}(\mathcal{X})$;
      $S_k \leftarrow \{x_{\text{rand}}\}$;
      $G_k \leftarrow \texttt{Extend}(G_{k-1}, x_{\text{rand}})$;
      for $x \in G_{k-1}$ **do**
        $\tilde{v}_k(x) \leftarrow v_{k-1}(x)$;
        if $\text{Flag}(x) == P$ **then**
          $S_k \leftarrow S_k \cup \{x\}$;
      for $x \in \mathcal{X}_{free} \setminus G_{k-1}$ **do**
        $\tilde{v}_k(x) \leftarrow v_{k-1}(\text{argmin}_{y \in G_{k-1}} \|x - y\|)$
      $(v_k, \pi_k, \text{Flag}) \leftarrow \texttt{ValueIteration}(G_k, \tilde{v}_k, \text{Flag})$

- `Collision Check`: Given a trajectory $x : [0, T] \to X$, the $\texttt{ObstacleFree}(x)$ returns true if x avoids collision with obstacles, i.e., $x(t) \notin X_{\texttt{obs}}$ for all $t \in [0, T]$, and returns false otherwise.

To find a quick initial solution, the simple RRT (unidirectional or bidirectional) is first used to connect the goal to the initial point. This also reduces a lot of unnecessary initial connections without affecting the optimality guarantees. Once, the tree reaches the goal, the `Near` algorithm and asynchronous value iterations are invoked to refine the solution. Specifically, the robot samples a point $x_{\texttt{rand}}$ from the free space and adds a point to its vertex set V_{k-1} by moving towards that point. This is achieved by solving the steering problem for movement between two points in the collision free space of the robot. Along with adding the point to its vertex set, we also add the corresponding input with the estimated cost of the same in the input set of the robot. Doing this we incrementally build the input set as well as the state-space of the robot. The robot keeps on building the graph till it reaches

Algorithm 4: The `Extend` Procedure

$V \leftarrow V_{k-1}$;
$E \leftarrow E_{k-1}$;
$x_{\text{nearest}} \leftarrow \texttt{Nearest}(V, x_{\text{rand}})$;
$(x_{\text{new}}, u_{\text{new}}, T_{\text{new}}) \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$;
▷ u_{new} is added to $U_k(x_{\text{new}})$
if $\texttt{ObstacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
  $\mathcal{X}_{\text{near}} \leftarrow \texttt{NearVertices}(V, x_{\text{new}}, |V|)$;
  $V \leftarrow V \cup \{x_{\text{new}}\}$;
  for $x_{\text{near}} \in \mathcal{X}_{\text{near}}$ **do**
    $(u_{\texttt{near}}, T_{\texttt{near}}) \leftarrow \texttt{Steer}(x_{\texttt{near}}, x_{\texttt{new}})$;
    $(u_{\texttt{new}}, T_{\texttt{new}}) \leftarrow \texttt{Steer}(x_{\texttt{new}}, x_{\texttt{near}})$;
    if $\texttt{ObstacleFree}(x_{\text{near}}, x_{\text{new}})$ **then**
      $E \leftarrow E \cup \{(x_{\text{near}}, x_{\text{new}}), (x_{\text{new}}, x_{\text{near}})\}$;
    $\text{Flag}(x_{\text{near}}) = P$;
▷ u_{near} is added to $U_k(x_{\text{near}})$ and u_{new} is added to $U_k(x_{\text{new}})$
return $G = (V, E)$

Algorithm 5: `ValueIteration`

for $x \in S_k$ **do**
  $v_k(x) \leftarrow \min_{u \in U_k(x)} \Big(g(x,u) + \alpha_k \tilde{v}_k(x + f(x,u)) \Big)$;
  $u_k(x) \leftarrow \text{argmin}_{u \in U_k(x)} \Big(g(x,u) + \alpha_k \tilde{v}_k(x + f(x,u)) \Big)$;
  $\text{Flag}(x) = 0$
for $x \in G_k \setminus S_k$ **do**
  $\text{Flag}(x) = \text{Flag}(x) + 1$;
  $v_k(x) = \tilde{v}_k(x)$
return $(\pi_k, v_k, \text{Flag})$

the goal set. Once the graph reaches the goal set, the value iteration updates begin (every node in the goal set is given a cost of 0). At every iteration, we perform a single value iteration update to get a better estimate of the cost-to-go from every node to the goal and thus, also find the corresponding optimal policy on the graph using the Bellman operator which is defined in the following equation:

$$T_k(\tilde{v}_k)(x) = \min_{u \in U_k(x)} \Big(g(x,u) + \alpha_k \tilde{v}_k(x + f(x,u)) \Big) \tag{5.5}$$

where, T_k is the Bellman operator defined on G_k, $\tilde{v}_k$ is the initialization of the values on G_k and $\alpha_k \in (0,1)$ is the discount factor for the kth iterate of the algorithm.

The discount factor is chosen such that $\lim_{k\to\infty} \alpha_k = 1$. $U_k(x)$ is the input set for node x which has been built by solving the steering function. Hence, using the proposed algorithms we maintain an estimate of the cost-to-go function for the robot and we refine it incrementally as we sample more points and perform more updates of the value iteration. The algorithm for the incremental policy synthesis are presented in Algorithms 3 through 8. Asynchronous value iterations are used for updating the value functions where every node is updated after a finite number of iterations; this is done reduce the computations [152]. The convergence of the value functions remains unaffected as long as every node is visited after every finite number of iterations of the algorithm (see next section 6.4 for the proof). In the asynchronous iterations, we maintain a counter for every node; a node is updated if it is the direct parent of a new sample or if it has not been updated since the last P updates. This helps save a lot of computations but doesn't affect the convergence guarantees. The algorithms show the asynchronous update rule.

5.5 Performance Analysis

In this section, we analyze and present results for the performance of the underlying algorithms. We show the asymptotic convergence of the value functions generated by the incremental feedback-policy algorithm presented earlier. The norm used in the following material in this section is defined as follows, $\|v\|_S = \sup_{v\in S} \|v\|$. The optimal value function for the robot in the original continuous state-space is denoted by $v^\star$ while the optimal value function is denoted by $v_k^\star$ for the graph G_k. Also, the definition of v_k is extended to $\mathcal{X}_{\text{free}}$ by the following interpolation: for $x \in G_k, \tilde{v}_{k+1}(x) = v_k(x)$ and $x \in \mathcal{X}_{\text{free}} \setminus G_k, \tilde{v}_{k+1}(x) = v_k(\text{argmin}_{y\in G_k} \|x - y\|)$. We show that the value functions estimated using the asynchronous value iterations converge to the optimal value functions for a finite P (see Algorithm 3).

Theorem 5.5.1 *For all $x \in G_k$, $v_k(x)$ is the value function (or the cost-to-go) for state x after k iterations. Then the following are true.*

1. $\lim_{k\to\infty} \|v_k - v_k^\star\|_{G_k} = 0$
2. *The sequence v_k converges to $v^\star$ point-wise for x_{init} i.e., it holds that*

$$\lim_{k\to\infty} |v^\star(x_{\text{init}}) - v_k^\star(x_{\text{init}})| = 0$$

Proof 5.5.1 *Consider the following:* $G_{k+1} \subseteq \bigcup_{\tau=0}^{P} S_{k+1+\tau}$ *(P is finite). Then we define an operator which is defined by the composition of Bellman operator defined earlier in Equation (5.5),* $\tilde{T}_P = T_{k+P} \circ T_{k+P-1} \circ \circ T_{k+1}$*. Then,* $\tilde{T}_P$ *is monotonic on* G_{k+1} *as it is the composition of monotonic operators. Consider,* $c = \|\tilde{v}_k - \tilde{v}^\star_{k+1}\|_{G_{k+1}}$*. Then the following is true point-wise on* G_{k+1}

$$|\tilde{v}_{k+P} - \tilde{v}^\star_{k+1}| \leq \zeta_{k+1}\zeta_{k+2}\dots\zeta_{k+P}\|\tilde{v}_k - \tilde{v}^\star_{k+1}\|_{G_{k+1}} \tag{5.6}$$

where $\zeta_{k+n} = 1$ *if* x *is not updated in iteration* $k+n$*, else it is equal to* $\alpha_{k+n} = e^{-\kappa_{k+n}}$ *. Then, taking the supremum on the LHS of equation (5.6) we get* $\|\tilde{v}_{k+P} - \tilde{v}^\star_{k+1}\|_{G_{k+1}}$*. On the RHS, taking the supremum we get* $\alpha_{k+1}\|\tilde{v}_k - \tilde{v}^\star_{k+1}\|_{G_{k+1}}$*. Thus we get the following inequality:*

$$\|\tilde{v}_{k+P} - \tilde{v}^\star_{k+1}\|_{G_{k+1}} \leq \alpha_{k+1}\|\tilde{v}_k - \tilde{v}^\star_{k+1}\|_{G_{k+1}} \tag{5.7}$$

Now, we define a subsequence $\tilde{v}_{k+mP} = \hat{v}_{k+m}$*. We assume that* k *is large enough that the subsequnce is well defined. Then, from the inequality (5.7) we get,* $\|\hat{v}_{k+1} - \tilde{v}^\star_{k+1}\|_{G_{k+1}} \leq \alpha_{k+1}\|\hat{v}_k - \tilde{v}^\star_{k+1}\|_{G_{k+1}}$*. Then using the triangle's inequality as above in Theorem 5.5.1, we get the following.*

$$\begin{aligned}
\|\hat{v}_{k+1} - \tilde{v}^\star_{k+1}\|_{G_{k+1}} &\leq \alpha_{k+1}\|\hat{v}_k - \tilde{v}^\star_{k+1}\|_{G_{k+1}} \\
&= \alpha_{k+1}(\|\hat{v}_k - \tilde{v}^\star_k + \tilde{v}^\star_k - \tilde{v}^\star_{k+1}\|_{G_{k+1}}) \\
&\leq \alpha_{k+1}(\|\hat{v}_k - \tilde{v}^\star_k\|_{G_{k+1}} + \|\tilde{v}^\star_k - \tilde{v}^\star_{k+1}\|_{G_{k+1}}) \\
&= \alpha_{k+1}(\|\hat{v}_k - \tilde{v}^\star_k\|_{G_k} + \|\tilde{v}^\star_k - \tilde{v}^\star_{k+1}\|_{G_k})
\end{aligned}$$

Hence, following Theorem 5.5.1, we get the expression: $\delta_{k+1} \leq \alpha_{k+1}\big(\delta_k + \gamma_k\big)$*, where* $\delta_k = \|\hat{v}_k - \tilde{v}^\star_k\|_{G_k}$ *and* $\gamma_k = \|\tilde{v}^\star_k - \tilde{v}^\star_{k+1}\|_{G_k}$*. The convergence of the subsequence* $\hat{v}_k$ *then follows from the steps shown in Theorem*4.1 *in [18]. The convergence of* $\tilde{v}_k$ *follows from the convergence of* $\hat{v}_k$*. The convergence of* v_k *follows from the convergence of* $\tilde{v}_k$*.*

(2) *This follows from the fact that* $\lim_{k\to\infty} v^\star_k(x_{\text{init}})$ *is the cost of the optimal trajectory obtained by RRG for a system with differential constraints from the initial point* x_{init}*. DPRM* introduced in [142] is a batch version of RRG algorithm (which is incremental) with the* `Near` *procedure defined in the last section (i.e.,* $r_k =$

$\beta\left(\frac{\log(|V|)}{|V|}\right)^{1/D}$*). Then, following Theorem* 6.4 *in [142] which guarantees asymptotic optimality of DPRM*, the RRG algorithm described here is asymptotically optimal. Let us denote the cost of the trajectory found by the RRG algorithm from initial point* x_{init} *as* $\text{Cost}_{\text{RRG}}(x_{\text{init}})$ *. Then, the estimate of the value function converges to the optimal cost of the trajectories found by RRG for* x_{init}*. To see this more clearly, consider the fact that* $\lim_{k\to\infty} v_k^\star(x_{\text{init}})$ *is the minimum achieved among all feasible policies, hence we have* $\lim_{k\to\infty} v_k^\star(x_{\text{init}}) \leq \text{Cost}_{\text{RRG}}(x_{\text{init}})$ *(where RRG can incorporate differential constraints). However, since RRG with differential constraint gives the minimum cost trajectory, hence* $\lim_{k\to\infty} v_k^\star(x_{\text{init}}) \geq \text{Cost}_{\text{RRG}}(x_{\text{init}})$*. Thus we have* $\lim_{k\to\infty} v_k^\star(x_{\text{init}}) = v^\star(x_{\text{init}})$.

Remark 5.5.1 *The first part of this theorem shows convergence of the value function to the optimal value function on the sampled graph when the state-space is incrementally built and a single update of the Bellman equation is done on the sampled graph in an asynchronous fashion. The second part argues the asymptotic convergence of the value function estimated on the sampled graph to the optimal value functions. We show that the limiting function approaches the optimal value function in the continuous state-space.*

5.6 Numerical Results and Discussion

In this section, we present results of numerical experiments using the proposed algorithm for various different systems. For simplicity and clarity of presentation, we first present the results for a point mass with trivial dynamics. Then we will present results for Reeds-Shepp and Dubins car dynamics which are well known examples of driftless affine system in the configuration space $\mathbb{R}^2 \times \mathbb{S}^1$ [136]. In the first example, the goal is a square region with sides of length 5 and we show the path obtained by the algorithm from different initial points and the anytime nature of the algorithm (i.e., the quality of the path improves with more number of iterations). In all of the examples for the point mass, the goal is fixed at coordinate $(25, 85)$ and the plots show the path obtained from different initial points to the goal region after 1500 iterations. And we also show the improvement in the resolution of the policies after another 1000 iterations (see Figure 5.1d).

The Reeds-Shepp car dynamics is given by the following equations.

$$\dot{x} = u_1 \cos\theta$$
$$\dot{y} = u_1 \sin\theta$$
$$\dot{\theta} = u_2$$

The tangent space for this system is spanned by the vector fields $g_1(x) = (\cos\theta, \sin\theta, 0)$, $g_2(x) = (0, 0, 1)$ and their Lie bracket $[g_1(x), g_2(x)] = (\sin\theta, -\cos\theta, 0)$. The priviledged coordinates at any point is defined by the same vectors with associated weight (1,1,2) (thus $D = \sum_{i=1}^{k} w_i = 4$, which is used in the `Near` procedure in the algorithm). For further simplification, the car has only two gears i.e., it is allowed at move at unity speed forward or backward. The control set for a radius of curvature R is thus given by, $U = \{-1, 1\} \times [-1/R, 1/R]$. The solution for the Reeds-Shepp car is found exactly by searching for the nearest neighbor in the sub-Riemannian ball induced by its dynamics. This is achieved by solving two-point boundary value problem for set of neighbors of each node sampled during graph construction. In the all the examples presented, P was chosen to be 100 and $x_{\text{init}} = (3, 3, \pi/9)$.

The analytic solutions for above described Reeds-Shepp steering problem is known for connecting any two states in collision-free configuration space of the robot [156]. The optimal solution is the time-optimal trajectory in the set of forty-six candidate solutions possible while trying to connect any two states in the configuration space. Even though the optimal solution to steering is known for Reeds-Shepp system, finding the optimal solution is still computationally intensive as it requires searching the set of time-optimal trajectories generated for any two states sampled during state-space construction of the system. Some tricks like symmetry about the origin can be used to speed-up the process or may be a classification scheme could be run in parallel by training a machine learning algorithm to quickly find the time-optimal trajectories and corresponding costs. However, as the optimal trajectories can be found, we calculate the exact sub-Riemannian ball while using the `Near Vertices` function described in Algorithm 4. It is noted that this step is critical to guarantee optimality of the value functions and ensures feasibility of the solutions.

In Figure 6.1, we show the estimated cost-to-go function obtained for the Reeds-Shepp dynamic system using the proposed algorithm with $R = 1$. The plots in

the figure shows the sub-Riemannian distance from the state-space to the goal for the robot. As the value function is a mapping from $\mathbb{R}^2 \times \mathbb{S}^1 \to \mathbb{R}_+$ hence we get a volume representing the cost-to-go from the state-space of the car to the goal. In Figure 5.2a, we show the variation in value functions in x, y and θ. To see things more clearly, in Figure 5.2b we show the estimated cost-to-go on a plane where the orientation of the car is parallel to the orientation at goal ($\theta_{\text{goal}} \sim 58^\circ$). The goal set of the car is a sub-Riemannian ball of radius 1.5 around the point $(15, 15, \pi/9)$ (however, during implementation the point where we can reach first inside the ball is assigned as the new goal). The shape of the level sets around the goal depict that the cost-to-go is higher in the transverse direction when compared to that along the longitudinal direction parallel to the orientation of the goal; this is for the Lie Bracket maneuvers that the car has to do to move to state in the transverse direction (as it has a non-zero turn radius). To save computations, the asynchronous updates of the value iteration was done where a node is updated if its not visited during the last P iterations or it is a parent of new sampled node or the new sampled node. As the asymptotic optimality guarantees remain unaffected by the initial estimates of the value functions, we initialize the graph by first starting with making a tree from the initial point to the goal using the simple RRT algorithm (i.e., the `Near` procedure is not invoked; this avoids a lot of unnecessary connections). Once, we reach the goal we start adding more connections using the `Near` procedure and perform value iterations to estimate the cost-to-go. One can also interpret these levels sets as the sub-Riemannian balls which are reachable by the Reeds-Shepp dynamic system from the goal point in a certain time interval.

In yet another simulation, the aforementioned environment includes three obstacles: all three obstacles are 4 m $\times$ 4 m in size, and their lower left vertexes have the coordinates $(3, 20), (7, 7), (20, 3)$, respectively. The goal configuration remains $(15, 15, \pi/9)$. The robot has the following geometric properties: length is 4.85 m, width 1.95 m, and the minimum turning radius is 4.321 m. The results of the path planning algorithm are illustrated in Figs. 5.3-5.5. Simulation demonstrates that the algorithm can compute a collision-free kinematic admissible path for the Reeds-Shepp model. The red box in these plots denotes the car.

In Figure 5.6, we show the estimate of the value functions obtained using the proposed algorithm for the Dubins car whose dynamics also lies in the $\mathbb{R}^2 \times \mathbb{S}^1$ space and is described by the same set of equations as the Reeds-Shepp car; however,

it can only move forward. As a result, the Dubins car is not time-reversible (i.e., $U = \{1\} \times [-1/R, 1/R]$). We follow the exact same procedure as followed for the Reeds-Shepp car to estimate the value functions; we find the distance between two points by finding the minimum in the set of six possible trajectories [136, 156] to connect two points. The goal set in this case is also the sub-Riemannian ball of radius 2 around the point $(15, 15, \pi/9)$ and $R = 1$ (again the goal is the point where we reach first inside the ball) . In Figure 5.6a we show the estimated cost-to-go on a plane where the orientation of the car is parallel to the orientation at goal ($\theta_{\text{goal}} \sim 32^\circ$). The estimate of the value functions show the more complicated level sets obtained for the Dubins car as it can not move backwards. Unlike the Reeds-Shepp system, the level sets are not symmetric about the origin; moreover, they are discontinuous. In figure 5.6b, we show the value functions on a plane where the initial orientation of the car is π.

5.6.1 Extensions

So far we have considered systems where the analytical solution to two-point boundary value problem is known and thus can be calculated. However, the feedback loop to estimate the value functions can be used to estimate value functions for a large class of systems and thus approximate the policies for a large class of motion planning problems. For example, a lot of work has been done to approximately solve the steering functions (see for example [145, 154, 157]). The algorithm presented in this chapter can be used to find the corresponding policies.

5.6.2 Experimental Validation on a Robotic Test Bed

The proposed incremental policy algorithm has been validated by implementation on a RC-car designed by in the robotics lab at Penn State. The platform is shown in Figure 5.7. A very brief discussion of the platform follows next.

The RC-car is built up on a factory finished chassis of the Tamiya TT-02 4WD fully independent suspension on-road chassis (more details could be found here (http://www.tamiyausa.com/items/radio-control-kits-30/rc-semi-assembled-chassis-35900/rc-tt02-chassis-57984). The experiments were done to demonstrate the capability of the proposed algorithms for disturbance rejection due to state-based feedback. In the following, we present comparison of the proposed feedback

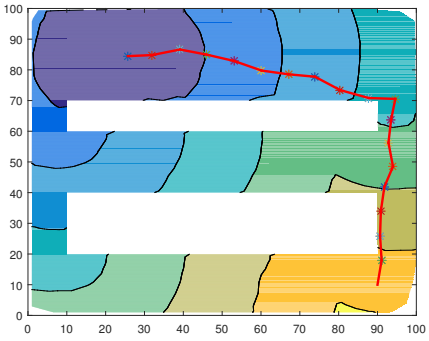
algorithm with the state-of-the-art algorithm RRT*.

In Figure 5.8, we show the implementation of the RRT* with PID control for trajectory following. As is shown in the figure, if the system is perturbed with significant disturbance, a re-planning is required for the system to avoid collision (for the very nature of trajectory following). A video of the experiments could be found here `https://www.youtube.com/watch?v=yRyNyXHkAXM` and `https://www.youtube.com/watch?v=RUhw1MWXqPw`, where we show that the RRT* with PID control collides with obstacles up on perturbed from the planned trajectory. In Figure 5.9, we show that the feedback nature of the algorithm allows the robot to use the current state information to use the optimal control from the state-space. The experiment video where we show the robot being perturbed from its position and still can find a collision-free path to the goal could be found here `https://www.youtube.com/watch?v=zcWrSm2oHao` and `https://www.youtube.com/watch?v=q6GGLtScxiU`. As the robot has a collision-free policy from the state-space, it reaches the goal efficiently (optimally) even from the perturbed state.

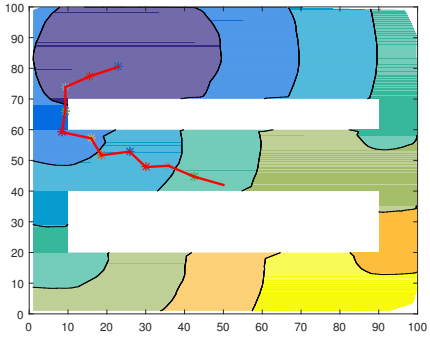
5.7 Conclusions and Future Work

Feedback motion planning is useful for various robotic missions in the presence of disturbances and/or process noise where the robot's state could be reconstructed using sensors. Synthesis of feedback controllers can simplify control of mobile robots as we can use state-based feedback instead of trajectory following. In this chapter we presented an anytime computation algorithm for feedback motion planning of nonholonomic systems. Using optimality results on sampling-based open-loop trajectory planning algorithms and dynamic programming, we presented an algorithm which can be used for feedback motion planning of nonholonomic systems and guarantees asymptotic optimality of the value functions. Some results of numerical simulations with Reeds Shepp and Dubins dynamic system are presented where the time-to-go costs are recovered using the proposed algorithm. We discussed an asynchronous update rule of the value functions for computational efficiency of the algorithms and proved that it retains asymptotic optimality guarantees.
Solution to problems like minimum complexity (instead of minimum time) feedback controllers for nonholonomic systems is a topic of future research. Also, we would

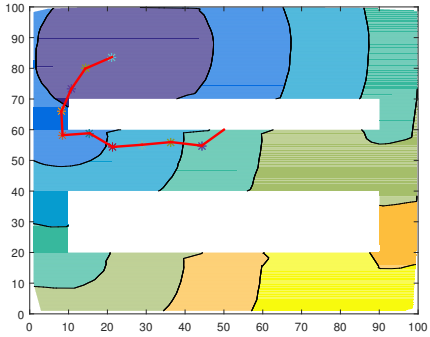
like to study feedback solutions for motion planning of nonholonomic systems in the presence of moving obstacles and uncertainty in the available environment maps.



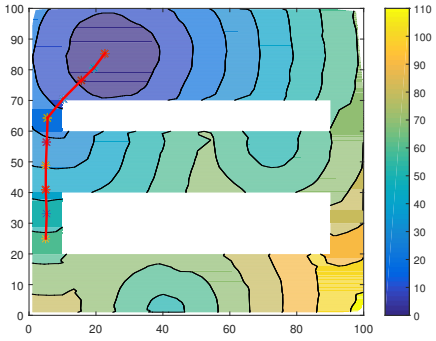
(a) The estimated cost-to-go function for the point mass



(b) The estimated cost-to-go function for the point mass

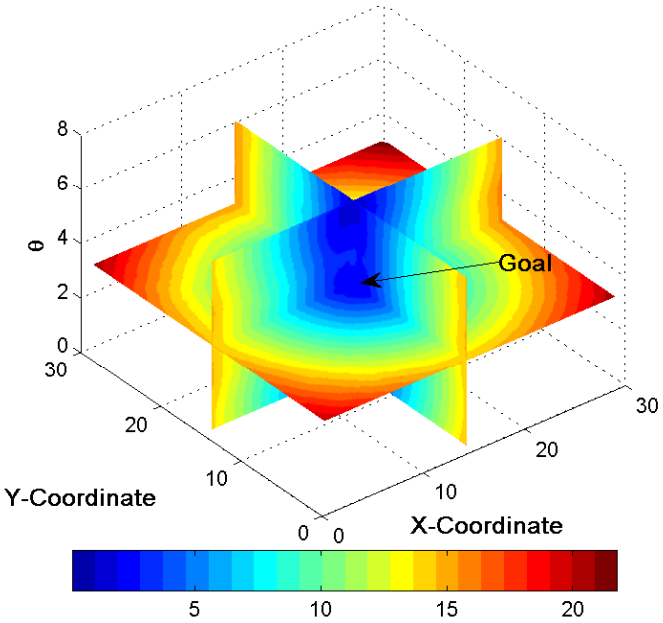


(c) The estimated cost-to-go function for the point mass

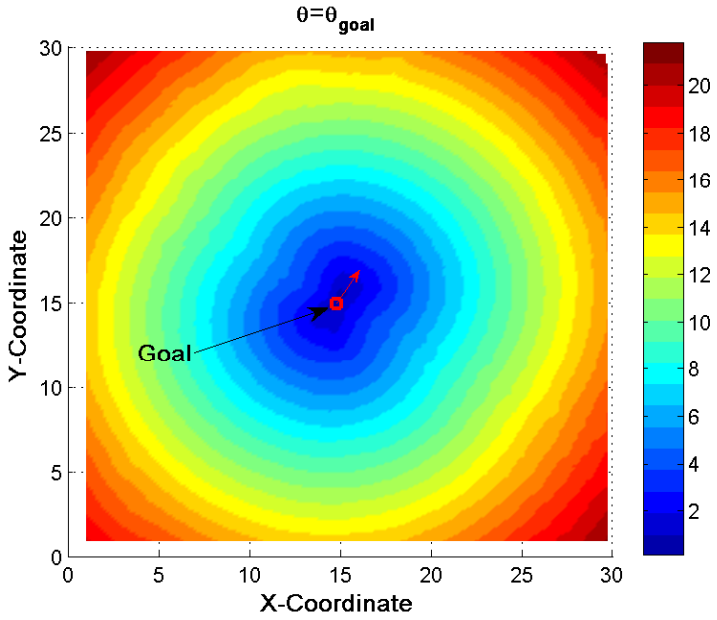


(d) The estimated cost-to-go function for the point mass after 2500 iterations

Figure 5.1: The estimated cost-to-go function obtained by the proposed incremental algorithm for point mass from different initial conditions.



(a) The estimated cost-to-go function for on three planes in $\mathbb{R}^2 \times \mathbb{S}^1$.



(b) The estimated cost-to-go function on the plane oriented with θ_{goal}.

Figure 5.2: The estimated cost-to-go function obtained by the proposed incremental algorithm for Reeds-Shepp system obtained after 19000 iterations. It shows the asymmetry and also the transverse movements possible due to the Lie Bracket operator. Notice that the levels sets are stretched in the longitudinal direction i.e., parallel to the orientation of the car at the goal. (Arrow shows the orientation of the car at the goal)

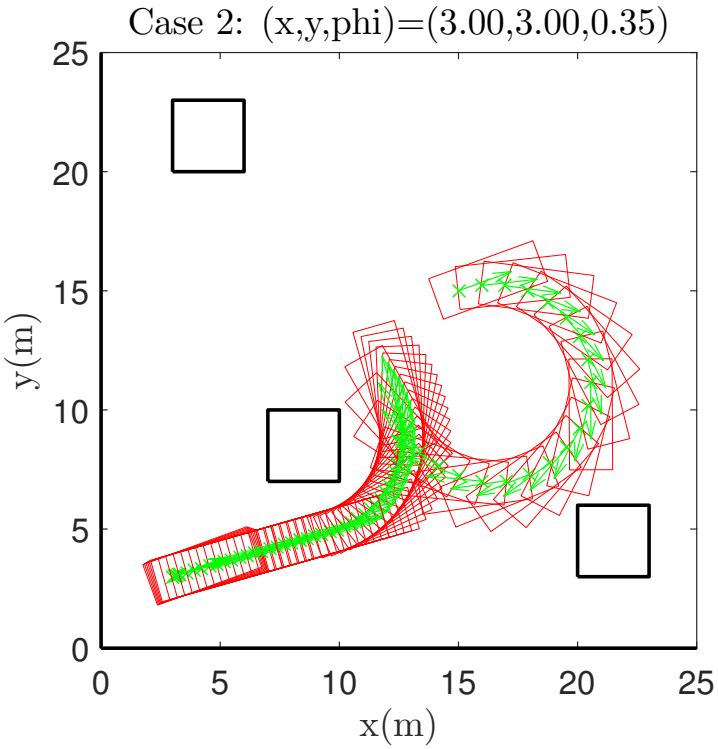


Figure 5.3: The path connecting $(3, 3, \pi/9)$ to $(15, 15, \pi/9)$.

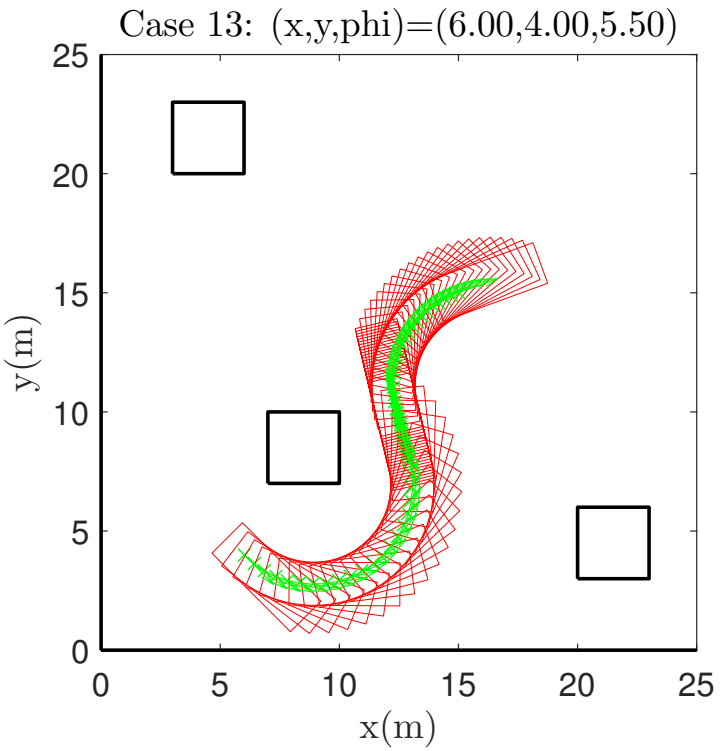


Figure 5.4: The path connecting $(6, 4, 5.5)$ to $(9, 9, \pi/2)$.

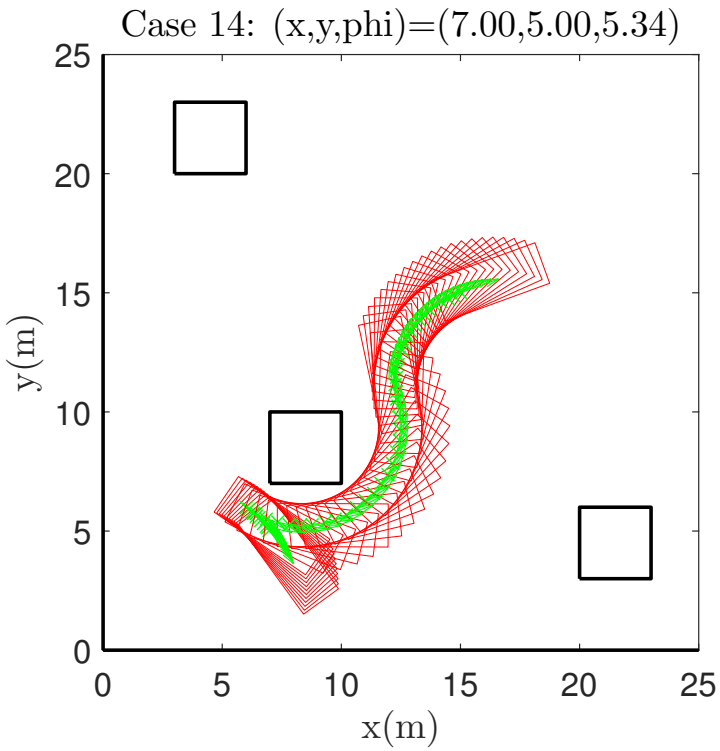
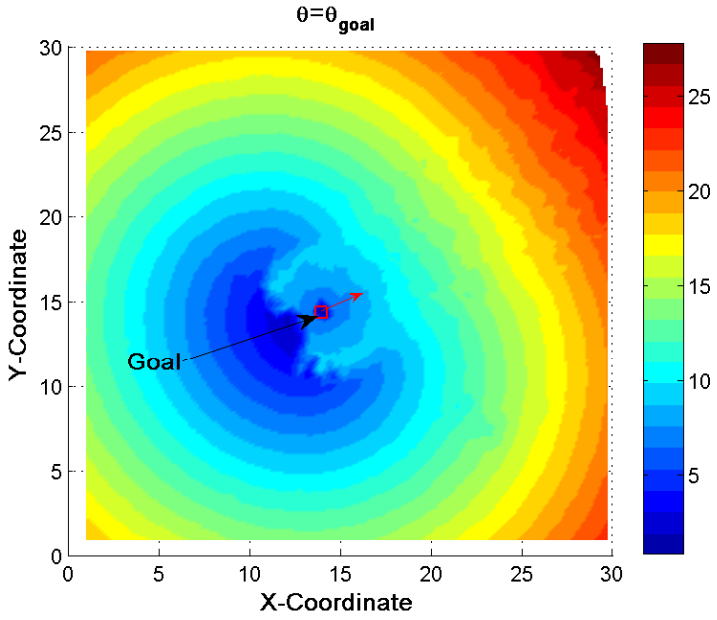
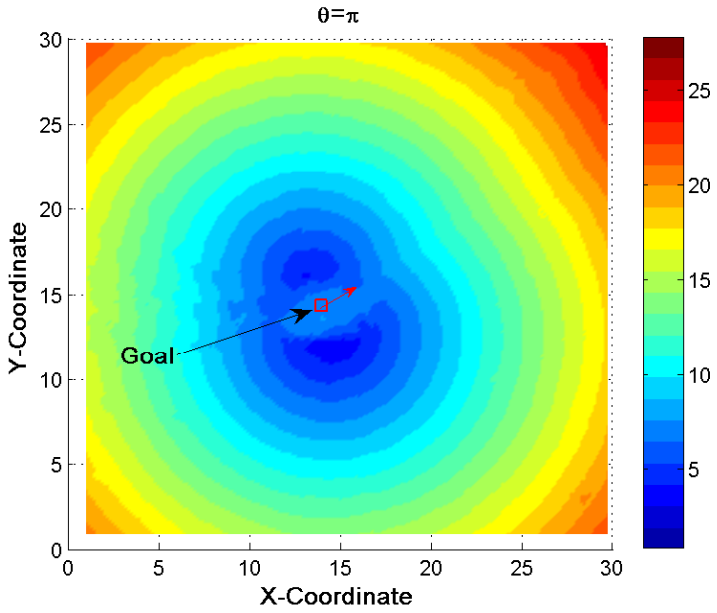


Figure 5.5: The path connecting $(7, 5, 5.34)$ to $(15, 15, \pi/9)$.



(a) The estimated cost-to-go function on plane oriented with the goal.



(b) The estimated cost-to-go function on plane with $\theta = \pi$.

Figure 5.6: The estimated cost-to-go function obtained by the proposed incremental algorithm for Dubins car system obtained after 19000 iterations. As the Dubins car can't move backwards, the value functions are discontinuous and it results in more complicated reachable sets in the sub-Riemannian manifold. (Arrow shows orientation of the car at the goal.)

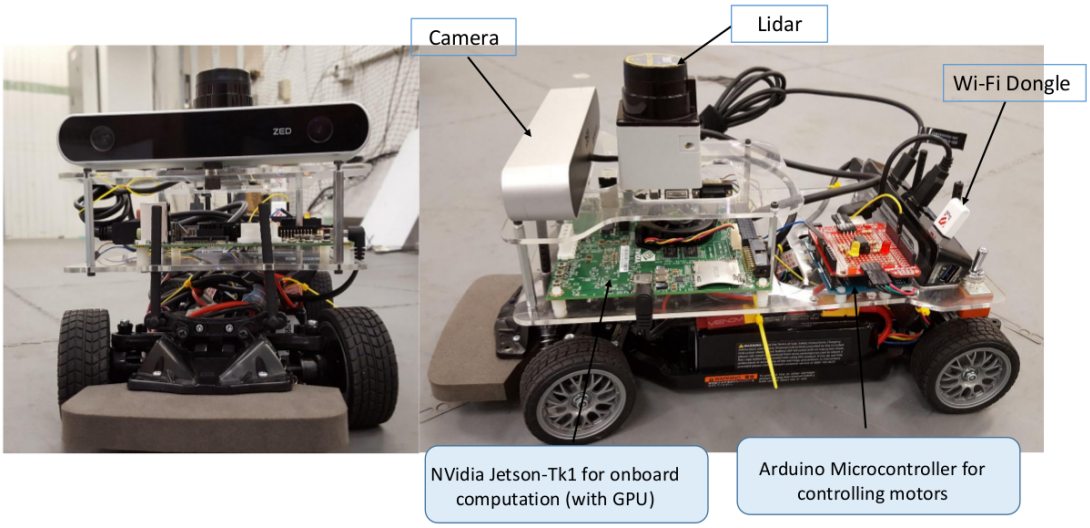
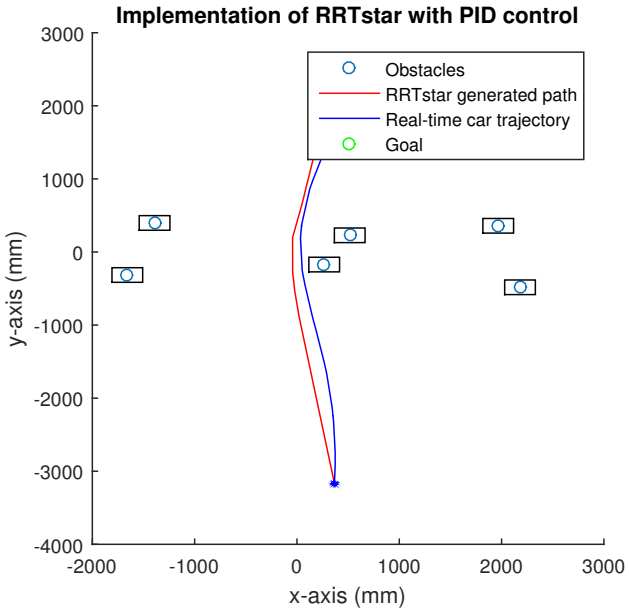
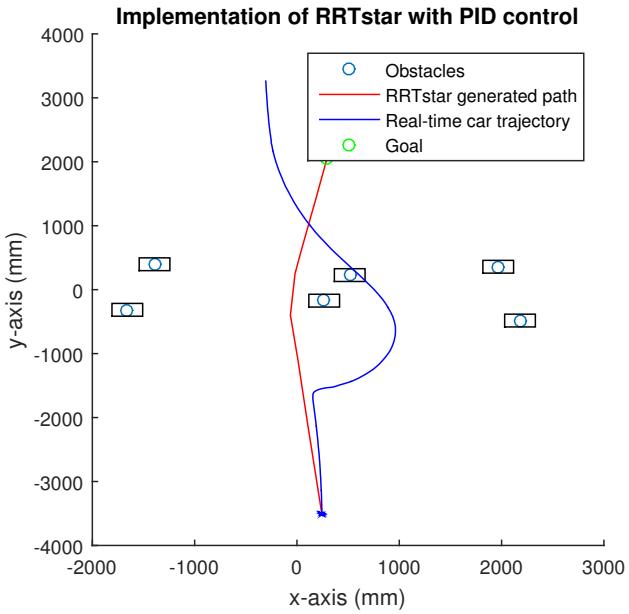


Figure 5.7: Picture of the RC car built in the robotics lab at Penn State

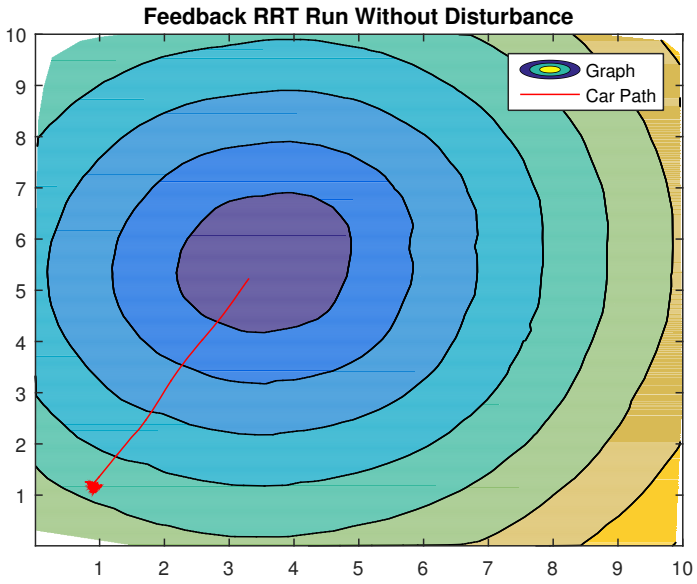


(a) RRT* with PID control in absence of disturbance

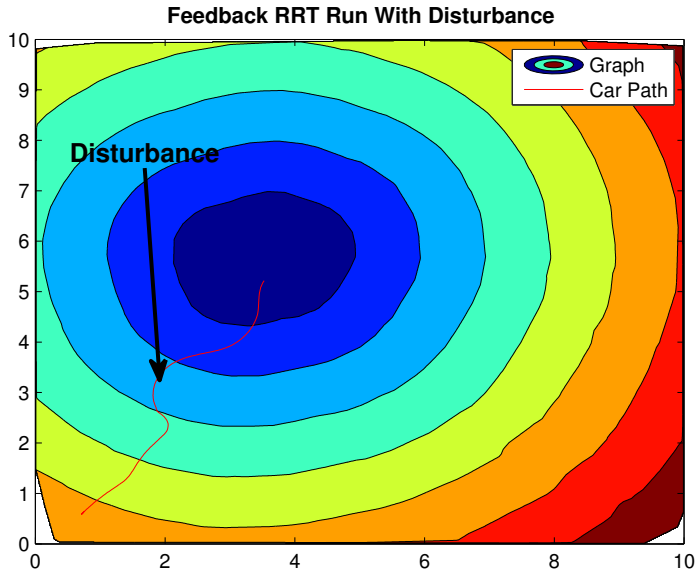


(b) RRT* with PID control in the presence of disturbance

Figure 5.8: This figure shows the performance of RRT* with PID control for trajectory following. In the presence of disturbance,a replanning is required for collision avoidance



(a) Feedback RRT in the absence of disturbance



(b) Feedback RRT in the presence of disturbance

Figure 5.9: This figure shows the performance of feedback RRT where re-planning is avoided for the inherent nature of state-based feedback.

Chapter 6

Data-Driven Algorithms for Anytime Motion Planning with Safety Guarantees

Safety in artificial intelligence systems is a very critical issue and has largely been ignored in software-inspired applications of machine learning. However, for learning in autonomous systems, safety can't be ignored for the associated costs of incorrect decisions and interpretations. In the last chapter, we saw some algorithms which allow state feedback-based control of robotic systems. These algorithms need to adapt to unmodeled parameters which can be estimated online from streaming data. This chapter presents a learning-based (i.e., data-driven) approach to motion planning of robotic systems. This is motivated by controller synthesis problems for safety critical systems where an accurate estimate of the uncertainties (e.g., unmodeled dynamics, disturbance) can improve the performance of the system. The state-space of the system is built by sampling from the state-set as well as the input set of the underlying system. The robust adaptive motion planning problem is modeled as a learning-based approach evasion differential game, where a machine-learning algorithm is used to update the statistical estimates of the uncertainties from system observations. The system begins with a conservative estimate of the uncertainty set to ensure safety of the underlying system and we relax the robustness constraints as we get better estimates of the unmodeled uncertainty. The estimates from the machine learning algorithm are used to refine

the estimates of the controller in an anytime fashion. We show that the values for the game converges to the optimal values with known disturbance given the statistical estimates on the uncertainty converges. Using confidence intervals for the unmodeled disturbance estimated by the machine learning estimator during the transient learning phase, we are able to guarantee safety of the robotic system with the proposed algorithms during transience.

6.1 Introduction

Motion planning is a classic problem in robotics and has received a lot of attention in robotics, computer science and control systems society. The basic problem of computing a collision-free trajectory connecting an initial configuration or state to a target region through a cluttered environment is well-understood and fairly well-solved [136, 143]. However, the shortcomings of the basic path planning algorithms are revealed when considering how these algorithms are used for controlling an autonomous robotic system using an auxiliary controller. A fundamental problem with control of autonomous robotic systems (e.g., self-driving cars) is safety control i.e., ensuring the system stays in the given safety sets while simultaneously achieving the given objectives. It becomes difficult to ensure safety for these systems in dynamically changing and uncertain environments while using the path planning algorithms with the decades-old *Sense Plan Act* paradigm as the reachable sets in the presence of dynamic uncertainties are not considered.

It is well-known that robotic motion planning is at least as difficult as the generalized piano mover's problem, which has been proven to be PSPACE-hard [135]. A wide spectrum of motion planning algorithms have been proposed, including, to name a few, discretized approaches such as A* [136]; continuous approaches involving navigation function and potential fields [136], and more recently, sampling-based geometric planning algorithms such as the rapidly-exploring random trees (RRT) [137], probabilistic roadmaps (PRM) [138] and their optimal variants RRT* and PRM* [139]. To date, the state-of-the-art motion planning algorithms are sampling-based and can guarantee asymptotic optimality [139]. More recently, some algorithms can provide convergence rates to the optimal solution [158].

Various algorithms have been proposed to solve related stochastic and robust motion planning in the presence of external disturbances and uncertainties. Some

approaches to solve robust planning can be found in [159–162] and stochastic control could be found in [14, 150, 163]. However, most of the proposed algorithms can not provide any performance guarantees for safety. Recently, a sampling-based approach-evasion game formulation of the motion planning problem was proposed in [152]. The proposed anytime algorithms guarantee asymptotic optimality for the underlying approach-evasion games and thus, provide a robust solution by finding the discriminating kernel [164] and the corresponding optimal control inputs for the system. Since [152] still adopts a robust approach, the algorithm performance could be conservative. A robust-adaptive motion planning algorithm in the presence of moving obstacles has been proposed in [165]. However, it is based on open-loop control and makes use of a model-based estimator.

Contributions: In this chapter, we propose data-driven anytime algorithms for motion planning with safety and performance guarantees for the underlying autonomous robotic system. In particular, we use incremental sampling to construct the state-space of the system and synthesize a robust controller by solving an approach-evasion differential game to reach the target set in the presence of unmodeled uncertainty in the system. The system uses a statistical estimator to estimate the unmodeled uncertainty based on observations of system trajectories. These estimates are then used by the controller to refine the estimates of the value functions for the underlying differential game to improve performance while preserving safety guarantees. This allows the system to learn from data and find better policies for performance enhancement while maintaining the desirable safety guarantees which are critical in modern autonomous robotic systems like self-driving cars etc.. To the best of authors' knowledge, such a data-driven, incremental sampling-based anytime algorithm for robust adaptive motion planning has not been presented before. Moreover, the proposed approach can be used for non-linear (e.g., nonholonomic systems) which we normally come across in motion planning problems. Guarantees on transient performance of the robust-adaptive controller is provided based on the confidence intervals for the disturbance function predicted by the statistical estimator.

Literature. Reinforcement learning has been widely used to design feedback controllers for systems without knowing full dynamics of the underlying systems [1, 166]. A lot of work has been done in reinforcement learning using different approaches like Hidden Markov Models [167, 168], Bayesian Methods [169],

Q-learning, Temporal Difference (TD) learning [170] etc.. However, a big concern in these algorithms is the transient learning phase when a bad initial policy could be disastrous for the autonomous systems. More recently, an elegant Model Predictive Control (MPC) formulation of the problem with safety guarantees was presented in [171] using reachability analysis of the dynamical system and Tube MPC for robust, learning-based control for linear systems with asymptotic performance guarantees. Some Hamilton-Jacobi-Issacs (HJI)-based approaches have been presented for control applications in [172, 173]. The basic idea in these papers is to first calculate the discriminating kernel and let the system explore and find better policies inside the discriminating kernel. However, there are no performance guarantees.

6.2 Problem Formulation

Consider a non-linear dynamical system governed by the following differential equation:

$$\dot{x}(t) = f(x, u, d) = g(x, u) + d(x), \tag{6.1}$$

where $x(t) \in \mathcal{X} \subseteq \mathbb{R}^N$ is the system state, and $u(t) \in \mathcal{U}$ is the control input of the system. Furthermore, $g : \mathbb{R}^N \times \mathbb{R}^M \to \mathbb{R}^N$ is locally Lipschitz continuous functions which represent the known part of the dynamics of the underlying system. The function $d : \mathbb{R}^N \to \mathbb{R}^N$ represents the unknown dynamics of the system. We assume that $d(x) \in \mathcal{D}$, where $\mathcal{D} \in \mathbb{R}^N$. For system (6.1), the set of admissible (feedback) control strategies is defined as:

$$\mathcal{U} \triangleq \{u(\cdot) \ : \ [0, +\infty) \to U, \text{ measurable}\},$$

where $U \subseteq \mathbb{R}^M$. Denote by $\phi(\cdot; x, u, d) \triangleq \{\phi(t; x, u, d)\}_{t \geq 0}$ the solution to system (6.1) given the initial state x, the controls of u and the unmodeled dynamics d.

Throughout this chapter, we impose the following assumption on system (6.1):

Assumption 6.2.1 *The following properties hold:*

(A1) The sets $\mathcal{X}$, U and $\mathcal{D}$ are compact.

(A2) The disturbance function $d(x)$ is locally Lipschitz continuous.

(A3) The function f is continuous in (x, u, d) and Lipschitz continuous in x for any $(u, d) \in U \times \mathcal{D}$.

(A4) For any pair of $x \in \mathcal{X}$ and $u \in U$, $F(x, u)$ is convex where the set-valued map $F(x, u) \triangleq \cup_{d \in \mathcal{D}} f(x, u, d)$.

The model represented by equation (6.1) is able to represent a large class of systems with certain unmodeled parameters which are difficult to model using the first principles of physics (e.g., air drag for aerial robots, tyre friction or slipping for ground wheeled car-like robots in rough or slippery terrain). The idea is that we estimate these parameters using some statistical methods which can be used by the controller to improve its performance while maintaining robustness. The problem of controller synthesis for the underlying system when $d(x)$ is unknown is formulated as an adaptive approach-evasion game with time-varying estimates on the disturbance.

The objective of the controller is to maximize the performance of the dynamical system which is affected by some disturbance $d(x)$. Disturbance (which in our case is the unmodeled dynamics) wants to maximize the cost of the control input u. We formalize the aforementioned objectives as follows. Define by $t(x, u, d)$ the first time when the trajectory $\phi(\cdot; x, u, d)$ hits $\mathcal{X}_{\text{goal}}$ while staying in $\mathcal{X}_{\text{free}}$ before $t(x, u, d)$. More precisely, the functional $t(x, u, d)$ is defined as follows:

$$t(x, u, d) \triangleq \inf\{t \geq 0 \mid \phi(t; x, u, d) \in \mathcal{X}_{\text{goal}}, \\ \phi(s; x, u, d) \in \mathcal{X}_{\text{free}}, \ \forall\ s \in [0, t]\}.$$

If $\phi(\cdot; x, u, d)$ leaves $\mathcal{X}_{\text{free}}$ before reaching $\mathcal{X}_{\text{goal}}$ or never reaches $\mathcal{X}_{\text{goal}}$, then $t(x, u, d) = +\infty$. This formulation leads to a zero-sum differential game between two players i.e., the controller and the disturbance. We call it the time-optimal-approach-evasion (TO-AE) differential game.

We use the notion of non-anticipating or causal strategy in the sense of [174] to define the value of the TO-AE differential game. The set Γ^a of such strategies for the controller is such that $\gamma^a : \mathcal{D} \to \mathcal{U}$ satisfies for any $T \geq 0$, $\gamma^a(d(t)) = \gamma^a(d'(t))$ for $t \in [0, T]$ if $d(t) = d'(t)$ for $t \in [0, T]$. The lower value of the TO-AE differential

game is given by:

$$T^*(x) = \inf_{\gamma^a(\cdot)\in\Gamma^a} \sup_{d(\cdot)\in\mathcal{D}} t(x, \gamma^a(d(\cdot)), d(\cdot)).$$

The function T^* is then referred to as the minimum time function. It is noted that $t(x, u, d)$ is potentially infinite and this may cause numerical issues. To deal with this, we normalize the hitting time by the Kružkov transform $\Psi(r) = 1 - e^{-r}$. With this nonlinear transform, we further define the discounted cost functional $J(x, u, d) = \Psi \circ t(x, u, d)$, and the discounted lower value v^* as follows:

$$v^*(x) = \inf_{\gamma^a(\cdot)\in\Gamma^a} \sup_{d(\cdot)\in\mathcal{D}} J(x, \gamma^a(d(\cdot)), d(\cdot)).$$

One can easily verify that $v^*(x) = \Psi \circ T^*(x)$ for $\forall x \in \mathcal{X}$. We will refer v^* to as the optimal value function.

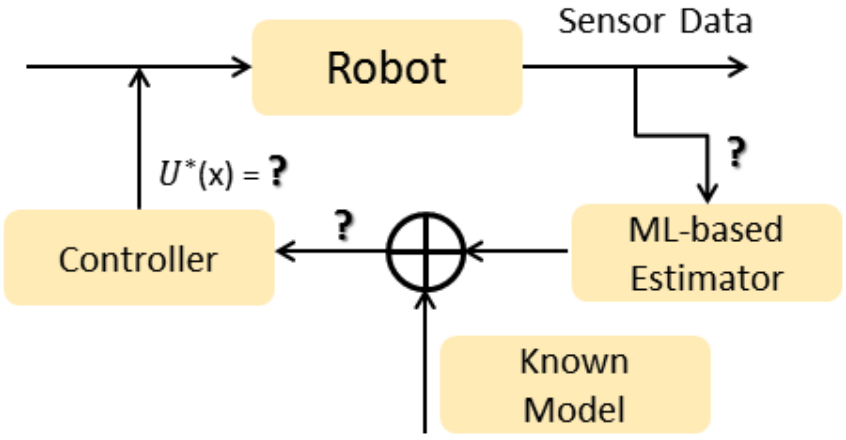


Figure 6.1: The Control Loop with the Machine Learning-based estimator for Motion Planning

Next under the TO-AE game setting, we allow the system to learn from observations and thus use new estimates on the disturbance function to make more accurate and better estimates on the minimum time function. To achieve this, we use a machine learning(ML)-based statistical estimator to estimate the function $d(x)$. Making use of observations for predicting the disturbance function results in inherent probabilistic estimates such that the point-wise functional estimates lie in some interval say $\tilde{\mathcal{D}}_k(x)$ at kth iteration with significance level $(1-\alpha)$ for $\alpha \in (0, 1)$.

A sufficient condition for the safety of the underlying system in the presence of the disturbance is that $d(x)$ lies in the estimated set $\tilde{\mathcal{D}}_k(x)$ for all $k \in \mathbb{N}$. While a possible solution to guarantee safety is to use the conservative bounds; our objective here is to estimate $d(x)$ or at least a tighter bound represented by the set $\tilde{\mathcal{D}}_k(x)$ so that we can improve system performance while maintaining provable-safety guarantees. The objective here is to optimize system performance by updating modeling uncertainties or exogenous disturbance interfering with the system while retaining safety guarantees. The difference from classical control theory-based robust adaptive control is that we use a machine learning-based statistical estimator (model-free) instead of a Kalman filter-type estimator.

Thus, our problem consists of two steps.

1. Making tighter estimates on the bounds of the disturbance function using a statistical method from observations on system trajectories.

2. Use the statistical estimates to make better estimate of minimum time function for the underlying approach-evasion game.

It is also shown as a feedback control system in Figure 6.1 where the different components of the problem i.e., how to analyze sensor data using a ML-based algorithm and consequently, how to use it for deciding the control law, are highlighted. The idea is that in the presence of a statistical estimator, we predict the intervals in which the disturbance function is expected to lie with a high degree of confidence. Due to the probabilistic estimates, we sacrifice some performance for safety of the system.

6.3 Data-Driven Anytime Robust-Adaptive Algorithm

In this section, we present the algorithms and ideas for solving the learning-based TO-AE differential game. We allow the controller and estimator to run in parallel, independent of each other using the separation principle. The key idea is that we ensure the safety of the system by solving the TO-AE game which calculates the discriminating kernel [152, 164] as well the optimal controller simultaneously for the underlying system; we further refine the controller by getting new updates on the

disturbances of the system using bootstrapping (as the value functions for the system might be non-differentiable, which is generally the case, we don't use gradient-based approaches for policy-learning). While with the gradient-based approaches it is possible to provide local optimality and convergence guarantees [1,166], it is difficult to provide global performance guarantees.

6.3.1 Machine Learning-based Estimator

A lot of work is available in Machine Learning literature for estimation of unknown generative functions (i.e., which can be used for prediction) from observed data using concepts from statistical learning theory; it is possible to provide convergence guarantees in probabilistic sense for such estimates [23,171]. We focus on the special case where we can measure all the states of our system and assume that there is no measurement noise. The key idea here is that to guarantee safety of the underlying system, we need to have deterministic convergence guarantees on the estimates provided by the statistical regression algorithm. While such guarantees might be elusive with finite amount of data, it is possible to estimate generative functions along with point-wise confidence intervals [175]. We try to estimate the unmodeled dynamics using non-parametric kernel-based regression technique (e.g., Support Vector Regression [93], Gaussian processes [176], Radial Basis Functions [177], etc.). The advantage with using such statistical estimators is that we can model a large number of generative functions (linear or non-linear) by using various kernels without the knowledge of the underlying model structure.

To estimate the unmodeled dynamics of the underlying system, we first numerically calculate the system state derivatives using observations on the system state and then, we calculate the residuals using the known part of the system model. More formally, the residuals are calculated as

$$\bar{d}(x) = \bar{g}(x, u) - g(x, u) \tag{6.2}$$

where $\bar{g}(x, u)$ represents the numerically calculated gradient using the observations on system trajectories and $g(x, u)$ is the known part of the system dynamics. The term $\bar{d}(x)$ in equation (6.2) represents anomaly in the system observation which can't be explained using the known model and is thus considered to be present due to the unmodeled disturbance (dynamics). These residuals are then used to predict

the unknown disturbance function using a least square support vector machine (LS-SVM) regression algorithm. Along-with estimating the underlying unknown function we also calculate the point-wise confidence bounds for the estimates. A least square support vector regression algorithm with confidence intervals was presented in [175]. We use the algorithms presented in [175] to estimate the unknown disturbance function. We use the following notation to describe the LS-SVM regression: the data set at any epoch n is the set $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ where $Y_i = \bar{d}(x_i)$ and $X_i = x_i$. Thus, X_i's are the independent state variables and Y_i's represent the corresponding observations corresponding to X_i's. The goal is to find the underlying generative function, with probabilistic bounds on accuracy of the same. The motivation behind using the LS-SVM regression is to be able to find the disturbance function with probabilistic bounds using confidence intervals for the function.

For the completeness of the chapter, we very briefly describe the key idea behind the regression-based estimation process. In particular, we model our data is being generated by $Y = d(X) + \sigma(X)\varepsilon$, where $\mathbf{E}[\varepsilon|X] = 0$, $\mathbf{Var}[\varepsilon|X] = 1$ and X & ε are independent (Y is the observation and X represents the independent variable in the domain of the unknown function). What we are interested is an estimate of the function $d(x)$ (we denote it by $\hat{d}_n(x)$ where n denotes the number of observations) and the corresponding confidence interval for $d(x)$ i.e., given $\alpha \in (0, 1)$ we want to find a bound η_α such that $\mathbf{P}(\sup_{x \in X} |\hat{d}_n(x) - d(x)| \leq \eta_\alpha) \leq 1 - \alpha$, where X is the domain of the function $d(\cdot)$. As such, we estimate the confidence interval for the unknown function $d(x)$ point-wise as well as the interval over the domain of the function. The LS-SVM problem is formulated as an optimization problem as follows

$$\min_{w,b,e} \mathcal{J}(w, b, e) = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{i=1}^{n} e_i^2 \tag{6.3}$$

such that $Y_i = w^T \psi(X_i) + b + e_i \,, i = 1, 2, \ldots, n$ where $e_i \in \mathbb{R}$ are assumed to be i.i.d. random variables with $\mathbf{E}[e|X] = 0$ and $\mathbf{Var}[e|X] < \infty$. We assume that the d is a smooth function and $\mathbf{E}[Y|X] = d(X)$, ψ is a feature map or kernel used in standard SVM. Based on the observations on the system trajectories, an estimate

of the unknown function (denoted as $\hat{d}$) using LS-SVM regression is obtained as

$$\hat{d}(x) = \sum_{i=1}^{n} \hat{\alpha}_i K(x, X_i) + \hat{b} \tag{6.4}$$

where, $K : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ represents a kernel function (e.g., Gaussian or radial basis functions), m is the dimension. The terms $\hat{\alpha}$ are the dual Lagrange multiplier obtained by solving the LS-SVM problem. In the next step we calculate the linear smoother matrix L for the LS-SVM regression such that we can estimate the conditional mean and variance for the estimated unmodeled function. The conditional mean and variance of the estimated function is given by the following expressions.

$$\mathbf{E}[\hat{d}(x)|X = x] = \sum_{i=1}^{n} l_i(x) m(x_i) \tag{6.5}$$

$$\mathbf{Var}[\hat{d}(x)|X = x] = \sum_{i=1}^{n} l_i(x)^2 \sigma^2(x_i) \tag{6.6}$$

Then, the expected value or the bias is approximately calculated by the following equation (see Theorem 2 in [175] for a proof).

$$\widehat{\texttt{bias}}[\hat{d}(x)|X = x] = L(x)^T \hat{d} - \hat{d}(x) \tag{6.7}$$

where, $\hat{d} = (\hat{d}(X_1), \ldots, \hat{d}(X_n))$ and L is the smoother matrix. The variance of the estimates is then calculated using the following equation (for a proof see Theorem 3 in [175]).

$$\mathbf{Var}[\hat{d}(x)|X = x] = L(x)^T \hat{\Sigma}^2 L(x) \tag{6.8}$$

where, $\hat{\Sigma} = \texttt{diag}(\hat{\sigma}^2(X_1), \ldots, \hat{\sigma}^2(X_n))$ and L is the smoother matrix. The term $\hat{\sigma}^2(x)$ is calculated using the following equation.

$$\hat{\sigma}^2(x) = \frac{S(x)^T \texttt{diag}(\hat{\varepsilon}\hat{\varepsilon}^T)}{1 + S(x)^T \texttt{diag}(LL^T - L - L^T)}$$

where, $S(x)$ is smoother vector at an arbitrary point X such that $S : \mathbb{R}^m \to \mathbb{R}^n$ and $S1_n = 1_n$, $\hat{\varepsilon}$ represents the residuals (i.e., deviation from the bias term) and $\texttt{diag}(A)$ represents the diagonal terms of A expressed as a column vector. The LS-SVM

guarantees that under some regularity conditions, the central limit theorem is valid and the following is true asymptotically. More formally we state the following theorem.

Theorem 6.3.1 *For finite variance of the residuals $\hat{\varepsilon}$ (calculated using equation* (6.8)*), the following is true.*

$$\lim_{n\to\infty} \frac{\hat{d}_n(x) - \mathbf{E}[\hat{d}_n(x)|X = x]}{\sqrt{\mathbf{Var}[\hat{d}_n(x)|X = x]}} \xrightarrow{D} \mathcal{N}(0, 1)$$

where $\xrightarrow{D}$ implies convergence in distribution.

Proof 6.3.1 *Follows from Theorem* 7.4 *in [178].*

Then, the point-wise $(1 - \alpha)$, where $\alpha \in (0, 1)$, confidence interval for the unknown function $d(x)$ is given by the following expression

$$\hat{d}_n(x) - \widehat{\texttt{bias}}[\hat{d}_n(x)|X = x] \pm z_{1-\alpha/2}\sqrt{\mathbf{Var}[\hat{d}_n(x)|X = x]} \tag{6.9}$$

where, the $\widehat{\texttt{bias}}$ is the correction term given by equation (6.7) and $z_{1-\alpha/2}$ denotes the $(1 - \alpha/2)$ quantile of the standard Gaussian distribution. For prediction at a new state $\tilde{x}$ the confidence interval is given by the following equation.

$$\begin{aligned}\hat{d}_n(\tilde{x}) - \widehat{\texttt{bias}}[\hat{d}_n(\tilde{x})|X = \tilde{x}] \\ \pm z_{1-\alpha/2}\sqrt{\hat{\sigma}^2(\tilde{x}) + \mathbf{Var}[\hat{d}_n(\tilde{x})|X = \tilde{x}]}\end{aligned} \tag{6.10}$$

With point-wise confidence interval for the statistical estimates, we can define the expected bound for disturbance with probabilistic significance level point-wise, i.e., the set $\tilde{\mathcal{D}}_k(x)$ which was introduced earlier in section 6.2. Using the sets $\tilde{\mathcal{D}}_k(x)$, we can construct the function D_k which contains the point-wise bounds for disturbance functions corresponding to a chosen significance level. This function is then used for synthesizing the control law. The original disturbance function then lies in this interval with significance level of $(1 - \alpha)$.

The statistical estimator can thus maintain an estimate of the unknown disturbance function with expected deviation from the actual function. Using the central limit theorem, the LS-SVM regression guarantees convergence of the expected

deviation of the function to the standard Gaussian distribution. This allows us to calculate the disturbance function bounds with probabilistic confidence intervals. Then, the idea is that we use these bounds corresponding to significantly high confidence intervals to relax the safety constraints on the system for improvement of performance.

6.3.2 Controller Synthesis using Iterative Incremental Game Algorithm

In the proposed approach, we decouple the controller and estimator such that sampling of estimates by the controller is independent of sampling rate of data used by the sensors and the estimator. Let the supremum of the disturbance vector norm in the set D_k be denoted by r_k, i.e., $r_k = \sup_{x \in \mathcal{X}} |D_k(x)|$. The estimate r_k is allowed to vary with time and it is not assumed to be monotonic. However, to establish the results the results presented in the chapter we make the following assumption.

Assumption 6.3.1 *The norm of the disturbance function is upper bounded by R which gives a conservative estimate of the disturbance i.e., $|d(x)| < R$ for all $x \in R$.*

The above assumption is not very restrictive in the sense that, in general, models with very high accuracy are available for engineered systems and a conservative bound for uncertainties involved with the environment could be approximated using statistical estimates of past experiences. However, this is important to ensure the safety of the system as otherwise the system might end up using policies which might result in very high penalties. With this structure on the estimates provided by the statistical estimator, a new TO-AE game is defined for a new estimate of the disturbance. This leads to a family of parametric differential game [179] in the sense that $v^\star_{r_k} \leq v^\star_{r_n}$ for any pair of $r_k \leq r_n$ and for all $x \in \mathcal{X}$ (where $v^\star_{r_i}$ denotes the estimates of optimal value functions parameterized by r_i). Thus when we get a new estimate of the disturbance, it can be used to solve a new TO-AE game parameterized by a new disturbance bound estimate using bootstrapping on the sampled graph, i.e., the values for the new game could be initialized by already existing estimates for the same parameterized by a different bound.

To begin with, we solve the approach-evasion game using the conservative bound on the unmodeled disturbance; thus we recover discriminating kernel and

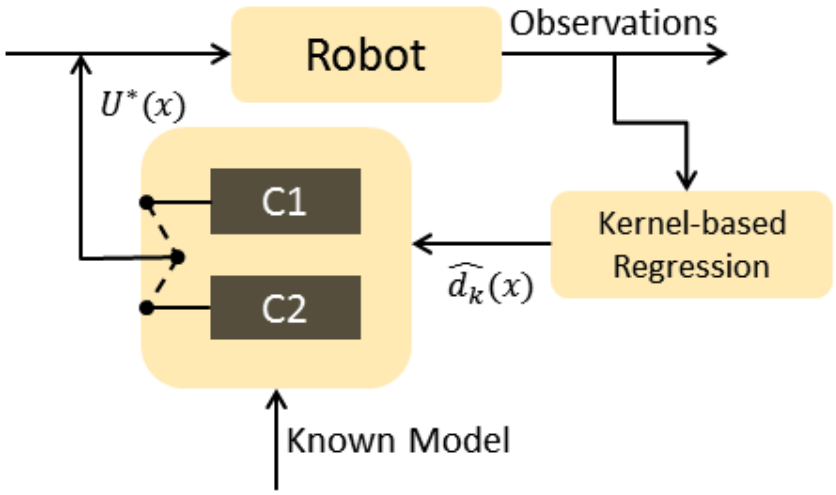


Figure 6.2: Controller synthesis using the kernel-based regression estimator

the corresponding optimal control inputs corresponding to the expected conservative disturbance bound. We estimate the unmodeled disturbance, based on the observations on system trajectory, using the the statistical estimator described in section 6.3.1. With a new estimate on the bounds of the unmodeled dynamics, we can initialize a new approach-evasion game and solve to find new control law. However, the statistical estimates of the bounds of deviation from the actual disturbance function may not monotonically converge to normal distribution (equation (6.9) implies asymptotic convergence; however no convergence rates are provided). Here, we would like to point out that if the estimator can provide monotonic, deterministic estimates on the bounds, we can use them to refine the estimates on the value functions for the system while ensuring safety. With probabilistic bounds, a risk-averse strategy is to maintain a bank of controllers for safer control during transience. The idea is shown as a schematic in Figure 6.2, where we maintain a bank of controllers and then we pick the control based on the current estimate provided by the kernel-based regression estimator. While the system uses the control given by the controller $C1$, $C2$ represents the bank of controllers. Given a new estimate by the ML-estimator, a new controller is pushed to $C1$ while $C1$ is moved back to $C2$ (we don't throw out any controller till estimator transience is over).

However, the problem becomes more complex with stochastic time-varying estimates on the bounds of disturbance function deviation and it requires a better

switching law to retain system safety. Nevertheless, if we can estimate the confidence intervals with high significance level, we can still ensure safety with very high confidence during transience. This motivates the use of confidence intervals with the statistical estimates. Since we have a conservative bound on the expected disturbance to the system, the actual discriminating kernel of the system corresponding to the actual disturbance function is thus a superset of the discriminating kernel calculated corresponding to the initial conservative bound. Thus the new estimates can still be used with low-confidence intervals to refine the value functions inside of the discriminating kernel and guarantee safety. At the boundaries of the discriminating kernel, using a bound with low confidence interval might have low safety guarantees. This observation can be used to further improve the performance of the system in some regions of the state-space. However, we use the estimates with confidence intervals corresponding to high significance levels for deciding new control laws to avoid unnecessary switching. While a deterministic estimate would lead to maximum performance, probabilistic estimates leads to a trade-off in performance for safety. However, this is a restriction imposed by the statistical model-free estimator.

To decide the control law, a user-defined significance level for statistical estimates is used to find the confidence interval for the unmoded disturbance function. Then, suppose that the controller samples new estimate of the unmodeled disturbance at an epoch n. Then, the uncertainty with a significance level $(1 - \alpha_{\text{CI}})$ (α_{CI} is user input) over the state-space is bounded by the following term, $r_n = \sup_{x \in \mathcal{X}} |D_n(x)|$.

The term r_n provides the bound on the expected deviation of the estimated function from the original disturbance function over its domain with a significance level $(1 - \alpha_{\text{CI}})$. Then, a new game could be initiated which is parameterized by r_n. Then, we get a series of such estimates and the corresponding value function which we arrange in an ordered fashion. We represent the sets by $E_k = \{R, r_1, r_2, \ldots, r_k\}$ and $V_k = \{v^\star_R, v^\star_{r_1}, v^\star_{r_2}, \ldots, v^\star_{r_k}\}$ for the estimates and the corresponding value functions respectively where, $R \geq r_1 \geq r_2 \geq \cdots \geq r_j \geq \ldots$ and it follows that $v^\star_R \geq v^\star_{r_1} \geq v^\star_{r_2} \geq \cdots \geq v^\star_{r_j} \geq \ldots$. Then, when a new estimate r_n is sampled by the controller with confidence level α_{CI}, it is compared to the elements of the existing elements of E_{n-1} and a controller corresponding to $v^\star_{r_k}$ is used where $r_k \geq r_n \geq r_{k-1}$. Consequently, the new estimate to the set E_k and a new game parameterized by r_n is initiated with $v^\star_{r_k}$ (technically we recover the optimal costs asymptotically;

with some abuse of notation we use the $v^{\star}_{r_k}$ to denote the converged numerical estimates). As the estimates for v_{r_n} converge, we switch to the control given by the same. With this we incrementally populate the set E_k and the set V_k. This process is repeated till the elements of set E_k converge. The controller is always decided by the latest estimates on the bounds of the disturbance function estimated by the ML-based estimator with significance level $(1 - \alpha_{\text{CI}})$. The adaptive controller synthesis is presented in algorithms 6 through 9. It is noted that since the controller and estimator are decoupled, the iteration number for them are different. The estimator is collecting data and making estimates based on the sampling rate of the sensors.

Algorithm 6: Data-Driven Anytime Control

Require: Initially, use the conservative bound R to solve iGame with values v_R and $E_0 \leftarrow \{R\}$. Pick a sampling interval T_s to get a new estimate for $d(x)$ using Algorithm 9

Ensure: Repeat the following steps at every iteration

if $n \mod T_s = 0$ **then**
  `flag` $= 1$
else
  `flag` $= 0$
if `flag` $== 1$ **then**
  $k = n/T$;
  $E_k = E_{k-1} \cup \{r_k = \sup_{x \in \mathcal{X}} |D_m(x)|\}$;
  Find m s.t. $r_m \geq r_k \geq r_{m+1}$ and order the set $E_k = E_{k-1} \cup \{r_k\}$;
  Initialize v_{r_k} with values from v_{r_m} on S_{n-1};
  $V_k \leftarrow V_{k-1} \cup \{v_{r_k}\}$;
  $u_n(x) \leftarrow$ solution to u from v_{r_m};
  go to Algorithm 7;
else
  go to Algorithm 7

The controller synthesis is based on the iGame algorithm earlier presented in [152] where the states-space for the robot is incrementally built by sampling from the free configuration space of the robot. The input set for the robot is also incrementally built by sampling from its input set. At every iteration of the algorithm, a single update of the value iteration is solved on the sampled graph for the corresponding pursuit-evasion game. Thus, by incremental sampling from the state-space a better refined discrete state-space for the robot is created and

Algorithm 7: The iGame Algorithm

$y_n \leftarrow \texttt{Sample}(\mathcal{X}, 1)$;
$S_n \leftarrow S_{n-1} \cup \{y_n\}$;
$h_n \leftarrow \zeta_n^{\frac{1}{1+\gamma}}$;
$\gamma_n \leftarrow 2\zeta_n + \ell h_n \zeta_n + M\ell h_n^2$;
$v_{n-1} = v_{r_m}$;
for $x \in S_{n-1}$ **do**
    $\tilde{v}_{n-1}(x) = v_{n-1}(x)$;
$\tilde{v}_{n-1}(y_n) = 1$;
for $x \in K_n \subseteq S_n \setminus \mathcal{B}(\mathcal{X}_{\text{goal}}, Mh_n + \zeta_n)$ **do**
    $(v_n(x), u_n(x)) \leftarrow \texttt{VI}(S_n, \tilde{v}_{n-1})$;
for $x \in S_n \setminus \big(K_n \cup \mathcal{B}(\mathcal{X}_{\text{goal}}, Mh_n + \zeta_n)\big)$ **do**
    $v_n(x) = \min_{y \in \mathcal{B}(x, \gamma_{n-1}) \cap S_{n-1}} v_{n-1}(y)$;
for $x \in S_n \cap \mathcal{B}(\mathcal{X}_{\text{goal}}, Mh_n + \zeta_n)$ **do**
    $v_n(x) = \tilde{v}_{n-1}(x)$;

Algorithm 8: $\texttt{VI}(S_n, \tilde{v}_{n-1})$

$U_n \leftarrow U_{n-1} \cup \texttt{Sample}(U, 1)$;
$v_n(x) \leftarrow 1 - e^{-\kappa_n} + e^{-\kappa_n} \max_{c_{m,\alpha_{\text{CI}}} \in D_m} \min_{u \in U_n} \min_{y \in \mathcal{B}(x + h_n f(x,u,d), \gamma_n) \cap S_n} \tilde{v}_{n-1}(y)$;
$u_n(x) \leftarrow$ the solution to u in the above step;

the estimates for the underlying pursuit-evasion game is further refined by solving value iteration once after a new sample and input is added to the corresponding state-space graph and input sets. Interested readers are referred to [152] for more details of the iGame algorithm. The iGame algorithm is anytime and guarantees asymptotic optimality for the robust motion planning with known disturbance bounds. Some of the notations in the iGame algorithm are defined as follows. The state dispersion ζ_n is the quantity such that for any $x \in \mathcal{X}$, there exists $x' \in S_n$ such that $\|x - x'\| \leq \zeta_n$. The other quantities for time discretization are defined as $h_n = \zeta_n^{\frac{1}{1+\gamma}}$ and $\kappa_n = h_n - \zeta_n$.

6.4 Performance Analysis

In this section, we discuss the performance of the data-driven anytime algorithms. The controller is anytime as the iGame algorithm which is used to calculate the system policy is anytime and thus can be terminated anytime after the sampled

Algorithm 9: Confidence Intervals

Input: The observation set $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ and significance level α_{CI}
Output: $\hat{d}_n(x)$ and the expected confidence intervals with significance level $(1 - \alpha)$

1 Given the data $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, calculate $\hat{d}$ using the equation (6.4);
2 Calculate the expectation or bias using equation (6.7);
3 Calculate the residuals as $\hat{\varepsilon}_k = Y_k - \hat{d}_k$, $k = 1, \ldots, n$;
4 Calculate the variance using equation (6.8);
5 Set the significance level $\alpha = \alpha_{\text{CI}}$;
6 Calculate the point-wise confidence intervals using equations (6.9) and (6.10), say $c_{n,\alpha_{\text{CI}}}(x)$;
7 Calculate the set D_n where $D_n(x) = c_{n,\alpha_{\text{CI}}}(x)$;

graph reaches the goal set of the robot. As the robust approach evasion game is solved with a conservative bound on the disturbance function, the existence of optimal policies leading to the goal set is guaranteed from the discriminating kernel of the dynamical system parameterized by the conservative bound. The asymptotic performance of the anytime algorithm is guaranteed under the assumption that the statistical estimates on the disturbance function converge (these estimates converge in probability; thus, we need the convergence of the variance for the estimates described earlier). Suppose that the error bounds on the statistical estimates converge to asymptotic bound denoted by $r^\star$. Then, the the values corresponding to the game parameterized by the sequence of bounds $\{r_k\}_{k\in\mathbb{N}}$ converges to the game with parameter $r^\star$, i.e., the following is true point-wise $\lim_{k\to\infty} \|v^\star_{r_k} - v^\star_{r^\star}\|_{\mathcal{X}} = 0$ (where the norm is defined as $\|v\|_S = \sup_{v\in S} \|v\|$). This follows from Theorem 3.1 in [152]. This shows that our controller is asymptotically optimal and safe, as the solutions are optimal for the underlying approach-evasion games.

During the transient behavior, i.e., when the estimates of the value functions and thus the discriminating kernel estimates are also based on the statistical estimates and the corresponding confidence intervals for deviation from the original function, the guarantees are based on the confidence interval and we provide probabilistic optimality with a high degree of confidence. During the transient phase, our guarantees for safety are based on the probability that the actual disturbance function lies inside the confidence interval provided by the statistical estimator. For example, with a significance level of 99%, there is a 1% chance that the actual disturbance function lies outside the predicted interval. Then, obviously there is a chance that the system might be overestimating the discriminating kernel;

however, the probabilities could be made arbitrarily small. It is expected that under the Lipschitz continuity assumptions on the disturbance function, the statistical estimator can achieve convergence rates on the regression; however, analysis of convergence rates and controller performance with such an estimator is a topic of future research.

6.5 Conclusions and Future Work

For fully autonomous operation of modern robotic systems, like self-driving cars, it is important that these systems be able to use observations to learn to improve performance while retaining safety guarantees. In general, some of the constraints that the system might come across during operation may suffer from lack of accurate models or in some cases, may not have any known model. A common approach is to use robust control which, even though restricts performance, can guarantee system safety. Machine learning offers a gamut of statistics-based approach to approximately model these constraints which can then be used for enhancement of system performance. In this chapter, we presented some initial results for data-driven anytime motion planning of robotic systems where we use observations on system trajectories to statistically estimate the unknown system dynamics which is then used to improve the controller performance while retaining system safety guarantee. The safety guarantees for the system during the transient phase in this paper are based on the confidence intervals provided by the statistical regression algorithm. For better deterministic guarantees, convergence rates for the statistical regression algorithms is required which is apparently elusive except for simple empirical density estimation algorithms [69].

Chapter 7

Conclusions and Future Work

In recent times, AI has seen much progress and is already making big strides in all aspects of our day-to-day lives. Access to large volumes of data made possible by low-cost sensing has led to rapid growth of software-based applications. However, its real potential could be only unleashed by synthesis of complex autonomous systems (e.g., self-driving cars, power-grids, smart buildings, medical haptics, etc.) which is a much more challenging problem. Even though it is still is in an early and nascent stage, it is potentially supposed to revolutionize engineering & science. AI brings together the best of statistical learning theory, machine learning, control theory and theoretical computer science for design and synthesis of systems with a very high degree of autonomy. While the state-of-the-art machine learning techniques have recently been able to achieve low-level representation of objects allowing some low-level autonomy (or higher degrees of automation), a high-degree of autonomy in large-scale complex systems is still a far-fetched goal. For higher degrees of autonomy, much progress is required for a higher-level understanding of the data from, possibly, distributed sources of information to learn complex representations of environment and reasoning.

7.1 Conclusions of Thesis Research

This thesis presented some algorithms and their validation for learning and decision optimization in autonomous systems. While the problems that were discussed in the thesis are mostly open-ended, several improvements for modeling of temporal

data and feedback planning of autonomous systems were presented in the thesis. The overall idea presented in the thesis is to bring together the modern control techniques to the statistical-learning based machine learning techniques (for information feedback). The machine learning module serves of the purpose of filtering and conditioning data from available sensors for decision and control. Several results on modeling, fusion of single as well as multi-dimensional temporal data were discussed. in particular, we have shown that symbolic analysis-based Markov modeling can serve as a useful, computationally-convenient method for representation and learning of time-series data. Several results from various different physical systems (like combustion instability, battery health degradation, target detection, fatigue initiation in alloys structures, etc.). Furthermore, we presented a technique for cross-modeling dependence between two different physical processes which can be used for fusion of multi-modal sensors for learning. Several results for these class of Markov modeling were presented related to selection of hyper-parameters, state-space reduction and their validation were presented.

Furthermore, the thesis also presented a technique for modeling of high-dimensional temporal data using deep learning and Gaussian processes. The proposed idea decomposed the problem of temporal learning in high-dimensional data into the problem of dimensionality reduction using deep learning and modeling the temporality using Gaussian processes. This allows make the best use of deep learning and Gaussian processes by limiting the number of parameters need to be learnt during the training process and thus, helps to get learn small models (which have several computational benefits).

The idea is to then use the information processed by the machine learning module for control using a feedback control system. However, designing optimal feedback controllers for modern autonomous systems is an open problem in literature. To this end, the thesis presents a technique for synthesis of feedback controllers for motion planning of robotic systems using sampling-based algorithms and dynamic programming. The proposed algorithms were analyzed for their performance and their optimality was established. Several extensions of the algorithm to multi-robot motion planning were discussed. Furthermore, for safety of the autonomous systems, an algorithm for safe learning and control in autonomous systems using game theory and statistical learning theory was presented and analyzed for its performance.

7.2 Future Research Directions

The research and ideas presented in the thesis open many new horizons for future research for autonomous systems. In particular, some of the important ones are delineated next.

1. **Optimization of hyper-parameters for Markov modeling time-series**: As the algorithms for order estimation and model inference presented in the thesis earlier discussed, partitions and order (or memory) are the two hyper-parameters required for symbolic representation of time-series data, a simultaneous optimization of the two parameters is required for inference of the right model structure. This problem can also, possibly, be solved using the ideas of model inference from minimum description length (MDL).

   Another important aspect is the scalability of the proposed techniques to higher dimensions. Partitioning a high-dimensional phase space with consistency guarantees is a hard problem and remains the bottleneck for extending the proposed technique to multiple dimensions. Synchronous use of deep learning and Markov models could be one possible direction to solve the multi-dimensional time-series modeling problem.

2. **Causal Modeling using Deep Learning**: Deep Learning is arguably the most successful machine learning algorithm which has recently found large-scale use in software industry. However, its use in systems industry and control is largely unexplored. Moreover, for the complexity of the network architectures, most of the features of the algorithms are not well mathematically understood. Deep Learning remains one of the biggest areas of research in machine learning and there are numerous possibilities of research. However, an important research direction which would make it more reliable to be used with control systems is allowing causally-constrained deep neural networks which can perform causal inference. These models would be able to encode knowledge about existence and non-existence of mechanisms without specification of the underlying mechanism. Furthermore, these networks would be able to learn differential physics-based model without prior knowledge. Another interesting area could be simultaneous optimization of the proposed deep learning and Gaussian process network instead of the composite process

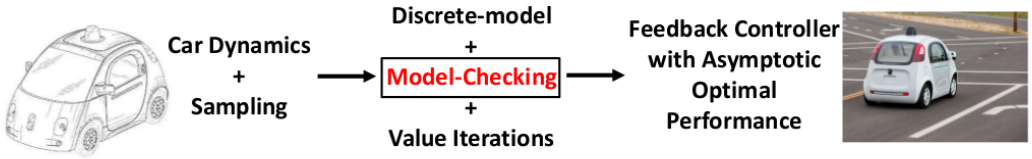


Figure 7.1: Schematic drawing of the the idea for synthesis of feedback controllers for motion planning under complex constraints

introduced in this thesis. Such a network could allow estimation of memory (or history) requirements for modeling of dynamical systems and hopefully, will help reduce overfitting of deep networks.

3. **Feedback Controllers with Complex Specifications** Recent times has seen a growing recognition of the need to use the formal mathematical model checking tools for design and analysis of cyber-physical systems with complex task specifications. Particularly, the increased interest in autonomous driving has led to the use of model checking tools for provably-correct controller synthesis for complex driving tasks. Consequently, there has been an increased interest in synthesizing motion planning controllers for more complex objectives like urban autonomous driving which requires human-like understanding and reasoning of constraints like the traffic rules. However, design and verification of these modern engineered autonomous driving systems, with a tight link between computational and physical elements, have become increasingly complex due to the interleaving between the high-level logics (e.g., the traffic rules) and low-level dynamics (e.g., differential constraints of the car) [180]. Designing even feasible controllers for such systems which can respect the constraints imposed by system dynamics and the specifications is challenging both algorithmically and computationally, leave aside the performance consistency.

   A possible solution idea is shown in figure 7.1 where, an additional model checking module is used to extend and generate a graph which satisfies the complex constraints. Then, value iterations could be performed to recover the value functions in an incremental fashion as was shown in the thesis.

4. **Safety Guarantees in Artificial Intelligence** This thesis discussed an anytime algorithm for motion planning with safety guarantees. The pro-

posed algorithm guarantees optimality as well safety of the underlying system. However, this problem remains largely unexplored where guarantees on convergence rates and computational trade-offs need to be further investigated. Thus future research will explore use of advanced analytic techniques for chance-constrained safe motion planning of intelligent systems. The proposed technique of safe reinforcement learning was based on the idea of value functions, the value-based techniques become intractable in high-dimensional, continuous action and state-spaces. Another common approach in reinforcement learning for robots is direct policy search which by-passes learning of values and learns different parameterization of policies. An important topic of future research is to use the proposed game theoretic approaches for safe policy search.

Appendix A

Instability in Combustion Systems

Combustion in gas turbine engines is a dynamic and complex non-linear phenomenon which requires deep understanding of numerous parameters and boundary conditions of a large multi-physics model, moreover, with various approximations. However, a completely model-driven diagnostics and prognostics of the process is so computationally expensive and sometimes intractable in its entirety, that it results in a poor real-time applicability. So, a data-driven approach is necessary which models multi-modal information from heterogeneous sensors for a holistic system understanding. In recent times, ultra-lean combustion is commonly used for reduction of oxides of nitrogen (NOx) and is susceptible to thermo-acoustic instabilities and lean blowout (LBO). It is well known that occurrence of combustion instabilities and LBO could be detrimental for operations of both land-based and aircraft gas turbine engines. In essence, a sudden decrease in the equivalence ratio or a rapid fluctuation in air flow rate due to inertia gap may lead to LBO in gas turbine engines, which could have serious consequences. This phenomenon calls for a real-time prediction of thermo-acoustic instability and LBO and adoption of appropriate measures to mitigate it.

It is noted that a priori determination of the LBO margin may not be feasible, because of flame fluctuations in the presence of thermoacoustic instability. From this perspective, quantifiable dynamic characteristics of flame preceding blowout have been exploited in the past as LBO precursors by a number of researchers in the combustion research community. De Zilwa *et al.* [181] investigated the occurrence of blowout in dump combustors with and without swirl. Chaudhuri and Cetegen [182,183] investigated the effects of spatial gradients in mixture composition

and velocity oscillations on the blowoff dynamics of a bluff-body stabilized burner that resembles representative of afterburners of gas turbine combustors. Chaudhuri *et al.* [184] and Stohr *et al.* [185] investigated LBO dynamics of premixed and partially premixed flames using combined particle image velocimetry/planar laser-induced fluorescence (PIV/PLIF)-based diagnostics. However, these work did not focus on developing strategies for mitigating LBO. Lieuwen, Seitzman and coworkers [186–189] and Yi and Gutmark [190] used time series data from acoustic and optical sensors for early detection and control of LBO in laboratory-scale gas turbine combustors via identifying blowout parameters using various (e.g., spectral, statistical, wavelet-based and threshold-based) techniques. However, both Lieuwen *et al.* and Gutmark demonstrated their techniques for LBO prediction in premixed combustors. Partially premixed combustion is a more realistic assumption (specially for aircraft gas turbine engine) and it's more challenging to predict LBO for partially premixed combustion due to the lack of precursor events [191]. On the other hand, Mukhopadhyay and co-workers [191,192] developed a number of techniques for early detection of LBO, which worked satisfactorily over a wide range of fuel-air premixing. Chaudhari *et al.* [192] used flame color to device a novel and inexpensive strategy for LBO detection. In a number of recent publications [193–195], the transition to LBO was correlated with changes in signature of the dynamic characteristics of the system by using nonlinear tools of dynamical systems analysis. Mukhopadhyay *et al.* [191] used chemiluminescence time series data for online prediction of LBO under both premixed and partially premixed operating conditions. The algorithm was built upon symbolic time series analysis (STSA), where the parameter D was set to 1 to construct the probabilistic finite state automata (PFSA), which implies that the memory of the underlying combustion process was restricted only to its immediate past state.

Combustion instability is a very undesirable phenomenon characterized by high-amplitude flame oscillations at discrete frequencies. These frequencies typically represent the natural duct/resonator acoustic modes. Combustion instability, in its most basic form arises when there is a positive coupling between the heat release rate oscillations and the pressure oscillations, provided this driving force is higher than the damping present in the system. The mechanisms of pressure-heat release rate coupling are system dependent and thus, the problem of combustion instability becomes very system specific. The complexity of this instability problem accrues

from the mutual interactions among the unsteady heat release rate, flow fields and acoustics, which outline the general features of combustion instability [196–198]. In order to predict and characterize the combustion instabilities, full-scale computational-fluid-dynamic (CFD) models and/or reduced-order models have been developed to identify unstable conditions; however, the simplifying assumptions and inherent complexity in system dynamics as well as computational restrictions may result in imperfect validation with experimental observations. The other approach relies on time series data from physical sensors to identify the features that are capable of exclusively characterizing combustion instabilities and thus, formulates strategies based on analysis of the acquired time series data. The prediction of combustion instability associated with swirl-stabilized flames is of particular interest in gas turbine combustion. The dynamics of three-dimensional swirl-stabilized flames are complex in nature as they are subjected to interactions of fluid-mechanical processes with the flame and acoustics, which give rise to complex growth and saturation mechanisms. Reviews of swirl-stabilized flames and their instabilities have been reported by Huang and Yang [102] and Candel et al. [199]. Significant studies in instabilities of swirl flames have also been conducted in the framework of

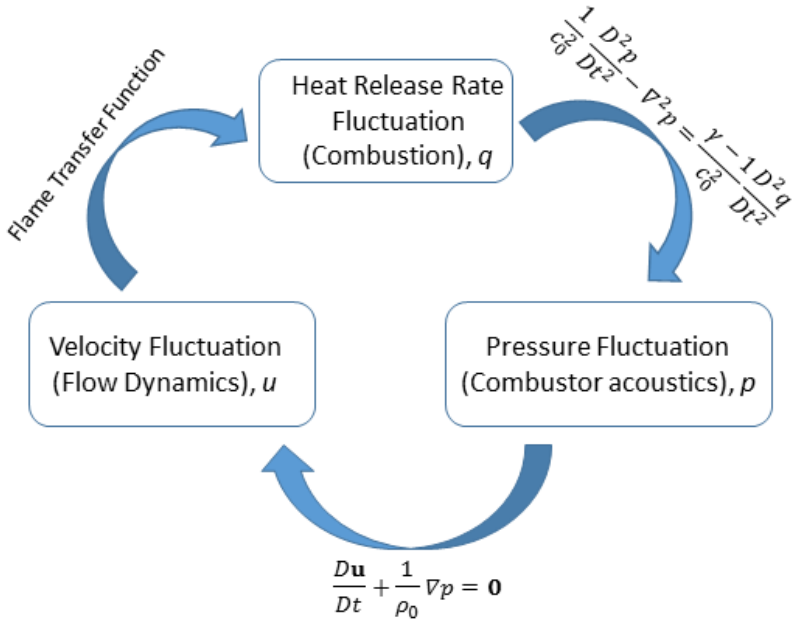


Figure A.1: A simplistic schematic diagram showing the self-excited feedback loop to describe combustion instability.

flame transfer functions in the context of frequency-amplitude dependence and the underlying flow physics [200, 201]. In particular, flame transfer functions provide low-order model-based tools that have been used to predict instabilities by solving the nonlinear dispersion relations as reported by Noiray et al. [202].

Most of the work available on detection and estimation of instability in combustion systems is mainly based on the idea of finding suitable thresholds for declaring unstable behavior under some operating conditions. As such, most of the work present in open literature is not generalizable to bigger sets of operating conditions. Nair and Sujith [203] have used the decay of Hurst exponents as precursors to impending combustion instabilities; other measures of instability like loss of chaotic behavior in the system dynamics have also been formulated [204]. Since the data-driven approach is, in general, expected to predict instabilities across the parameter space, within which the combustor operates. A key parameter is the inlet Reynolds number (Re), a variation of which has been observed to clearly mark the stable and unstable regimes of a combustor. Chakravarthy et al. [205] have reported the onset of "lock-on", a term representing the frequency locking of natural duct acoustics and a dominant hydrodynamic mode as an essential feature of combustion instability. The hallmark feature of Re variations is the break in the dominant frequency, which corresponds to onset of combustion instability. Thus, the frequency break can be thought of as a reliable precursor for combustion instability. However, a major drawback is the bifurcation in the combustor behavior at the frequency break-point; therefore, it cannot be used as a tool of early warning, thus motivating the need for a fast predictive tool that is generic. An experimental investigation on the onset of thermoacoustic oscillations via decreasing fuel equivalence ratio has been reported by Gotoda et al. [206]. This recent study has shown the possibility of encountering low-dimensional chaotic oscillations in combustors by employing several methods of nonlinear time series analysis. Sujith and co-workers [203, 204] showed that combustion noise is chaotic fluctuation of moderately high dimension that undergoes transition to high amplitude oscillations characterized by periodic behavior. This information was used to determine the loss of chaos in the system dynamics as a precursor for prediction of thermoacoustic instability. Recently, Nair et al. [108] have utilized Recurrence Quantization Analysis (RQA) for prediction of instability, which detects a route of intermittency filled with random bursts of high amplitude oscillations. All of these methods generally exploit single sensor

data and sometimes, lack modeling and representation of data which could be misleading when the results need to be generalized. These are also validated on a narrow range of parameter space, which is based sensitive thresholding making them vulnerable to sensor noise. This calls for a generic data-driven approach which can extract nonlinear features from multi-modal sensor data (e.g., pressure transducers and chemiluminescence sensors) and utilize their causality to obtain a real-time predictor of instability, that is robust to sensor noise.

Another hypothesis, along with loss of chaos while the instability kicks in, is the shedding of coherent structure in the flow. Coherent structures are fluid mechanical structures associated with coherent phase of vorticity, high levels of vorticity among other definitions [207]. The interesting case of the natural shedding frequency of these structures, causing acoustic oscillations, has been observed by Chakravarthy et al. [208–210]. Recently, proper orthogonal decomposition (POD) [128] and dynamic mode decomposition (DMD) [129] are used to detect coherent structure, which use tools from spectral theory to derive spatial structural modes. In summary, combustion instability has attracted a lot of attention and is still an active area of research. However, most of the results using data analysis tools are susceptible to dependence on the operating conditions of the process and thus, can not be generalized to bigger sets of operating conditions.

Appendix B

Fatigue Crack Initiation in Polycrystalline alloys

Microstructural degradation (due to fatigue) in polycrystalline alloys during the crack initiation phase is both difficult to sense accurately and suffers from high levels of uncertainty. The life of polycrystalline alloy structures that are subjected to cyclic loading patterns is broadly classified into two phases: 1. *the crack initiation* and 2. *the crack propagation.* This classification implies that there is a phase transition when the micro-structural damage in the form of surface and subsurface deformities (e.g., voids, slip bands, inclusions etc.) develop into micro-cracks; some of these micro-cracks coalesce together to form a macro-crack that propagates under oscillating load [211]. Despite of great success in detection and prediction of cracks during the propagation phase, the crack initiation phase is still an intriguing phenomenon for scientists due to sensing inaccuracies and lack of reliable models. However, for effective life-extending control [212] of mechanical components, it is important to detect and predict changes during the crack-initiation phase as a significant amount of service life of components made of these alloys is spent during the initiation phase.

The fatigue damage evolution is critically dependent on the initial defects present in the materials, which may form crack nucleation sites. A random distribution of micro-structural flaws produce a wide uncertainty in the crack initiation phase even under similar loading conditions. Also, accurate identification and estimation of initial conditions (defects vary from specimen to specimen) is infeasible. The

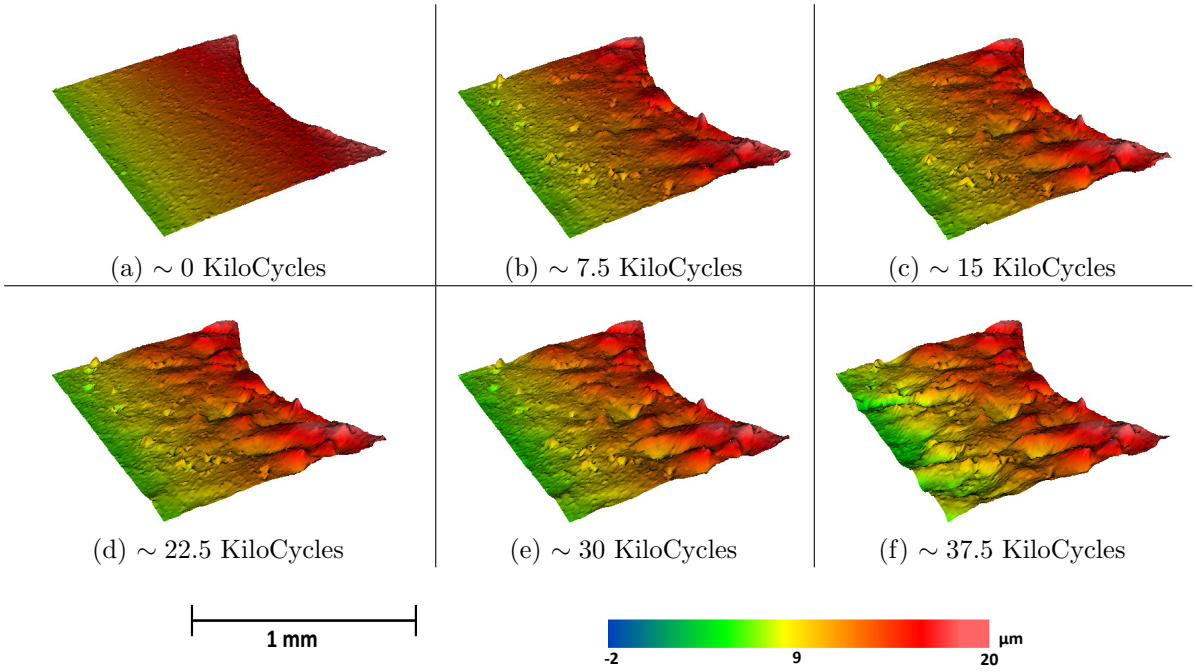


Figure B.1: Evolution of the Surface Profile with Microstructural Damage

current state-of-the-art techniques for material behavior identification rely heavily on empirical modeling, which is acutely insufficient for online monitoring. Furthermore, to the best of author's knowledge, there is not any method reported in literature for prediction and estimation of service life in the crack-initiation phase. Figure B.1 shows the evolution of surface damage during fatigue crack initiation which are measured using a high-precision interferometer. The challenging part of crack detection is to detect it in early stages when it can not be observed on the surface. The ultrasonic sensing provides a low-cost active technique for sensing development of cracks and through processing of the signals obtained by ultrasonic sensors, we show that we can detect crack initiation in early stages. For more details, interested readers are directed to [211].

Many model-based techniques have been reported in recent literature to study the fracture process. For example, Qian et al. [213] presented a x-ray computed micro-tomography study of the ductile fracture process of an aluminum alloy specimen. Zhang et al. [214] investigated the atomistic mechanism of crack-initiation in α-alumina. Stochastic approaches have been reported for modeling 2D crack propagation in heterogeneous materials [215] and characterization of surface profiles [216]. While some studies correlate the mechanical properties with the fractured

surfaces [217] [218], several attempts have been made for damage modeling in the range of short cracks [219] [220]. However, the existing physics-based models can not adequately capture the dynamical behavior of damage evolution in the crack initiation phase.

Due to the lack of applicability of the physics-based models for runtime monitoring, several data-driven methods have been reported based on different sensing methods, such as, eddy currents [221], acoustic emission [222] and ultrasonics [223] [81]. However, the use of these methods is limited to the crack propagation phase.

Bibliography

[1] Vamvoudakis, K. G., P. J. Antsaklis, W. E. Dixon, J. P. Hespanha, F. L. Lewis, H. Modares, and B. Kiumarsi (2015) "Autonomy and machine intelligence in complex systems: A tutorial," in *American Control Conference (ACC), 2015*, IEEE, pp. 5062–5079.

[2] Campbell, M., M. Egerstedt, J. P. How, and R. M. Murray (2010) "Autonomous driving in urban environments: approaches, lessons and challenges," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **368**(1928), pp. 4649–4672.

[3] Gao, Y., A. Gray, H. E. Tseng, and F. Borrelli (2014) "A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles," *Vehicle System Dynamics*, **52**(6), pp. 802–823.

[4] Anderson, S. J., S. C. Peters, T. E. Pilutti, and K. Iagnemma (2010) "An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios," *International Journal of Vehicle Autonomous Systems*, **8**(2-4), pp. 190–216.

[5] Moore, M. and D. Zuby "Collision avoidance features: initial results," .

[6] Liu, C. and M. Tomizuka (2016) "Enabling safe freeway driving for automated vehicles," in *2016 American Control Conference (ACC)*, pp. 3461–3467.

[7] Jha, D., A. Srivastav, and A. Ray (2016) "Temporal Learning in Video data using deep learning and Gaussian processes," *International journal of prognostics and health management*, **7**(022), p. 11.

[8] Sarkar, S., D. Jha, A. Ray, and Y. Li (2015) "Dynamic data-driven symbolic causal modeling for battery performance & health monitoring," in *Information Fusion (Fusion), 2015 18th International Conference on*, IEEE, pp. 1395–1402.

[9] Jha, D. K., A. Srivastav, K. Mukherjee, and A. Ray (2015) "Depth estimation in Markov models of time-series data via spectral analysis," in *American Control Conference (ACC), 2015*, IEEE, pp. 5812–5817.

[10] Li, Y., D. K. Jha, A. Ray, and T. A. Wettergren "Information Fusion of Passive Sensors for Detection of Moving Targets in Dynamic Environments," .

[11] Jha, D. K., T. A. Wettergren, A. Ray, and K. Mukherjee (2015) "Topology optimisation for energy management in underwater sensor networks," *International Journal of Control*, **88**(9), pp. 1775–1788.

[12] Li, Y., D. K. Jha, A. Ray, and T. A. Wettergren (2015) "Feature level sensor fusion for target detection in dynamic environments," in *2015 American Control Conference (ACC)*, IEEE, pp. 2433–2438.

[13] Chattopadhyay, P., D. K. Jha, S. Sarkar, and A. Ray (2015) "Path planning in GPS-denied environments: A collective intelligence approach," in *2015 American Control Conference (ACC)*, IEEE, pp. 3082–3087.

[14] Jha, D., Y. Li, T. Wettergren, and A. Ray (2015) "Robot Path Planning in Uncertain Environments: A Language Measure-theoretic Approach," *ASME Journal of Dyn. Sys., Meas., Control*, **137**, p. 0345003.

[15] Jha, D. K., A. Ray, K. Mukherjee, and S. Chakraborty (2013) "Classification of Two-Phase Flow Patterns by Ultrasonic Sensing," *Journal of Dynamic Systems, Measurement, and Control*, **135**(2), p. 024503.

[16] Jha, D. K., M. Zhu, Y. Wang, and A. Ray (2016) "Data-driven anytime algorithms for motion planning with safety guarantees," in *American Control Conference (ACC), 2016*, American Automatic Control Council (AACC), pp. 5716–5721.

[17] Jha, D. K., P. Chattopadhyay, S. Sarkar, and A. Ray (2016) "Path planning in GPS-denied environments via collective intelligence of distributed sensor networks," *International Journal of Control*, **89**(5), pp. 984–999.

[18] Jha, D., M. Zhu, and A. Ray (2015) "Game Theoretic Controller Synthesis for Multi-Robot Motion Planning-Part II: Policy-based Algorithms," in *5th IFAC Workshop on Distributed Estimation and Control in Networked Systems*, IFAC, p. Accepted for Publication.

[19] Seto, Y., N. Takahashi, D. K. Jha, N. Virani, and A. Ray (2016) "Data-driven robot gait modeling via symbolic time series analysis," in *American Control Conference (ACC), 2016*, American Automatic Control Council (AACC), pp. 3904–3909.

[20] Li, Y., D. K. Jha, A. Ray, and T. A. Wettergren (2016) "Sensor selection for passive sensor networks in dynamic environment: A dynamic data-driven approach," in *American Control Conference (ACC), 2016*, American Automatic Control Council (AACC), pp. 4924–4929.

[21] Sonti, S., N. Virani, D. Jha, K. Mukherjee, and A. Ray (2013) "Language measure-theoretic path planning in the presence of dynamic obstacles," in *American Control Conference (ACC), 2013*, pp. 5110–5115.

[22] Box, G. E., G. M. Jenkins, G. C. Reinsel, and G. M. Ljung (2015) *Time series analysis: forecasting and control*, John Wiley & Sons.

[23] Bishop, C. M. (2006) *Pattern recognition and machine learning*, Springer.

[24] Beim Graben, P. (2001) "Estimating and improving the signal-to-noise ratio of time series by symbolic dynamics," *Physical Review E*, **64**(5), p. 051104.

[25] Daw, C. S., C. E. A. Finney, and E. R. Tracy (2003) "A review of symbolic analysis of experimental data," *Review of Scientific Instruments*, **74**(2), pp. 915–930.

[26] Ray, A. (2004) "Symbolic dynamic analysis of complex systems for anomaly detection," *Signal Processing*, **84**(7), pp. 1115–1130.

[27] Mukherjee, K. and A. Ray (2014) "State splitting and merging in probabilistic finite state automata for signal representation and analysis," *Signal Processing*, **104**, pp. 105–119.

[28] Hirata, Y., K. Judd, and D. Kilminster (2004) "Estimating a generating partition from observed time series: Symbolic shadowing," *Physical Review E*, **70**(1), p. 016215.

[29] Buhl, M. and M. Kennel (2005) "Statistically relaxing to generating partitions for observed time-series data," *Physical Review E*, **71**(4), p. 046213.

[30] Adler, R. (1998) "Symbolic dynamics and Markov partitions," *Bulletin of the American Mathematical Society*, **35**(1), pp. 1–56.

[31] Kennel, M. B. and M. Buhl (2003) "Estimating good discrete partitions from observed data: Symbolic false nearest neighbors," *Physical Review Letters*, **91**(8), p. 084102.

[32] Rajagopalan, V. and A. Ray (2006) "Symbolic Time Series Analysis via Wavelet-based partitioning," *Signal Processing*, **86**(11), pp. 3309–3320.

[33] Lin, J., E. Keogh, L. Wei, and S. Lonardi (2007) "Experiencing SAX: a novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, **15**(2), pp. 107–144.

[34] Sarkar, S., A. Srivastav, and M. Shashanka (2013) "Maximally Bijective Discretization for Data-driven Modeling of Complex Systems," *American Control Conference, Washington, D.C., USA*, pp. 2680–2685.

[35] Virani, N., J.-W. Lee, S. Phoha, and A. Ray (2016) "Information-Space Partitioning and Symbolization of Multi-Dimensional Time-Series Data using Density Estimation," in *American Control Conference (ACC), 2016*, IEEE.

[36] Yang, Y. and G. I. Webb (2002) "A comparative study of discretization methods for naive-bayes classifiers," in *Proceedings of PKAW*, vol. 2002.

[37] Fleuret, F. (2004) "Fast binary feature selection with conditional mutual information," *The Journal of Machine Learning Research*, **5**, pp. 1531–1555.

[38] Liu, H., F. Hussain, C. Tan, and M. Dash (2002) "Discretization: An Enabling Technique," *Data Mining and Knowledge Discovery*, **6**, pp. 393–423.

[39] Garcia, S., J. Luengo, J. A. Saez, V. Lopez, and F. Herrera (2012) "A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning," *IEEE Transactions on Knowledge and Data Engineering*, **99**(PrePrints).

[40] Sarkar, S., P. Chattopdhyay, A. Ray, S. Phoha, and M. Levi (2015) "Alphabet size selection for symbolization of dynamic data-driven systems: An information-theoretic approach," in *American Control Conference (ACC), 2015*, IEEE, pp. 5194–5199.

[41] Steuer, R., L. Molgedey, W. Ebeling, and M. A. Jimenez-Montaño (2001) "Entropy and optimal partition for data analysis," *The European Physical Journal B-Condensed Matter and Complex Systems*, **19**(2), pp. 265–269.

[42] Dougherty, J., R. Kohavi, and M. Sahami (1995) "Supervised and unsupervised discretization of continuous features," in *Machine learning: proceedings of the twelfth international conference*, vol. 12, pp. 194–202.

[43] Pazzani, M. J. (1995) "An Iterative Improvement Approach for the Discretization of Numeric Attributes in Bayesian Classifiers." in *KDD*, pp. 228–233.

[44] Barron, A., J. Rissanen, and B. Yu (1998) "The minimum description length principle in coding and modeling," *Information Theory, IEEE Transactions on*, **44**(6), pp. 2743–2760.

[45] Csiszár, I. and Z. Talata (2006) "Context tree estimation for not necessarily finite memory processes, via BIC and MDL," *Information Theory, IEEE Transactions on*, **52**(3), pp. 1007–1016.

[46] Cover, T. M. and J. A. Thomas (2012) *Elements of information theory*, John Wiley & Sons.

[47] Tong, H. (1975) "Determination of the order of a Markov chain by Akaike's information criterion," *Journal of Applied Probability*, pp. 488–497.

[48] Katz, R. W. (1981) "On some criteria for estimating the order of a Markov chain," *Technometrics*, **23**(3), pp. 243–249.

[49] Csiszár, I. and P. C. Shields (2000) "The consistency of the BIC Markov order estimator," *The Annals of Statistics*, **28**(6), pp. 1601–1619.

[50] Csiszár, I. (2002) "Large-scale typicality of Markov sample paths and consistency of MDL order estimators," *Information Theory, IEEE Transactions on*, **48**(6), pp. 1616–1628.

[51] Morvai, G. and B. Weiss (2007) "On estimating the memory for finitarily Markovian processes," in *Annales de l'Institut Henri Poincare (B) Probability and Statistics*, vol. 43, Elsevier, pp. 15–30.

[52] ——— (2008) "Estimating the lengths of memory words," *Information Theory, IEEE Transactions on*, **54**(8), pp. 3804–3807.

[53] ——— (2013) "Universal Tests for Memory Words," *Information Theory, IEEE Transactions on*, **59**(10), pp. 6873–6879.

[54] Papapetrou, M. and D. Kugiumtzis (2013) "Markov chain order estimation with conditional mutual information," *Physica A: Statistical Mechanics and its Applications*, **392**(7), pp. 1593–1601.

[55] Zhao, L., C. Dorea, and C. Gonçalves (2001) "On determination of the order of a Markov chain," *Statistical inference for stochastic processes*, **4**(3), pp. 273–282.

[56] Merhav, N., M. Gutman, and J. Ziv (1989) "On the estimation of the order of a Markov chain and universal data compression," *Information Theory, IEEE Transactions on*, **35**(5), pp. 1014–1019.

[57] Rajagopalan, V., A. Ray, R. Samsi, and J. Mayer (2007) "Pattern identification in dynamical systems via symbolic time series analysis," *Pattern Recogn.*, **40**(11), pp. 2897–2907.

[58] Srivastav, A. (2014) "Estimating the size of temporal memory for symbolic analysis of time-series data," *American Control Conference, Portland, OR, USA*, pp. 1126–1131.

[59] Chakraborty, S., A. Ray, A. Subbu, and E. Keller (2010) "Analytic signal space partitioning and symbolic dynamic filtering for degradation monitoring of electric motors," *Signal, Image and Video Processing*, **4**(4), pp. 399–403.
URL `http://dx.doi.org/10.1007/s11760-009-0133-4`

[60] Vidal, E., F. Thollard, C. de la Higuera, F. Casacuberta, and R. Carrasco (2005) "Probabilistic Finite-State Machines- Part I," *IEEE Trans. Pattern Anal. Mach. Intell.*, **27**(7), pp. 1013–1025.

[61] ——— (2005) "Probabilistic Finite-State Machines- Part II," *IEEE Trans. Pattern Anal. Mach. Intell.*, **27**(7), pp. 1026–1039.

[62] Van Trees, H. L. (2004) *Detection, estimation, and modulation theory*, John Wiley & Sons.

[63] Poor, H. V. (2013) *An introduction to signal detection and estimation*, Springer Science & Business Media.

[64] Laurence, B. (1993) "A complete sufficient statistic for finite-state Markov processes with application to source coding," *IEEE transactions on information theory*, **39**(3), p. 1047.

[65] Lind, D. and B. Marcus (1995) *An Introduction to Symbolic Dynamics and Coding*, Cambridge University Press.

[66] Virani, N., J.-W. Lee, S. Phoha, and A. Ray (2015) "Learning context-aware measurement models," in *American Control Conference (ACC), 2015*, IEEE, pp. 4491–4496.

[67] Sarkar, S., K. Mukherjee, X. Jin, D. S. Singh, and A. Ray (2012) "Optimization of symbolic feature extraction for pattern classification," *Signal processing*, **92**(3), pp. 625–635.

[68] Virani, N., D. K. Jha, and A. Ray (2016, Submitted) "Sequential Hypothesis Tests Using Markov Models of Time Series Data," in *Workshop on Machine Learning for Prognostics and Health Managament at 2016 KDD, San Francisco, CA*.

[69] Vapnik, V. N. (1998) *Statistical learning theory*, vol. 1, Wiley New York.

[70] Vapnik, V. (2013) *The nature of statistical learning theory*, Springer Science & Business Media.

[71] Poor, H. V. (1994) *An Introduction to Signal Detection and Estimation*, 2nd ed., Springer, New York, NY.

[72] Wald, A. and J. Wolfowitz (1948) "Optimum character of the sequential probability ratio test," *The Annals of Mathematical Statistics*, pp. 326–339.

[73] Shiryaev, A. (1978) *Optimal Stopping Rules*, Springer-Verlag, New York, NY.

[74] Burkholder, D. and R. Wijsman (1963) "Optimum properties and admissibility of sequential tests," *The Annals of Mathematical Statistics*, **34**(1), pp. 1–17.

[75] Bishop, C. M. (2006) "Pattern Recognition," *Machine Learning*.

[76] Grossi, E. and M. Lops (2008) "Sequential detection of Markov targets with trajectory estimation," *Information Theory, IEEE Transactions on*, **54**(9), pp. 4144–4154.

[77] Chen, B. and P. Willett (2000) "Detection of hidden Markov model transient signals," *Aerospace and Electronic Systems, IEEE Transactions on*, **36**(4), pp. 1253–1268.

[78] Fuh, C.-D. (2003) "SPRT and CUSUM in hidden Markov models," *Annals of Statistics*, pp. 942–977.

[79] Papoulis, A. and S. U. Pillai (2002) *Probability, random variables, and stochastic processes*, Tata McGraw-Hill Education.

[80] Wen, Y., K. Mukherjee, and A. Ray (2013) "Adaptive Pattern Classification for Symbolic Dynamic Systems," *Signal Processing*, **93**, pp. 252–260.

[81] Gupta, S., A. Ray, and E. Keller (2007) "Symbolic Time Series Analysis of Ultrasonic Data for Early Detection of Fatigue Damage," *Mechanical Systems and Signal Processing*, **21**(2), pp. 866–884.

[82] Li, Y., P. Chattopadhyay, A. Ray, and C. Rahn (doi: 10.1016/j.jpowsour.2015.03.068) "Identification of the Battery State-of-Health Parameter from Input-Output Pairs of Time Series Data," *Journal of Power Sources*.

[83] Li, Y., D. K. Jha, A. Ray, and T. A. Wettergren (2016) "Information Fusion of Passive Sensors for Detection of Moving Targets in Dynamic Environments," .

[84] Li, Y., D. Jha, A. Ray, and T. Wettergren (2015) "Feature Level Sensor Fusion for Target Detection in Dynamic Environments," in *American Control Conference, Chicago, IL, USA*.

[85] Dasarathy, B. (1997) "Sensor fusion potential exploitation-innovative architectures and illustrative applications," *Proceedings of the IEEE*, **85**(1), pp. 24–38.

[86] Choi, S.-S., S.-H. Cha, and C. C. Tappert (2010) "A survey of binary similarity and distance measures," *Journal of Systemics, Cybernetics and Informatics*, **8**(1), pp. 43–48.

[87] Tibshirani, R., G. Walther, and T. Hastie (2001) "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**(2), pp. 411–423.

[88] MacQueen, J. (1967) "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley symposium on mathematical statistics and probability*, vol. 1, California, USA, pp. 281–297.

[89] Xu, R. and D. Wunsch (2005) "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, **16**(3), pp. 645–678.

[90] Arthur, D. and S. Vassilvitskii (2007) "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, pp. 1027–1035.

[91] Lutkepol, H. (1997) *Handbook of matrices*, Wiley interscience, New York, NY, USA.

[92] Poor, H. (1988) *An Introduction to Signal Detection and Estimation*, Springer-Verlag, New York, NY, USA.

[93] Shalizi, C. R. and K. L. Shalizi (2004) "Blind Construction of Optimal Nonlinear Recursive Predictors for Discrete Sequences," in *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pp. 504–511.

[94] Chakrabarty, A. and J. Langelaan (2013) "UAV flight path planning in time varying complex wind-fields," in *American Control Conference*, pp. 2568–2574.

[95] Vidal, E., F. Thollard, C. de la Higuera, F. Casacuberta, and R. Carrasco (2005) "Probabilistic Finite-State Machines– Part I," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**(7), pp. 1013–1025.

[96] Geiger, B. C., T. Petrov, G. Kubin, and H. Koeppl (2015) "Optimal Kullback–Leibler Aggregation via Information Bottleneck," *Automatic Control, IEEE Transactions on*, **60**(4), pp. 1010–1022.

[97] Vidyasagar, M. (2012) "A metric between probability distributions on finite sets of different cardinalities and applications to order reduction," *Automatic Control, IEEE Transactions on*, **57**(10), pp. 2464–2477.

[98] Xu, Y., S. M. Salapaka, and C. L. Beck (2014) "Aggregation of graph models and Markov chains by deterministic annealing," *Automatic Control, IEEE Transactions on*, **59**(10), pp. 2807–2812.

[99] O'Connor, J., V. Acharya, and T. Lieuwen (2015) "Transverse combustion instabilities: Acoustic, fluid mechanic, and flame processes," *Progress in Energy and Combustion Science*, **49**, pp. 1–39.

[100] Sé, b. Ducruix, T. Schuller, D. Durox, Sé, and b. Candel (2003) "Combustion dynamics and instabilities: Elementary coupling and driving mechanisms," *Journal of Propulsion and Power*, **19**(5), pp. 722–734.

[101] Candel, S., D. Durox, T. Schuller, J.-F. Bourgouin, and J. P. Moeck (2014) "Dynamics of swirling flames," *Annual review of fluid mechanics*, **46**, pp. 147–173.

[102] Huang, Y. and V. Yang (2009) "Dynamics and stability of lean-premixed swirl-stabilized combustion," *Progress in Energy and Combustion Science*, **35**(4), pp. 293–364.

[103] Moeck, J. P., J.-F. Bourgouin, D. Durox, T. Schuller, and S. Candel (2012) "Nonlinear interaction between a precessing vortex core and acoustic oscillations in a turbulent swirling flame," *Combustion and Flame*, **159**(8), pp. 2650–2668.

[104] Banaszuk, A., P. G. Mehta, C. A. Jacobson, and A. I. Khibnik (2006) "Limits of achievable performance of controlled combustion processes," *Control Systems Technology, IEEE Transactions on*, **14**(5), pp. 881–895.

[105] Banaszuk, A., P. G. Mehta, and G. Hagen (2007) "The role of control in design: From fixing problems to the design of dynamics," *Control Engineering Practice*, **15**(10), pp. 1292–1305.

[106] Jha, D. K., A. Srivastav, and A. Ray (2016, Submitted) "Temporal Learning in Video Data using Deep learning and Gaussian Processes," in *Workshop on Machine Learning for Prognostics and Health Managament at 2016 KDD, San Francisco, CA*.

[107] Sarkar, S., S. R. Chakravarthy, V. Ramanan, and A. Ray (2016) "Dynamic data-driven prediction of instability in a swirl-stabilized combustor," *International Journal of Spray and Combustion Dynamics*, p. 1756827716642091.

[108] Nair, V., G. Thampi, and R. Sujith (2014) "Intermittency route to thermoacoustic instability in turbulent combustors," *Journal of Fluid Mechanics*, **756**, pp. 470–487.

[109] Murugesan, M. and R. Sujith (2015) "Combustion noise is scale-free: transition from scale-free to order at the onset of thermoacoustic instability," *Journal of Fluid Mechanics*, **772**, pp. 225–245.

[110] Grunwald, P. (2004) "A tutorial introduction to the minimum description length principle," *arXiv preprint math/0406077*.

[111] Hansen, M. H. and B. Yu (2001) "Model selection and the principle of minimum description length," *Journal of the American Statistical Association*, **96**(454), pp. 746–774.

[112] Xu, R. and D. Wunsch (2005) "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, **16**(3), pp. 645–678.

[113] Akaike, H. (1974) "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, **19**(6), pp. 716–723.

[114] Schwarz, G. (1978) "Estimating the Dimension of a Model," *Ann. Statist.*, **6**(2), pp. 461–464.

[115] Gray, R. M. (1990) *Entropy and information*, Springer.

[116] Marton, K. (1996) "Bounding $\bar{d}$-distance by informational divergence: a method to prove measure concentration," *The Annals of Probability*, **24**(2), pp. 857–866.

[117] "Prognostic Data Repository: Bearing Data Set NSF I/UCRC Center for Intelligent Maintenance Systems, 2010," .
URL `http://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/`

[118] Tobon-Mejia, D. A., K. Medjaher, N. Zerhouni, and G. Tripot (2012) "A data-driven failure prognostics method based on mixture of Gaussians hidden Markov models," *IEEE Transactions on reliability*, **61**(2), pp. 491–503.

[119] Qiu, H., J. Lee, J. Lin, and G. Yu (2006) "Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics," *Journal of sound and vibration*, **289**(4), pp. 1066–1090.

[120] Kim, K., J. Lee, B. Quay, and D. Santavicca (2010) "Response of partially premixed flames to acoustic velocity and equivalence ratio perturbations," *Combustion and Flame*, **157**(9), pp. 1731–1744.

[121] LeCun, Y., Y. Bengio, and G. Hinton (2015) "Deep learning," *Nature*, **521**(7553), pp. 436–444.

[122] Bengio, Y., A. Courville, and P. Vincent (2013) "Representation learning: A review and new perspectives," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **35**(8), pp. 1798–1828.

[123] Hinton, G. E. and R. R. Salakhutdinov (2006) "Reducing the dimensionality of data with neural networks," *Science*, **313**(5786), pp. 504–507.

[124] Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012) "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105.

[125] Lee, H., R. Grosse, R. Ranganath, and A. Y. Ng (2009) "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, pp. 609–616.

[126] Rasmussen, C. E. and C. Williams (2006) *Gaussian processes for machine learning*, The MIT Press, Cambridge, MA, USA.

[127] Jha, D., N. Virani, S. Xiong, and A. Ray (2016, Accepted) "Markov moldeing of time-series data via spectral analysis," in *2016 International Conference on Computational Science*.

[128] Berkooz, G., P. Holmes, and J. L. Lumley (1993) "The proper orthogonal decomposition in the analysis of turbulent flows," *Annual review of fluid mechanics*, **25**(1), pp. 539–575.

[129] Schmid, P. J. (2010) "Dynamic mode decomposition of numerical and experimental data," *Journal of fluid mechanics*, **656**, pp. 5–28.

[130] Sarkar, S., K. G. Lore, S. Sarkar, V. Ramanan, S. R. Chakravarthy, S. Phoha, and A. Ray (2015) "Early Detection of Combustion Instability from Hi-speed Flame Images via Deep Learning and Symbolic Time Series Analysis," .

[131] Sarkar, S., D. Jha, K. Lore, S. Sarkar, and A. Ray (2016, Accepted) "Multimodal Spatiotemporal Information Fusion using Neural-Symbolic Modeling for Early Detection of Combustion Instabilities," in *2016 American Control Conference*.

[132] Sarkar, S., K. G. Lore, and S. Sarkar (2015) "Early detection of combustion instability by neural-symbolic analysis on hi-speed video," in *Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches (CoCo@ NIPS 2015), Montreal, Canada*.

[133] Hauser, M., Y. Li, J. Li, and A. Ray (2016, Accepted) "Real-time Combustion State Identification via Image Processing: A Dynamic Datat-Driven Approach," in *2016 American Control Conference*.

[134] Le Cun Y., B. B., J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1990) "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*.

[135] Reif, J. (1979) "Complexity of the mover's problem and generalizations," in *Proceedings of the 20th Annual IEEE Conference on Foundations of Computer Science*, pp. 421–427.

[136] LaValle, S. M. (2006) *Planning algorithms*, Cambridge university press.

[137] LaValle, S. M. and J. J. Kuffner (2001) "Randomized kinodynamic planning," *The International Journal of Robotics Research*, **20**(5), pp. 378–400.

[138] Kavraki, L., P. Svestka, J.-C. Latombe, and M. Overmars (1996) "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, **12**(4), pp. 566–580.

[139] Karaman, S. and E. Frazzoli (2011) "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, **30**(7), pp. 846–894.

[140] Murray, R. M. and S. S. Sastry (1993) "Nonholonomic motion planning: Steering using sinusoids," *Automatic Control, IEEE Transactions on*, **38**(5), pp. 700–716.

[141] Laumond, J.-P., S. Sekhavat, and F. Lamiraux (1998) *Guidelines in nonholonomic motion planning for mobile robots*, Springer.

[142] Schmerling, E., L. Janson, and M. Pavone (2014) "Optimal sampling-based motion planning under differential constraints: the driftless case," *arXiv preprint arXiv:1403.2483*.

[143] LaValle, S. (2011) "Motion Planning," *Robotics Automation Magazine, IEEE*, **18**(1), pp. 79–89.

[144] Perez, A., R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez (2012) "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, pp. 2537–2542.

[145] Webb, D. J. and J. van den Berg (2013) "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, pp. 5054–5061.

[146] Karaman, S. and E. Frazzoli (2013) "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, pp. 5041–5047.

[147] Tedrake, R., I. R. Manchester, M. Tobenkin, and J. W. Roberts (2010) "LQR-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*.

[148] Tedrake, R. (2009) "LQR-Trees: Feedback motion planning on sparse randomized trees," .

[149] Maeda, G. J., S. P. Singh, and H. Durrant-Whyte (2011) "A tuned approach to feedback motion planning with RRTs under model uncertainty," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, pp. 2288–2294.

[150] Huynh, V. A., S. Karaman, and E. Frazzoli (2012) "An incremental sampling-based algorithm for stochastic optimal control," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, pp. 2865–2872.

[151] Chaudhari, P., S. Karaman, D. Hsu, and E. Frazzoli (2013) "Sampling-based algorithms for continuous-time POMDPs," in *American Control Conference (ACC), 2013*, pp. 4604–4610.

[152] Mueller, E., M. Zhu, S. Karaman, and E. Frazzoli (2013) "Anytime computation algorithms for approach-evasion differential games," *arXiv preprints*.
URL `http://arxiv.org/abs/1308.1174`

[153] Yershov, D., M. Otte, and E. Frazzoli (2015) "Planning for Optimal Feedback Control in the Volume of Free Space," *arXiv preprint arXiv:1504.07940*.

[154] Jean, F., G. Oriolo, and M. Vendittelli (2005) "A globally convergent steering algorithm for regular nonholonomic systems," in *IEEE Conference on Decision and Control*, IEEE, pp. 7514–7519.

[155] Montgomery, R. (2006) *A tour of subriemannian geometries, their geodesics and applications*, 91, American Mathematical Soc.

[156] Sutton, R. and A. Barto (1998) *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, USA.

[157] Chitour, Y., F. Jean, and R. Long (2013) "A global steering method for nonholonomic systems," *Journal of Differential Equations*, **254**(4), pp. 1903–1956.

[158] Janson, L., E. Schmerling, A. Clark, and M. Pavone (2015) "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International Journal of Robotics Research*, p. 0278364915577958.

[159] Majumdar, A. and R. Tedrake (2013) "Robust online motion planning with regions of finite time invariance," in *Algorithmic foundations of Robotics X, Springer Berlin Heidelberg*, pp. 543–558.

[160] Bry, A. and N. Roy (2011) "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, pp. 723–730.

[161] Luders, B., M. Kothari, and J. P. How (2010) "Chance constrained RRT for probabilistic robustness to environmental uncertainty," in *AIAA guidance, navigation, and control conference (GNC), Toronto, Canada*.

[162] Luders, B. D., S. Karaman, and J. P. How (2013) "Robust sampling-based motion planning with asymptotic optimality guarantees," in *AIAA Guidance, Navigation, and Control Conference (GNC), Boston, MA*.

[163] Blackmore, L., M. Ono, A. Bektassov, and B. C. Williams (2010) "A probabilistic particle-control approximation of chance-constrained stochastic predictive control," *Robotics, IEEE Transactions on*, **26**(3), pp. 502–517.

[164] Aubin, J.-P. and H. Frankowska (2009) *Set-valued analysis*, Springer Science & Business Media.

[165] Virani, N. and M. Zhu (2015, Submitted) "Robust Adaptive Motion Planning in the presence of dynamic obstacles," in *2016 American Control Conference*.

[166] Kolter, J. Z. and A. Y. Ng (2009) "Policy search via the signed derivative." in *Robotics: science and systems*.

[167] Abbeel, P. and A. Y. Ng (2004) "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, ACM, p. 1.

[168] Kolter, J. Z., P. Abbeel, and A. Y. Ng (2007) "Hierarchical apprenticeship learning with application to quadruped locomotion," in *Advances in Neural Information Processing Systems*, pp. 769–776.

[169] Poupart, P., N. Vlassis, J. Hoey, and K. Regan (2006) "An analytic solution to discrete Bayesian reinforcement learning," in *Proceedings of the 23rd international conference on Machine learning*, ACM, pp. 697–704.

[170] Bertsekas, D. P. and J. N. Tsitsiklis (1995) "Neuro-dynamic programming: an overview," in *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, vol. 1, IEEE, pp. 560–564.

[171] Aswani, A., H. Gonzalez, S. S. Sastry, and C. Tomlin (2013) "Provably safe and robust learning-based model predictive control," *Automatica*, **49**(5), pp. 1216–1226.

[172] Akametalu, A. K., S. Kaynama, J. F. Fisac, M. N. Zeilinger, J. H. Gillula, and C. J. Tomlin (2014) "Reachability-based safe learning with Gaussian processes," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, IEEE, pp. 1424–1431.

[173] Gillula, J. H. and C. J. Tomlin (2012) "Reducing conservativeness in safety guarantees by learning disturbances online: iterated guaranteed safe online learning," *Robotics: Science and Systems*.

[174] Varaiya, P., R. Elliott, E. Roxin, and N. Kalton (1972) "The existence of value in differential games," *American Mathematical Society*, (126).

[175] De Brabanter, K., J. De Brabanter, J. A. Suykens, and B. De Moor (2011) "Approximate confidence and prediction intervals for least squares support vector regression," *Neural Networks, IEEE Transactions on*, **22**(1), pp. 110–120.

[176] Williams, C. K. and C. E. Rasmussen (1996) "Gaussian processes for regression," .

[177] Chen, S., C. F. Cowan, and P. M. Grant (1991) "Orthogonal least squares learning algorithm for radial basis function networks," *Neural Networks, IEEE Transactions on*, **2**(2), pp. 302–309.

[178] De Brabanter, K. (2011) "Least squares support vector regression with applications to large-scale data: a statistical approach," .

[179] Mueller, E., S. Z. Yong, M. Zhu, and E. Frazzoli (2013) "Anytime computation algorithms for stochastically parametric approach-evasion differential games," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 3816–3821.

[180] Wongpiromsarn, T., U. Topcu, and R. Murray (2012) "Receding Horizon Temporal Logic Planning," *Automatic Control, IEEE Transactions on*, **57**(11), pp. 2817–2830.

[181] Zilwa, S. D., J. H. Uhm, and J. H. Whitelaw (2000) "Combustion oscillations close to the lean flammability limit," *Combustion Science and Technology*, **160**, pp. 231–258.

[182] Chaudhuri, S. and B. M. Cetegen (2008) "Blowoff characteristics of bluff body stabilized conical premixed flames under upstream mixture gradients and velocity oscillations," *Combustion and Flame*, **153**, pp. 616–633.

[183] ——— (2009) "Blowoff characteristics of bluff body stabilized conical premixed flames in a Duct with upstream mixture gradients and velocity oscillations," *Combustion Science and Technology*, **181**, pp. 555–569.

[184] Chaudhuri, S., S. Kostska, M. W. Renfro, and B. M. Cetegen (2010) "Blowoff dynamics of bluff body stabilized turbulent premixed flames," *Combustion and Flame*, **157**, pp. 790–802.

[185] Stohr, M., I. Boxx, C. Carter, and W. Meier (2011) "Dynamics of lean blowout of a swirl-stabilized flame in a gas turbine model combustor," *Proceedings of the Combustion Institute*, **33**, p. 2953 2960.

[186] Nair, S., T. M. Muruganandam, R. Olsen, A. Meyers, J. Seitzman, B. T. Zinn, T. Lieuwen, T. Held, and H. Mongia (2004) "Lean blowout detection in a single nozzle swirl cup combustor," in *42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, USA*, pp. AIAA 2004–0138.

[187] Muruganandam, T. M., S. Nair, D. Scarborough, Y. Neumeier, J. Jagoda, T. Lieuwen, J. Seitzman, and B. T. Zinn (2005) "Active control of lean blowout for turbine engine combustors," *Journal of Propulsion and Power*, **21**, pp. 807–815.

[188] Nair, S. and T. Lieuwen (2005) "Acoustic detection of blowout in premixed flames," *Journal of Propulsion and Power*, **21**, pp. 32–39.

[189] Prakash, S., S. Nair, T. M. Muruganandam, Y. Neumeier, T. Lieuwen, J. M. Seitzman, and B. T. Zinn (2005) "Acoustic based rapid blowout mitigation in a swirl stabilized combustor," in *ASME Turbo Expo, Reno, USA*.

[190] Yi, T. and E. J. Gutmark (2007) "Real-time prediction of incipient lean blowout in gas turbine Combustors," *AIAA Journal*, **45**, pp. 1734–1739.

[191] Mukhopadhyay, A., R. Chaudhuri, T. Paul, S. Sen, and A. Ray (2013) "Prediction of Lean Blow-out in Gas Turbine Combustors using Symbolic Time Series Analysis," *Journal of Propulsion and Power (AIAA)*, **29**, pp. 950–960.

[192] Chaudhari, R. R., R. Sahu, S. Ghosh, A. Mukhopadhyay, and S. Sen (2013) "Flame Color as a Lean Blowout Predictor," *International Journal of Spray and Combustion Dynamics*, **5**, pp. 49–66.

[193] Kabiraj, L., R. Sujith, , and P. Wahi (2012) "Investigating dynamics of combustion driven oscillations leading to lean blowout," *Fluid Dynamics Research*, **44**.

[194] Kabiraj, L. and R. I. Sujith (2012) "Nonlinear self-excited thermoacoustic oscillations: intermittency and flame blowout," *J. Fluid Mech.*, **713**, pp. 376–397.

[195] Gotoda, H., Y. Shinoda, M. Kobayashi, Y. Okuno, and S. Tachibana (022910 (2014)) "Detection and control of combustion instability based on the concept of dynamical system theory," *Phys. Rev. E*, **89**.

[196] Lieuwen, T., H. Torres, C. Johnson, and B. T. Zinn (2000) "A Mechanism of Combustion Instability in Lean Premixed Gas Turbine Combustors," *Journal of Engineering for Gas Turbines and Power*, **123**(1), pp. 182–189.

[197] McManus, K., T. Poinsot, and S. M. Candel (1993) "A review of active control of combustion instabilities," *Progress in Energy and Combustion Science*, **19**(1), pp. 1–29.

[198] Dowling, A. P. and S. Hubbard (2000) "Instability in lean premixed combustors," *Journal of power and energy*, **214**(4), pp. 317–332.

[199] Candel, S., D. Durox, T. Schuller, J. F. Bourgouin, and J. P. Moeck (2014) "Dynamics of swirling flames," *Annual review of fluid mechanics*, **46**, pp. 147–173.

[200] Palies, P., T. Schuller, D. Durox, and S. Candel (2011) "Modelling of premixed swirling flame transfer functions," *Proceedings of the combustion institute*, **33**(2), pp. 2967–2974.

[201] Bellows, B., M. Bobba, A. Forte, J. Seitzman, and T. Lieuwen (2007) "Flame transfer function saturation mechanisms in a swirl stabilized combustor," *Proceedings of the combustion institute*, **31**(2), pp. 3181–3188.

[202] Noiray, N., D. Durox, T. Schuller, and S. Candel (2008) "A unified framework for nonlinear combustion instability analysis based on flame describing function," *Journal of Fluid Mechanics*, **615**, pp. 139–167.

[203] Nair, V. and R. I. Sujith (2014) "Multifractality in combustion noise: predicting an impending combustion instability," *J. Fluid Mech.*, **747**, pp. 635–655.

[204] Nair, V., G. Thampi, S. Karuppusamy, S. Gopalan, and R. I. Sujith (2013) "Loss of chaos in combustion noise as a precursor of impending combustion instability," *International Journal of Spray and Combustion Dynamics*, **5**, pp. 273–290.

[205] Chakravarthy, S. R., R. Sivakumar, and O. J. Shreenivasan (2007) "Vortex acoustic lock on in bluff body and backward facing step combustors," *Sadhana*, **32**, pp. 145–154.

[206] Gotoda, H., H. Nikimoto, T. Miyano, and S. Tachibana (2011) "Dynamic properties of combustion instability in a lean premixed gas-turbine combustor," *Chaos*, **21**, p. 013124.

[207] Hussain, A. K. M. F. (1983) "Coherent structures - Reality and myth," *Physics of Fluids*, **26**(10), pp. 2816–2850.

[208] Chakravarthy, S. R., O. J. Shreenivasan, B. Bhm, A. Dreizler, and J. Janicka (2007) "Experimental characterization of onset of acoustic instability in a nonpremixed half-dump combustor," *Journal of the Acoustical Society of America*, **122**, p. 120127.

[209] Syred, N. (2006) "A Review of Oscillation Mechanisms and the Role of Precessing Vortex Core (PVC) in Swirl Combustion Systems," *Progress in Energy and Combustion Science*, **32**(2), p. 93161.

[210] Paschereit, C. O., E. Gutmark, , and W. Weisenstein (1998) "Structure and Control of Thermoacoustic Instabilities in a Gas Turbine Combustor," *Combustion Science and Technology*, **138**(1-6), p. 213232.

[211] Jha, D. K., D. S. Singh, S. Gupta, and A. Ray (2012) "Fractal analysis of crack initiation in polycrystalline alloys using surface interferometry," *EPL (Europhysics Letters)*, **98**(4), p. 44006.

[212] Tangirala, S., M. Holmes, A. Ray, and M. Carpino (1998) "Life-extending control of mechanical structures: Experimental verification of the concept," *Automatica*, **34**(1), pp. 3–14.

[213] Qian, L. and H. Toda (2005) "Application of synchrotron x-ray microtomography to investigate ductile fracture in Al alloys," *Applied Physics Letters*, **87**.

[214] Zhang, C., R. Kalia, A. Nakano, and P. Vashishta (2007) "Fracture initiation mechanisms in alpha-alumina under hypervelocity impact," *Applied Physics Letters*, **91**.

[215] Katzav, E., M. Adda-Bedia, and B. Derrida (2007) "Fracture surfaces of heterogeneous materials: A 2D solvable model," *Europhysics Letters*, **78**(4).

[216] Waechter, M., F. Riess, H. Kantz, and J. Peinke (2003) "Stochastic analysis of surface roughness," *Europhysics Letters*, **64**(5), pp. 579–585.

[217] Herrmann, H., J. Kertsèsz, and L. Arcangelis (1989) "Fractal shapes of deterministic cracks," *Europhysics Letters*, **10**(2), pp. 147–152.

[218] Gabrielli, A., R. Cafeiro, and G. Caldarelli (1999) "Statistical properties of fractures in damaged materials," *Europhysics Letters*, **45**(1), pp. 13–19.

[219] Ishihara, S. and A. McEvily (2002) "Analysis of short fatigue crack growth in cast aluminum alloys," *International Journal of Fatigue*, **24**(11), pp. 1169–1174.

[220] Bjerkén, C. and S. Melin (2003) "A tool to model short crack fatigue growth using a discrete dislocation formulation," *International Journal of Fatigue*, **25**(6), pp. 559–566.

[221] Zilberstein, V., K. Walrath, D. Grundy, D. Schlicker, N. Goldfine, E. Abramovici, and T. Yentzer (2003) "MWM eddy-current arrays for crack initiation and growth monitoring," *International Journal of Fatigue*, **25**(9), pp. 1147–1155.

[222] Lysak, M. (1996) "Development of the theory of acoustic emission by propagating cracks in terms of fracture mechanics," *Engineering Fracture Mechanics*, **55**(3), pp. 443–452.

[223] Rokhlin, S. and J.-Y. Kim (2003) "In situ ultrasonic monitoring of surface fatigue crack initiation and growth from surface cavity," *International journal of fatigue*, **25**(1), pp. 41–49.

Vita

Devesh Kumar Jha

Devesh Jha was born in India on October 1st 1987. He graduated with B.E. in Mechanical Engineering from Jadavpur University, Kolkata, India in June, 2010. Devesh joined the Mechanical Engineering Department at Penn State in August 2010. He got concurrent Master degrees in Mathematics and Mechanical Engineering in May, 2016. Devesh was a research intern at Mitsubishi Electric Research Laboratories, Cambridge, MA from June 2015 to September 2015 where he was working on feedback motion planning of nonholonomic systems for automatic driving. His research interests are in the areas of Machine Learning, Deep Learning, Time Series Analysis and Motion Planning in Robotics. Based on his research at Penn State, he has already published more than 25 papers in archival journals and conference proceedings. He was a recipient of a best paper award at a workshop on Machine Learning for Prognostics and Health Management at KDD 2016. He has also been the recipient of several travel grants to conferences by IEEE, ASME and the Department of Mechanical Engineering, Penn State. As a graduate student at Penn State, Mr. Jha has successfully led multi-disciplinary teams in multiple research projects sponsored by Air Force Office of Scientific Research (AFOSR) and Office of Naval Research (ONR). He is a student member of IEEE, ASME and RAS. When not working, he likes to be outdoors and take pictures.