Robust Learning of Probabilistic Hybrid Models

by

Stephanie Gil

B.S., Cornell University (2006)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2008

Author . . . . . .
Department of Aeronautics and Astronautics
September 4, 2008

Certified by . . . . . .
Brian Williams
Professor
Thesis Supervisor

Accepted by . . . . . .
Prof. David L. Darmofal
Associate Department Head
Chair, Department Committee on Graduate Theses

Robust Learning of Probabilistic Hybrid Models

by

Stephanie Gil

Submitted to the Department of Aeronautics and Astronautics
on September 4, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Advances in autonomy, in the fields of control, estimation, and diagnosis, have improved immensely, as seen by spacecraft that navigate toward pinpoint landings, or speech recognition enabled in hand-held devices. Arguably the most important step to controlling and improving a system, is to understand that system. For this reason, accurate models are essential for continued advancements in the field of autonomy. Hybrid stochastic models, such as JMLS and LPHA, allow for representational accuracy of a general scope of problems. The goal of this thesis is to develop a robust method for learning accurate hybrid models automatically from data. A robust method should learn a set of model parameters, but should also avoid convergence to locally optimal solutions that reduce accuracy, and should be less sensitive to sparse or poor quality observation data. These three goals are the focus of this thesis.

We present the HML-LPHA algorithm that uses approximate EM for learning maximum likelihood model parameters of LPHA, given a sequence of control inputs $\{u\}_0^T$, and outputs, $\{y\}_1^{T+1}$. We implement the algorithm in a scenario that simulates the mechanical wheel failure of the MER Spirit rover wheel and demonstrate empirical convergence of the algorithm.

Local convergence is a limitation of many optimization approaches for multimodal functions, including EM. For model learning, this can mean a severe compromise in accuracy. We present the kMeans-EM algorithm, that iteratively learns the locations and shapes of explored local maxima of our model likelihood function, and focuses the search away from these areas of the solution space toward undiscovered maxima that are promising apriori. We find the kMeans-EM algorithm demonstrates iteratively increasing improvement over a Random Restarts method with respect to learning sets of model parameters with higher likelihood values, and reducing Euclidean distance to the true set of model parameters.

Lastly, the AHML-LPHA algorithm is an active hybrid model learning approach that augments sparse, and/or very noisy training data, with limited queries of the discrete state. We use an active approach for adding data to our training set, where we query at points that obtain the greatest reduction in uncertainty of the distribution over the hybrid state trajectories. Empirical evidence indicates that querying only 6% of the time reduces continous state squared error and MAP mode estimate error of the discrete state. We also find that when the passive learner, HML-LPHA, diverges due to poor initialization or training data, the AHML-LPHA algorithm is capable of convergence; at times, just one query allows for convergence, demonstrating a vast improvement in learning capacity with a very limited amount of data augmentation.

Thesis Supervisor: Brian Williams
Title: Professor

Acknowledgments

I would like to thank my advisor Brian Williams, and the generous fellowships, the Bell Labs Graduate Research Program and the NSF Graduate Research Program, that have supported my graduate work and have made my degree a possibility. I would also like to thank other academic mentors that have provided amazing guidance and impetus when I needed it most, and have provided me with amazing opportunities, Thank You to Suresh Goyal, Jim Bell, Dave Mangus, Tom Stengle, John Callas, and Nicholas Zabaras.

My warmest gratitude is for my family who have made everything I do a possibility. Mom, Dad, and Jenny, I feel like I can do anything with you behind me and I'll try to remind you every day that what I accomplish is because of you. A la patota entera de los Gil y los Gómez, sé que siempre tengo su apoyo, ya sea cuando llenaron medio salón en mi primera conferencia, o para mi último grado donde tuvimos que alquilar un bús para que todos queparan. Los adoro mucho, y soy muy afortunada y estoy muy agradecida de tener una familia tán preciosa.

Emmanuel, thank you for all of your support and guidance. In many ways, you were also a mentor to me and I'm really happy for that, oh and of course, thank you for the spaghetti. Lars, you were a really big help in getting me started on this master's work and in the group. To others in the group, particularly Paul Robertson, thank you for being around when I needed to talk about research or otherwise.

To all of the wonderful friends that I've been fortunate enough to meet over the last two years, Hoda, Miriam, Steph, Sandra, Antonio,Yvonne, and Sam, you guys have filled my days, nights, and weekends with endless entertainment and amazing memories.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Increased attention to pushing the envelop in the field of autonomous systems has led to an explosion of more capable systems. In recent history we have seen space missions that owe their successful landings to onboard systems that pinpoint landing targets, advances in speech recognition that allow us to communicate with GPS systems or even dial contacts on our mobile phones, and autonomous navigation systems that can steer unmanned underwater to areas of scientific interest.

Central to the improved performance and accuracy of these systems is the ability to handle the uncertainties inherent to the environment they work in. In order for autonomy to be useful in a practical setting, these systems must be capable of handling disturbances, component failures, unknown system dynamics, and unknown locations of science targets and/or obstacles. In addition, autonomous systems often cannot directly observe their own state resulting in uncertainty in its current state estimates. The answer to many of these challenges comes via a more flexible and representative model of the environment and of the system itself.

Stochastic models account for the uncertainty of real-world environments in which these systems must operate. Furthermore, hybrid stochastic models are sufficient to model a general class of systems that exhibit both continuous and discrete dynamic behaviors. Jump Markov Linear Systems (JMLS) are a class of models that model transitions between discrete states as a Hidden Markov Model (HMM) where the transitions are stochastic and assume the Markov property. The continuous state is modeled by a set of Linear Differential Equations assigned to each discrete state.

However, these models assume independence between discrete state transitions and the continuous state, which, is often limiting. A schematic drawing of a JMLS for a system with two discrete modes is shown in Figure 1-1.

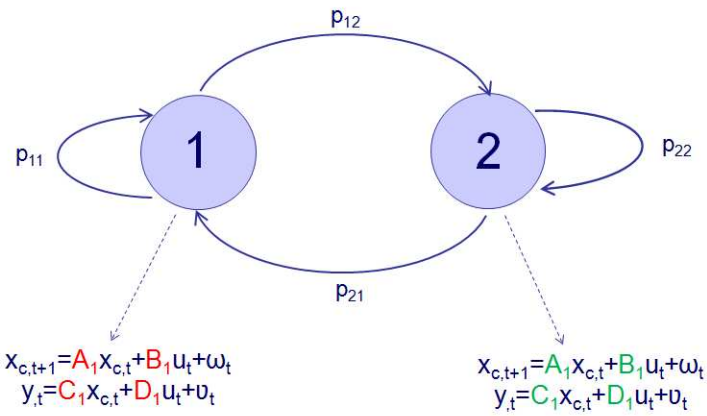

Figure 1-1: A schematic representation of a JMLS with two discrete modes.

In many cases the continuous state of a system may actually have a large influence on the discrete state. This was the case for one of the NASA 2003 Mars Exploration Rover (MER) vehicles, Spirit, which experienced intermittent mechanical failure in one of its wheels, Figure 1-2. The wheel would transition between two operational modes where sometimes it would stick, and other times it would work nominally. It was found that driving the wheel backwards would lessen its tendency to stick. A JMLS model of the rover wheel subsystem would be capable of modeling the two discrete modes, **nominal** and **stuck**, as well as the continuous dynamics of the angular velocity of the wheel, $\dot{\theta}$, but would *not* model the higher probability of transitioning into the **stuck** mode during forward operation of the wheel, when $\dot{\theta} < 0$.

We can extend JMLS to include discrete mode transition dependence on the continuous state. We refer to these models as Linear Probabilistic Hybrid Automata (LPHA). Figure 1-3 shows a schematic of the rover wheel subsystem modeled as a LPHA where the transition probabilities between discrete states are conditioned on the continuous state $\dot{\theta}$. We call this conditioning a *guard*, C, that is satisfied, $C = T$, or not satisfied, $C = F$. In this example the guard is of the form of an inequality where $C = T$ if $\dot{\theta} < 0$.

Because of their generality and ability to accurately represent such a wide spectrum of physical phenomena, JMLS and LPHA have enjoyed the attention of research in areas as diverse as activity

Figure 1-2: Image of the NASA Mars Exploration Rover, Spirit, which experienced mechanical failure in one of its wheels causing it to stick. Photo credit: NASA/JPL-Caltech/Cornell

recognition in machine vision, fault detection and diagnosis, and econometrics [10, 32, 31, 21, 7, 19]. However, this improvement comes at significant cost - the difficulty of modeling. This is a substantial barrier that research in these areas must tackle before benefiting from the great potential in representation accuracy and increased robustness to uncertainty offered by this rich class of models and the algorithms that employ them.

Acquiring an accurate model of complex systems is a challenging, and often times manual, task. These systems also change or degrade with time such that the initial specifications of the discrete and continuous behaviors become obsolete. Manually re-deriving accurate models can be expensive and even impossible in most cases, particularly if the system itself is overly complex or has limited accessibility, as in the case of the MER rover. As a result, we loose the ability for high fidelity control of the system, along with many other important capabilities that rely on accurate system models. With the Spirit wheel failure, scientists and engineers at JPL had to devise new methods of driving the rover with a handicapped wheel by trial and error in a mock-up testbed; while loosing precious time and project money that would have been better spent exploring science targets on Mars. What we would like to do, is to be able to autonomously learn hybrid system models from input and output data. This thesis addresses the challenge of model learning for LPHA by introducing methods for learning these models automatically from data.

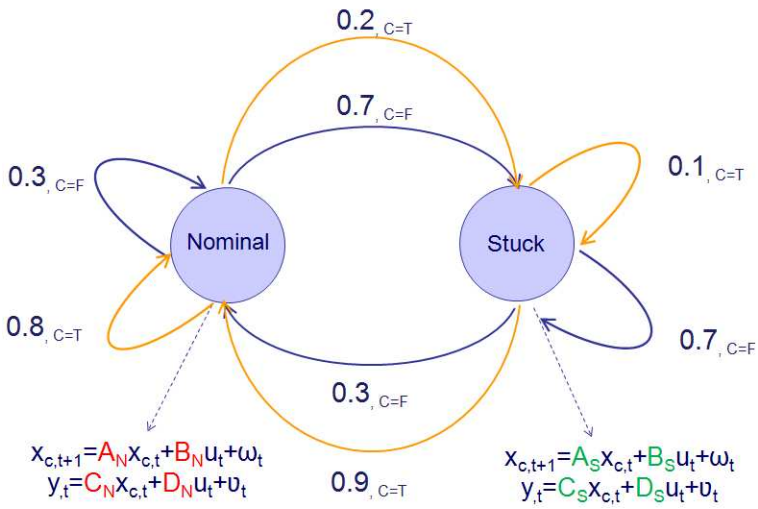

Figure 1-3: Schematic drawing of an LPHA model for MER rover's wheel subsystem. The two discrete modes represent the **nominal** mode and the failed **stuck** mode.

1.2 Background

The key factors that make the problem of hybrid model learning most challenging are 1) partially observable state 2) noisy observations and 3) multi-dimensional model parameter space. Previous work addressed the issue of hybrid state estimation for JMLS [14, 29] and LPHA [19, 18]. An Active Estimation approach extended this work to incorporate the addition of control sequences that would further disambiguate the current hybrid state, thus making the hybrid estimation problem easier [7, 28]. Hybrid estimation itself is an important area for localization, fault detection, diagnosis, and is an integral part of hybrid model learning. An Expectation Maximization Approach for model learning of non-switching Linear Dynamical Systems (LDS) was introduced where the continuous model parameters for these systems is learned automatically from data [35, 15]. This approach was later extended to the JMLS case [16, 11, 4].

Model learning and estimation for JMLS and LPHA is particularly difficult because of the magnitude of the hybrid trajectory space. The number of unique hybrid states grows exponentially in time, making any exact learning algorithm for the model parameters intractable with current computational limitations. In response to this difficulty, existing hybrid model learning techniques are forced to incorporate approximations and sometimes tracking of only the most likely state trajectory [18, 9, 16].

1.3 Contributions

The goal of this thesis is to present a robust learning capability for hybrid discrete-continuous systems. A robust method should learn a set of model parameters, but should also avoid convergence to locally optimal solutions that reduce accuracy, and should be less sensitive to sparse or poor quality observation data. These three goals are the focus of this thesis. We develop:

1. A principled approach to LPHA model learning based on approximate, soft-EM.

2. A guided restart method that avoids getting stuck in locally optimal solutions, and instead explores new hills by learning the bounds of its explored parameter space, converging to optimal or near-optimal learned model solutions.

3. A query based active learning method for augmenting sparse or very noisy observation data that would otherwise lead to poor quality of the learned LPHA.

We present a hybrid model learning algorithm that we refer to as HML-LPHA. This algorithm yields the maximum likelihood parameter estimates for both the discrete and continuous model parameters. This approach extends previous approaches in that it is capable of handling the dependence of discrete transitions on the continuous state, learning the dependence on continuous control inputs, and uses an k-best approach that tracks several most likely hybrid state trajectories, referred to as *soft* EM. However, similar to many other hybrid model learning capabilities and EM-based optimization methods, it is susceptible to convergence to local maxima of the log-likelihood function. This can lead to severe compromises of accuracy of the learned model, particularly if the starting guess of parameters is far from the global maximum and the objective function has many local maxima.

To address the limitation of local convergence, we develop an algorithm that iteratively learns the shapes and locations of explored local maxima of the log-likelihood function, and uses this map to focus the search away from visited areas of the solution space. By identifying new sets of model parameters that seem a priori promising, and that are outside of the bases of explored hills, we avoid being stuck in any single local maximum and target learned models that are the optimal, or near-optimal solutions to the maximization of the likelihood function. We present this algorithm as the kMeans-EM algorithm.

Another limitation of many hybrid learning capabilities is a reduction in quality of a learned model due to sparsity of training data, and/or very noisy observations. In these cases, the accuracy of the learned models can suffer substantially, and in the worse case, the learning algorithm can diverge altogether. Our final technical contribution focuses on this problem by augmenting sparse or poor quality data with a modest amount of labeled data obtained via queries of the discrete state of the LPHA. These queries are limited and are carefully chosen to be only at time instances when the action of querying the discrete state reduces uncertainty in the state estimate the most. The ability of the learner to choose which data to add to its training set is referred to as Active Learning [12]. We develop an active model learning approach for hybrid systems that we refer to as the Active Hybrid Model Learning for LPHA (AHML-LPHA) algorithm.

In the following sections we summarize the contributions of this thesis by introducing the three algorithms HML-LPHA, kMeans-EM, and AHML-LPHA, the precise problem statement of each of these and pertinent results.

1.3.1 Hybrid Model Learning for LPHA

Having an accurate model for a hybrid system is necessary for making inferences, controlling, or task planning for that system. As motivated in Section 1.1 with the MER rover, losing an accurate representation of the system we are trying to control can result in wasted resources such as project time or money, and in the worse case, total loss of the system. Our objective for the HML-LPHA algorithm is to automatically learn the continuous and discrete model parameters, $\theta = [\theta_c \cup \theta_d]$, for *Linear Probabilistic Hybrid Automata*, or JMLS with Autonomous Mode Transitions, from noisy data. Figure 1-3 demonstrates the MER rover wheel subsystem modeled as a LPHA. Our problem statement becomes:

Problem Statement: Given a set of continuous observations $\mathbf{y_1^{T+1}}$ *and a sequence of control inputs* $\mathbf{u}_0^T$*, determine the set of model parameters that maximize the likelihood of the data,* $f(\theta) = p\left(\mathbf{y}_1^{T+1}|\theta\right)$*. Where* $\mathbf{y_1^{T+1}}$ *is the observation sequence* y *from time* $t = 1$ *to* $t = T + 1$*. More specifically, find*
$\theta = \arg\max_\theta p\left(\mathbf{y}_1^{T+1}|\theta\right)$

We perform this objective by employing an iterative local optimization technique, Expectation

Maximization (EM). The EM algorithm is used to find a set of model parameters that maximizes the likelihood of its probabilistic model. This algorithm is particularly useful for situations where direct maximization of the objective function is hard or impossible due to the presence of unobserved (latent) variables. This is the case for many non-trivial maximization problems. The algorithm treats the available data (observations $\mathbf{y}$) as incomplete. It assumes that the *complete data* $\mathbf{z}$ is partially hidden and is generated by the probability distribution $p(\mathbf{z}|\theta)$. In our application for hybrid model learning, the latent variables are the hidden hybrid states $x_t = [x_d, x_c]_t$ over the time trajectory that we are interested in, and the objective function is $f(\theta)$. The EM algorithm is composed of an initialization stage and two steps that are iterated until convergence to a local maximum of $f(\theta)$:

1. **Initialization:** Initialize the current guess of parameters $\theta^k, k = 0$

   Iterate until convergence of the log-likelihood:

2. **Expectation Step:** Given θ^k, calculate the lower bound $h\left(\theta|\theta^k\right) = E\left[\log p\left(\mathbf{z}|\theta\right)|\mathbf{y}, \theta^k\right]$

3. **Maximization Step:** Given the lower bound $h\left(\theta|\theta^k\right)$, maximize to find a new guess of parameters θ^{k+1}
   $\theta^{k+1} = \arg\max_\theta h\left(\theta|\theta^k\right)$
   $k = k + 1$

Figure 1-4 shows the iteration steps in EM for convergence. At each iteration, k, a lower bound to the objective function $g(\theta)$ is constructed in the E-step. The lower bound $h(\theta|\theta^k)$ is maximized in the M-step to find a new guess of model parameters, θ^{k+1} for the next iteration. This process repeats until convergence to a local maximum of $g(\theta)$ is achieved.

Because the number of unique hybrid state trajectories grows exponentially in time, exact hybrid estimation in the E-step becomes intractable, instead we employ an approximate E-step. Many hybrid estimation methods in the literature approximate by either collapsing or pruning the possible trajectories [19, 9, 16]. We use an k-best approach that tracks only the k most likely hybrid state trajectories and assigns the probability of all other trajectories to zero.

We find that in practice, the majority of the probability mass lies within the first few trajectories and thus in many cases k-best is a good approximation. In fact, our empirical results show that the

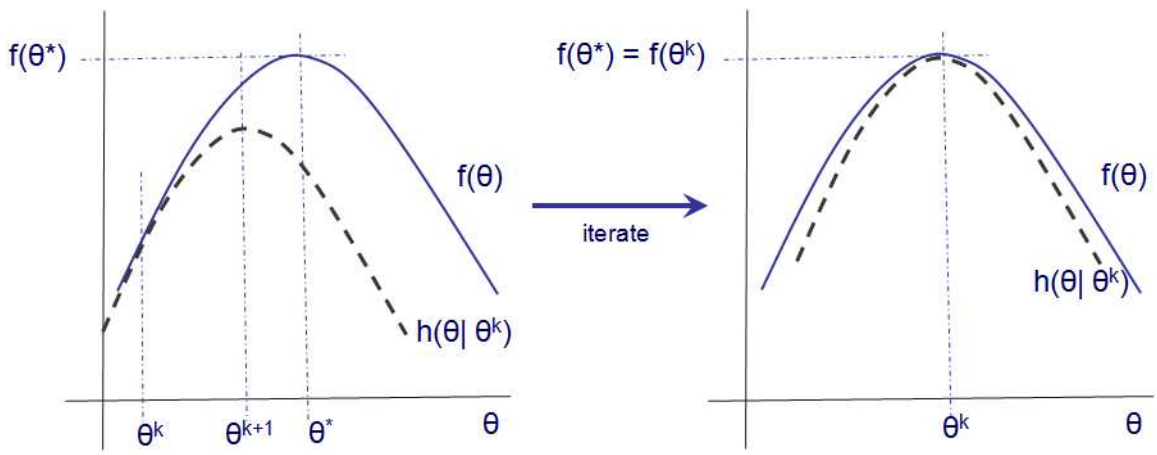

Figure 1-4: Figure demonstrating convergence of EM to local maximum of $f(\theta)$.

number, k, of trajectories does not have a significant influence on squared error in the continuous state or MAP mode estimation error.

The approximate EM method does not share the proof of local convergence as in classical EM, due to the approximation over the distribution of the hybrid state. However, we do show empirical convergence of the algorithm through a tracking the change in the lower bound at each iteration. The HML-LPHA algorithm unfortunately shares some of the less desirable properties inherent to many non-convex optimization techniques, including convergence to local maxima and sensitivity to initialization.

We implement our HML-LPHA algorithm on a simulation scenario inspired by the intermittent mechanical failure of the wheel on the MER rover Spirit. We model the wheel subsystem as a LPHA as shown in Figure 1-3. We use an autonomous guard conditioned on the angular velocity of the wheel to reflect the dependence of the discrete mode transition on the direction of motion of the wheel.

1.3.2 K-Means Clustering for Jumping out of Local Maxima in EM

Many optimization methods for multi-modal functions, where the function itself has many local maxima, are susceptible to local convergence and sensitivity to initialization. Often times a locally optimal solution can be far from the global maximum. For model learning of hybrid systems, this can result in a severe compromise in accuracy and a learned set of model parameters that are of little practical use. Local convergence and sensitivity to initialization proved to be the case for our

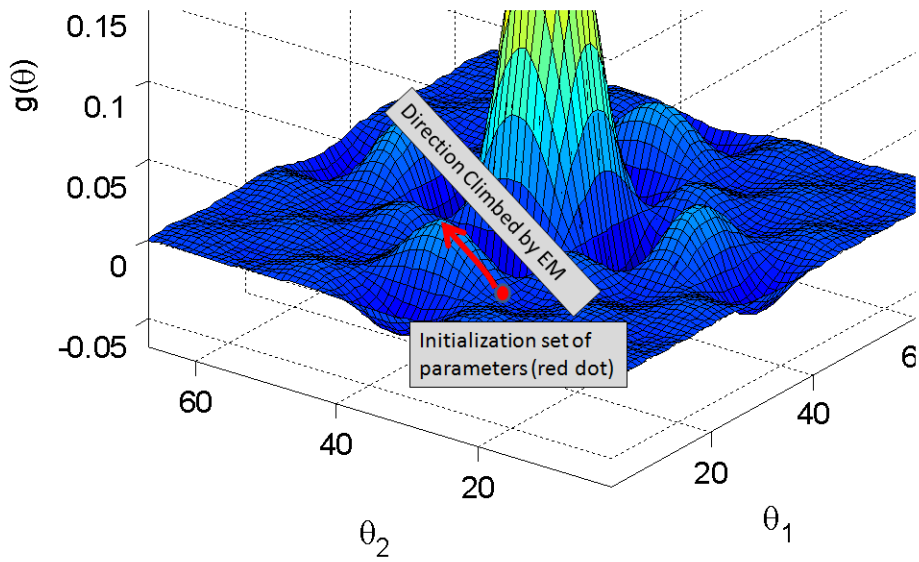

Figure 1-5: Schematic drawing of hypothetical three-dimensional objective function, showing local convergence and sensitivity to initialization of EM.

hybrid model learning algorithm, HML-LPHA. See figure 1-5.

A standard stochastic method for jumping out of these local maxima is a restart method that samples from a distribution over the model parameters to find a new initialization point for the local optimization algorithm. This Random Restarts method has no way of guiding the search away from explored regions of the search space, resulting in the wasted computational expense of re-discovering the same local solutions. Figure 3-2 shows a schematic diagram of random restarts leading to convergence of the same local maxima of the log-likelihood objective function, $g(\theta)$.

Hence, the key technical challenge is to select restart points that seem a priori promising, but lie outside of the bases of hills that have already been discovered. This would lead to an algorithm that has a higher probability of discovering model parameter estimates that are optimal, or near-optimal solutions to the maximization of $g(\theta)$.

Problem Statement: Iteratively learn the shapes and locations of discovered local maxima of the log-likelihood objective function, $g(\theta)$, and use this as a map to guide the search toward new maxima that are a priori more likely to be optimal solutions. We target the learning of continuous model parameters, θ_c, that are the optimal or near-optimal solutions to the maximization of the log-likelihood function $g(\theta) = \log(p(y|\theta))$.

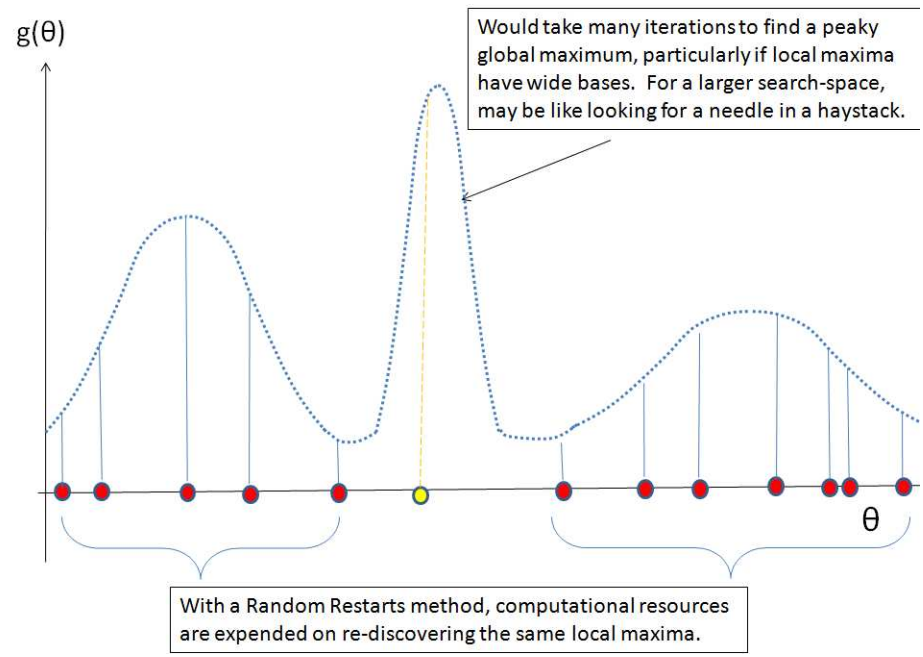

Figure 1-6: Schematic drawing of hypothetical two-dimensional log-likelihood objective function, $g(\theta)$, showing initialization points leading to repeated discovery of same local maxima.

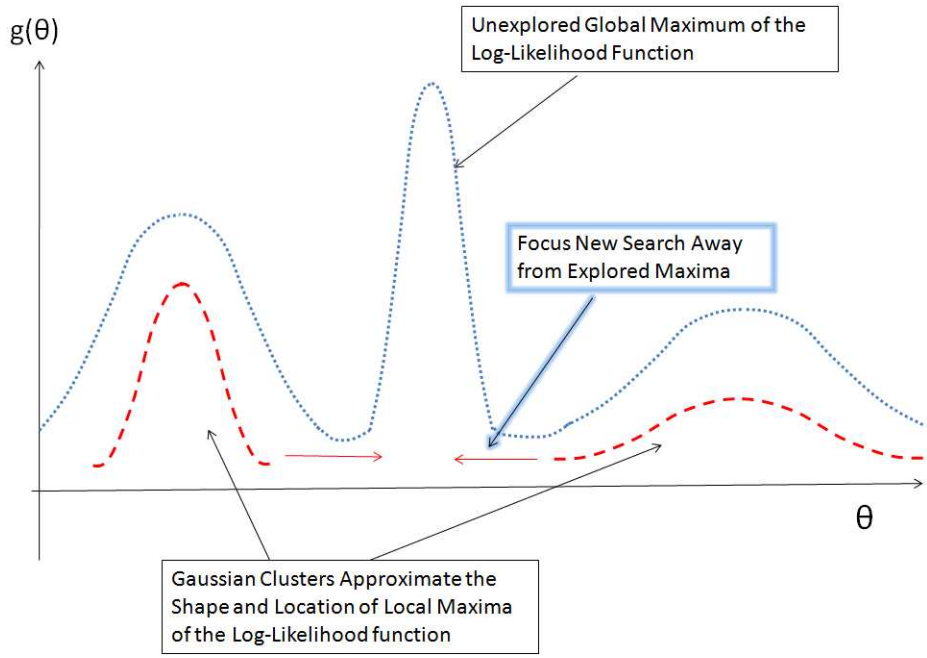

Figure 1-7: Schematic drawing of hypothetical two-dimensional log-likelihood objective function, $g(\theta)$, with explored local maxima approximated by Gaussian clusters.

We propose an algorithm that combines two optimization methods, Expectation Maximization and K-Means Clustering, for allowing a thorough search over the space of model parameters and a systematic method of forcing EM to jump away from local maxima of the log-likelihood function. This algorithm iterates three phases, a clustering phase, an optimization phase, and a labeling phase. These phases are described below:

1. **Initialization Phase:** In the initialization phase, EM is used to provide an initial labeling for a pool of different sets of parameters $\{\theta_{0i}\}_{i=1}^{n}$ where n is the total number of models and the label is a converged set of model parameters, θ_{f_i}. We set the number of clusters, k, to 1.

2. **Clustering Phase:** In the clustering phase, the labeled parameters, $\{\theta_{f_i}\}_{i=1}^{n}$, are clustered into k groups where the clustering uses Euclidean distance between sets of model parameters. Each of these clusters is defined by a Gaussian distribution whose mean is the centroid of the cluster, and whose variance is the empirical variance calculated over the members of each cluster. The resulting Gaussian clusters are used as representations of local maxima hills of the log-likelihood function, $g(\theta)$. See Figure **??**

3. **Optimization Phase:** The goal of this phase is to find a new set of model parameters that is an *a priori* likely set of model parameters and that has a low probability of converging to any of the explored local maxima of $g(\theta)$. We can frame this as a maximization problem over the function $s(\theta)$ that uses the Gaussian cluster information and the prior over model parameters to produce a locally optimal solution, $\widetilde{\theta}_0$, that is both a likely set of model parameters and is not likely to converge to known local maxima.

4. **Labeling Phase:** This phase uses EM to label the new initialization point $\widetilde{\theta}_0 \rightarrow \widetilde{\theta}_f$. If $\widetilde{\theta}_f$ belongs to a new cluster, then augment the number of known clusters, $k = k + 1$. Return to the clustering phase.

We use an Autonomous Underwater Vehicle (AUV) simulation in Matlab to implement the kMeans-EM algorithm, where the true AUV dynamics model is a model of the AUVs used at the Monterey Bay Aquarium Research Institute in California,USA. Our algorithm is applied to learning the linearized longitudinal dynamics of this AUV using noisy observations of the true continuous state trajectory.

We demonstrate empirical evidence that the kMeans-EM algorithm outperforms a Random Restarts method with regards to finding learned sets of model parameters that have higher average likelihood values. This improvement increases at each iteration as the algorithm learns a better map of the likelihood function and updates its prior over model parameters. We also find that the kMeans-EM algorithm finds learned model parameters that are closer to the global optimum in the Euclidean sense, than a Random Restarts method. Finally, we find that clustering accuracy influences the performance of the kMeans-EM algorithm, however, even with a non-optimal clustering scheme, the algorithm outperforms a Random Restarts method.

1.3.3 Active Hybrid Model Learning for LPHA

Learning models for stochastic, partially observable systems, such as JMLS or LPHA, is a challenging problem. This problem combined with sparse training data, or very noisy observations, can cause divergence of a hybrid learning algorithm altogether, which makes learning an accurate model of the hybrid system an impossibility. Unfortunately, many systems that we are interested in modeling, rely on instruments to produce observation data, and these instruments can also degrade over time, which, leads to poorer quality data. In the field of machine vision, JMLS and LPHA are often used to model the dynamics of an object being tracked or performing some activity [32, 10, 36]. The data used to acquire the model, or make inferences about the system, can be obtained from video sequences whose pixel quality is also affected by noise.

Our final technical chapter focuses on this problem by augmenting sparse or poor quality data with a modest amount of labeled data obtained via queries of the discrete state of the LPHA. Queries are strategically chosen at time instances where they can provide the most reduction in uncertainty of the distribution over the hidden state; resulting in higher quality learned models. The ability of the learner to select training data that would most reduce a cost objective, in this case uncertainty of the current distribution over the hybrid state, is referred to as active learning [12]. We present an active learning approach to hybrid model learning for the model parameters of LPHA. We refer to this algorithm as Active Learning Hybrid Model Learning for LPHA (AHML-LPHA).

Problem Statement: To improve the accuracy of parameter model learning for LPHA via active learning. Active Learning in this context includes the ability to query the discrete state of the sys-

tem, or add labeled data, to enhance learning capability.

Given an observation sequence $\mathbf{y_1^{T+1}}$ and a control input sequence $\mathbf{u_0^T}$
1) find the time points t^* where the information gain from querying is highest

$$t^* = \arg\max_t \text{VOI}$$

and
2) find the Maximum Likelihood set of model parameters given the optimal queries $Q(t^*) = q$, $q \in [1, \ldots, m]$ and m is the total number of discrete states.

Query based active learning has many applications in estimation and model learning where the query is a request for additional information about the discrete state of the system. This information can be provided by a human supervisor; an example in activity recognition from video sequences may be the answer to a query "was the person running or walking at time $t = 100$?" A query may be the result of a specialized measurement as with active probing in the fields of medical or hardware diagnosis [33], or a query may be a request for GPS coordinates for localization [20].

Performing a query however, can be expensive, and thus we must limit the number of queries made while choosing each query such that we maximize the amount of information obtained. The definition of "maximum value" is problem dependent. For our problem we use an entropy-based loss function such that queries with the maximum value correspond to those that reduce the entropy, or uncertainty, in the hidden state distribution. Each query is labeled with its associated *Value of Information*(VOI). An entropy based VOI takes the form of the mutual information between the distribution over the hybrid state trajectory and the result of performing a query at time t, Q_t.

$$\begin{aligned}
\text{VOI}(Q_t) =& \mathcal{H}(\pi) - \mathcal{H}(\pi|Q_t) \\
=& \mathcal{H}(Q_t) - \mathcal{H}(Q_t|\pi) \text{ by symmetry of mutual information} \qquad (1.1) \\
\pi =& \left[x_{d_t}, x_{c_{t+1}}\right]_{t=0}^{t=T} \text{ is a state trajectory}
\end{aligned}$$

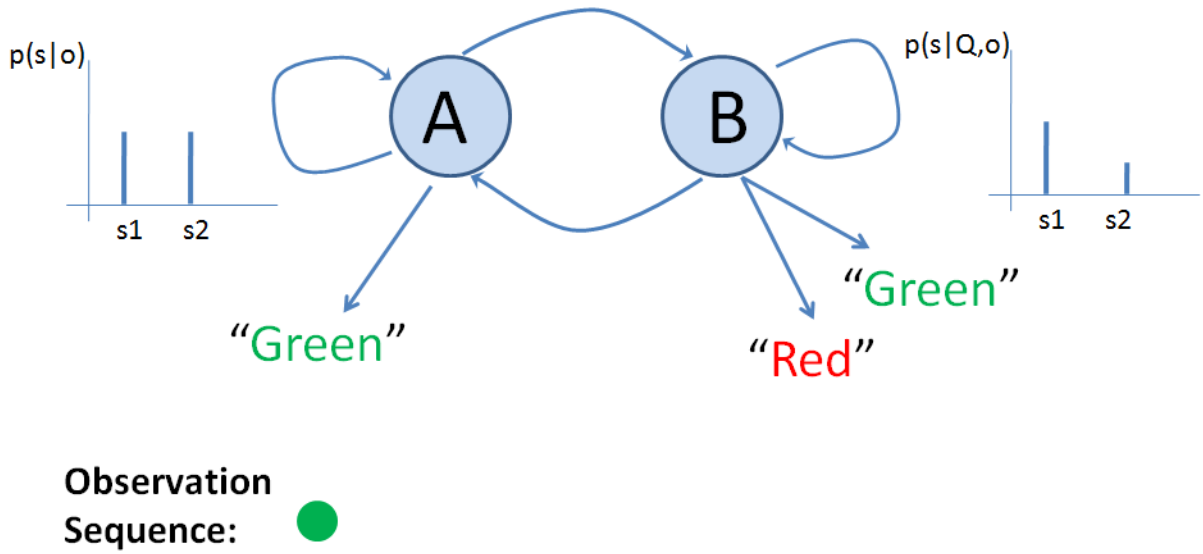

Figure 1-8: Schematic diagram of a discrete HMM where the VOI of a query would be high due to maximum uncertainty in the distribution over the discrete state.

Anderson and Moore [1] present a query and VOI-based active learning approach for purely discrete systems, or HMMs. The reduction in entropy of the distribution over the discrete states after obtaining the result of a query is shown in Figure 1-8 for a two-state HMM where observing "Green" results in a uniform posterior distribution over the hidden states.

We are able to extend active model learning for the purely discrete case to hybrid model learning by making the key observation that *given the discrete state at time t, the exact posterior distribution over the continuous state is provided via a Kalman Smoother.* Thus, we can perform hybrid active learning effectively while only querying the discrete component of the state, which reduces cost over having to query the full hybrid state.

The main steps in the AHML-LPHA algorithm are sketched below:

1. FOR *all observations* y_t, $t \in [1, \ldots, T]$*:*

2. Evaluate the, VOI(Q_t), the value of making a query at time t.

3. Store all maximum value queries in $\{Q^*\}$

4. ENDFOR

5. FOR *all optimal queries in* $\{Q^*\}$

6. Update the distribution over the hybrid state trajectories,$p(x_{c0}^{T+1}, x_{d0}^{T}|Q_t^* = q, y_1^t, \theta)$, to reflect the result of the query $Q^*(t)$.

7. Run a backwards pass on the data to obtain the full posterior distribution over hybrid state trajectories, $p(x_{c0}^{T+1}, x_{d0}^{T}|\{Q_t^*\}, y_1^{T+1}, \theta)$

8. ENDFOR

9. M-Step Maximization: Maximize $f(\theta) = \int p(x_{c0}^{T+1}, x_{d0}^{T}, y_1^{T+1}|Q_t^* = q, \theta)dx_d dx_c$ w.r.t. θ using EM to yield maximum likelihood model parameters θ^*

We implement our algorithm on a simulated exploration vehicle that is driving over different types of terrain in a testbed and is autonomously learning a LPHA model of its dynamics when its wheels are slipping or not slipping. Empirical results indicate, that with as little as 6% of the total observations are used for queries, we obtain a reduction in squared error between the estimated continuous state trajectory and true continuous state trajectory. The same is true for a reduction in MAP mode estimation error in the estimated discrete state trajectory. We also find that when the EM-based hybrid model learning technique diverges due to poor initial conditions, in some cases, as little as one query of the discrete state prevents divergence and vastly improves learning ability.

1.4 Thesis Roadmap

The thesis is organized such that each chapter describes the problem statement, technical approach, and empirical results for each algorithm discussed. Chapter 2 presents the HML-LPHA algorithm, Chapter 3 presents the kMeans-EM algorithm, and Chapter 4 presents the AHML-LPHA algorithm. Finally, we conclude the thesis with Chapter 5.

Chapter 2

Expectation Maximization for Hybrid Model Learning

In this chapter we review the basic theory behind the Expectation Maximization optimization method, then we present the application of Expectation Maximization for Parameter Estimation of linear dynamical systems, and finally we present the extension of Expectation Maximization to Switching Linear Dynamical Systems. We present simulation results of parameter estimation for switching linear dynamical systems and discuss some of the limitations of this approach, including local convergence, and initialization sensitivity. This section of the thesis was part of a joint work with Lars Blackmore and many of the derivations can be found in his PhD thesis, [8], and in our joint paper, [6].

2.1 Introduction to Hybrid Model Learning

We begin by presenting motivations, the problem statement, and definitions for our problem. The use of Switching Linear Dynamical Systems has been become more prevalent for modeling many phenomena, from tracking in machine vision [32], to economic growth models [21], and diagnosis [19, 7]. A Switching Linear Dynamical System (SLDS) provides a framework for modeling hybrid discrete-continuous systems where the discrete and continuous system behaviors are coupled. SLDS combine Hidden Markov Models (HMM) and Linear Dynamical Systems (LDS), to form a generalized modeling capability for systems exhibiting *both* discrete and continuous behaviors. We will

provide a formal definition of Switching Linear Dynamical Systems in the next section. For many problems of interest, the actual hybrid state of the system is not directly observable, or hidden, and the evolution of the hidden state is stochastic, as modeled by the HMM. An accurate model for a hybrid system allows for the ability to perform inference, and also allows for the application of optimal controllers and/or adaptive control, amongst many other applications.

There is a body of literature in the areas of estimation and control of these stochastic hybrid discrete-continuous models [29, 4, 2]. However, the effectiveness and feasibility of these methods relies on accurate models of the hybrid system. Thus, in this chapter of the thesis we present a method for learning the Maximum Likelihood parameter estimates for Switching Linear Dynamical Systems using an approximate Expectation Maximization (EM) approach.

2.1.1 Previous Work and Relation to this Thesis

Previous work developed methods for learning Maximum Likelihood parameter estimates for the continuous case of a Linear Dynamical System via EM [15]. In this case, EM can be applied exactly and thus the characteristics of EM, ie. proof of convergence to a local maximum of the objective function, hold. The extension of EM applied to hybrid models for parameter estimation has also been an active topic of research. Much work has been done in this area for Jump Markov Linear Systems, which are the combination of Linear Dynamical Systems (LDS) with Hidden Markov Models (HMM), where the continuous and discrete state are assumed to be independent [16, 4]. The extension of Jump Markov Linear Systems for the incorporation of dependence of the discrete transitions on the continuous state have also been investigated. We refer to these models as Linear Probabilistic Hybrid Automata. Estimation of the hybrid state for these systems [19], as well as hybrid model learning for these systems using heuristic approaches and approaches where only the most likely state trajectories are tracked [18], have also been considered. The contribution that we present in this thesis extends previous work in the following ways:

1. We allow for the dependence of the discrete state on the continuous state through guarded transitions. We refer to this dependence as *autonomous mode transitions*,

2. We learn the effects of control inputs on the system, and

3. We consider *soft* EM, where we track the k-best or most likely trajectories, as opposed to

hard EM where only the single most likely trajectory is tracked.

We demonstrate our approach in simulation and present and discuss results as well as limitations of our method.

Definitions

In this section we provide formal definitions for Hidden Markov Model, Linear Dynamical System, Switching Linear Dynamical System, and Autonomous Mode Transition.

Hidden Markov Model: A Hidden Markov model [23] is a tuple $\langle x_d, y, T, B \rangle$ where $x_d \in \{1, \ldots, m\}$ is a set of m non-observable discrete states, y is a set of possible discrete observations, $T \in R^{mxm}$ is the transition probability between discrete states $p\left(x_{d_i} | x_{d_j}\right)$, and B is the observation probability $p\left(y_{di} | x_{d_i}\right)$.

Linear Dynamical System: A Linear Dynamical System [3] is a system whose continuous state $x_c \in R^{nx1}$ evolves in time according to a linear set of equations

$$x_{ct+1} = \mathbf{A} x_{c_t} + \mathbf{B} u_t + \omega_t$$
$$y_t = \mathbf{C} x_{c_t} + \mathbf{D} u_t + \upsilon_t$$

where

- $x_{c_t} \in R^{\text{nx1}}$ is a vector containing the continuous state at time t.
- $y_t \in R^{\text{pxn}}$ where $p \in 1, \ldots, n$ is a vector of continuous observations at time t.
- $u_t \in R^{\text{qx1}}$ where $q \in 1, \ldots, n$ is a vector of continuous control inputs at time t.
- $\omega_t \in R^{\text{nx1}}$ is the process noise assumed to be white and zero mean Gaussian with covariance $\mathbf{Q} \in R^{\text{nxn}}$.
- $\upsilon_t \in R^{\text{nx1}}$ is the measurement noise assumed to be white and zero mean Gaussian with covariance $\mathbf{R} \in R^{\text{nxn}}$.
- $\mathbf{A} \in R^{\text{nxn}}$ is a matrix relating $x_{c_{t+1}}$ to x_{c_t}.

- $\mathbf{B} \in R^{nxq}$ is a matrix relating x_{c_t} to the control inputs.
- $\mathbf{C} \in R^{pxn}$ is a matrix relating x_{c_t} to y_t.
- $\mathbf{D} \in R^{nxq}$ is the feedthrough input matrix.

Switching Linear Dynamical System: A Switching Linear Dynamical System has both a continuous and discrete state. We denote this hybrid state as $x = [x_c, x_d]$ The discrete state is modeled by a Hidden Markov Model and the discrete state transitions are conditioned on the continuous state via Autonomous Mode Transitions, that are defined below. Each discrete state has its own set of Linear Dynamical Equations that govern the evolution of the continuous state. In particular,

$$x_{c_t+1} = \mathbf{A}(x_{d_t})x_{c_t} + \mathbf{B}(x_{d_t})u_t + \omega_t$$
$$y_t = \mathbf{C}(x_{d_t})x_{c_t} + \mathbf{D}(x_{d_t})u_t + \upsilon_t$$

where the definitions for continuous state, observation, control inputs, and noise vectors are the same as for LDS, except that the matrices $\mathbf{A}(x_{d_t})$, $\mathbf{B}(x_{d_t})$, $\mathbf{C}(x_{d_t})$, $\mathbf{D}(x_{d_t})$, and the noise covariances $\mathbf{Q}(x_{d_t})$ and $\mathbf{R}(x_{d_t})$ have an explicit dependence on the discrete state x_d. In turn, the discrete state transitions also depend on the continuous state. As a result of this definition, we find that the distributions over the continuous state and continuous observation are both Gaussian. We also define the distribution over the discrete state x_{d_t}.

$p\left(x_{c_t+1}|x_{c_t}, x_{d_t}, u_t\right) \sim \mathcal{N}\left(\mathbf{A}(x_{d_t})x_{c_t} + \mathbf{B}(x_{d_t})u_t, \mathbf{Q}(x_{d_t})\right)$

$p\left(y_{c_t}|x_{c_t}, x_{d_t}, u_t\right) \sim \mathcal{N}\left(\mathbf{C}(x_{d_t})x_{c_t} + \mathbf{D}(x_{d_t})u_t, \mathbf{R}(x_{d_t})\right)$

$p\left(x_{d_0}, x_{c_0}\right)$ is the distribution over the initial hybrid state and is a sum of Gaussians,where

$p\left(x_{c_0}|x_{d_0}\right) \sim \mathcal{N}\left(\mu\left(x_{d_0}\right), \mathbf{V}\left(\mathbf{x_{d_0}}\right)\right)$

Autonomous Mode Transitions: We define guard conditions $c_i \in \mathcal{G}$, where each guard condition has an associated guard region $C_i \subset \mathbb{R}^n$ and a transition probability matrix T_i such that $T_i\left(i,j\right) = p\left(x_{d_t+1} = i|x_{d_t} = j\right)$. The guards form a partition of the space $\mathbb{R}^n$. In this thesis, we

consider guards that are defined over regions of the continuous state. The guard itself is a boolean variable that indicates whether it is satisfied or not and this is conditioned on the continuous state. The regions of satisfiability, C_i, can be general. An example would be a linear guard, or satisfaction of an inequality. Autonomous Mode Transitions are depicted in Figure 1-3.

Linear Probabilistic Hybrid Automata (LPHA): We refer to LPHA [19] as SLDS or Jump Markov Linear systems that incorporate Autonomous Mode Transitions. In other words, we generalize SLDS to allow for the dependence of discrete state transitions on the continuous state.

Model Parameters θ: We define θ to be model parameters. For the hybrid case $\theta = \theta_{\mathbf{c}} \cup \theta_{\mathbf{d}}$, where $\theta_{\mathbf{c}} = \langle \mathbf{A}(x_{d_t}), \mathbf{B}(x_{d_t}), \mathbf{C}(x_{d_t}), \mathbf{D}(x_{d_t}), \mathbf{Q}(x_{d_t}), \mathbf{R}(x_{d_t}), \mathbf{V}(x_{d_t}), \mu(x_{d_t}) \rangle$ are the continuous model parameters and $\theta_{\mathbf{d}} = \langle \mathbf{T_i} \rangle$ are the discrete model parameters where the transition matrix $\mathbf{T_i}$ is defined for the guard condition $c_i \in \mathcal{G}$.

2.1.2 Problem Statement

Our objective is to learn the continuous and discrete model parameters, θ, for *Linear Probabilistic Hybrid Automata* (LPHA). Formally, given a sequence of control inputs $\mathbf{u}_0^T$, and a set of continuous observations $\mathbf{y}_1^{T+1}$, determine the set of model parameters θ that maximize the likelihood of the data $p\left(\mathbf{y}_1^{T+1}|\theta\right)$. We define $\mathbf{u}_0^T$ to be the sequence from $t = 0$ to $t = T$. More specifically, find $\theta = \arg\max_\theta p\left(\mathbf{y}_1^{T+1}|\theta\right)$.

Note that the learning of the guard regions is not in the scope of this thesis. Learning the guard regions requires integration over a multivariate Gaussian distribution, that represents your continuous state estimate, for an arbitrary guard region. For many forms of the guard region, this integration does not have a closed-form solution. There are numerical methods for computing this integration, and an efficient such method would have to be used for learning the guard regions. This is a topic of future research.

2.2 Review of Expectation Maximization

In this section we review the basic theory behind Expectation Maximization optimization methods. We introduce fundamental tools and mechanisms that will be used throughout the chapter. A common approach to computing Maximum Likelihood (ML) estimates of model parameters is to

maximize the probability distribution of the observations available given the set of model parameters, $p(\mathbf{y}|\theta)$, with respect to the unknown model parameters. This can be achieved by differentiating the probability distribution with respect to the parameter of interest and setting this derivative to zero. This results in equations that are hard and intractable to solve for many non-trivial problems. The Expectation Maximization algorithm was designed as an optimization method for such cases [13]. This algorithm is an iterative procedure that exploits the structure of the probability distribution to efficiently search for a local maximum of this function. The algorithm treats the available data, or observations $\mathbf{y}$, as incomplete data and assumes that the *complete data* $\mathbf{z}$ is partially hidden, that is, non-observable. The complete data is assumed to be generated by the probability distribution $p\left(\mathbf{z}|\theta\right)$. We wish to compute the ML estimate of the parameter θ given the observed data $\mathbf{y}$. We can write the distribution over our observations as a marginalization over the hidden variables $\mathbf{x}$, where the complete data $\mathbf{z} = [\mathbf{x}, \mathbf{y}]$.

$$f\left(\theta\right) \triangleq p\left(\mathbf{y}|\theta\right) = \int p\left(\mathbf{x}|\mathbf{y},\theta\right) p\left(\mathbf{y}|\theta\right) d\mathbf{x} \tag{2.1}$$

We are looking to maximize the objective function $f(\theta)$. The log of this function is monotonic and thus will have the same maximum as the function itself. Taking the log of both sides of Equation (2.1)

$$g(\theta) \triangleq \log p\left(\mathbf{y}|\theta\right) = \log \int p\left(\mathbf{x}|\mathbf{y},\theta\right) p\left(\mathbf{y}|\theta\right) d\mathbf{x} \tag{2.2}$$

This integral is hard, or intractable to compute in the general case. However, we can find a lower bound to this equation via an application of Jensen's inequality that *is* tractable to work with.

$$g(\theta) = \log p\left(\mathbf{y}|\theta\right) = \log \int p\left(\mathbf{x}|\mathbf{y},\theta\right) p\left(\mathbf{y}|\theta\right) d\mathbf{x} \tag{2.3}$$

$$\geq \int p(\mathbf{x}|\mathbf{y},\theta^k) \log \frac{p(\mathbf{y},\mathbf{x}|\theta)}{p(\mathbf{x}|\mathbf{y},\theta^k)} d\mathbf{x} \tag{2.4}$$

$$\triangleq h(\theta|\theta^k) \tag{2.5}$$

Where θ^k denotes the guess of model parameters, θ, at iteration k of the EM algorithm. The lower bound, $h(\theta|\theta^k)$, to our objective function, $g(\theta)$, can be rewritten in terms of an expectation plus an entropy term.

$$h(\theta|\theta^k) = E_{p(\mathbf{x}|\mathbf{y},\theta^k)}[\log p(\mathbf{z}|\theta)] + \mathcal{H} \tag{2.6}$$

where $\mathcal{H}$ represents the entropy term. This bound constitutes the tightest possible bound to $g(\theta)$ in that at the current guess of model parameters, the value of this bound equals the value of $g(\theta)$. See Figure 1-4. Because $h(\theta|\theta^k)$ is a tight lower bound to $g(\theta)$, maximizing $h(\theta|\theta^k)$ at each iteration of EM guarantees a solution set of model parameters θ^{k+1} that increases the value of the true objective function $g(\theta)$. The EM algorithm stops when the value of the objective function reaches a local maximum [25]. In summary, the EM algorithm is an iteration between two simple and intuitive steps, the **E-Step** and **M-Step**, where a cost function, the *expectation* of the logarithm of the complete data is maximized. The advantage of EM is that many times this objective cost function is easier to maximize than maximizing the likelihood function itself $p(\mathbf{y}|\theta)$. Below we summarize the steps in the EM algorithm:

1. **Initialization:** Initialize the current guess of parameters $\theta^k, k = 0$
   Iterate until convergence of the log-likelihood:
2. **Expectation Step:** Given θ^k, calculate the lower bound $h\left(\theta|\theta^k\right)$
3. **Maximization Step:** Given the lower bound $h\left(\theta|\theta^k\right)$, maximize to find a new guess of parameters θ^{k+1}
   $\theta^{k+1} = \arg\max_\theta h\left(\theta|\theta^k\right)$
   $k = k + 1$

2.3 Expectation Maximization for Switching Linear Dynamical Systems

In the previous section we introduced EM and its underlying theory. We also discussed some of the properties of EM that make it so effective and applicable, 1) the ability to maximize a

tractable lower bound of an intractable objective function, and 2) guarantee of convergence to a locally optimal estimate for the parameters of interest. In this section we present Expectation Maximization applied to parameter estimation of Switching Linear Dynamical Systems. Previous work in this area includes Expectation Maximization for parameter estimation of non-switching Linear Dynamical Systems where the state is purely continuous [5, 15, 35]. In Section 2.3.1 we discuss the application of exact EM to learning the model parameters for the switching case, in Section 2.3.3 we argue that exact EM is actually not possible in the switching case, and in Section A.0.1 we propose an approximate EM approach to learning the ML parameter estimates for the hybrid model case that we refer to as the HML-LPHA algorithm. In the last sections of this chapter we present empirical results and discussion of these results for the HML-LPHA algorithm presented in A.0.1 and also discuss limitations.

2.3.1 Exact EM for Hybrid Model Learning

We now apply the framework of EM presented in 2.2 to learning the model parameters for Linear Probabilistic Hybrid Automata. The reader is also referred to [6, 8] for many of the analytical derivations presented in this section. Per the problem statement, our objective is to find a set of model parameter estimates, θ, for a LPHA that maximizes the function $f(\theta) = p\left(\mathbf{y}_1^{T+1}|\theta\right)$. Thus we can use EM as a tool to perform this maximization. For the hybrid case, our hidden state is defined as $\mathbf{x} = [\mathbf{x_c}, \mathbf{x_d}]$. The hidden state sequence is comprised of the continuous state sequence, ${x_{c0}}^{T+1}$, and the discrete state sequence ${x_{d0}}^{T}$, the observed data consists of the observation sequence y_1^{T+1}. The bound in this case is written as

$$h\left(\theta|\theta^k\right) = E\left[\log p\left(\mathbf{z}|\theta\right)\right] + \mathcal{H} \tag{2.7}$$

$$= E\left[\log p\left(y_1^{T+1}, {x_{c0}}^{T+1}, {x_{d0}}^{T}|\theta\right)\right] + \mathcal{H} \tag{2.8}$$

Where the expectation is taken with respect to the distribution over the hidden state, $p\left({x_{c0}}^{T+1}, {x_{d0}}^{T}|y_1^{T+1}, \theta^k\right)$.

2.3.2 Summary of Approach and Theoretical Results

In this section of the thesis, we summarize the theoretical results of the E-step and the M-step for EM applied to model learning for LPHA as achieved by the HML-LPHA algorithm. The full technical details of these results can be found in [6, 8] and in the appendix of this thesis.

First we present results of the E-step applied to the problem of model parameter learning. We find that in the case of a LPHA, the number of unique hybrid states grows exponentially in time. This exponential growth causes an intractability in the calculation of the distribution over the hidden state, $p\left(x_{c0}^{T+1}, x_{d0}^{T} | y_1^{T+1}, \theta^k\right)$. As a result, we can no longer compute the exact bound $h(\theta|\theta^k)$ presented in Equation (2.8). This explosion of the hybrid state space is well-known and well documented in the literature [16, 18, 19, 9]. We propose a solution to this intractability by pruning the space of state trajectories and tracking only the k most likely trajectories. The results in an *approximate* EM algorithm for hybrid model learning that we call the HML-LPHA algorithm. We provide a more in-depth discussion of the intractability of exact EM for hybrid models in Section 2.3.3.

Following the discussion on the intractability of exact EM for model parameter learning of hybrid systems, we present the main results for the approximate E-step, and the approximate M-step, for the HML-LPHA algorithm. The detailed derivations and equations for these sections can be found in the appendix of this thesis.

2.3.3 Intractability of Exact EM for Hybrid Models

As mentioned in the previous section, exact EM cannot be implemented for model learning in the hybrid case. Instead, we can use the EM framework to implement an approximate version for which we *can* still obtain parameter estimates for LPHA. The exponentially growing number of unique state trajectories prevents us from being able to calculate the lower bound presented in (2.8).

For the purposes of clarity and discussion, we briefly discuss the application of dynamic programming approaches to a similar problem of exponential trajectory growth in the purely discrete case. If our state definition had only a discrete component, we would still be in the realm of problems where the number of unique trajectories grows exponentially in time. This is a well-known phenomena encountered with many HMM applications, such as those solved by Forward-Backwards type algorithms, the Viterbi, and Baum Welch algorithms to name a few. In the case of a purely

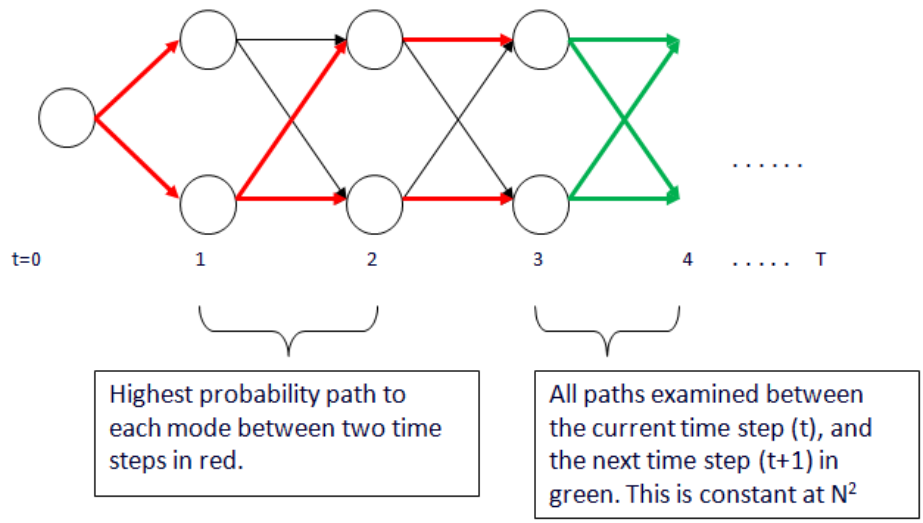

Figure 2-1: Diagram of a Purely Discrete System with a Constant Trellis

discrete state, however, although the number of unique trajectories explodes in time, the *trellis* diagram, or the number of unique *states* at each time remains constant. Figure 2-1 depicts the Trellis structure for a discrete system with two discrete states, 1 and 2, over time. Exploiting this structure, dynamic programming approaches are able to provide closed form solutions for problems such as finding the Maximum A Posteriori state trajectory, or the Maximum Likelihood model parameters for the underlying HMM.

The distinction between the aforementioned case where the state is discrete and has a constant trellis structure and the current case of the hybrid state, is that the fact of the state having a discrete *and* continuous component does not allow for a constant trellis structure where the number of unique states per timestep is constant. In contrast, for the hybrid case, the trellis structure itself is growing exponentially in time, as do the unique number of hybrid states.

In the next section we propose an approximation to exact EM that allows us to follow the framework provided by EM while handling the intractability of the exponentially growing number of trajectories.

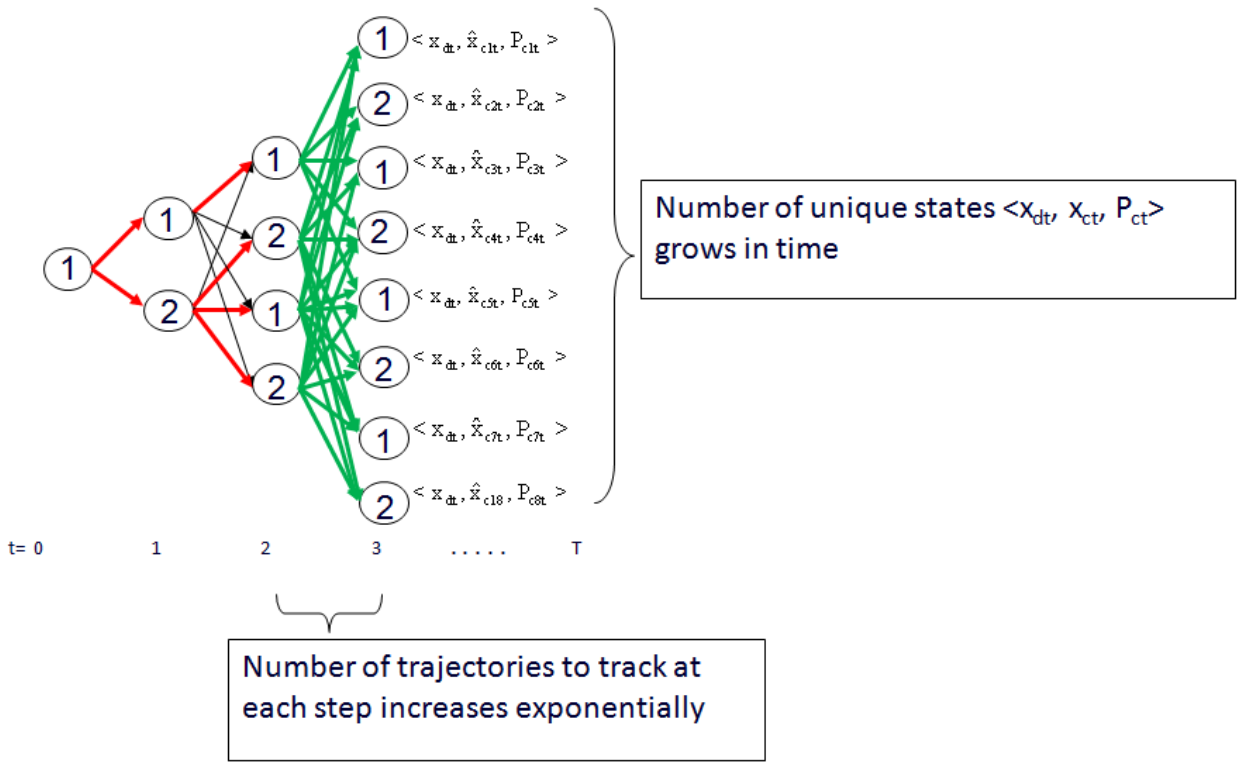

Figure 2-2: Diagram of a Hybrid System with a Non-Constant Trellis

2.3.4 Approximate EM for Hybrid Model Learning

Approximate E-Step for Hybrid Model Learning

In order to address the problem of the exponentially growing number of trajectories, we restrict our attention to a subset of the trajectories $\mathcal{S}$ where we include in $\mathcal{S}$ only the k most likely trajectories. We call this k-best enumeration. We therefore approximate the lower bound in (2.8) by restricting the summation to be over all sequences $x_{d_0}^T \in \mathcal{S}$. The probability assigned to all sequences falling outside of the set $\mathcal{S}$ will be set to zero, and thus posterior probability of each sequence within $\mathcal{S}$ cannot be evaluated exactly. This is a well-known problem in approximate inference and a standard approach is to choose the factor such that the sequences remaining in $\mathcal{S}$ sum to one. We therefore find our approximate distribution over the discrete state sequences to be

$$\widetilde{p}(x_{d_0}^T|y_1^{T+1},\theta) = \frac{1}{c}p(x_{d_0}^T|y_1^{T+1},\theta) \tag{2.9}$$

$$c = \sum_{x_{d_0}^T \in \mathcal{S}} p(x_{d_0}^T|y_1^{T+1},\theta) \tag{2.10}$$

and our resulting approximation to the lower bound in (2.8) to be

$$\begin{aligned}&\widetilde{h}\left(\theta|\theta^k\right) = \\ &\sum_{x_{d_0}^T \in \mathcal{S}} \left(\widetilde{p}\left(x_{d_0}^T|y_1^{T+1},\theta^k\right) \int p\left(x_{c_0}^{T+1}|x_{d_0}^T, y_1^{T+1},\theta^k\right) * \log p\left(y_1^{T+1}, x_{c_0}^{T+1}, x_{d_0}^T|\theta\right) dx_{c_0}^{T+1}\right) \\ &+ \widetilde{\mathcal{H}}\end{aligned} \tag{2.11}$$

As a result of our approximation, we no longer achieve the tightest possible lower bound of the log-likelihood function $g(\theta)$ and thus the analytical proof of local convergence does not extend to our method. However, we do demonstrate empirical convergence of our method and in the vast majority of cases a large amount of the probability mass over the discrete trajectories can be captured by the k-best trajectories. We demonstrate empirical results in 2.3.5 that analyze more in depth the impact of k on algorithm performance.

Approximate M-Step for Hybrid Model Learning

The maximization step involves a maximization of the bound $\widetilde{h}(\theta|\theta^k)$ with respect to θ. We can break this maximization up into two main parts, one being maximization of the lower bound with respect to the continuous model parameters, θ_c, and the other being the maximization of the lower bound with respect to the discrete model parameters, θ_d. We present the results of the approximate M-step here and include the derivations in the Appendix.

A. *Maximization Step for Continuous Model Parameters*

In order to find the next guess of continuous model parameters, θ_c^{k+1}, we must find the set of parameters θ_c that maximizes the lower bound expression in A.9. To find the maximum of $\tilde{h}(\theta|\theta^k)$ we look for its extremum by setting its derivative with respect to θ_c to zero. We can write this derivative as a summation over the derivative for each mode sequence [6]:

$$\begin{aligned}\frac{\partial \tilde{h}(\theta|\theta^k)}{\partial \theta_c} = \sum_{x_{d_0}^T \in \mathcal{S}} \Bigg(& \tilde{p}(x_{d_0}^T | y_1^{T+1}, \theta^k) \\ & * \frac{\partial}{\partial \theta_c} \int_{x_{c_0}^{T+1}} p(x_{c_0}^{T+1} | x_{d_0}^T, y_1^{T+1}, \theta^k) \\ & * \log p(y_1^{T+1}, x_{c_0}^{T+1}, x_{d_0}^T | \theta) dx_{c_0}^{T+1} \Bigg) = 0. \end{aligned} \tag{2.12}$$

The optimal values for $A(x_d)$ and $B(x_d)$ are found by performing weighted sum of the LTI results from [11] over the mode sequences in $\mathcal{S}$ to give the following equations:

$$\begin{aligned} & \sum_{x_{d_0}^T \in \mathcal{S}} \left(\tilde{p}(x_{d_0}^T | y_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(x_{d_0}^T)} P_{t+1,t}(x_{d_0}^T) \right) = \\ & A^*(x_d) \sum_{x_{d_0}^T \in \mathcal{S}} \left(\tilde{p}(x_{d_0}^T | y_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(x_{d_0}^T)} P_t(x_{d_0}^T) \right) \\ & + B^*(x_d) \sum_{x_{d_0}^T \in \mathcal{S}} \left(\tilde{p}(x_{d_0}^T | y_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(x_{d_0}^T)} u_t \hat{x}_{c_{t+1}}(x_{d_0}^T) \right) \end{aligned} \tag{2.13}$$

$$\sum_{x_{d_0}^T \in \mathcal{S}} \left(\tilde{p}(x_{d_0}^T | y_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(x_{d_0}^T)} \hat{x}_{c_{t+1}} u_t^{'} \right) =$$
$$A^*(x_d) \sum_{x_{d_0}^T \in \mathcal{S}} \left(\tilde{p}(x_{d_0}^T | y_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(x_{d_0}^T)} \hat{x}_{c_{t+1}}(x_{d_0}^T) u_t^{'} \right)$$
$$+ B^*(x_d) \sum_{x_{d_0}^T \in \mathcal{S}} \left(\tilde{p}(x_{d_0}^T | y_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(x_{d_0}^T)} u_t u_t^{'} \right), \tag{2.14}$$

where $\mathcal{F}(x_{d_0}^T)$ is the set of time steps in the sequence $x_{d_0}^T$ for which the mode is x_d. Solving the set of linear equations (2.13), (2.14) yields the optimal values for $A(x_d)$ and $B(x_d)$. Similarly, the equations yielding the optimal values for the remaining continuous parameters are provided below. $C(x_d)$ and $D(x_d)$ are obtained by solving the following set of linear equations:

$$\sum_{\mathbf{x_d}_0^T} \left(\tilde{p}(\mathbf{x_d}_0^T | \mathbf{y}_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(\mathbf{x_d}_0^T)} \mathbf{y}_{t+1} \hat{\mathbf{x}}_{t+1}{}'(\mathbf{x_d}_0^T) \right) =$$
$$C^*(\mathbf{x_d}) \sum_{\mathbf{x_d}_0^T \in \mathcal{S}} \left(\tilde{p}(\mathbf{x_d}_0^T | \mathbf{y}_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(\mathbf{x_d}_0^T)} P_{t+1}(\mathbf{x_d}_0^T) \right)$$
$$+ D^*(\mathbf{x_d}) \sum_{\mathbf{x_d}_0^T \in \mathcal{S}} \left(\tilde{p}(\mathbf{x_d}_0^T | \mathbf{y}_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(\mathbf{x_d}_0^T)} \mathbf{u}_t \hat{\mathbf{x}}_{t+1}{}'(\mathbf{x_d}_0^T) \right) \tag{2.15}$$
$$\sum_{\mathbf{x_d}_0^T \in \mathcal{S}} \left(\tilde{p}(\mathbf{x_d}_1^T | \mathbf{y}_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(\mathbf{x_d}_0^T)} \mathbf{y}_{t+1} \mathbf{u}_t{}' \right) =$$
$$C^*(\mathbf{x_d}) \sum_{\mathbf{x_d}_0^T \in \mathcal{S}} \left(\tilde{p}(\mathbf{x_d}_0^T | \mathbf{y}_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(\mathbf{x_d}_0^T)} \hat{\mathbf{x}}_{t+1}(\mathbf{x_d}_0^T) \mathbf{u}_t{}' \right)$$
$$+ D^*(\mathbf{x_d}) \sum_{\mathbf{x_d}_0^T \in \mathcal{S}} \left(\tilde{p}(\mathbf{x_d}_0^T | \mathbf{y}_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(\mathbf{x_d}_0^T)} \mathbf{u}_t \mathbf{u}_t{}' \right). \tag{2.16}$$

The optimal noise covariance matrices are

$$Q^*(\mathbf{x_d}) = \sum_{\mathbf{x_d}_0^T \in \mathcal{S}} \Bigg(\frac{\tilde{p}(\mathbf{x_d}_0^T | \mathbf{y}_1^{T+1}, \theta^k)}{|\mathcal{F}(\mathbf{x_d}_0^T)|} \sum_{t \in \mathcal{F}(\mathbf{x_d}_0^T)} \Big(P_{t+1}(\mathbf{x_d}_0^T)$$

$$-A^*(\mathbf{x_d}) P_{t,t+1}(\mathbf{x_d}_0^T) - B^*(\mathbf{x_d}) \mathbf{u}_t \hat{\mathbf{x}}_{t+1}{}'(\mathbf{x_d}_0^T) \Big) \Bigg) \tag{2.17}$$

$$R^*(\mathbf{x_d}) = \sum_{\mathbf{x_d}_0^T \in \mathcal{S}} \Bigg(\frac{\tilde{p}(\mathbf{x_d}_0^T | \mathbf{y}_1^{T+1}, \theta^k)}{|\mathcal{F}(\mathbf{x_d}_0^T)|} \sum_{t \in \mathcal{F}(\mathbf{x_d}_0^T)} \Big(\mathbf{y}_{t+1}$$

$$-C^*(\mathbf{x_d}) \hat{\mathbf{x}}_{t+1}(\mathbf{x_d}_0^T) - D^*(\mathbf{x_d}) \mathbf{u}_t \Big) \mathbf{y}_{t+1}{}' \Bigg). \tag{2.18}$$

Finally, the parameters specifying the distribution over the initial state are

$$\mu^*(\mathbf{x_d}) = \sum_{\{\mathbf{x_d}_0^T | \mathbf{x_d}_0 = \mathbf{x_d}\}} \tilde{p}(\mathbf{x_d}_0^T | \mathbf{y}_1^{T+1}, \theta^k) \hat{\mathbf{x}}_0{}'(\mathbf{x_d}_0^T) \text{ and}$$

$$V^*(\mathbf{x_d}) = \sum_{\{\mathbf{x_d}_0^T | \mathbf{x_d}_0 = \mathbf{x_d}\}} \tilde{p}(\mathbf{x_d}_0^T | \mathbf{y}_1^{T+1}, \theta^k) P_{0,0}(\mathbf{x_d}_0^T), \tag{2.19}$$

In the above equations, we let

$$\hat{\mathbf{x}}_t(\mathbf{x_d}_0^T) = E\big[\mathbf{x}_t | \mathbf{x_d}_0^T, \mathbf{y}_1^{T+1}, \theta^k\big] \text{ and}$$

$$P_{t_1,t_2}(\mathbf{x_d}_0^T) = E\big[\mathbf{x}_{t_1} \mathbf{x}_{t_2}{}' | \mathbf{x_d}_0^T, \mathbf{y}_1^{T+1}, \theta^k\big]. \tag{2.20}$$

This concludes the M-Step for learning the optimal continuous model parameters θ_c. We now focus on the M-Step for learning the optimal discrete model parameters.

B. *Maximization Step for Discrete Hybrid Model Parameters*

In this section we demonstrate maximization of the lower bound in A.9 with respect to θ_d to yield the optimal discrete model parameters. We first note that the discrete model parameters are defined for each guard condition $c_i \in \mathcal{G}$. Each guard condition has a corresponding transition matrix T_i. In order for our transition probability matrices to be valid, we must perform a constrained

optimization using a Lagrangian Multiplier for all of the possible discrete states $x_d \in \mathcal{X}_d$. The result of this optimization yields the optimal transition probabilities between discrete states conditioned on the current continuous state. See Appendix for accompanying derivation.

Completing the optimization we obtain the optimal value of $T_i(j, x_d)$:

$$T_i^*(j, \mathbf{x_d}) = \frac{\sum_{\mathbf{x_d}_0^T \in \mathcal{S}} \tilde{p}(\mathbf{x_d}_0^T | \mathbf{y}_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(\mathbf{x_d}_0^T)} p_{c_i}(\mathbf{x_d}_0^T)}{\sum_{\mathbf{x_d} \in \mathcal{X}_d} \left(\sum_{\mathbf{x_d}_0^T \in \mathcal{S}} \tilde{p}(\mathbf{x_d}_0^T | \mathbf{y}_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(\mathbf{x_d}_0^T)} p_{c_i}(\mathbf{x_d}_0^T) \right)}. \tag{2.21}$$

This Maximum Likelihood solution takes into account Autonomous Mode Transitions, and can be interpreted as a weighted number of transitions from a source mode x_{t-1} to a target mode x_t for each guard condition c_i.

2.3.5 Simulation

In this section we present results from typical runs of our learning algorithm. We consider the subsystem consisting of a motor and a wheel. An intermittent fault causes the wheel to 'stick' at random, and the probability of the wheel sticking is different depending on whether the wheel is being driven forwards or backwards. When stuck, the wheel experiences increased friction. The wheel subsystem is modeled as a LPHA with two modes. In Mode 1 the wheel operates normally, while Mode 2 the wheel is stuck. The hidden continuous state $\mathbf{x}$ is $\begin{bmatrix} i & \dot{\theta} \end{bmatrix}^T$ where i is the current in the motor and $\dot{\theta}$ is the angular velocity of the wheel. Noisy observations y of the wheel velocity are available through an encoder. The input u is the voltage applied to the driver circuit.

The true continuous parameters are given by:

$$\begin{aligned} A(1) &= \begin{bmatrix} -0.0044 & -0.0203 \\ 0.0366 & 0.1665 \end{bmatrix} \quad B(1) = \begin{bmatrix} 0.92 \\ 0.81 \end{bmatrix} \\ A(2) &= \begin{bmatrix} -0.0032 & -0.0142 \\ 0.0256 & 0.1106 \end{bmatrix} \quad B(2) = \begin{bmatrix} 0.93 \\ 0.71 \end{bmatrix} \\ C(1) = C(2) &= \begin{bmatrix} 0 & 1 \end{bmatrix} \quad C(1) = D(2) = 0. \end{aligned} \tag{2.22}$$

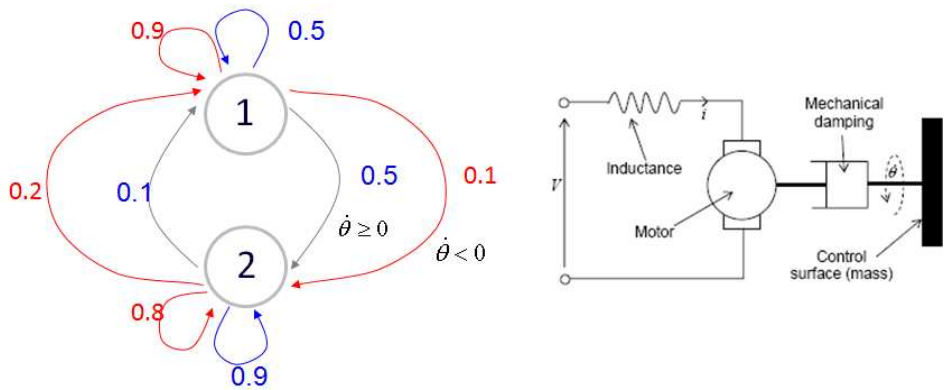

Figure 2-3: Schematic Diagram of Rover Wheel Subsystem as a LPHA

The true guard conditions are given by:

$$\mathcal{C}_1 = [-\infty \; 0] \;\; T_1 = \begin{bmatrix} 0.9 & 0.2 \\ 0.1 & 0.8 \end{bmatrix}$$
$$\mathcal{C}_2 = [0 \; \infty] \;\; T_2 = \begin{bmatrix} 0.5 & 0.1 \\ 0.5 & 0.9 \end{bmatrix}, \tag{2.23}$$

where the guard regions $\mathcal{C}_1$ and $\mathcal{C}_2$ are defined over $\dot{\theta}$.

2.3.6 Results and Discussion

Because of the intractability of this problem as imposed by the exponentially growing number of trajectories we are no longer able to track the value of the log-likelihood and thus we show empirical convergence of the algorithm by tracking the change in the lower bound of the log-likelihood objective function, Figure 2-4. We find that in most cases, the learned transition probabilities converge close to the true transition probabilities for each guard region as shown for a typical run in Figure 2-6. We also find that the mean squared error in the continuous state estimate, and the MAP mode error decreases with iteration number, further supporting convergence, 2-7.

We find that there is no clear correlation between the number of trajectories tracked and the MAP mode error for the discrete state sequences 2-8. Although this result seems non-intuitive, closer examination shows that for many cases the majority of the probability mass over the discrete trajectories lies in the first few trajectories, more than half of which is placed on the first trajectory.

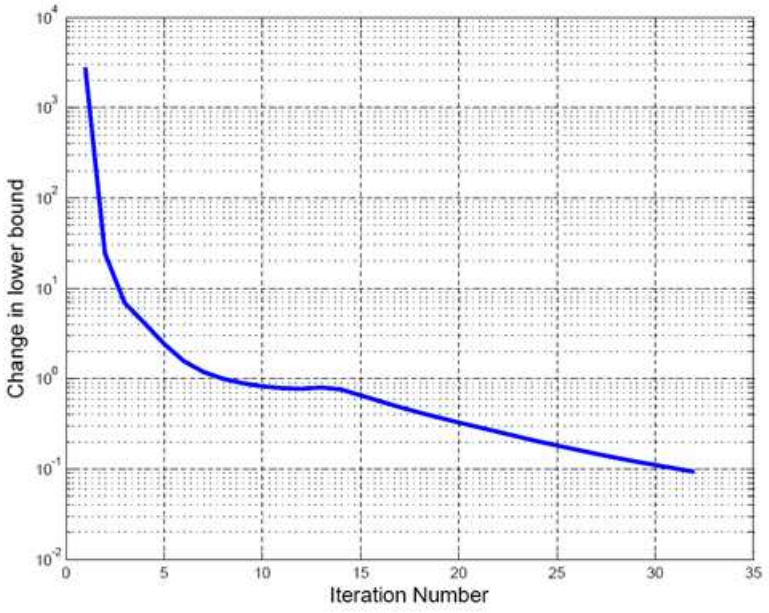

Figure 2-4: Emprical Convergence of the HML-LPHA Algorithm

This was shown in Figure 2-5. This result motivates cruder approximations, such as using hard EM where only the most likely trajectory is used for model learning.

We also show a strong empirical validation of the sensitivity of EM to model initialization. We find that in the approximate EM case, this sensitivity can be particularly detrimental to the performance, causing divergence of the transitions probability matrices and MAP mode estimation errors as show in Figures 2-9 and 2-10. The reason for which our approximate EM approach is particularly sensitive to initialization has to do with the pruning of the trajectory space where a low observation probability resulting from a poorly chosen initialization set can cause pruning of the true trajectory early on in the EM process. Subsequent chapters of this thesis, in particular Chapter 4 discusses how to improve the difficulty of the learning problem by taking advantage of available information in an efficient manner, and Chapter 3 discusses how to improve the chances of converging to the global maximum of the log-likelihood function by enabling a more expansive search of the parameter space.

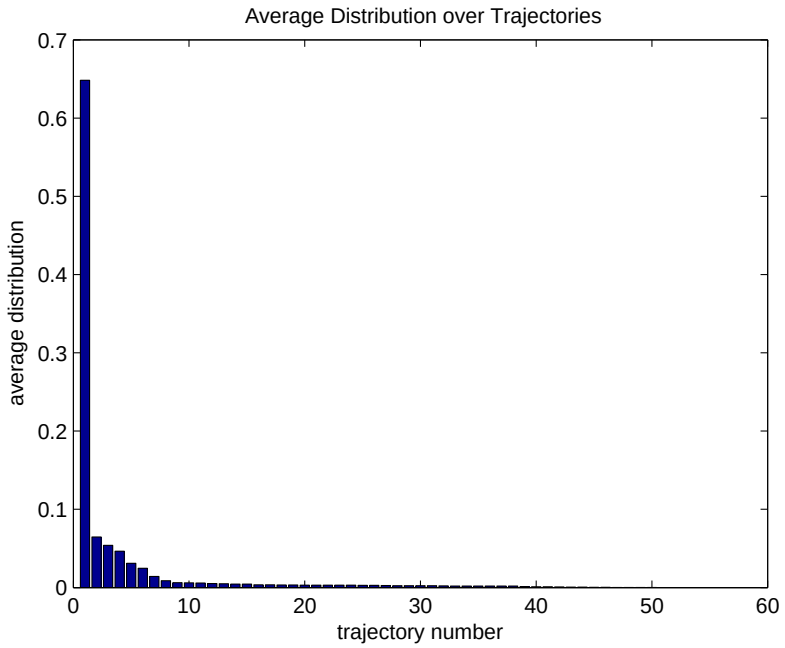

Figure 2-5: Distribution over Fifty Most Likely Trajectories, Averaged over 100 Runs

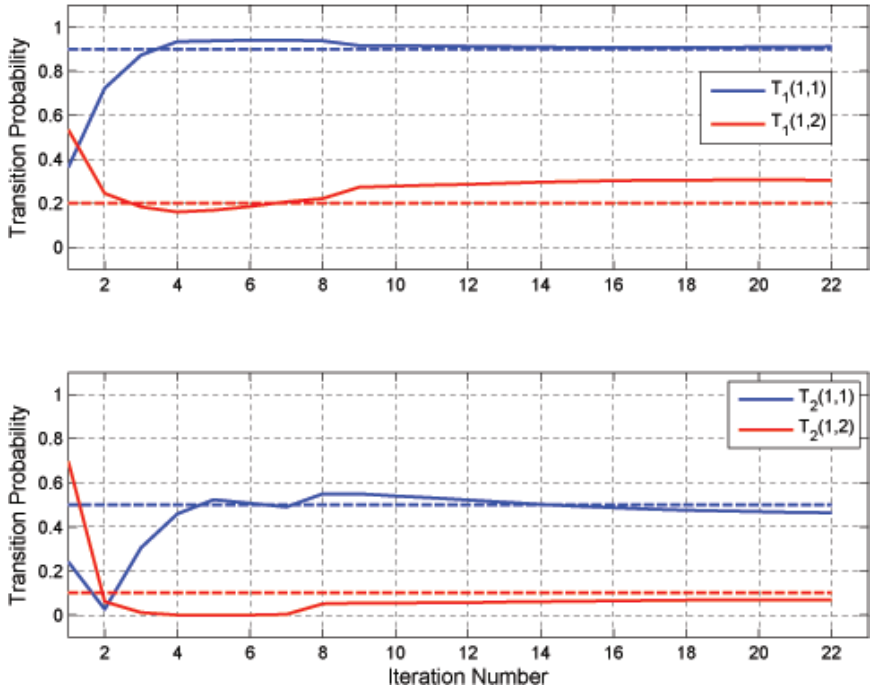

Figure 2-6: Empirical Convergence of Learned Transition Probabilities to True Transition Probabilities for Two Guard Regions

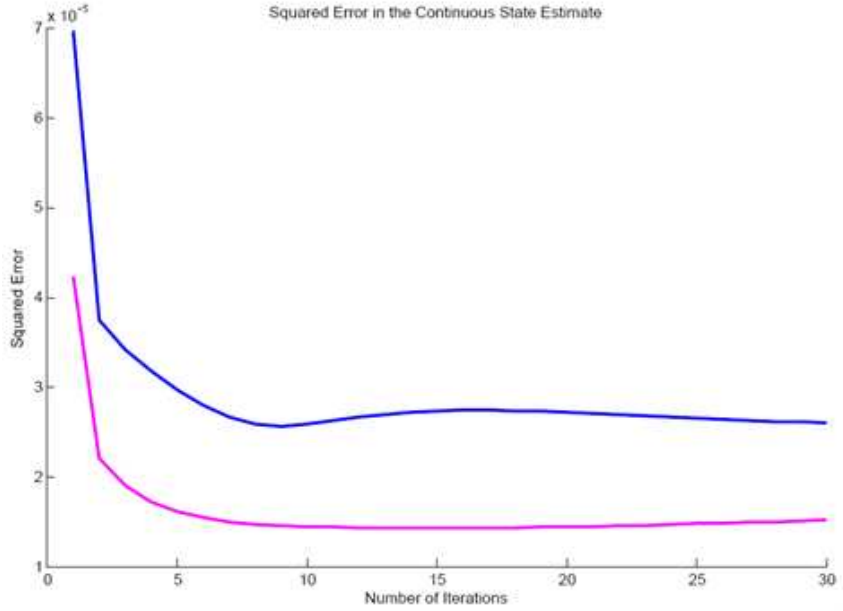

Figure 2-7: Convergence of Squared Error in the Continuous State Estimate for State 1 (blue) and State 2 (magenta)

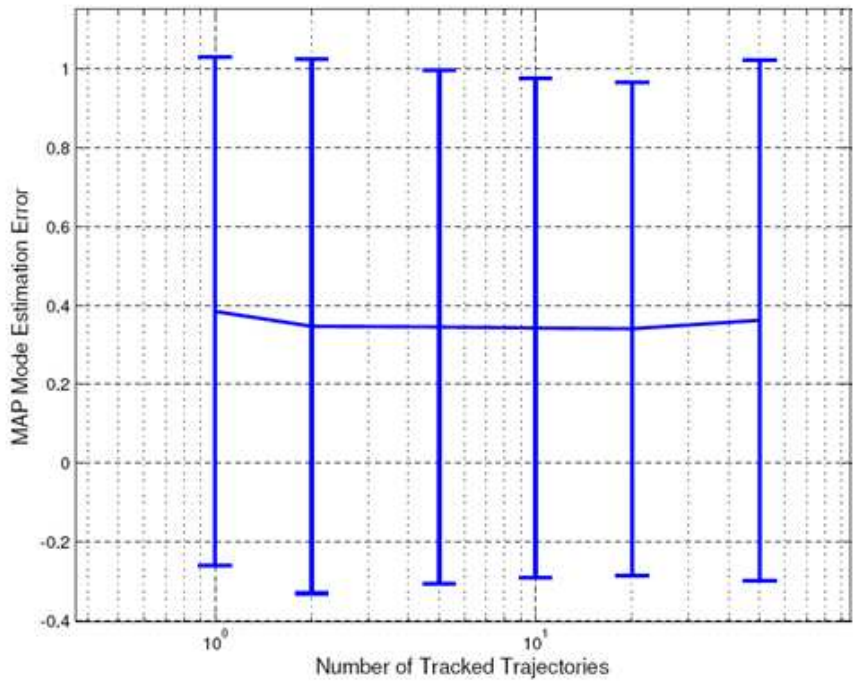

Figure 2-8: Aggregate Plot of MAP Mode Estimation Error versus Number of Trajectories

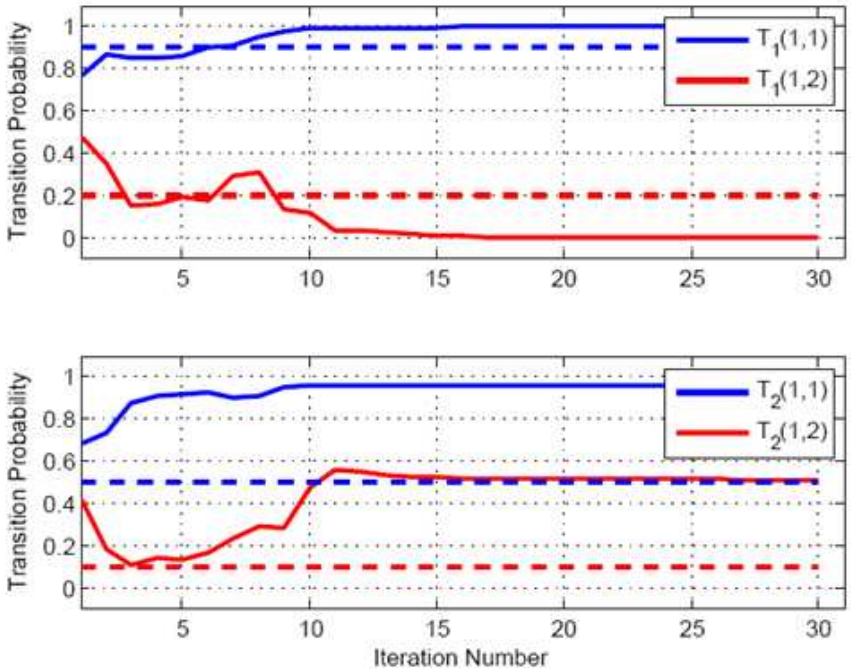

Figure 2-9: Plot Showing Divergence of the Learning Transition Probabilities due to Poor Quality of Initialization Model Parameters

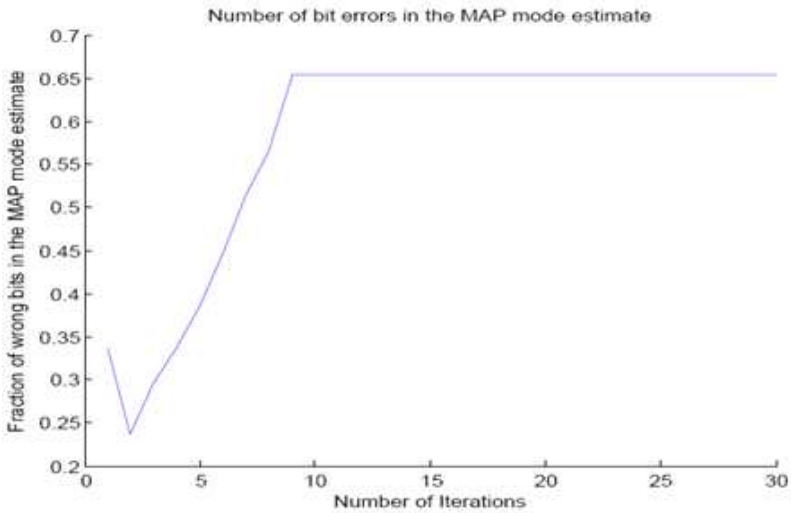

Figure 2-10: Plot Showing Divergence of the MAP Mode Estimate (2 discrete modes) due to Poor Quality of Initialization Model Parameters

2.4 Conclusions

We have demonstrated in simulation a new approach to learning the hybrid model parameters of LPHA using an approximate Expectation Maximization algorithm. This approach

1. Incorporates Autonomous Mode Transitions which allow for the dependence of the discrete state transitions on the continuous state
2. Allows for learning the effects of control inputs on the system.
3. Considers *soft* EM where we track the k-best or most likely trajectories, as opposed to *hard* EM where only the single most likely trajectory is tracked.

A summary of the main results of this chapter are found below:

- We have presented a new approximate EM approach for learning the hybrid model parameters for LPHA
- We demonstrate empirical convergence of our approximate EM method
- We use a best k-search, where only the k most likely trajectories are tracked, to allow for a tractable E-step in the hybrid case
- We find that the MAP mode estimation error and MSE continuous state error do not show a clear correlation with number of trajectories tracked. We find that a major contributing factor for this result is that on average about 65% of the probability mass is contained in the most-likely trajectory, thus supporting cruder approximations such as hard EM where only the most-likely trajectory is tracked.

Chapter 3

K-Means Clustering for Jumping out of Local Maxima in EM

As discussed in the previous chapter, Expectation Maximization based model learning techniques provide promising methods that converge to a locally optimal set of model parameters. EM itself however, does not guarantee globally optimal solutions for problems like hybrid parameter estimation in which the likelihood function $g(\theta) = p(y|\theta)$ is multimodal, where, conceptually the function has multiple hills or local maxima. In this case, an EM-based parameter estimation technique can get stuck at the top of a local maximum hill of the likelihood function, far from the best parameter estimate. In our experiments, this proved to be the case for the EM-based HML-LPHA algorithm (see discussion in section 2.3.6). Figure 3-1 shows a hypothetical three-dimensional objective function where EM will climb to the top of the local maximum hill whose base contains the initialization parameters.

A standard stochastic method for jumping out of these local maxima is a restart method that samples from a distribution over the model parameters, often a uniform distribution, to find a new initialization point for the local optimization algorithm; in our case, Expectation Maximization. Restarting can be effective in general, if there is a high likelihood that the restart will jump to a new hill and that this hill has a higher peak than those discovered thus far. Our experimentation with a random restart method that selects a new set of initialization parameters from a uniform distribution resulted in many initializations to the same local maximum. In other words, much computational effort was expended on re-discovering the same hills in our $g(\theta)$, or objective,

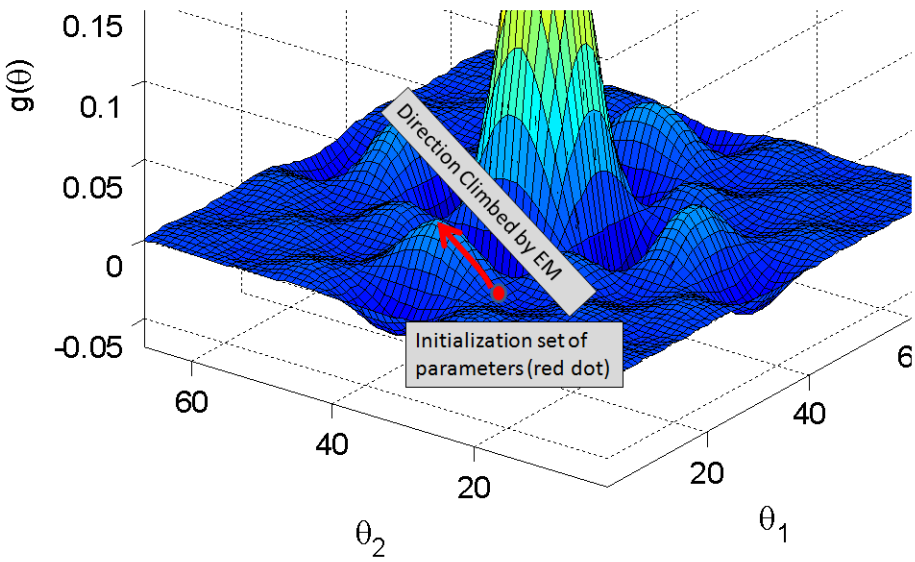

Figure 3-1: Plot of a hypothetical three-dimensional $g(\theta)$ function with many local maxima and one global maximum, showing local optimization performed by EM.

function. See Figure 3-2.

Hence, the key technical challenge is to develop a restart method that has a high likelihood of generating restart points, or initialization parameters, that lead to maxima outside of those already explored. Our approach is to select restart points that seem a priori promising, but lie outside of the bases of hills that have already been discovered. This would lead to an algorithm that has a higher probability of discovering model parameter estimates that are optimal, or near-optimal solutions to the maximization of the likelihood function. To do this we introduce an approach that uses k-means clustering to learn the probability that a point is within the base of a hill that has already been explored. We then propose a probabilistically based method for updating our prior probability of a promising initialization point based on our learned distributions of the discovered maxima hills.

3.1 Overview of Technical Approach and Chapter Organization

We propose an algorithm that combines approximate Expectation Maximization and K-Means Clustering that allows for a thorough search over the space of model parameters and a systematic method of forcing EM to jump away from local maxima of the log-likelihood function. This

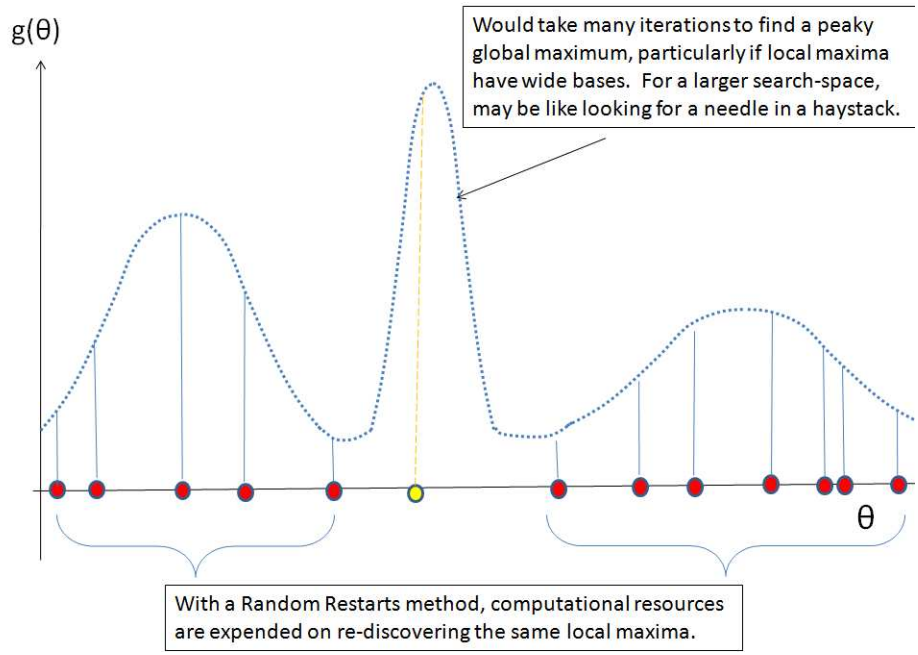

Figure 3-2: Schematic drawing of hypothetical two-dimensional log-likelihood objective function, $g(\theta)$, showing initialization points leading to repeated discovery of same local maxima.

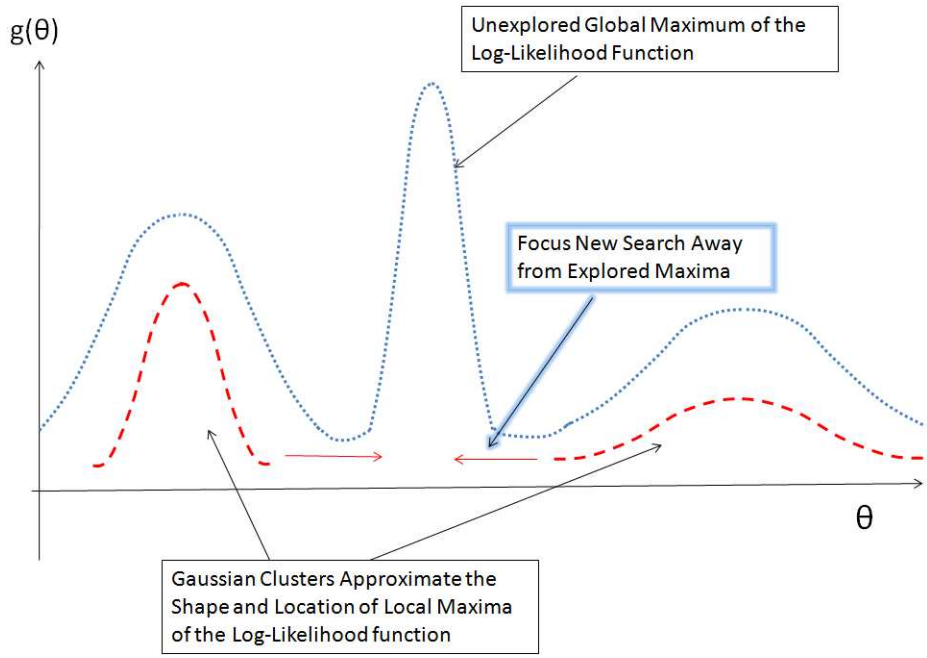

Figure 3-3: Schematic drawing of hypothetical two-dimensional log-likelihood objective function, $g(\theta)$, with explored local maxima approximated by Gaussian clusters.

algorithm consists of an initialization phase followed by three additional phases that are iterated to completion: a clustering phase, an optimization phase, and a labeling phase. We consider the algorithm to have reached completion either when the search space has been exhausted, or when a certain measure of performance has been met; such as a bound on squared error or a cluster with a desired $g(\theta)$ value has been found. We provide a skeleton of this process below and the full kMeans-EM algorithm in Figure 3.3.2.

1. **Initialization Phase:** In the initialization phase, EM is used to provide an initial labeling for a pool of different sets of parameters $\{\theta_{0i}\}_{i=1}^{n}$ where n is the total number of models and the label is a converged set of model parameters, θ_{f_i}. We set the number of clusters, k, to 1.

2. **Clustering Phase:** In the clustering phase, the labeled parameters, $\{\theta_{f_i}\}_{i=1}^{n}$, are clustered into k groups where the clustering uses Euclidean distance between sets of model parameters. Each of these clusters is defined by a Gaussian distribution whose mean is the centroid of the cluster, and whose variance is the empirical variance calculated over the members of each cluster. The resulting Gaussian clusters are used as representations of local maxima hills of the log-likelihood function, $g(\theta)$.

3. **Optimization Phase:** The goal of this phase is to find a new set of model parameters that is an *a priori* likely set of model parameters and that has a low probability of converging to any of the explored local maxima of $g(\theta)$. We can frame this as a maximization problem over the function $s(\theta)$ that uses the Gaussian cluster information and the prior over model parameters to produce a locally optimal solution, $\widetilde{\theta}_0$, that is both a likely set of model parameters and is not likely to converge to known local maxima.

4. **Labeling Phase:** This phase uses EM to label the new initialization point $\widetilde{\theta}_0 \rightarrow \widetilde{\theta}_f$. If $\widetilde{\theta}_f$ belongs to a new cluster, then augment the number of known clusters, $k = k + 1$. Return to the clustering phase.

Figure 3-4: Sketch of kMeans-EM Algorithm

We assume that the log-likelihood function, $g(\theta)$, that we are trying to map via Gaussian clusters, is smooth and that initializations that are closer to the peak of one local maximum, say ${\theta^*}_1$, will converge to this local maximum versus another, ${\theta^*}_2$, that is farther away. This phenomenon is common for smooth, non-convex but bounded variance objective functions and the convergence of EM to a local maximum hill that is nearest to its initialization point is well documented and is often referred to as "Initialization Sensitivity". Generally, this is not a desired property of EM, however in the kMeans-EM algorithm we take advantage of this. For this reason we choose Euclidean distance as a measure for clustering model parameters into a common group. We also choose Gaussian distributions to approximate the shape and location of local maxima in $g(\theta)$ because a

Gaussian distribution will assign a higher probability mass to the event of a set of model parameters θ converging to ${\theta^*}_1$ if it is closer to this maximum point in the Euclidean sense. There are other distributions that may also achieve this effect, however we choose Gaussian because the large body of existing results for Gaussian distributions make them easier to work with in many cases.

The last phase of the algorithm requires one to determine whether or not $\widetilde{\theta}_f$ belongs to a new cluster. Currently, we use properties of the clustering algorithm to determine whether or not this is the case. One can compute the silhouette value of a cluster, or the normalized separation between clusters, to determine degree of accuracy in the groupings. A positive silhouette value between 0 and 1 indicates the confidence value of the current cluster arrangement. This value can be used as a rule of thumb to determining the correct number of current clusters. Other approaches include evaluating the probability of the labeled set of parameters, $\widetilde{\theta}_f$, belonging to any of the existing clusters and declaring a new cluster if this value is below some threshold.

We discuss the clustering part of the algorithm in Section 3.3 where we discuss how we cluster the data using K-Means clustering, our definitions of a cluster, and the Gaussian distribution that we define for each cluster. The optimization phase of the algorithm is discussed in Section 3.3.4 where we present the criteria for a suitable $s(\theta)$ objective function and derive three such objective functions. The last phase consists mainly of using the EM algorithm to provide a labeling, or a mapping to a converged set of model parameters, $\widetilde{\theta}_0 \rightarrow \widetilde{\theta}_f$. This was discussed in Chapter 2, section 2.2. Empirical results for the clustering and optimization phases, as well as aggregate results that characterize the performance of the kMeans-EM algorithm against a Random Restarts method are found in Section 3.4.

3.2 Background on Stochastic Methods for Jumping out of Local Maxima

Our problem with EM being stuck in a local maximum hill whose peak corresponds to a suboptimal parameter estimate is an instance of the general problem of finding a global optimum for non-convex objective functions. Because convergence to local maxima of the objective function is so prevalent in numerical optimization methods, there exist a large array of methods to combat this problem. Amongst these are random restart methods, Genetic Algorithms, and Simulated Annealing type

methods [22, 26]. Random restart methods typically incorporate a uniform sampling of the search space. The idea is that, being lucky enough, one could find a set of initialization parameters that is at the foot of a global maxima of the objective function that one is optimizing. One pitfall of this method is that many starting points lead to local maxima that have already been found and thus much computation time is wasted. Simulated Annealing is another restart method that attempts to converge to the global maximum of the objective function by climbing its hills while allowing for systematic jumps to other areas of the search space. The rate at which these jumps are allowed are regulated by the annealing temperature and other such parameters. Other optimization techniques, such as Genetic Algorithm (GA) based or Simultaneous Perturbation approaches (SP) are also promising for finding local or global maxima of the objective function [37]. However, these referenced algorithms, along with many others, require evaluations of the objective function and/or evaluations of the gradient of the objective function, both of which are not available in the case of hybrid model learning.

The current chapter presents a new stochastic optimization method that does not rely on evaluating the objective function, $g(\theta) = \log(p(y|\theta))$, but provides guidance of the search away from explored local maxima of the objective function. Thus, the method does not suffer from the wasteful computational expense of sampling from within searched regions of the parameter space as do Random Restarts, and does not require evaluations of the objective function that in many cases is hard or impossible to evaluate. For the case of LPHA, evaluation of the objective function $\log(p(y|\theta))$ is indeed intractable as discussed in Chapter 2 of this thesis.

3.3 K-Means Clustering and EM

3.3.1 Review of K-Means Clustering

This section provides a succinct overview of k-means clustering. A comprehensive exposition of k-means clustering can be found in [17]. The goal of k-means clustering is to group data into clusters where each of these clusters is characterized by a mean and a variance.

K-Means Clustering Problem Statement:

Given a set of n data points in d-dimensional space R^d, and an integer k, determine a set of k

centroid points c_j,$j \in [1, \ldots, k]$ in R^d so as to minimize the mean squared distance from each data point in a set $\mathcal{S}$ to its nearest center c_j where $j \in \{1, \ldots, k\}$.

An outline of the k-means clustering algorithm is presented below:

1. WHILE (centroids $c_j's$ keep moving)
2. FOR $i \in \mathcal{S}$ where $\mathcal{S}$ is the set of all data points x_i,$i \in \{1, ..., n\}$
3. Assign $G_j \leftarrow i$ where G_j is the group of data points containing c_j such that $c_j = \arg\min_c \sum_{j=1}^{k} \left\| x_i^{(j)} - c \right\|^2$
4. ENDFOR
5. When all i's have been assigned, recalculate the positions of the centroids c_j, $j \in [1, \ldots, k]$.
6. ENDWHILE

Figure 3-5: K-Means Clustering Algorithm

Thus, k-means clustering is an iterative algorithm that repeatedly calculates cluster centers by finding the data point $c_j \in G_j$ that minimizes the distance from it to any other data point $i \in G_j$, where $j \in [1, \ldots, k]$.

Assuming that the data points are generated independently from k different multivariate distributions in R^d, let $\{\mathbf{X}\} = [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}]$ denote a random vector from this distribution. We can group this data into k clusters each with mean μ_j and covariance Λ_j. Furthermore, we can evaluate the pdf of this distribution for any data point x_n:

$$p(x_n) = \frac{1}{\sqrt{2\pi^d |\Lambda_j|}} \exp(\frac{-1}{2}(x_n - \mu_j)^T \Lambda_j^{-1} (x_n - \mu_j))$$

The value of the pdf at x_n gives us an idea of how likely x_n is to belong to the Gaussian cluster with mean μ_j and covariance Λ_j. This is an important observation and is key for our application of k-means clustering to jumping out of local maxima in EM. We present the extension of k-means clustering to the EM local convergence problem in the next section.

3.3.2 K-Means Clustering Applied to the Local Convergence Problem in EM-based Model Learning

In this section we combine a k-means clustering approach with Expectation Maximization for Linear Dynamical Systems as presented in 2.3. The extension of this method to Linear Probabilistic Automata (LPHA) will be discussed in 3.4.3. On the highest level, this algorithm iterates between two recursive steps:

for each data point $i \in \mathcal{S}$

1. **Estimate Probability**: Given the parameters of the Gaussian clusters, compute the probability that i belongs to each of the k Gaussians

2. **Estimate Model**: Given the set of data points $\{i\}_j$ belonging to cluster j where $j \in \{1, \ldots, k\}$, estimate the mean and variance of each of the k Gaussians.

We wish to use k-means clustering to identify when an initialization θ_0 will converge to the same θ_f and thus, to the same local maxima of the log-likelihood function. The initial estimate for the Gaussian model generating the data will be provided via EM and then k-means clustering will be utilized as a means of iterating through the two steps presented above. A more explicit description of the algorithm is presented below:

The first step consists of running EM n times, once for each set of initialization parameters. The generation of the initial set of parameters $\{\theta_{0_i}\}_{i=1}^n$ is independent of the rest of the algorithm. One way of producing this initial set is via a uniform distribution over the parameter space. The reader is referred to 2.3 for implementation details for the EM algorithm in step 1. Initially, $k = 1$, meaning that we assume the existence of only one cluster. As the algorithm iterates, the number of clusters will change to reflect the number of local minima identified in the log-likelihood function. Steps 2,3, and 4 are key components in uniting the EM for parameter estimation and k-means clustering algorithms to form the KMeans-EM algorithm. These steps include fitting k Gaussian clusters to the $\{\theta_{f_i}\}_{i=1}^n$, and generating $\widetilde{\theta}_0$ such that running EM on $\widetilde{\theta}_0$ would converge to a $\widetilde{\theta}_f$ corresponding to a local maxima of the log-likelihood function that had not previously been visited. These steps are discussed more in detail in the subsequent sections.

1. Set k=1. Run EM on many different initializations to label, or map, each initialization with a corresponding converged set of parameters θ_f
   $\theta_{0_1} \rightarrow \theta_{f_1}$
   $\theta_{0_2} \rightarrow \theta_{f_2}$
   $\vdots$
   $\theta_{0_n} \rightarrow \theta_{f_n}$
2. Apply k-means clustering algorithm to group $\{\theta_{f_i}\}_{i=1}^n$ into k clusters each with centroids c_j and covariances Λ_j
3. Generate a new guess of parameters $\widetilde{\theta}_0$ that is likely to converge to a new local maxima of $g(\theta)$ by maximizing the objective function $s(\theta)$
4. If $\widetilde{\theta}_0$ has a low probability of belonging to any of the existing k clusters, go to step 5, else return to step 3.
5. Run EM using $\widetilde{\theta}_0$ as initial guess of parameters to find $\widetilde{\theta}_f$. Set $n = n + 1$ and $k = k + 1$ and go to step 2.

Figure 3-6: The kMeans-EM Algorithm

3.3.3 Clustering the $\{\theta_{f_i}\}_{i=1}^n$

The goal in this section is to provide a characterization of a cluster defined over sets of parameters $\{\theta_{f_i}\}_{i=1}^n$.

Definition of a Cluster over a Set of Parameters:

We define a cluster G_j over a set of parameters to be a Gaussian distribution with a mean μ_j and a covariance Λ_j such that

$\mu_j = c_j =$ *centroid for cluster j*

$\Lambda_j =$ *empirical covariance for cluster j*

The mean of each cluster is equivalent to the centroid of each cluster computed using the standard k-means clustering algorithm 3.3. We now discuss the calculation of the empirical covariance.

In Section 2.3 we defined the set of parameters θ to be composed of various matrices A, B, C, D, Q, R. In this section, we must define a *vector* of parameters $\vec{\theta}$ that decomposes all matrices component-wise so that we can compute the empirical variance of each cluster.

Definition: $\vec{\theta}$

We define the vector of parameters $\vec{\theta}$ such that

$$\vec{\theta} = [a_{11}, a_{12}, \ldots, a_{nn}, b_{11}, \ldots, b_{1n}, c_{11}, c_{12}, \ldots, c_{nn}, d_{1n}, \ldots, d_{1n}, q_{11}, q_{12}, \ldots, q_{nn}, r_{11}, r_{12}, \ldots, r_{nn}]^T$$

where $A \in \mathcal{R}^{nxn}, B \in \mathcal{R}^{nx1}, C \in \mathcal{R}^{nxn}, D \in \mathcal{R}^{nx1}, Q \in \mathcal{R}^{nxn}$, and $R \in \mathcal{R}^{nxn}$.

If we view each set of parameters θ as a vector $\vec{\theta}$ and we have n such vectors, we now have n observations of each parameter value and can compute the empirical variance. This calculation becomes

$$\Lambda_j = \frac{1}{n-1}\left(\sum_{i=1}^{n}\left(\vec{\theta_{f_i}} - \vec{\mu_j}\right)\left(\vec{\theta_{f_i}} - \vec{\mu_j}\right)^T\right)$$

Knowing the mean and covariance of each Gaussian cluster, we can now evaluate the pdf of a given cluster G_j for any vector set of parameters $\vec{\theta_0}$

$$p\left(\vec{\theta_0}\right) = \frac{1}{\sqrt{(2\pi)^d\,|\Lambda_j|}} exp\left(-\frac{1}{2}\left(\vec{\theta_0} - \vec{\mu_j}\right)^T (\Lambda_j)^{-1}\left(\vec{\theta_0} - \vec{\mu_j}\right)\right)$$

3.3.4 Generating $\widetilde{\theta}_0$ that Converge to new Local Maxima

In the last subsection we discussed how to cluster sets of parameters into Gaussians whose means and covariances approximate the peak of, and width of, the local maxima of the log-likelihood function respectively. We can use this as a blueprint of where the discovered local maxima hills are and as a guide for where to look for any unexplored maxima. We can develop different heuristics as guides for searching the space of possible model parameters. The properties that we wish to embody in any heuristic are the following

1. **Search Outside of Explored Local Maxima:** We wish to seed the next iteration of our model learning algorithm with a set of model parameters that have a high probability of not converging to the same local maxima of the log-likelihood function $g(\theta) = \log(p(y|\theta))$. This way we can perform a thorough search of the solution space while minimizing waste of computational resources. In order to achieve this, we use a heuristic that takes into account

cluster shape and covariance.

2. **Constrain search to Be Within Feasible Areas of the Search Space:** If our only goal was to find a set of model parameters far from visited local maxima of the log-likelihood function, we could trivially achieve this by always choosing a set of parameters at the edges of the search space, no matter how infeasible these solutions may be. For this reason we need to enable a constraint that will guide the optimization toward areas of the search space that are likely to be valid sets of model parameters.

In this section of the thesis we develop three heuristics that have these desired properties. We first derive a principled approach to computing an objective function whose maximization yields a set of parameters $\widetilde{\theta}_0$ that has a high probability of not belonging to any of the existing clusters while maximizing the probability of belonging to a prior distribution over valid model parameters. We then develop two approximations to this objective function, one being a function of rational polynomials and the other being a quadratic function, that also embody these properties but may in certain cases be easier to work with.

Minimizing the Probability of Belonging to an Existing Local Maximum

In this section we develop an approach for finding a set of parameters $\widetilde{\theta}_0$ that minimizes the probability of belonging to any existing cluster. We assume the probability of $\widetilde{\theta}_0$ belonging to any cluster is independent of any other cluster. We define the probability that a certain set of model parameters θ belongs to cluster j as $q_j(\theta)$.

$$q_j(\theta) = P\left\{\theta \in G_j\right\} \tag{3.1}$$

where G_j denotes cluster j

We wish to minimize the probability of θ belonging to any of the existing clusters and we can thus phrase this as a maximization of the probability that θ belongs to *none* of the clusters. Thus we define our objective function to be $s(\theta)$ and our optimization problem becomes

$$\widetilde{\theta}_0 = \arg\max_{\theta} s(\theta) = \arg\max_{\theta} \prod_{j=1}^{k} (1 - q_j(\theta)) \tag{3.2}$$

If we attempt to perform this maximization we quickly find that the resulting $\widetilde{\theta}_0$ is always at the edge of the search space and this is not what we are looking for. In order to bias our maximization to search in areas that are not at the edges of the search space we introduce a prior distribution to our objective function $s(\theta)$. We use a Gaussian prior that is updated at each iteration of the kMeans-EM algorithm where the mean is an average over all converged sets of model parameters labeled by EM. This prior is constructed using no prior knowledge of where valid model parameters lie, but instead uses information that is updated during each use of EM where converged model parameters are labeled and high log-likelihood areas of the search space are identified. If more *a priori* information is available about where valid model parameters can be found in the search space, this can be used to focus the search via the prior. If we denote our prior distribution over parameters to be $p_0(\theta)$, our maximization becomes

$$s(\theta) = \arg\max_{\theta} p_0(\theta) \prod_{j=1}^{k} (1 - q_j(\theta)) \tag{3.3}$$

$$\widetilde{\theta}_0 = \arg\max_{\theta} s(\theta) \tag{3.4}$$

where

$$p_0(\theta) \sim \mathcal{N}(\mu, \Lambda) \tag{3.5}$$

The prior distribution $p_0(\theta)$ can have a large influence on the result of the maximization and thus one should be careful when choosing a suitable distribution. The influence of the prior can be gradually diminished by adjusting the covariance to be arbitrarily large. This will be discussed in further detail in the subsection regarding objective function evaluation. We now discuss how to evaluate the probability of θ belonging to any cluster j where $j \in [1, \ldots, k]$.

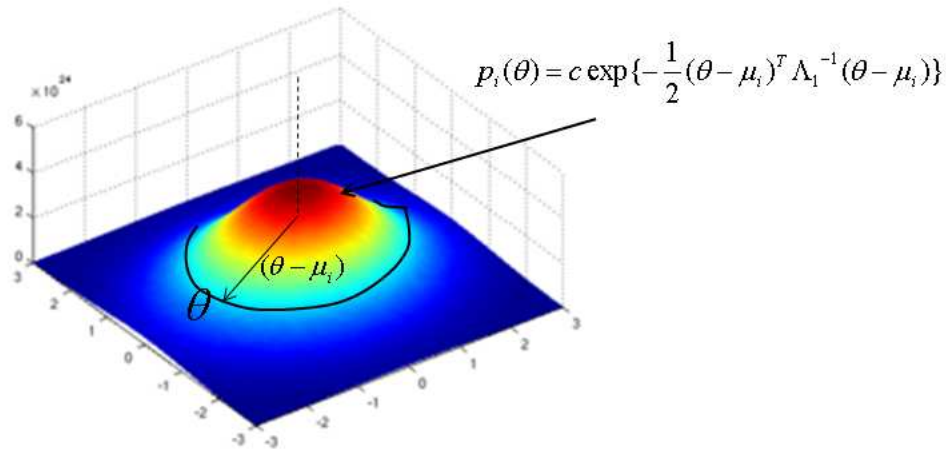

Figure 3-7: Gaussian cluster with mean μ_i and covariance Λ_i. The area of the Gaussian within the black ellipse passing through θ and centered at μ_i is the compliment of the probability that the set of parameters θ belongs to cluster j.

Finding $q_j(\theta)$: the Probability of Belonging to Cluster j

In order to find $q_j(\theta)$ we integrate over the pdf of the Gaussian cluster j. We define $\mathcal{E}$, an ellipse that is centered at the mean of the cluster, passes through the point θ, and whose major and minor axes are the eigenvectors of the covariance matrix for Gaussian cluster j. Note that in the special case where the covariance matrix for cluster j is a multiple of the identity matrix, this elliptical region becomes a circle. The integral of the pdf of cluster j over this elliptical region becomes the probability that θ *does not* belong to that cluster. This makes sense if we think in the asymptotic sense where θ is at infinity and thus the probability that θ does not belong to cluster j approaches 1. We define $q_j(\theta)$ mathematically as

$$q_j(\theta) = \int_{\mathcal{E}} p_j(\theta) \tag{3.6}$$

where

$$p_j(\theta) \sim \mathcal{N}(m_j, K_j) \tag{3.7}$$

Integrating over Gaussians cannot generally be done in closed form, however, integrating a Gaussian over the area of a disc or ellipse (for elliptical Gaussians) *is* possible in closed form. We take advantage of this to compute $q_j(\theta)$ in closed form.

We begin with the simpler case of integrating a two-dimensional Gaussian over a disc of radius α.

$$\int_{v_1^2+v_2^2\leq\alpha} \frac{1}{2\pi} \exp^{-\frac{1}{2}v_1^2} \exp^{-\frac{1}{2}v_2^2} dv_1 dv_2 \tag{3.8}$$

$$= \int_{\theta\in[0,2\pi]} \int_{r\in[0,\alpha]} \frac{1}{2\pi} \exp^{\frac{-1}{2}r^2} \tag{3.9}$$

$$= 1 - \exp^{\frac{-1}{2}\alpha^2} \tag{3.10}$$

Note that we integrate assuming that the Gaussian cluster has zero mean and we integrate over $r \in [0, \alpha]$. This does not affect the integration and a non-zero Gaussian mean can be easily accounted for after performing the integration. We now extend this analysis to the case where our Gaussian is multi-dimensional and we are integrating over the elliptical area denoted by $\mathcal{E}$.

$$P\left\{\theta \text{ being outside of cluster } j\right\} = \int_{\mathcal{E}} p_j(x)dx \tag{3.11}$$

$$p_j(x) = \frac{1}{\sqrt{(2\pi)^d det(\Lambda_j)}} \exp^{\frac{-1}{2}x^T\Lambda_j^{-1}x} \tag{3.12}$$

$$\Lambda = U^T\Sigma U \text{ where } \Sigma \text{ is diagonal, and } U \text{ is an orthogonal matrix} \tag{3.13}$$

$$\mathcal{E} = \left\{x : x^T\Lambda^{-1}x \leq \theta^T\Lambda^{-1}\theta\right\} \tag{3.14}$$

and d is the dimension of x

We make the following substitutions.

$$x^T\Lambda^{-1}x$$

$$= x^TU^T\Sigma^{-1}Ux$$

$$= y^T\Sigma^{-1}y$$

$$= y_1^2\lambda_1^{-1} + y_2^2\lambda_2^{-1} \text{ for the two-dimensional case}$$

Our integral becomes

$$\int_{y^T\Sigma^{-1}y\leq\theta^T\Lambda_j^{-1}\theta} \frac{1}{\sqrt{(2\pi)^d det(\Sigma)}} \exp^{\frac{-1}{2}y^T\Sigma^{-1}y} dy_1\cdots dy_n \tag{3.15}$$

$$\tag{3.16}$$

Expanding out the above integral for the two-dimensional case, our integral becomes:

$$\int_{y_1^2\lambda_1^{-1}+y_2^2\lambda_2^{-1}\leq\alpha} \frac{1}{\sqrt{2\pi\lambda_1}} \exp^{\frac{-y_1^2}{2\lambda_1}} \frac{1}{\sqrt{2\pi\lambda_2}} \exp^{\frac{-y_2^2}{2\lambda_2}} dy_1 dy_2 \tag{3.17}$$

where

$\alpha = \theta^T\Lambda^{-1}\theta$ as before

Completing the integral in polar coordinates and generalizing to the multivariate Gaussian case with non-zero mean μ_j we find the probability of any set of parameters θ belonging to cluster j, which we denote G_j.

$$P\left\{\theta\in G_j(\mu_j,\Lambda_j)\right\} = \exp^{\frac{-1}{2}\left[(\theta-\mu_j)^T\Lambda_j^{-1}(\theta-\mu_j)\right]^2} = q_j(\theta) \tag{3.18}$$

Thus we find a closed form solution for the probability of a set of parameters θ being in cluster j. We can now use this result in our objective function expression.

Evaluating the Objective Function $s(\theta)$

In order to find a new guess for initial model parameters, $\widetilde{\theta}_0$, we must maximize our objective function. Using the results of the last subsection we can write our maximization as:

$$s(\theta) = p_0(\theta)\prod_{i=1}^{k}(1-\exp^{\frac{-1}{2}\left[(\theta-\mu_i)^T\Lambda^{-1}(\theta-\mu_i)\right]^2}) \tag{3.19}$$

$$\widetilde{\theta}_0 = \arg\max_{\theta} s(\theta)$$

Our objective function is of the form of a Gaussian prior multiplying a product of exponential functions. The form of this objective function makes an analytical solution very challenging. Although a closed form solution is not available, an off-the-shelf numerical optimization algorithm does provide a local solution. It is also important to note that the maximization of this objective function provides a heuristic for guiding the search of the next initialization set of parameters. Other objective functions can be used that also capture the local maxima information provided by the Gaussian clusters and may be easier to maximize than the objective function presented in (3.20). Two such options will be investigated in the next subsection but for now we will focus on the objective function $s(\theta)$. We will also discuss the role of the prior distribution $p_0(\theta)$ in $s(\theta)$.

We demonstrate a few plots of $s(\theta)$ for the case where we have two elliptical Gaussian clusters centered at $x = 5, y = 0$ and $x = 0, y = 5$ respectively, and a Gaussian prior centered at $x = 0, y = 0$. For the first plot, figure 3-8, we allow for the Gaussian prior to have a covariance of $\Lambda_0 = 5\mathcal{I}$, where $\mathcal{I} \in \mathcal{R}^{2\text{x}2}$ is the identity matrix. Keeping in mind that we wish for our next set of initialization parameters $\widetilde{\theta}_0$ to be far from the center of the Gaussian clusters, but close to our Gaussian prior $p_0(\theta)$, we mark the areas that we wish our objective function to have local maxima. This plot is shown in figure 3-9.

We show the plotted objective function in figure 3-10 and show that indeed the local maxima of the function can be found in the areas where the Gaussian cluster pdf values are low and the Gaussian prior pdf value is high. This plot also shows the influence of the Gaussian prior on $s(\theta)$. In particular, solutions near the mean of the Gaussian prior are largely favored. This may not be the desired effect of the Gaussian prior if one does not have an idea *a priori* where the true set of model parameters may be, which is likely the case. However, because a reason for which we use a prior is to avoid the maximization always returning solutions that are at the edge of the search space, we can make our Gaussian prior have a covariance that is arbitrarily large. A large covariance for the Gaussian prior allows for the effect of the prior to be gradually diminished as the relatively high peaks of the Gaussian clusters now have a much higher repelling effect than the attracting effect of the prior. This is shown in plot 3-11 where the Gaussian prior was assigned to have a covariance of $\Lambda_0 = 50\mathcal{I}$. The local maxima of the objective function are now located farther from the Gaussian clusters and cover a wider portion of the search space.

Figure 3-12 shows a plot of the objective function when the Gaussian prior is assigned to have

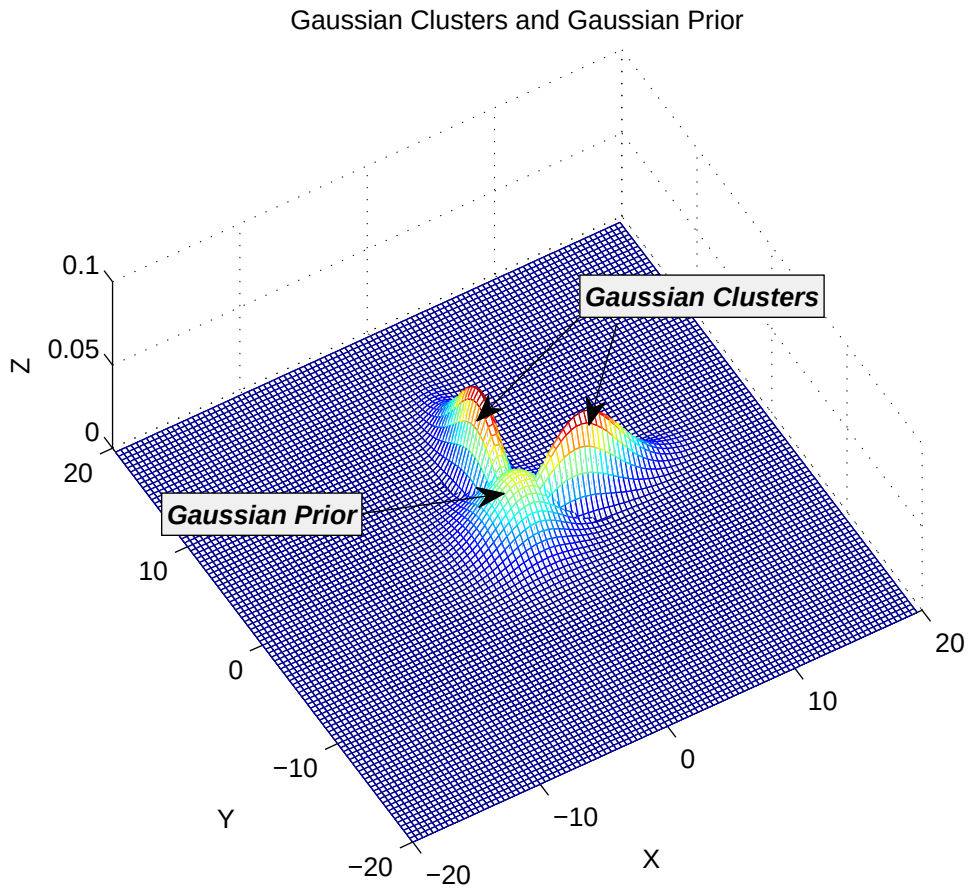

Figure 3-8: Schematic of two possible Gaussian Clusters centered at $x = 5, y = 0$ and $x = 0, y = 5$ with a Gaussian prior centered at $x = 0, y = 0$.

an even larger covariance of $\Lambda_0 = 500\mathcal{I}$. In this case we see that this effect of the objective function favoring solutions that are more spread out amongst the search space and farther from the Gaussian clusters, is even more dramatic. In conclusion, we can allow the Gaussian prior to have a large influence on the search for the next set of initialization parameters if we have a good idea of where we wish to focus the search, or if we do not have a good guess for an a priori distribution over likely model parameters then we can diminish this effect by widening our Gaussian prior and thus approaching a more uniform prior over model parameters.

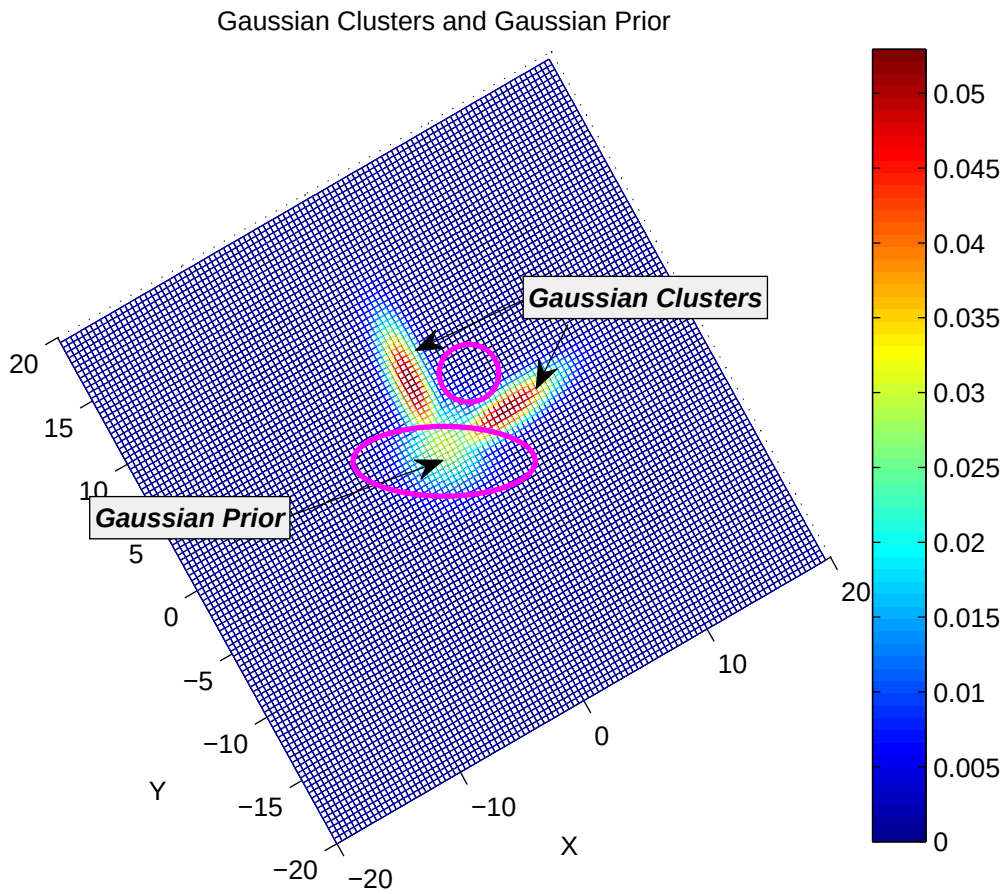

Figure 3-9: Contour plot of hypothetical Gaussian clusters with prior demonstrating areas (marked with purple ellipses) where we wish to focus the search for new model parameters.

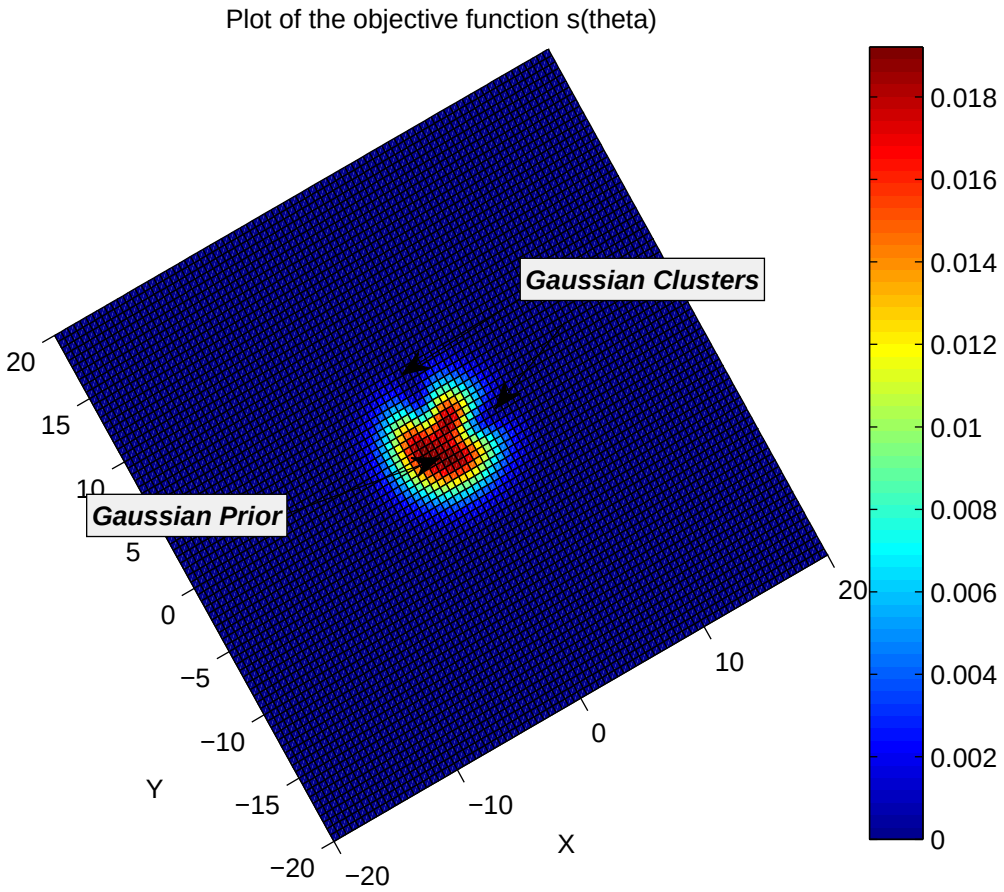

Figure 3-10: Contour plot of the objective function $s(\theta)$ whose maximization yields a new guess of model parameters away from existing clusters and toward a priori interesting areas of the search space as indicating by the Gaussian prior over model parameters.

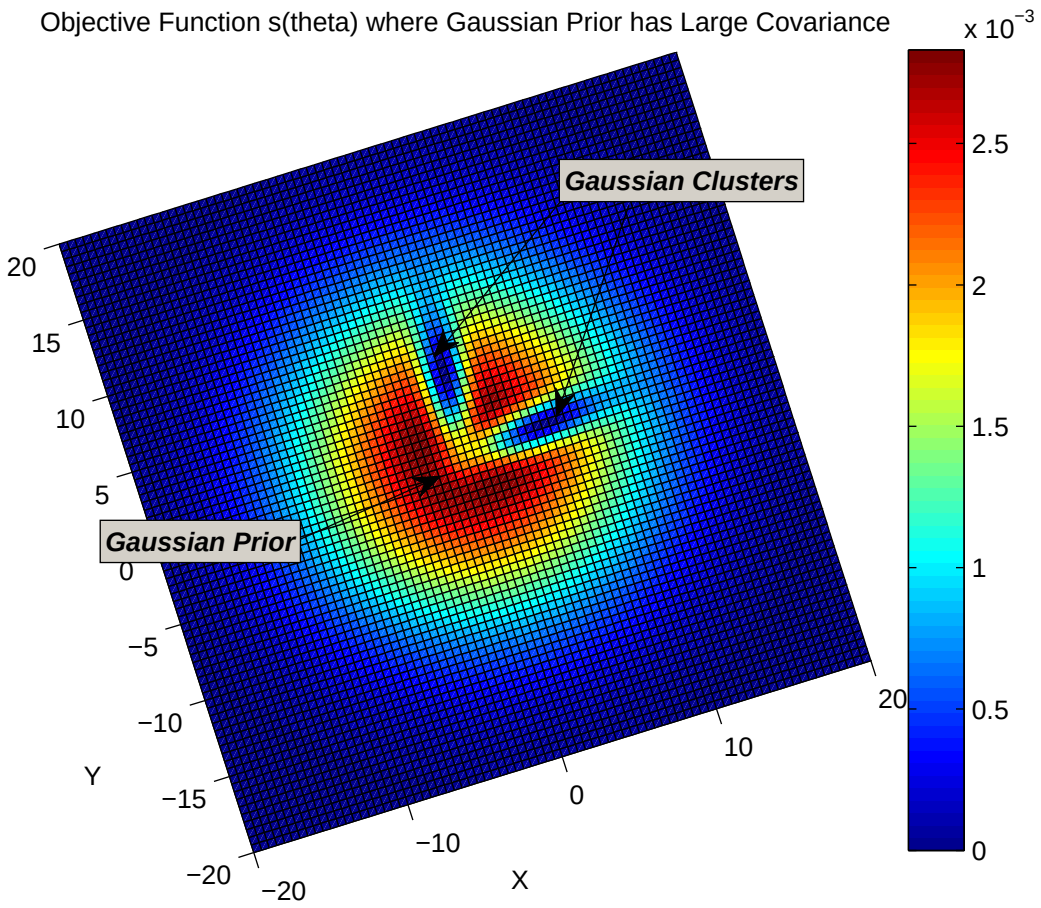

Figure 3-11: Contour plot showing the influence of the Gaussian prior on the objective function, $s(\theta)$. A wider prior covariance of $\Lambda_0 = 50\mathcal{I}$ produces maxima of $s(\theta)$ that are widely spread over the search space.

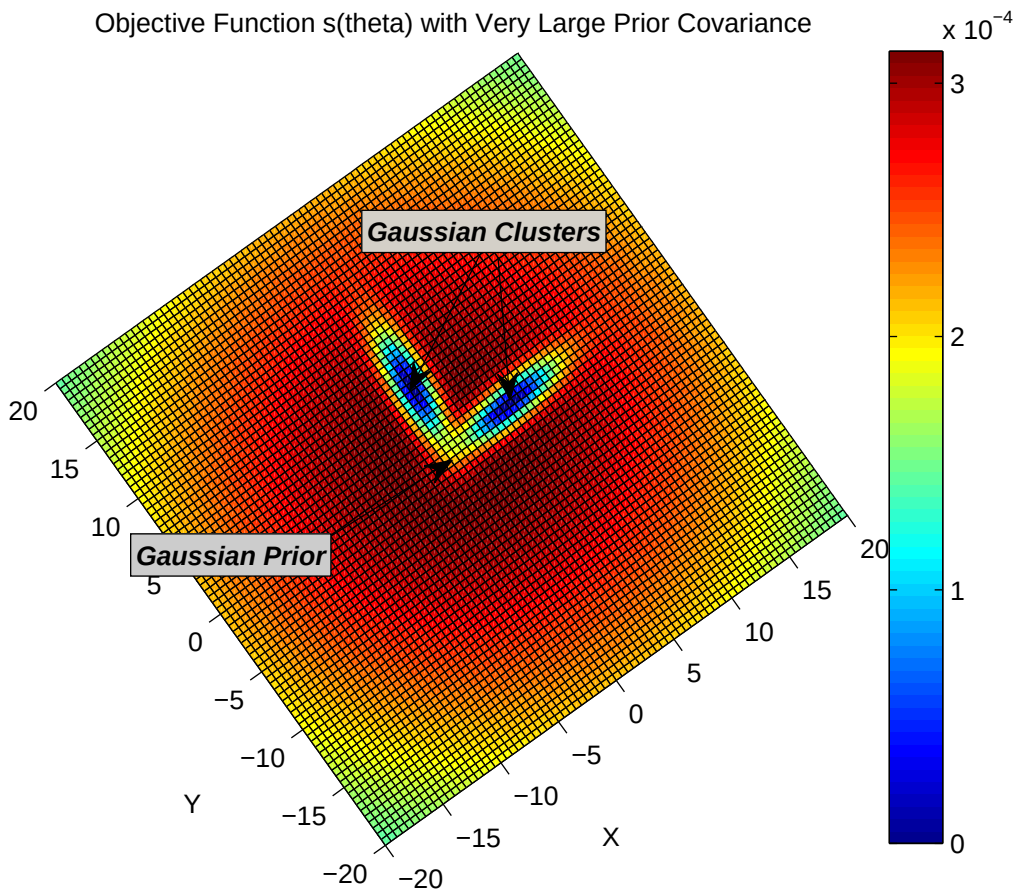

Figure 3-12: Second contour plot showing the influence of the Gaussian prior on the objective function, $s(\theta)$. A very wide prior covariance of $\Lambda_0 = 500\mathcal{I}$ begins to approximate the effect of a uniform distribution over the feasible search space region.

Approximations to the Objective Function $s(\theta)$

In the previous section we derived an objective function, $s(\theta)$, whose local maxima are outside of explored maxima of the log-likelihood function, $g(\theta) = \log(p(y|\theta))$, and are in the feasible areas of the search space as indicated by the Gaussian prior over model parameters. We also investigated the effect of the Gaussian prior on the objective function. In this subsection we present two approximations, or alternatives, to this objective function.

i. *Using Rational Functions of Polynomials as the Objective Function*

The expression for our objective function $s(\theta)$ is composed of a Gaussian prior distribution multiplying a product of exponential functions. We repeat this definition here for convenience.

$$s(\theta) = p_0(\theta) \prod_{j=1}^{k} (1 - \exp^{\frac{-1}{2}\left[(\theta-\mu_j)^T \Lambda_j^{-1} (\theta-\mu_j)\right]^2})$$

We emphasize that this objective function is not unique and that we can find an objective function that is of the form of a rational function of polynomials that has comparable performance to $s(\theta)$. We choose a rational function of second-order polynomials to approximate our Gaussian prior, and a rational function of first-order polynomials to approximate the exponential functions. We choose the form of these rational functions such that the function is constrained to remain between 0 and 1. For the rational function replacing the exponential, we further desire that the function approach 1 as the argument of the function approaches infinity. We present this approximation function as $s_R(\theta)$.

$$s_R(\theta) = \frac{1}{1 + {\alpha_0}^2} \prod_{i=1}^{k} \frac{\alpha_i}{i - \alpha_i} \tag{3.20}$$

where

$$\alpha_j = \frac{-1}{2} \left[(\theta - \mu_j)^T \Lambda_j^{-1} (\theta - \mu_j) \right] \tag{3.21}$$

The plot of this objective function evaluated using the same descriptions for the Gaussian clus-

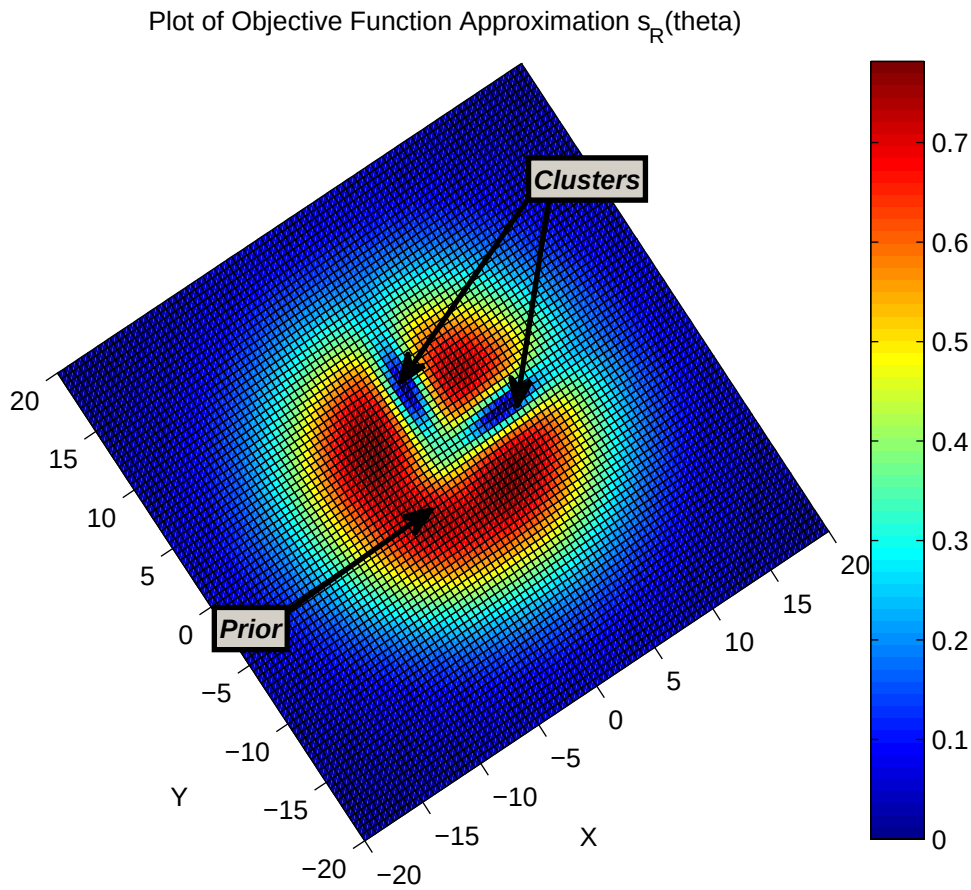

Figure 3-13: This contour plot shows the resulting approximation to the objective function $s_R(\theta)$, evaluated for two Gaussian clusters with means $x = 5, y = 0$ and $x = 0, y = 5$ respectively, and a Gaussian prior centered at $x = 0, y = 0$. We allow for the Gaussian prior to have a large covariance of $\Lambda_0 = 50\mathcal{I}$.

ters and prior as in the previous $s(\theta)$ plots, figure 3-8, is found in figure 3-13. Note that for this plot we allow for the Gaussian prior to have a covariance $\Lambda_0 = 50\mathcal{I}$. As can be seen by a quick comparison of the plots from the original objective function $s(\theta)$, figure 3-11, and the approximated objective function $s_R(\theta)$,figure 3-13, the locations of the local maxima of these functions are similar.

ii. *Using a Quadratic Objective Function*

We can also design an objective function that is of the form of a quadratic function. The advantage to maximizing a quadratic objective function is that quadratic programming software is

widely available and can provide exact solutions to the maximization. One limitation however, is that the constraints that are provided for the optimization must be linear. The objective function $s_Q(\theta)$ that we will develop is subject to quadratic, not linear constraints. However, because of the convex nature of a quadratic constraint, one can approximate this by a union of linear constraints and still use quadratic programming to solve the optimization problem. We do not show how to do this as part of this thesis. We define $s_Q(\theta)$ as

$$s_Q(\theta) = \|\theta_i - \mu_i\|^2_{\Lambda_i^{-1}} \tag{3.22}$$

So that our optimization now becomes

$$\widetilde{\theta}_0 = \arg\max_{\theta} s_Q(\theta) \tag{3.23}$$

subject to

$$\|\theta_i - \mu_i\|^2_{\Lambda_i^{-1}} = \|\theta_j - \mu_j\|^2_{\Lambda_j^{-1}} \; \forall j \in [1, .., k], j \neq i \tag{3.24}$$

where

$$\|\theta_i - \mu_i\|^2_{\Lambda_i^{-1}} \equiv (\theta - \mu_i)^T \Lambda^{-1} (\theta - \mu_i) \tag{3.25}$$

In words, this objective function finds the set of model parameters that is equidistant to all identified clusters. In this context the term equidistant takes into account the shape of the Gaussian clusters as well as the mean of each of the clusters. The norm operator is defined with respect to the covariance, specifically, the norm is taken in the coordinate system defined by the eigen vectors of the covariance matrix. The effect of taking this norm is that the shape and width of each cluster is respected when finding a suitable candidate of parameters $\widetilde{\theta}_0$. As an example, given two clusters where one of them has a very small covariance and thus has a peaky distribution, and the second cluster has a very large covariance and is thus flatter, a naive way to choose the set of parameters that is equidistant from both clusters is to choose the mean between the two cluster centers. This would likely result in a solution that is still within the area of the wider cluster and will thus converge to the same local maxima of this discovered hill. If instead a new set of parameters is

chosen by taking the point that is equidistant in the sense presented in equation (3.25), then the fact that one cluster is wider than the other is taken into account and the chance of converging to either of the two known hills is minimized. Also, because you are always finding the point equidistant to all clusters, the optimization will not return a set of parameters that is at the edge of the search space.

3.4 Simulation Results and Discussion

In this section we present simulation results for the kMeans-EM algorithm. We use an Autonomous Underwater Vehicle (AUV) simulation in Matlab where the true AUV dynamics model is a model of the AUVs used at the Monterey Bay Aquarium Research Institute in California,USA. The AUV dynamics model is derived and explained in full in [24]. The linearized AUV discrete time longitudinal dynamics are used for the purposes of this simulation. As discussed in [24], the linearized equations of motion for the AUV are split into a fourth order sway/yaw set, and a fourth order heave/pitch set. These equations are linearized about the solution for a constant AUV speed. We refer the reader to the MBARI AUV reference for further information on the AUV dynamics model.

We provide empirical results that demonstrate the resulting accuracy of Gaussians fit to clustered sets of model parameters, the result of optimizing the objective function $s(\theta)$ to find a new initialization point, and aggregate results characterizing the performance of the overall kMeans-EM algorithm. Our aggregate results are averaged over several trials and are plotted against algorithm iteration number. We find that the kMeans-EM algorithm consistently outperforms a Random Restarts method in the areas of best achieved log-likelihood value, and in minimizing Euclidean distance to the true model parameters. We also find that there is a positive trend in performance with iteration number of the kMeans-EM algorithm. This is because the algorithm gains a better understanding of where the existing local maxima are located, and thus generates $\widetilde{\theta}$ that help guide the search toward unexplored regions of the search space. In turn, as more area of the search space is explored, optimal or near-optimal solutions are found. Aggregate results plots presented in this section support this trend of improvement in performance with iteration number of the kMeans-EM algorithm, demonstrating discovery of better learned model parameters. In contrast, the Random Restarts method produces similar or fluctuating performance with iteration number, demonstrating

inefficient rediscovery of the same local maxima hills or discovery of non-optimal local maxima.

We find that the normalized separation between clusters fluctuates around the average value of 0.6 which indicates that the clusters may not be currently separated in an optimal manner. Ideally we would like this value to be closer to 1. We note that severe mis-classification of clusters would negatively influence the accuracy of our objective function $s(\theta)$ and could lead to performance reduced to that of a Random Restarts method. How to improve or optimize the effectiveness of the clustering algorithm is a topic of future work. Our aggregate results demonstrate however, that the kMeans-EM algorithm outperforms a Random Restarts method for our application, even if the clustering is non-optimal.

3.4.1 Clustering of Model Parameters and Finding New $\widetilde{\theta}_0$

Our empirical results, Figure 3-14, demonstrate that the Gaussian distributions formed over each cluster, with an empirical mean and covariance, are good approximations to the actual model parameters. These Gaussian distributions capture the locations and shapes of the clusters of converged model parameters; this is necessary for obtaining an accurate objective function $s(\theta)$ whose maximization corresponds to a new set of initialization model parameters $\widetilde{\theta}_0$. Figures 3-15,and 3-16 are example plots of the new initialization point, $\widetilde{\theta}_0$, resulting from the maximization of $s(\theta)$ for a typical iteration of the kMeans-EM algorithm. These plots show that $\widetilde{\theta}_0$, shown as a purple star, lies outside of the Gaussian distributions defined over identified cluster regions while staying in the active area of the search space for model parameters as desired. The "active area" of the search space in this context is used to refer to the area of the search space as defined by the Gaussian prior p_0 that contains higher likelihood model parameters and is not at the edges of the search space.

3.4.2 Aggregate Results Characterizing Performance of kMeans-EM Algorithm

Log-Likelihood Value: $g(\theta)$ We also demonstrate aggregate results that characterize the performance of the kMeans-EM algorithm compared against a Random Restarts method, averaged over several trials. The first performance criteria that we use is log-likelihood value, $g(\theta)$. Figure 3-18 shows the best log-likelihood value achieved versus iteration number for kMeans-EM and a Random Restarts method. This plot shows that for our application, the kMeans-EM algorithm consistently

outperforms the Random Restarts method and demonstrates improvement in the best log-likelihood value achieved with iteration number. We would hope to see this positive correlation in performance with iteration number of the kMeans-EM algorithm.

Euclidean Distance from True Model Parameters We are also interested in monitoring Euclidean distance of the best model parameters found and the true model parameters versus iteration number. Because EM is known to converge to the local maximum of $g(\theta)$ that is nearest to its point of initialization, we are interested in the Euclidean distance between the best model parameters found thus far, and the true model parameters. For continuous functions whose local maxima can be approximated by Gaussian distributions, as we believe $g(\theta)$ to be, we assume that Euclidean distance is a good measure of proximity between the current set of model parameters and its nearest hilltop. We also assume that the true set of model parameters will be the global optimum of $g(\theta)$. Figure 3-17 demonstrates the Euclidean distance between the true model parameters and the best set of learned model parameters versus iteration number. This plot shows a positive trend in a decrease of Euclidean distance with iteration number for the kMeans-EM algorithm. The Random Restarts method does not show the same positive trend and seems to fluctuate about a consistent value of Euclidean distance.

Normalized Separation Between Clusters Lastly we present a plot of the normalized separation between clusters that we call the silhouette plot. The silhouette value over clusters is an indication of the confidence level of the current clustering scheme. Two clusters that are not well separated will have a low silhouette value, indicating that there is likely to be a more appropriate clustering scheme with more or less clusters than currently present. In Figure 3-19 we present a plot of the silhouette values versus iteration number, averaged over twenty runs. The silhouette value fluctuates about the value 0.6. Ideally we would like the silhouette value to be as close to 1 as possible. This plot demonstrating an average silhouette value of 0.6 indicates that the clustering portion of the algorithm may not be as effective at creating accurate clusters of model parameters. How to improve the effectiveness of the clustering algorithm is a topic of future work. Our presented aggregate results demonstrate however, that the kMeans-EM algorithm outperforms a Random Restarts method for our application, even if the clustering is non-optimal.

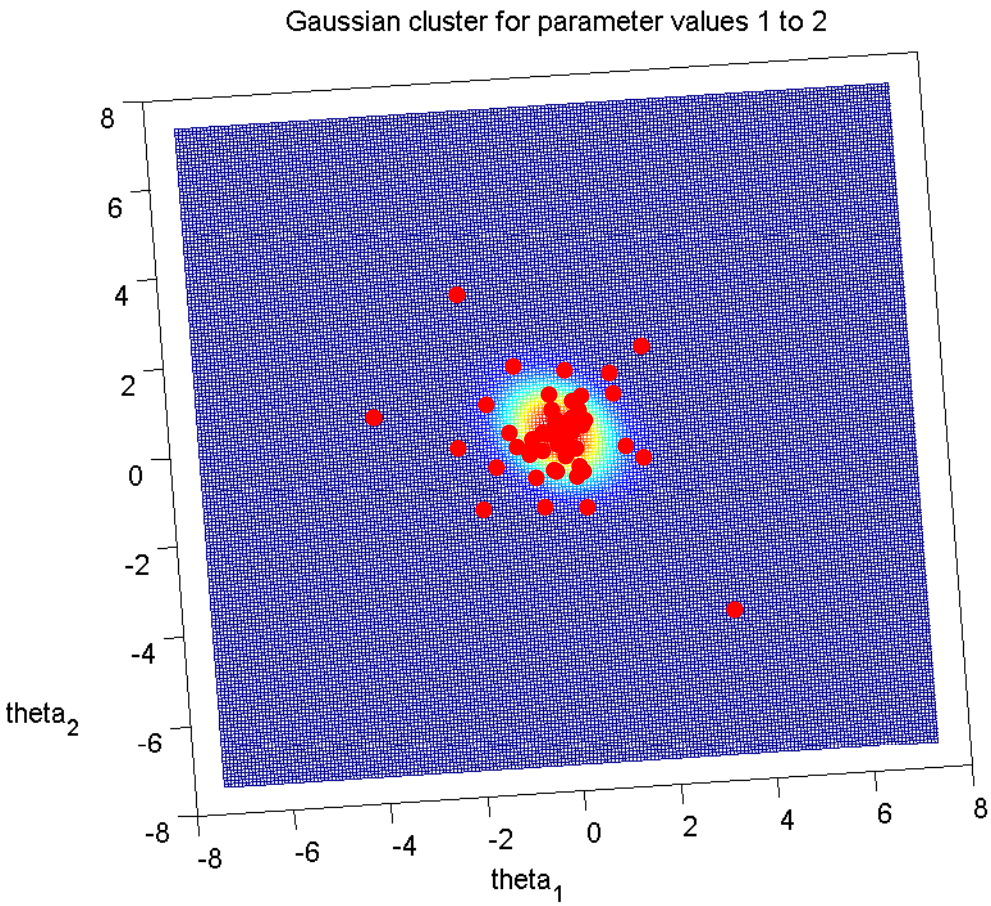

Figure 3-14: Plot of Gaussian cluster approximating shape and location of the first and second parameters for all initialization models $\{\theta_{0i}\}_{i=1}^{n}$.

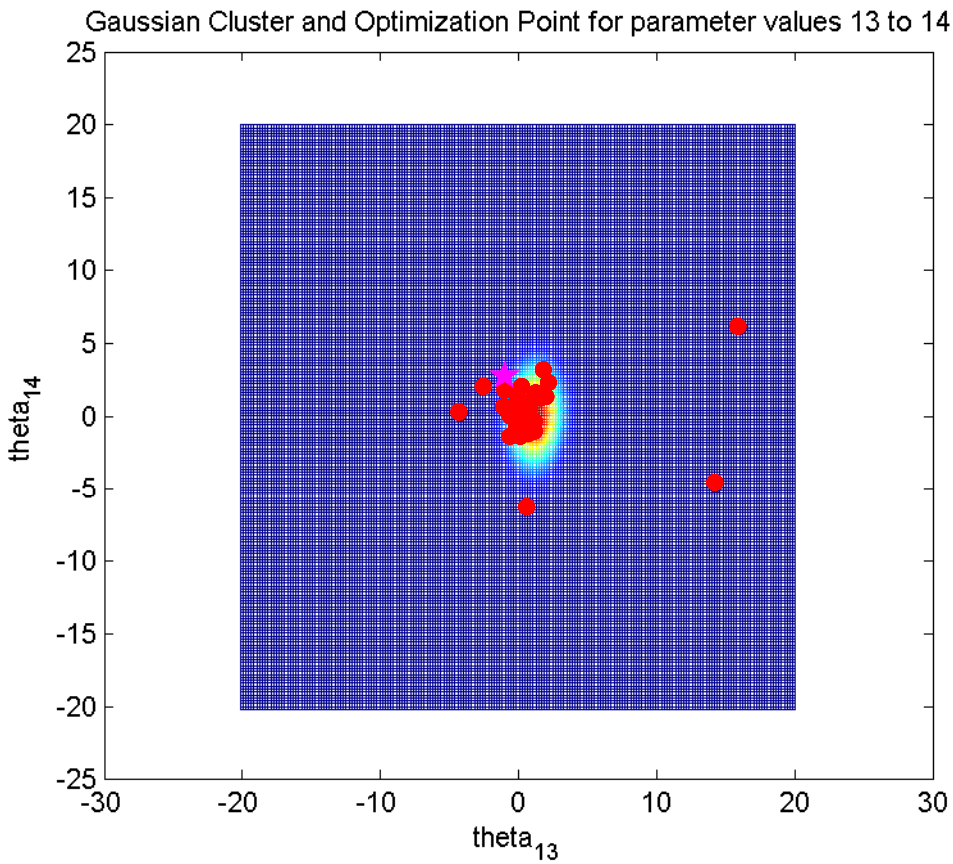

Figure 3-15: Cluster of the thirteenth and fourteenth parameters of all initialization models $\{\theta_{0i}\}_{i=1}^{n}$ with the optimal initialization point, found via a maximization of $s(\theta)$, shown as the purple star.

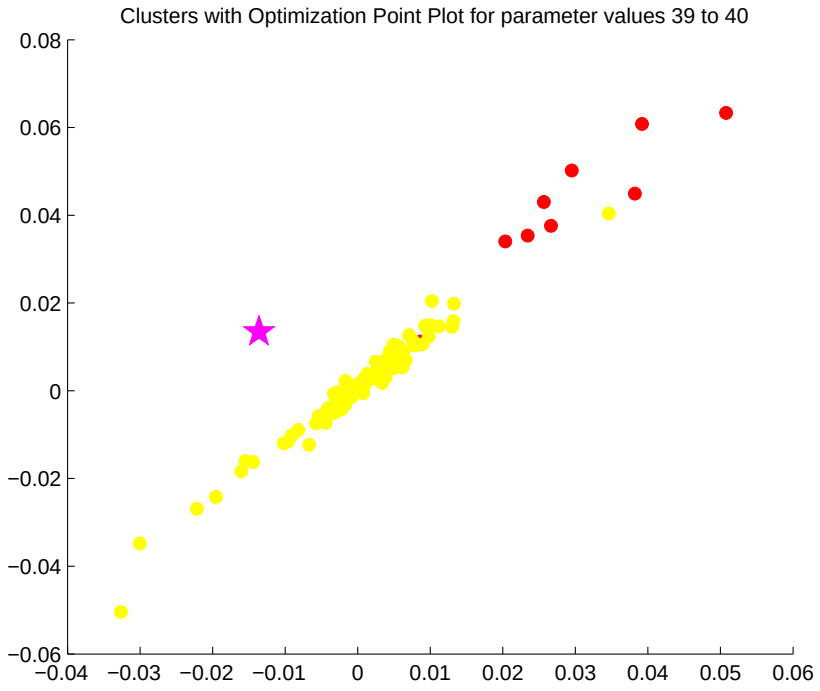

Figure 3-16: Plot of two clusters (yellow and red) with the optimal initialization point, found via a maximization of $s(\theta)$, shown as the purple star. Note that the new initialization lies outside of existing clusters, while remaining inside high probability areas of the search space.

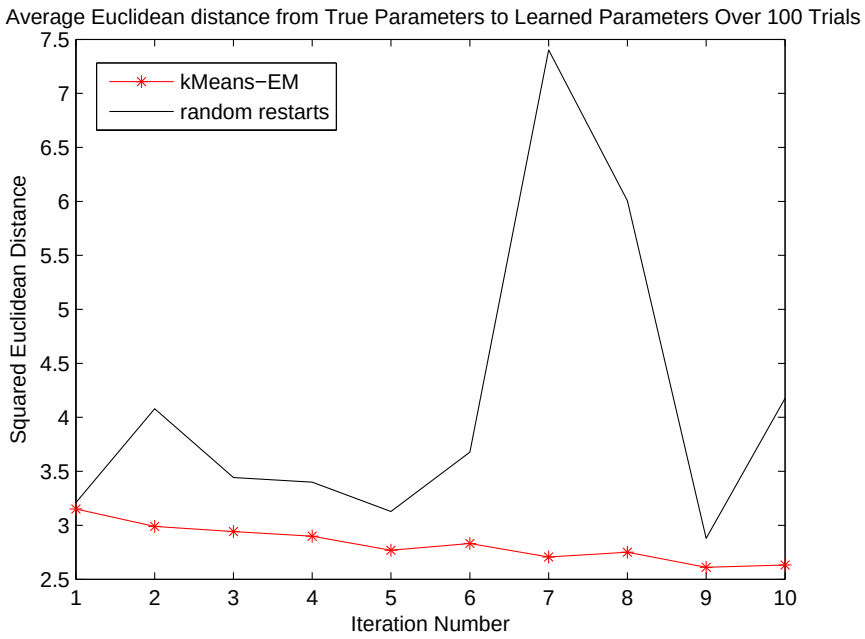

Figure 3-17: Euclidean distance between best learned model and true model versus iteration number, averaged over 100 trials.

3.4.3 Conclusion and Extension to the Switching Case

In this chapter we presented a kMeans-EM algorithm for avoiding being stuck in locally optimal solutions characteristic of the Expectation Maximization Algorithm. The kMeans-EM avoids convergence to a locally optimal solution by learning the approximate locations and shapes of the log-likelihood function, $g(\theta)$, and seeds the next iteration of EM with a set of model parameters outside of these regions of the search space. In this manner, the kMeans-EM algorithm explores a larger area of the feasible search space and seeks global, or near-optimal solutions. Our empirical results indicate that the kMeans-EM algorithm performs better than a Random Restarts method, and that performance of the algorithm improves with number of iterations. This positive trend indicates that the algorithm learns a better map of the search space with each iteration and performs a more thorough search over time.

Chapter Summary

We summarize the main results and conclusions of the chapter.

1. **Identifying Local Maxima through Gaussian Clusters:** We use a kMeans clustering

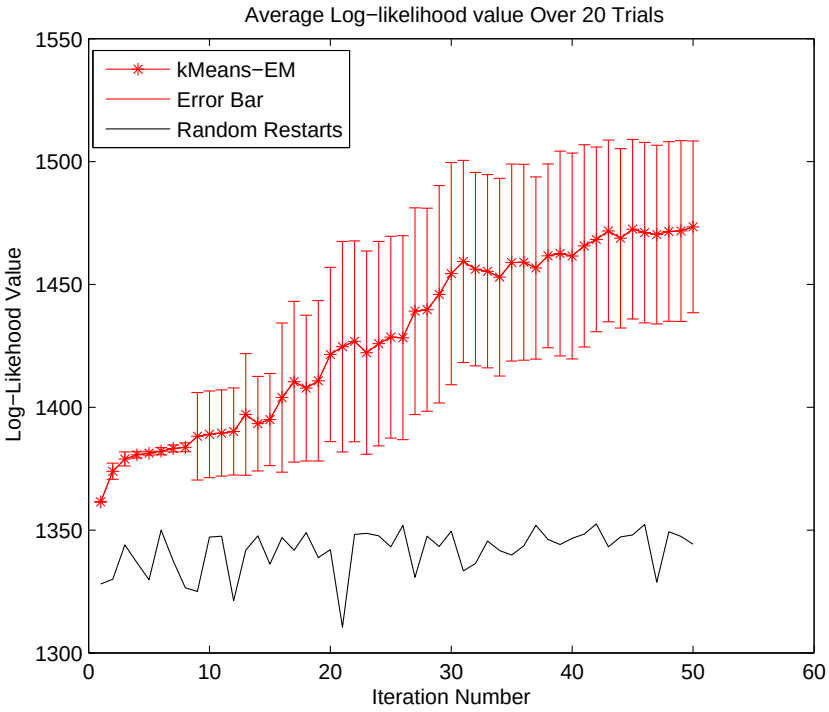

Figure 3-18: Plot of the average likelihood value of the best learned set of model parameters versus iteration number, for 50 clusters averaged over 20 trails.

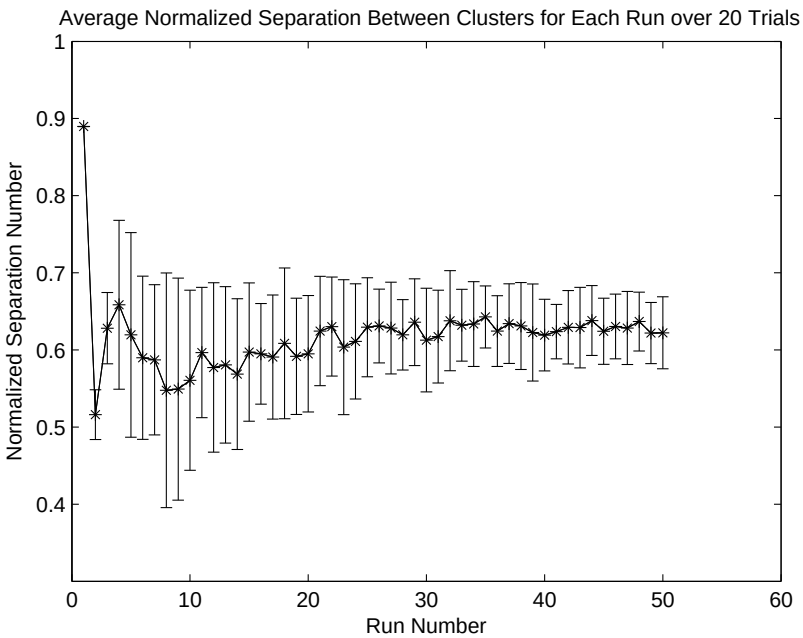

Figure 3-19: This plot shows the normalized separation between clusters, or mean silhouette value, averaged over 50 clustering iterations and 20 trials of the kMeans-EM algorithm.

algorithm to assign Gaussian clusters to the sets of model parameters $\{\theta_{f_i}\}_{i=1}^{n}$ learned by the EM-based model learning algorithm. These Gaussian clusters provide an approximate map for the location and shape of explored local maxima of the log-likelihood function $g(\theta)$.

2. **Finding $\widetilde{\theta}_0$ that Converge to New Local Maxima:** We derive an objective function $s(\theta)$ that takes into account the Gaussian cluster information, and a Gaussian prior over feasible model parameters. The maximization of this function yields a locally optimal guess for the next set of initialization model parameters,$\widetilde{\theta}_0$, for the EM portion of the algorithm. The resulting $\widetilde{\theta}_0$ has the property that it maximizes the probability of being within the feasible region of model parameters as designated by the Gaussian prior, while minimizing the probability of belonging to an existing Gaussian cluster.

3. **Non-Uniqueness of the Objective Function:** We derive alternatives to the objective function $s(\theta)$ that we call $s_R(\theta)$ and $s_Q(\theta)$. These alternative functions may be easier to optimize in some cases, and the quadratic function $s_Q(\theta)$ can be solved using a quadratic programming approach by linearizing the quadratic constraints.

4. **Influence of Gaussian Prior on Optimization:** We find that the kMeans-EM algorithm is greatly influenced by the Gaussian prior assigned over the model parameters. This is not desirable in the case where little or no information is available with regards to feasible, or likely, model parameters. We demonstrate however, that even with an ad hoc prior, constructed over labeled data using no *a priori* information, the kMeans-EM algorithm consistently outperforms a Random Restarts Method with respect to log-likelihood value, and Euclidean distance to the true model parameters. If more information is available *a priori*, this can be incorporated into the prior distribution to improve performance.

5. **Influence of Clustering Efficiency on kMeans-EM Algorithm:** The performance of the kMeans-EM algorithm is highly dependent on the effectiveness of the clustering algorithm and thus is subject to all of the limitations inherent to kMeans clustering. In the case that the clustering does not perform well, the algorithm performance reduces to that of a random restarts method. We demonstrate an average normalized separation between clusters of around 0.6. While we would prefer a value closer to 1, our positive empirical results of comparison to a Random Restarts method demonstrate that non-optimally clustered data still produces good results. We believe however, that improved clustering would further improve performance of the algorithm.

6. **Empirical Results:**

   a. We demonstrate empirically that with respect to improving the average log-likelihood value, and Euclidean squared distance between learned model parameters and true model parameters, the kMeans-EM algorithm performs better than a Random Restarts method. We also found that the performance of the kMeans-EM algorithm improves steadily with iteration number, indicating that the algorithm learns a better map of local maxima of the log-likelihood function and produces learned sets of model parameters that are closer in optimality to the global maximum than a Random Restarts method.

   b. We find that the kMeans-EM algorithm is sensitive to clustering accuracy, however, Figure 3-19 and Figure 3-18 demonstrates even with a sub-optimal silhouette value, or normalized separation between clusters, the kMeans-EM algorithm demonstrates superior performance to a Random Restarts method. We expect performance to increase further with more effective clustering techniques.

Extension to the Switching Case

Extension of this algorithm to the switching case where the model to be learned is hybrid as in the other two chapters of this thesis is straightforward. The hybrid case introduces new challenges because of the fact that evaluation of the log-likelihood value for a set of parameters is not possible. This algorithm however, is based solely on Euclidean distance between model parameters and thus does not rely on the evaluation of $g(\theta)$. The values of the log-likelihood for sets of model parameters were used in this chapter as a means of evaluating algorithm performance. Therefore, the theory presented in this chapter should carry over to the switching case. This however, is a topic of future research.

Chapter 4

Active Hybrid Model Learning

The methods presented throughout this thesis are focused on autonomously and accurately learning LPHA dynamical models. In Chapter 2 of the thesis we presented the HML-LPHA algorithm, a framework for hybrid model learning that assumes knowledge of only a sequence of control inputs and the resulting sequence of observations. In Chapter 3 we introduced the kMeans-EM algorithm that improves the quality of learned model parameters by avoiding being trapped in locally optimal solutions typical of EM-based algorithms. We haven't however, addressed the effect of sparsity of training data and very noisy observations, on the quality of the learned model. In these cases, the accuracy of the learned models can suffer substantially, and in the worse case, the learning algorithm can diverge altogether.

Our final technical chapter focuses on this problem by augmenting sparse or poor quality data with a modest amount of labeled data obtained via queries of the discrete state of the LPHA. Queries are strategically chosen at time instances where they can provide the most reduction in learning uncertainty; resulting in higher quality learned models. The ability of the learner to select training data that would most reduce a cost objective, in this case uncertainty of the current distribution over the hybrid state, is referred to as active learning [12]. We present an active learning approach to hybrid model learning for the model parameters of LPHA. We refer to this algorithm as Active Learning Hybrid Model Learning for LPHA (AHML-LPHA).

We show that the AHML-LPHA algorithm greatly improves learned model accuracy over the HML-LPHA algorithm, even when the number of requests for additional information are as few as 6% of the total number of timesteps, and when the answer to the query itself is noisy. In fact, we

show that in some cases we are able to converge to an accurate model of the hybrid system when the HML-LPHA algorithm diverges.

First we provide an introduction to the chapter where we define our problem statement. Second we define the concept of active learning, and provide a review of active learning for discrete systems. Finally, we present the extension of active learning for discrete systems to active learning for LPHA where we allow for a limited number of noisy queries of the discrete state, chosen at time points with high Value of Information (VOI). We also discuss simulation results.

4.1 Related Work

For many learning problems there is a natural source of information that, if used efficiently, can greatly improve learning performance. This is true for many machine vision tasks, such as Human activity recognition [32, 10, 30], and clustering during text classification tasks [27]. It is also true for modeling in the field of econometrics [21], and for supervised tasks, such as testing planetary rovers in a sandbed. These tasks share the common goal of accurately identifying a model in the form of a Switching Linear Dynamical System or LPHA, from which to perform inference about the underlying system.

Active learning for state estimation of hybrid systems has been investigated in the form of selection of a particular control sequence that used for disambiguating the current state [7, 28]. This is often referred to as Auxilliary Signal Design, of Detection Signal design and has applications in failure detection or model selection. Extending this to model learning for the SLDS or LPHA case in an interesting area and a topic of future research. In this chapter we focus on active hybrid model learning via querying of the discrete state.

When the learning task is made more challenging due to sparsity of training data, or large amounts of observation noise, the quality of the learned model may be severely compromised. In these cases, if additional information is available, for example the answer to a query, the learning task becomes substantially easier and the resulting learned model can be far more accurate. Furthermore, if this additional information is provided at critical time points, for example, when the algorithm is most *uncertain*, the learning algorithm has a much higher potential for better performance.

For some learning problems, the concept of answering a query is a natural one. Take the example

of human activity recognition and tracking. Much attention has been dedicated to this field where data is collected via video sequences and the task is to infer the state of the person or object filmed [32, 10]. When learning activity models, if information is requested about a certain time instant, say "was the person running or walking at time t=100," a human supervisor can inspect the video and provide an answer. These types of queries, where input is requested from a human, are also common in facilitating other learning tasks, such as in text classification. Other types of queries may not involve humans and instead may require the use of expensive to operate sensors; an example being requesting GPS coordinates for a localization task [20]. Due to the additional expense of querying, we would like to limit the amount of additional information requested and only request that information that would be most beneficial. [1] developed an Active Learning approach for learning of discrete systems modeled as HMMs, that utilizes the concept of "queries" and "Value of Information." In this chapter we extend to hybrid discrete/continuous systems, namely, SLDS.

4.1.1 Problem Statement

To recapitulate, our goal is to improve the accuracy of parameter model learning for hybrid discrete-continuous linear dynamical models via active learning. Active Learning in this context consists of the ability to query the system, ie. add labeled data, to enhance learning capability. There is a significant cost to answering queries, hence, the challenge is to identify time instances in which querying is most valuable. The Value of Information (VOI) of a query q can be defined in terms of a cost that we are attempting to minimize, for example, uncertainty in the state estimate. The VOI is the resulting minimization of this cost given the answer to the query q. We wish to select to query at time instances where the VOI is maximized.

Problem Statement:

Given an observation sequence $\mathbf{y_1^{T+1}}$ and a control input sequence $\mathbf{u_0^T}$

1) find the time points t^* where the information gain from querying is highest

$$t^* = \arg\max_t \text{VOI}$$

and

2) find the Maximum Likelihood set of model parameters given the optimal queries $Q(t^*)$. We formally define Value of Information (VOI) and queries for hybrid systems in section 4.4.

4.2 Active Learning

The field of active learning is defined by Cohn as the study of the closed-loop phenomenon of a learner selecting actions or making queries that influence what data are added to its training set [12]. Active learning is an integral component for many natural learning tasks, as well as medical and hardware diagnosis tasks via active probing [33].

A contrast can be made to distinguish active learning from passive learning. *Passive* learning is an attempt at the learning task by processing and using only available data. The learning algorithm presented in chapter 2 is an example of passive learning where the hybrid model is learned using only the provided data, observations $\mathbf{y}$ and control inputs $\mathbf{u}$. The majority of learning tasks in the fields of machine learning and autonomy are passive. Some examples of using active learning to enhance a learning task are selecting torques or joint angles to learn the kinematics of dynamics of robotic arm, querying an oracle for robot localization [20], or selecting a control input sequence for hybrid state estimation [7].

4.3 Review of Active Learning for Discrete Systems

Anderson and Moore [1] presented an active learning approach for Hidden Markov Models using the concept of queries and Value of Information (VOI). Discrete queries are defined as follows:

Discrete Query Q_{d_t}: A discrete query is an *unavailable* observation of the current discrete, or otherwise hidden, state whose value can be requested at specific time steps. The revelation of a query value can incur cost and/or be noisy. A noisy query refers to an observation that has imperfect information about the state at the time a query is made. In the case of a perfect query, $b_i(i) = 1$. The probability of a query taking the value q at time t, given the discrete state x_d at

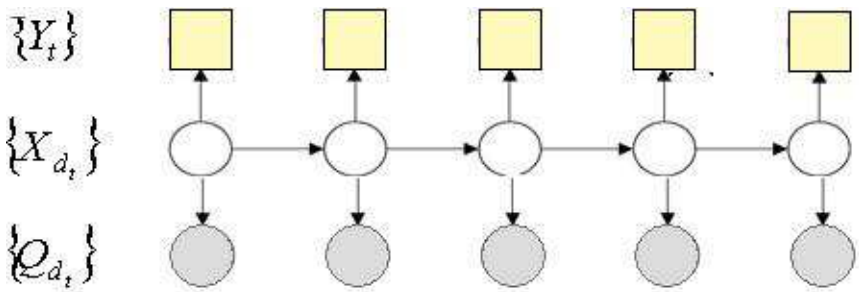

Figure 4-1: Schematic diagram showing the relationship between a sequence of queries $\{Q_{dt}\}$, hidden discrete state $\{X_{dt}\}$, and observations $\{Y_t\}$.

time t

$$p(Q_{d_t} = q | X_{d_t} = i, y_1^T) = b_i(q) \tag{4.1}$$

In the discrete setting, a query at time t provides information about the discrete mode at the specified time instant. Figure 4-1 shows the relationship between the query variables for each discrete state, the observations for each discrete state, and the discrete states themselves. From this diagram one can see that the query at time t, Q_{d_t}, is independent of all other queries and states given the state at time t. In the next section we demonstrate how this idea of a query for purely discrete Markov Models is extended to hybrid Markov Models.

The main purpose of active learning for discrete systems is to identify those queries for which the greatest benefit can be obtained for the learning task, and then use the results of these queries to improve the performance of the learner. We define a concept of "usefulness", or VOI of a query at time t. There are many different ways to define the VOI of a query, which is more appropriate depends on the problem. In cases where one is particularly concerned with error reduction, one may choose a VOI that selects queries that are expected to reduce future classification error the most [34]. Alternatively, one may choose to select data that minimizes learner variance [12], or maximizes Kullback-Leibler Divergence between two competing hypothesis densities [38]. For our goal of hybrid model learning, we choose an entropy based VOI definition that chooses queries at time instances where the uncertainty in the current hybrid state estimate is highest. An interesting future topic of research would be to contrast and compare different VOI definitions for the hybrid model learning case. Before we can formally define VOI, we must first introduce the concept of a loss function.

Loss Function $L(p)$: The function $L(p)$ is a general function that is defined to capture the element of the problem that we wish to minimize. For example, if one wishes to reduce uncertainty, then $L(p)$ may be defined as the entropy over the discrete state distribution.

Now that we have introduced loss functions, we can formally define VOI.

Value of Information VOI: The value of information of a query is the expected gain, as measured by the reduction in the loss function, that is obtained by performing the query. It is the loss function minus the expected value of the loss function given the result of the query.

$$\text{VOI}(Q;p) = L(p) - E_Q[L(p|Q=q)] \tag{4.2}$$

For entropy based VOI, the loss function takes the form of the entropy of the distribution over the hidden state X_d. Specifically,

$$\begin{aligned}\text{VOI}(Q;p) =& \mathcal{H}(p(X_d)) - E_Q\left[\mathcal{H}(p(X_d|Q=q))\right] \\ =& \mathcal{H}(X_d) - \mathcal{H}(X|Q)\end{aligned} \tag{4.3}$$

We motivate active learning for discrete systems with a simple Hidden Markov Model example. One of the important learning tasks associated with HMMs is determining the state of the HMM at each timestep. The goal in this problem is to correctly determine as many probabilistic states in a given state sequence as possible given an observation sequence. Consider an HMM with two discrete states A and B where state A activates a green light, and state B has a non-zero probability of activating a green or a red light. Given an observation sequence of red and green lights, one can say with certainty that a red light indicates being in state B, however, observation of a green light does not offer such a clear conclusion. Figure 4-2 shows the setup of our HMM problem.

In terms of entropy, given a red light, the state is uniquely determined to be B and there is zero entropy associated with the hidden state. However, given a green light observation, we are no longer sure of which state we are in and thus there is a higher entropy associated with the

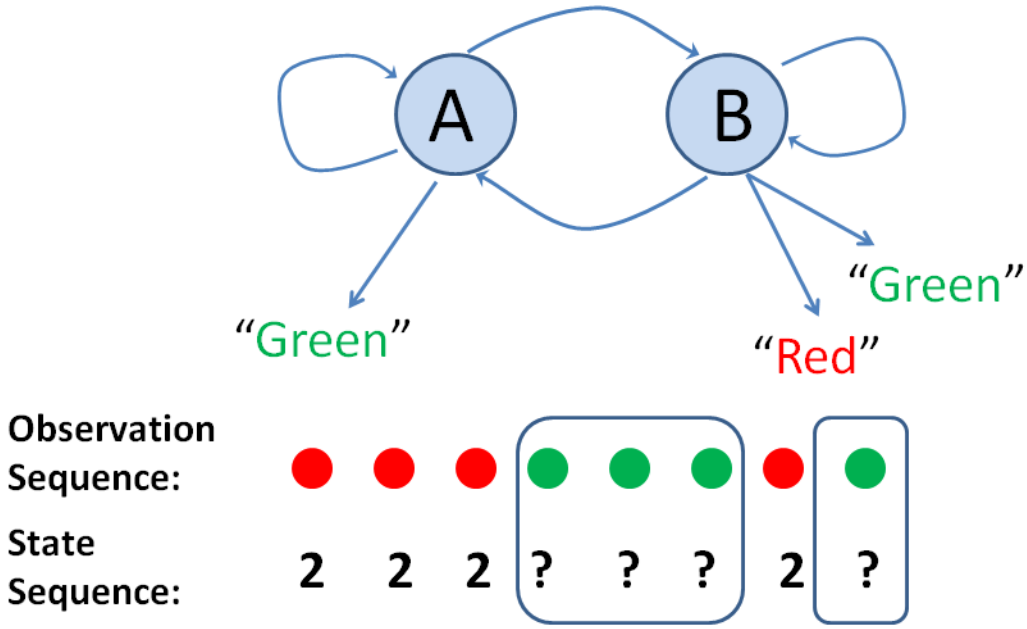

Figure 4-2: HMM with two discrete states and "Red" or "Green" observations. The problem is to infer the hidden state given a sequence of observations.

hidden state. In the most unfortunate case, the posterior probability distribution over the discrete states given the green light observation can be uniform, indicating maximum uncertainty and thus a maximum entropy value of 1. Thus, querying the discrete state at this point would give us extra information about the distribution over the two states; this would be indicated by a high VOI. In contrast, it is also easy to see that querying the discrete state given a red light observation would be a waste of a query and in fact would result in a VOI value of zero.

This simple problem motivates the need to query *intelligently*, and gives us the intuition for why querying only a small fraction of the total number of timesteps can provide enhanced an learning ability.

4.4 Active Model Learning for Switching Linear Dynamical Systems

In the previous section we introduced the key concepts of active learning using queries and VOI, by reviewing active learning for purely discrete systems. We now use these tools to develop an active learning approach for hybrid model learning. The reader interested in a more comprehensive

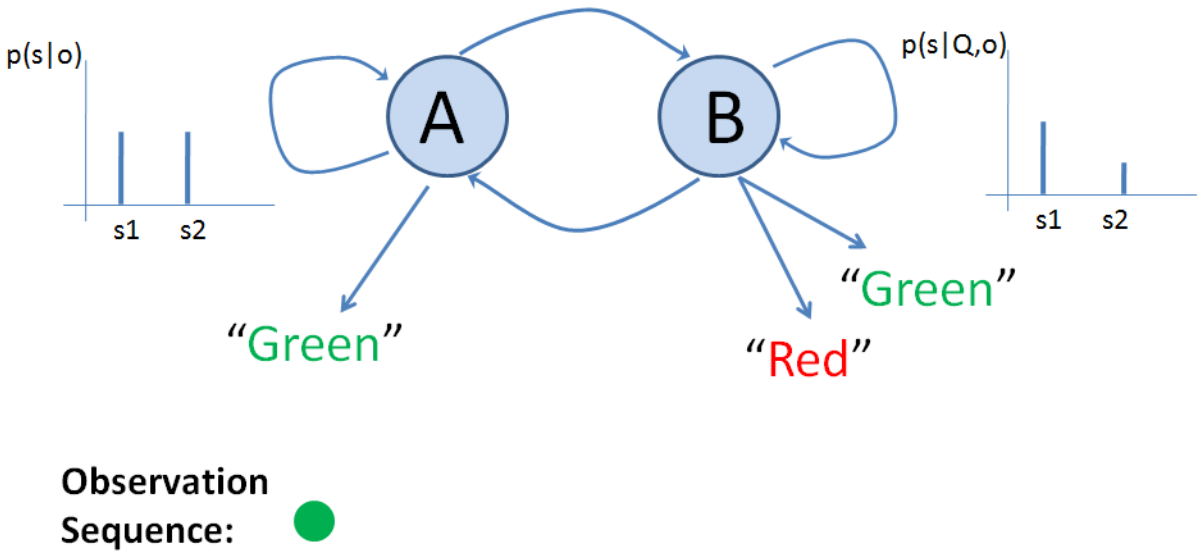

Figure 4-3: Schematic diagram of a discrete HMM where the VOI of a query would be high due to maximum uncertainty in the distribution over the discrete state.

presentation of active learning for HMMs or a broader discussion of different types of loss functions is referred to [1].

In a straightforward generalization of the above approach to hybrid HMMs, a query should provide information on the complete hybrid state, *both* the continuous and discrete components. This would be quite difficult in general. To illustrate, consider the example of tracking a person walking vs. jogging, where the task is to identify the dynamics of certain points on the person's body and to identify transition probabilities for the two discrete modes; *walking*, or *jogging*. One could visually identify the discrete hidden state at a query point by observing the video sequence, but if asked to additionally provide values for the noisy continuous state, say position and velocity of each of the tracked points, this task would become much more difficult. For complex systems with large numbers of continuous hidden states, the task of answering a query can become significantly more challenging. Thus, for any system whose discrete hidden state may be more easily identified than the continuous hidden state, active model learning for hybrid systems would be much more cost effective if a query of the complete state was not necessary. This observation motivates our key intuition for extending active model learning to hybrid systems.

4.4.1 Key Intuition for Active Hybrid Model Learning

Key Intuition: We can perform hybrid active learning effectively while only querying the discrete component of the state.

The reason that querying only the discrete state is sufficient is that given the discrete state, the Kalman Filter utilized in the E-step of the HML-LPHA algorithm, will provide an exact estimate of the continuous state distribution for the given time step. Referring back to the Expectation Maximization Algorithm in Chapter 2, we see that the distribution over the state trajectory $p(\mathbf{x_d}_0^T, \mathbf{x}_0^{T+1}|\mathbf{y}_1^{T+1}, \theta)$ is calculated by the E-step. This is the distribution over which the expectation is taken to find the lower bound to the log-likelihood function. We repeat the definition of the lower bound to the log-likelihood function here for convenience.

$$h(\theta|\theta^k) = E\left[\log(p(\mathbf{x_d}_0^T, \mathbf{x}_0^{T+1}|\mathbf{y}_1^{T+1}, \theta))\right] + \mathcal{H} \tag{4.4}$$

4.4.2 Queries, VOI, and Belief-State Update for Active Hybrid Model Learning

Next we develop an algorithm for performing active hybrid model learning for LPHA, we define queries, VOI, and belief-state update for this hybrid case.

Queries

Hybrid Query: A query for a SLDS on the discrete component of the hybrid state. The probability of a query at time t given the state at time t is

$$p(Q_t = q|x_{d_t} = i, x_{c_{t+1}}, \mathbf{y}_1^{T+1}) = b_i(q) \tag{4.5}$$

where $i, q \in \{1, ..., m\}$ with m being the total number of discrete states

Notice that q is conditioned on the hybrid state X. Also notice that we only require knowledge of the discrete component, $x_{d_t} = i$, while the distribution for $x_{c_{t+1}}$ is found via the Kalman Filter. For the perfect query case, $b_i(q) = 1$.

Value of Information VOI for the Hybrid Case: Recall we use an entropy based VOI

definition. For the case where the cost function is the entropy, the VOI at time t takes the form of the mutual information between the query at time t and the hybrid state trajectory.

$$\begin{aligned}\text{VOI}(Q_t) =& \mathcal{H}(\pi) - \mathcal{H}(\pi|Q_t) \\ =& \mathcal{H}(Q_t) - \mathcal{H}(Q_t|\pi) \text{ by symmetry of mutual information} \\ \pi =& \left[x_{d_t}, x_{c_{t+1}}\right]_{t=0}^{t=T} \text{ is a state trajectory}\end{aligned} \tag{4.6}$$

Note that to compute VOI(Q_t), we must compute two key values, $\mathcal{H}(Q_t)$, and $\mathcal{H}(Q_t|\pi)$.

$$\mathcal{H}(Q_t) = -\sum_q p(Q_t = q|y_1^{T+1}, \theta) \log(p(Q_t = q|y_1^{T+1}, \theta)) \tag{4.7}$$

$$p(Q_t = q|y_1^T, \theta) = \sum_{x_{c_0}^{T+1}, x_{d_0}^T} p(Q_t = q|x_{c_0}^{T+1}, x_{d_0}^T, y_1^{T+1}, \theta) p(x_{c_0}^{T+1}, x_{d_0}^T|y_1^{T+1}, \theta) \tag{4.8}$$

$$= \sum_i \sum_{x_{c_0}^{T+1}, x_{d_0}^T : x_{dt}=i} p(Q_t = q|x_{c_0}^{T+1}, x_{d_0}^{t-1}, x_{dt} = i, x_{d_{t+1}}^T, y_1^{T+1}, \theta) \times \tag{4.9}$$

$$\times\, p(x_{c_0}^{T+1}, x_{d_0}^{t-1}, x_{dt} = i, x_{d_{t+1}}^T|y_1^{T+1}, \theta) \tag{4.10}$$

The query Q_t is independent of all other variables given x_{dt} so $p(Q_t|y_1^{T+1}, \theta)$ becomes

$$p(Q_t|y_1^{T+1}, \theta) = \sum_i \sum_{x_{c_0}^{T+1}, x_{d_0}^T : x_{dt}=i} p(Q_t|x_{dt} = i) p(x_{c_0}^{T+1}, x_{d_0}^T|y_1^{T+1}, \theta) \tag{4.11}$$

We compute the distribution over hybrid state trajectories $p(x_{c_0}^{T+1}, x_{d_0}^T|y_1^{T+1}, \theta)$.

$$p(x_{c_0}^{T+1}, x_{d_0}^T|y_1^{T+1}, \theta) = p(x_{d_0}^T|y_1^{T+1}, \theta) p(x_{c_0}^{T+1}|x_{d_0}^T, y_1^{T+1}, \theta) \tag{4.12}$$

$$p(x_{d_0}^T|y_1^{T+1}, \theta) = \frac{p(y_1^{T+1}|x_{d_0}^T, \theta) p(x_{d_0}^T|\theta)}{\sum_{x_{d_0}^T} p(y_1^{T+1}|x_{d_0}^T, \theta) p(x_{d_0}^T|\theta)} \tag{4.13}$$

Where the distribution over the continuous state trajectory, $p(x_{c_0}^{T+1}|x_{d_0}^T, y_1^{T+1}, \theta)$, is computed by the Kalman Smoother in the E-step of the EM algorithm.

The second key term in our VOI(Q_t) calculation is the entropy of a query Q_t, given the trajectory over the hybrid state, $\mathcal{H}(Q_t|\pi)$. Recalling that $\pi = [x_{c_0}^{T+1}, x_{d_0}^T]$:

$$\mathcal{H}(Q_t|x_{c0}^{T+1}, x_{d0}^{T}) = -\sum_{x_{c0}^{T+1}, x_{d0}^{T}} p(x_{c0}^{T+1}, x_{d0}^{T}|y_1^{T+1}, \theta) \sum_q p(q|x_{c0}^{T+1}, x_{d0}^{T}) \log p(q|x_{c0}^{T+1}, x_{d0}^{T}) \tag{4.14}$$

$$= -\sum_i \sum_{x_{c0}^{T+1}, x_{d0}^{T}: x_{dt}=i} p(x_{c0}^{T+1}, x_{d0}^{T}|y_1^{T+1}, \theta) \times$$

$$\times \sum_q p(Q_t = q|x_{dt} = i) \log(p(Q_t = q|x_{dt} = i)) \tag{4.15}$$

We've now presented how to compute the VOI for the case of Switching Linear Dynamical Systems. Evaluating the VOI for possible queries at each timestep identifies optimal queries. However, we have not yet discussed how to incorporate the result of making a query on the distribution over the hybrid state trajectories. This is the topic of the next subsection.

Updating the Distribution Over Hybrid State Trajectories

In this subsection, we present the update to the distribution over hybrid state trajectories given the result of a query $Q_t = q$. We compute this update using Bayes Rule:

$$p(x_{c0}^{T+1} x_{d0}^{T}|Q_t = q, y_1^{T+1}, \theta) = \frac{p(Q_t = q, y_1^{T+1}|x_{c0}^{T+1} x_{d0}^{T}, \theta) p(x_{c0}^{T+1} x_{d0}^{T}|y_1^{T+1}, \theta)}{\sum_{x_{c0}^{T+1} x_{d0}^{T}} p(Q_t = q, y_1^{T+1}|x_{c0}^{T+1} x_{d0}^{T}, \theta) p(x_{c0}^{T+1} x_{d0}^{T}|y_1^{T+1}, \theta)} \tag{4.16}$$

Where $p(x_{c0}^{T+1} x_{d0}^{T}|y_1^{T+1}, \theta)$ is the original distribution over our hybrid trajectories that was presented in Equation (4.13)

Putting these pieces together, our algorithm for Active Hybrid Model Learning for SLDS, or LPHA is:

4.4.3 Simulation Results and Discussion

In this section we discuss the results of implementing our Active Hybrid Model Learning algorithm in simulation [1]. We use the example of the rover from Chapter 2 where this time the rover is driven through different types of terrain and thus can experience wheel slippage. Therefore, the two discrete states of our system are *slipping* and *not slipping*. Our continuous state is composed of i, the current supplied to the wheels of the rover, and $\dot{\theta}$, the angular velocity of the wheels. In the

[1] The empirical results of this chapter were obtained with the help of John Nham.

1.Initialization
$k = 1$
$\theta_k = \theta_0$
2.WHILE:Not converged
3.$p(x_{d0}^{T}, x_{c0}^{T+1} | y_1^{T+1}, \theta_k) \leftarrow$ Compute E-Step
4. FOR($t = 1 : T$)
5.VOI(t) $= \mathcal{H}(Q_t) - \mathcal{H}(Q_t | X_t)$
ENDFOR
6.$\theta_k \leftarrow$ Compute M-Step
ENDWHILE
7.$Q^* \leftarrow$ VOI(Q_t) $\geq$ *threshold* % Keep track of maximum VOI points
8.$FOR(t = 1 : T)$
9.$FOR(q = Q^*)$
10.doBeliefUpdateForward % Update hybrid belief for all max VOI points
11.$FOR(s = t^* : -1 : 0)$
12.doBeliefUpdateBackward % Smooth hybrid belief for max VOI points
$ENDFOR$
$ENDFOR$
$ENDFOR$
13.Return θ^*, the learned hybrid model

Figure 4-4: Active Hybrid Model Learning Algorithm

mode *not slipping*, the current supplied increases the angular velocity of the rover, whereas in the *slipping* mode, the angular velocity will actually be greater for the same amount of current, due to slippage. We can also incorporate autonomous mode transitions to help us model the problem. If we place a guard condition on the angular velocity, $\dot{\theta}$, we can specify that the transition probabilities between the modes are such that mode *slipping* is more likely if $\dot{\theta}$ is $\geq$ *tolerance*. Where the tolerance value can be arbitrarily chosen.

As a means of comparison, we ran the simulations under four conditions, using (1) No Queries, (2) All Queries, (3) Random Queries, and (4) Queries with highest VOI value. We use the "All Queries" case as a benchmark for comparison because in this case the algorithm is given information about the discrete state at every timestep. The "No Queries" case corresponds to the passive learning algorithm presented in Chapter 2. For the case where we use highest VOI points we

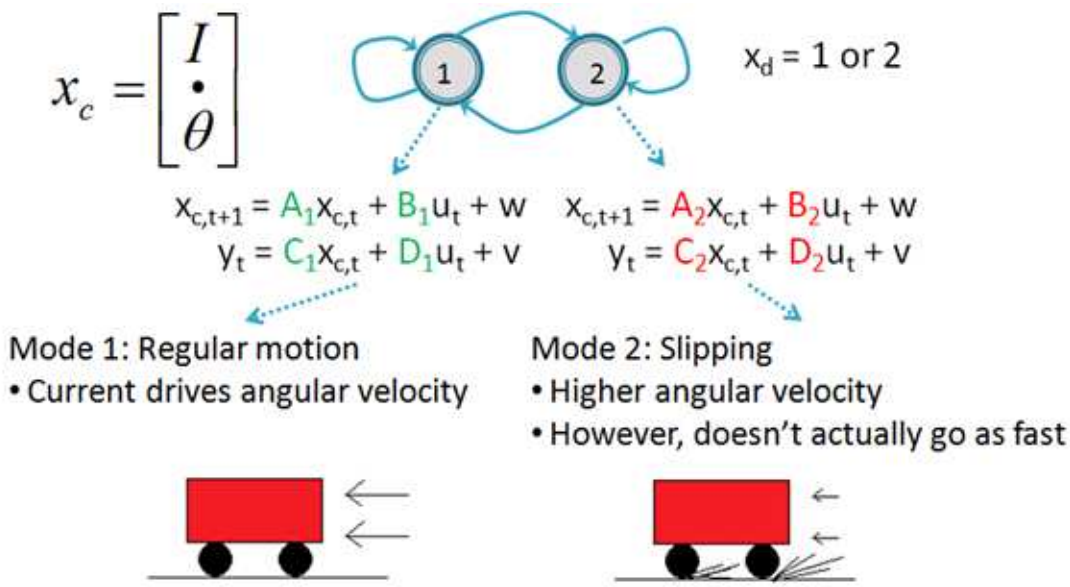

Figure 4-5: Schematic diagram of AHML-LPHA simulation setup with two discrete modes, *slipping* and *not slipping*, and continuous being current I, and angular velocity $\dot{\theta}$ of the wheel.

choose to query at only those points whose VOI value was above the threshold of 0.5 where this threshold was chosen arbitrarily. The "Random Queries" case queried the same amount of times as the VOI case, but did not use information from the VOI to choose queries intelligently.

Figure 4-6 shows a plot of the VOI values at each timestep for a typical run. This figure shows the sparsity of the VOI graph where only about 2% of the total number of timesteps had a VOI value above the threshold of 0.5. For a case where the VOI graph is so sparse, there seems to be a clear advantage to choosing queries intelligently versus choosing queries randomly as in the "Random Queries" case. This graph also motivates the conclusion that Active Hybrid Model Learning can be performed with only a small amount of queries because the queries with VOI$<$ 0.5 offer little value to the learning task.

Figure 4-7 shows the MAP mode estimation error of each run of the active learning algorithm averaged over 100 trials. The MAP mode estimation error reflects the number of wrong bit transitions in the Maximum A Posterior discrete mode trajectory. The "All Queries" case has zero MAP mode estimation error for all iterations of the algorithm because it was given perfect information of the discrete state at each timestep. This plot demonstrates an improvement in MAP mode error of the VOI case over the Random Queries case and a more significant improvement over the passive learning case shown in red. For the data gathered in these plots, only 5 out of 150 timesteps were

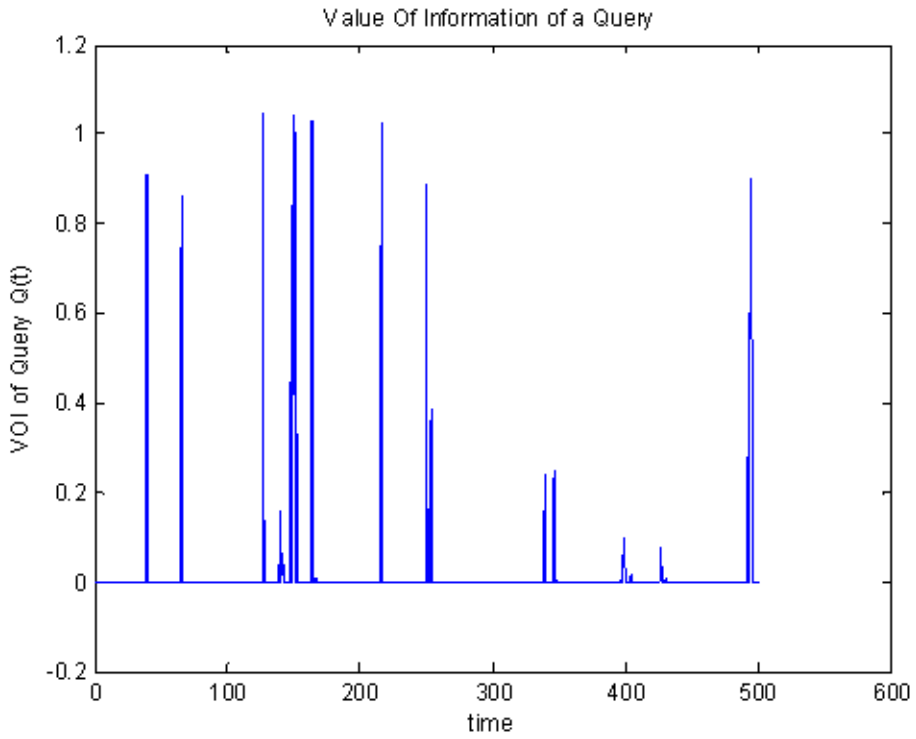

Figure 4-6: Plot of the VOI value for each timestep for a typical run.

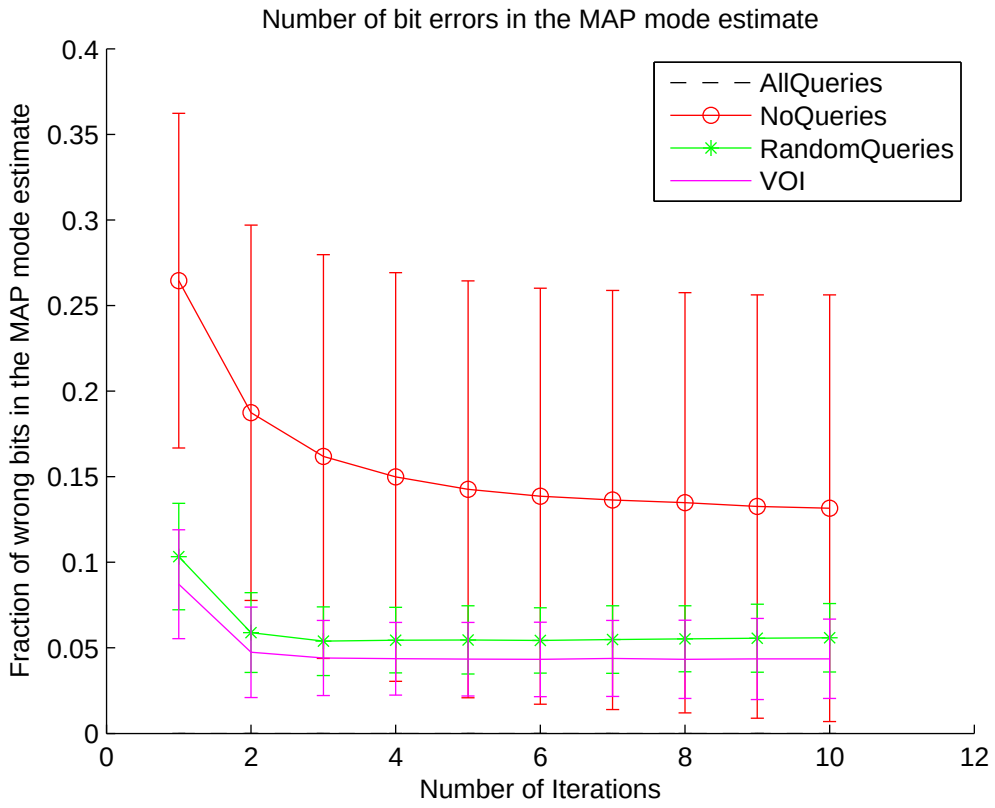

Figure 4-7: Comparison plot of the MAP mode estimation error for all four types of runs.

queried. The VOI plot shows an average of 10% improvement over the passive learner for only 3% of the total number of timesteps used for queries. Both the "Random Queries" and the "VOI" case use the same number of queries during each run, the main difference is that the "VOI" case uses the VOI information to query only at points of highest uncertainty.

Figures 4-8 and 4-9 plot the squared error in the continous state estimate vs. iterations of the AHML-LPHA algorithm, averaged over 100 trials of the AHML-LPHA algorithm. These plots show an improvement of the "VOI" case over the "Random Queries" case, and a more significant improvement over the "No Queries" case. In the cases where querying is allowed, the average squared error in the continuous state estimate reduces dramatically.

The comparison plot for the transition probability convergence 4-10 shows that the "All Queries" case performs best as expected from a baseline measure. We see that the "VOI" case performs slightly better than the "No Queries" and "Random Queries" cases that have comparable performance to each other.

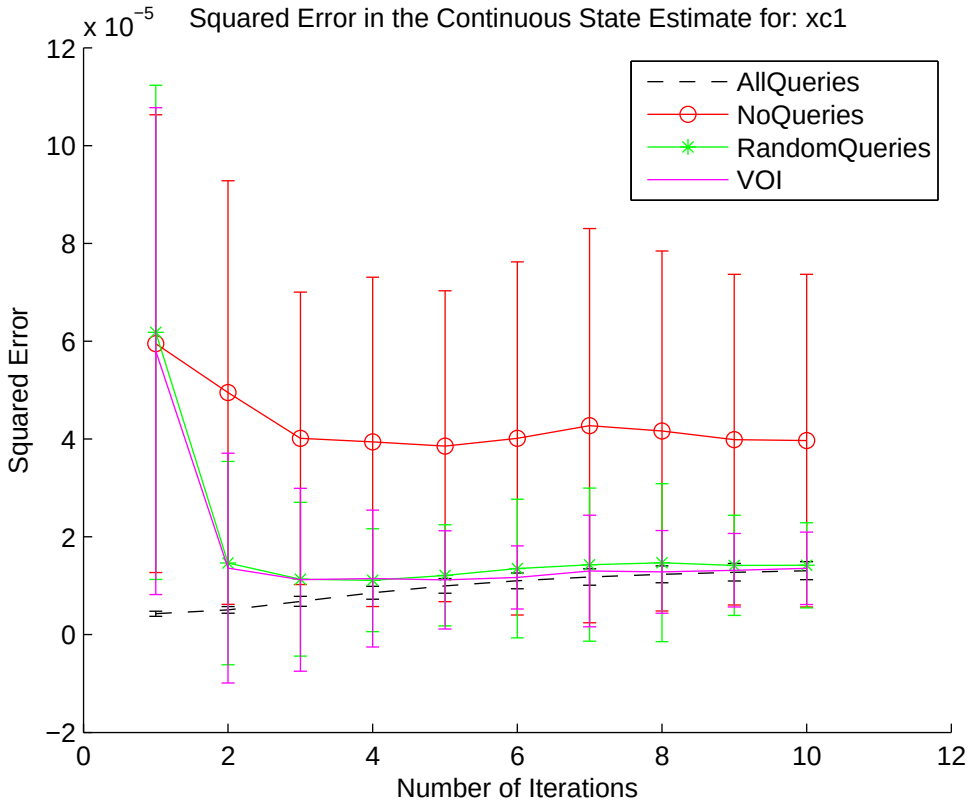

Figure 4-8: Comparison plot of the squared error for the continuous state estimate of $\dot{\theta}$ for all four types of runs, averaged over 100 trials of the AHML-LPHA algorithm.

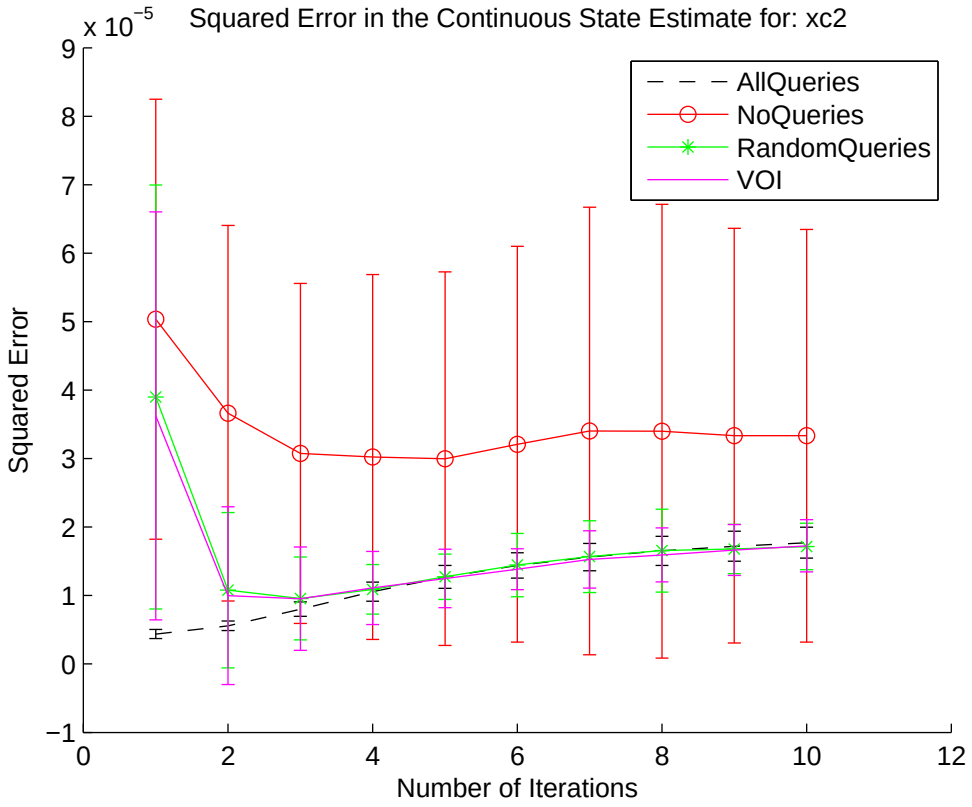

Figure 4-9: Comparison plot of the squared error for the continuous state estimate of I for all four types of runs.

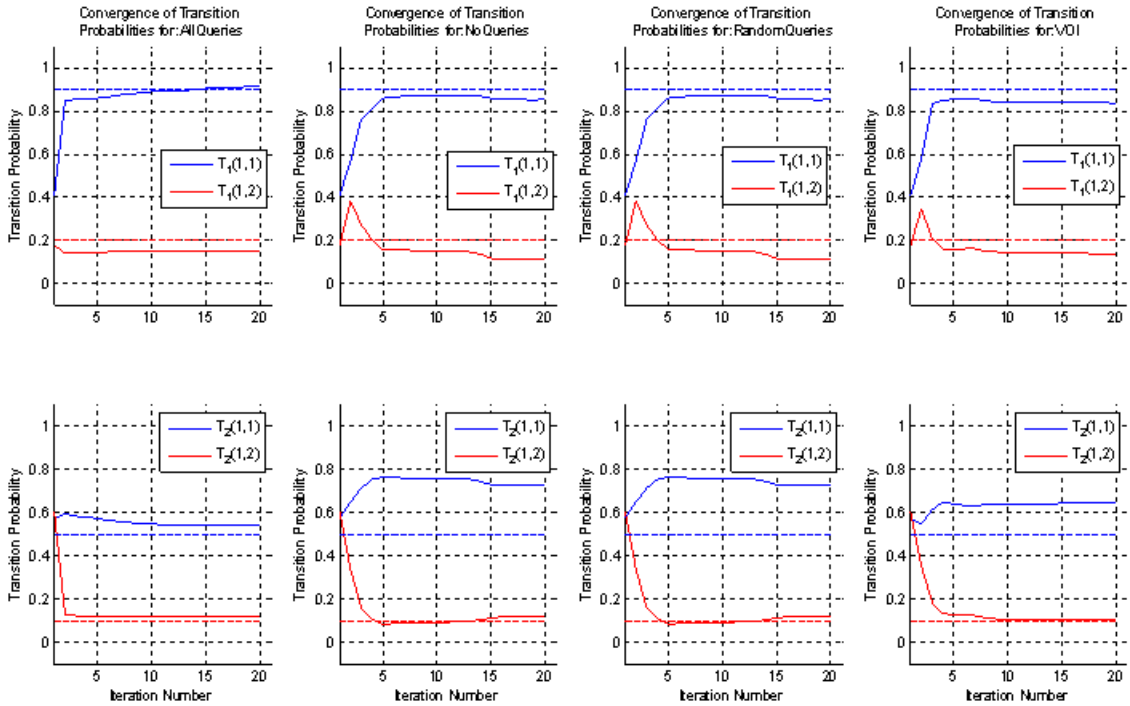

Figure 4-10: Comparison plot of the transition probability convergence for the two guard conditions. The guard was set at $\dot{\theta} > threshold$

Active Hybrid Model Learning Convergence when EM Diverges

Figure 4-11 shows a case where the initial conditions were poor enough to cause a divergence of the EM algorithm. We also show that the use of active learning allows for convergence of the algorithm even in the case of poor initialization so that the HML-LPHA algorithm, or "No Queries" case, diverges. These diverged EM results were obtained by perturbing the continuous parameters by 100% of their true values to seed EM, and the initial discrete parameters, or transition probabilities, were chosen uniformly in $[0, 1]$. Figure 4-11 shows the "No Queries" case diverging to the single mode case where transition probabilities tend to zero. The VOI case shown in this plot used only one query point and shows a vast improvement over the "No Queries" case. Moreover, with just one query, the "VOI" case shows comparable performance to the "All Queries" motivating the conclusion that even minimal information seems to be "good enough". We also show similar results in the continuous state estimate error plots.

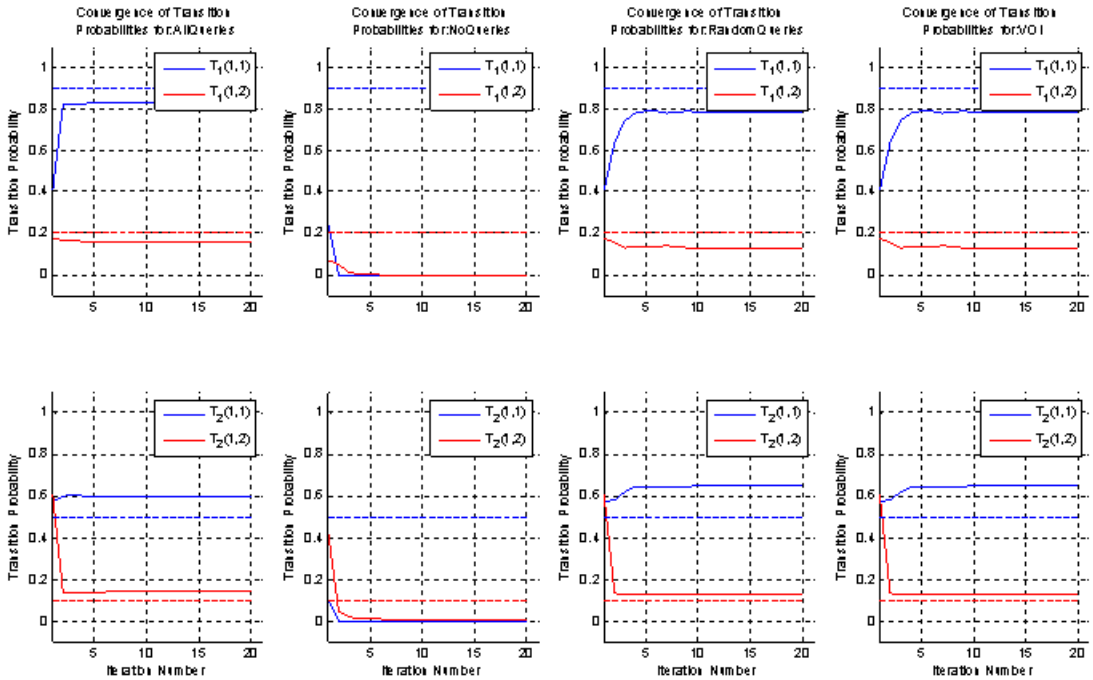

Figure 4-11: Comparison plot of the transition probability convergence for the two guard conditions. This plot shows divergence of the algorithm in the "No Queries" case due to poor initialization conditions.

4.5 Conclusions

In this chapter we presented an approach to active learning for hybrid models with autonomous mode transitions. We used entropy of the distribution of the hybrid state as a loss function for which the resulting VOI is the mutual information of the query and the hybrid state. We were able to extend active learning for the purely discrete case with HMMs to the hybrid case with SLDS, this employs the key intuition that in the hybrid case, we need only to query the discrete state, because given the discrete state, we can a Kalman Smoother to provide us with an exact estimate of the continuous state. We implemented our algorithm in simulation and presented the empirical results. Below we summarize the important findings of this chapter:

1. We presented an approach for active hybrid model learning that incorporates autonomous mode transitions where the discrete state transitions are dependent on the continuous state.

2. We utilized the concept of queries and Value of Information where we query only the discrete state at timesteps where the VOI value is high, where VOI takes the form of Mutual

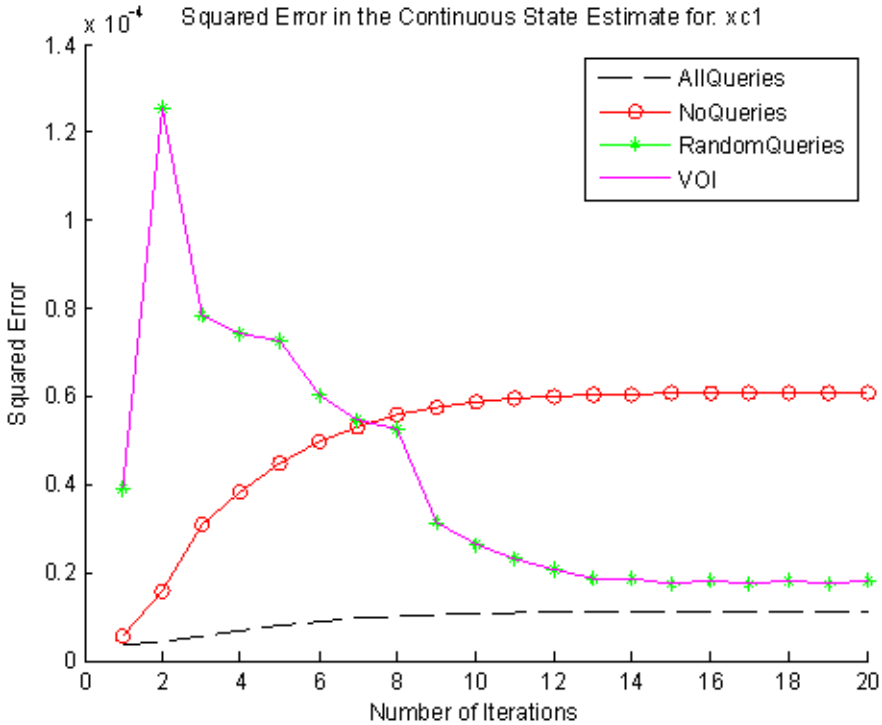

Figure 4-12: This plot shows the squared error in the continuous state xc1. This plot shows divergence of the algorithm in the "No Queries" case due to poor initialization conditions.

Information for our problem.

3. We find that in many cases the VOI plot is sparse, meaning that the number of values for which the VOI is high are few compared to the number of total timesteps. This motivates the conclusion that random querying is as effective as not querying at all for these cases, and that a moderate amount of well-chosen queries can be sufficient for performing active hybrid model learning. Our plots however, show that VOI and Random Querying can have comparable performance in some cases, motivating further investigation of an appropriate loss function.

4. We find that by querying only 3% of the time, and by choosing these queries to be optimal, such that the VOI is maximized, we gain an improvement in MAP mode estimation error and squared error of the continuous state trajectory over the "Random Queries" case and a more significant improvement over the purely passive, or "No Queries" case. Our empirical evidence bolsters the idea that augmenting sparse or very noisy observations with even a

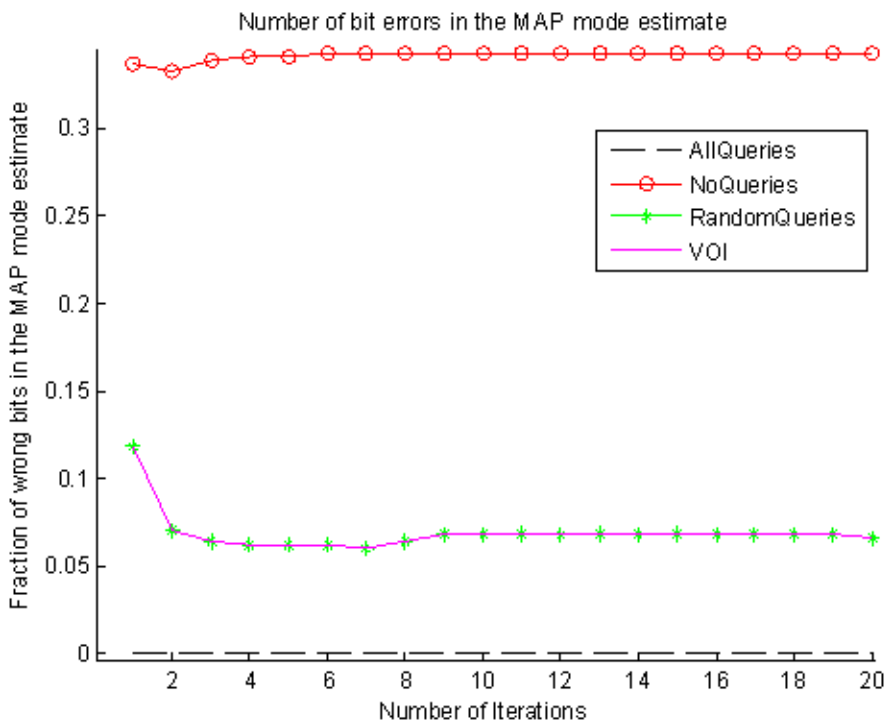

Figure 4-13: This plot shows the MAP Mode Estimation error. This plot shows divergence of the algorithm in the "No Queries" case due to poor initialization conditions.

modest number of well-chosen queries can greatly improve quality of the learned model.

5. In cases where hybrid model learning is particularly sensitive to diverging, such as poor initial conditions or large amounts of noise, we find that active hybrid model learning *is* capable of converging and in fact demonstrates performance comparable to the "All Queries" case. In the extreme case, we found that for these particularly troublesome runs, even one query is enough to prevent divergence of the algorithm and vastly improve learning ability. However, we note that poor initialization conditions and large amounts of noise may also affect the accuracy of the VOI calculation and thus this may also introduce some difficulties for performing active hybrid model learning.

Chapter 5

Summary and Future Work

5.1 Summary

In this thesis we have presented algorithms for learning models of systems that exhibit both continuous and discrete dynamic behaviors. We also allow for modeling of systems whose transition probabilities between discrete modes is conditioned on the continuous state via Autonomous Mode Transitions. In the first chapter we presented a hybrid model learning algorithm for these Linear Probabilistic Hybrid Automata that we call the HML-LPHA algorithm. We find that this algorithm, while successfully demonstrating empirical convergence via a tracking of the change in the lower bound to the objective function, $g(\theta)$, is susceptible to convergence to local maxima of $g(\theta)$ and often converges to local maxima that are close to the initial guess of model parameters. This is not a desirable property for practical applications where a locally optimal set of learned model parameters may be far from the true model parameters of the system being modeled. Therefore, we dedicate the rest of the technical focus of the thesis, chapters 3 and 4, to developing the kMeans-EM and Active Hybrid Model Learning for LPHA algorithms that address these limitations of the HML-LPHA algorithms. Our kMeans-EM algorithm is an approach for learning continuous model parameters that aims to avoid being trapped in locally optimal estimates to which the HML-LPHA algorithm is prone. By learning the area of the known maxima of $g(\theta)$ and jumping to potential solutions in the parameter space outside of these hills, the kMeans-EM algorithm is less susceptible to converging to locally optimal solutions. The Active Hybrid Model Learning algorithm aims to improve the accuracy of parameter model learning for LPHA via active learning. Active Learning

in this context includes the ability to query the discrete state of the system, or add labeled data, to enhance learning capability and accuracy.

We demonstrate empirically that both the kMeans-EM and AHML-LPHA algorithms show improvements in the accuracy of the learned model parameters and in sensitivity to initialization. The kMeans-EM algorithm consists of clustering a set of model parameters that were initially labeled by the EM portion of the algorithm. Once these parameters are clustered, the algorithm derives Gaussian distributions that approximate the means and covariances of these clusters. We show empirically that these derived Gaussians provide accurate fits to the clustered parameter data. These Gaussian clusters provide a map of the shape and location of the explored local maxima of the log-likelihood function $g(\theta) = p(y|\theta)$, where y is a sequence of noisy observations that we are learning from. We derive an objective function $s(\theta)$ that incorporates a Gaussian prior over feasible model parameters and the Gaussian cluster distributions. Maximization of $s(\theta)$ provides a new set of initialization parameters $\widetilde{\theta_0}$ that is within the feasible region of the search space of model parameters, while maximizing the probability that $\widetilde{\theta_0}$ will not converge to any of the local maxima of $g(\theta)$ that have already been discovered.

We demonstrate empirically that the kMeans-EM algorithm outperforms EM-based model learning that is initialized using a random restarts method. We present aggregate results that demonstrate that with each iteration, the kMeans-EM algorithm learns sets of model parameters with higher average likelihood values, whereas the random restarts method rediscovers the same local maxima and provides solutions with the same likelihood values on average . We also present aggregate results that demonstrate a reduction in Euclidean distance from the true set of model parameters in the case of the kMeans-EM algorithm. We find that the algorithm is sensitive to the accuracy of the clustering of model parameters, all of our presented results were obtained using clusters that were sub-optimal with a normalized separation between clusters of only about 60%. Our positive results indicate that even with sub-optimal clustering schemes of the model parameters, kMeans-EM performs better than a Random Restarts method with regards to obtaining higher likelihood solutions, and solutions that are closer to the global maximum in the Euclidean sense.

The Active Hybrid Model Learning algorithm learns the hybrid model parameters, both discrete and continuous, for LPHA. This algorithm takes as input a sequence of noisy observations, a sequence of control inputs, and the results of noisy queries of the discrete state performed at a

limited number of timesteps. We choose the timesteps for performing these queries by maximizing an entropy based Value of Information function. We derive an entropy based VOI function for the hybrid model learning case. This VOI function is of the form of the Mutual Information between the current hybrid state distribution and the query at time t. We also derive an update to the current hybrid belief state given the result of the query and run a Kalman smoothing step to find the updated probabilities of the hybrid state trajectories.

We find that in many cases the VOI plot is sparse, meaning that the number of values for which the VOI is high are few compared to the number of total timesteps. This motivates the conclusion that a moderate amount of well-chosen queries can be sufficient for performing active hybrid model learning. Empirical results bolster our conclusion that a minimal number of queries, querying as little as 6% of the time, demonstrate improvement in squared error of the continuous state and We find that performing even less queries, 3% of the time, we find an average error reduction of 10% over the no queries case. We find that using entropy-based VOI to choose the queries does not demonstrate vast improvement over the random queries case, although querying, whether using VOI or random, does provide vast improvement over the no queries case. This motivates a further investigation of an appropriate VOI definition for hybrid model learning. For situations where hybrid model learning is particularly sensitive to diverging, such as poor initial conditions or large amounts of noise, we find that active hybrid model learning *is* capable of converging to a set of learned parameters. In the extreme case, even one query is enough to prevent divergence of the algorithm and vastly improve learning ability. However, we note that poor initialization conditions and large amounts of noise may also affect the accuracy of the VOI calculation and thus this may also introduce some difficulties for performing active hybrid model learning.

5.2 Future Work

The work performed in this thesis brought to light many interesting areas for extensions and future work. Because of the generality and widespread applicability of Jump Markov Linear Systems and Linear Probabilistic Hybrid Automata, attention to the important problem of accurately learning these models is well warranted. Below we suggest a few areas of future work in this area of research that were brought to our attention as a result of working on the Hybrid Model Learning problem.

1. *Applying Sampling Methods to the E-step in the HML-EM Algorithm*: One of the most challenging aspects of this work is the exponential growth of the hybrid state space due to the switching nature of the underlying model. This forces the use of some approximative method for finding the distribution over hybrid state trajectories and prevents the calculation of a tight bound to the log-likelihood function $g(\theta)$ in the E-step of the algorithm. This approximation may work well in practice but prevents one from providing a theoretical proof of convergence as in the classic EM algorithm. There exist algorithms, such as Monte Carlo EM [ref:MCEM, SEM], that replace the E-step with a sampling method for situations where the E-step is intractable. To our knowledge, this has not been applied to the hybrid case where the log-likelihood function $g(\theta)$ itself cannot be evaluated. Sampling techniques such as the Metropolis Hastings algorithm allows convergence to the true distribution from which the samples are drawn. The hybrid model learning algorithm can substitute the E-step with a MH-type algorithm that allows for convergence to the true hybrid state distribution. Given the true distribution, $g(\theta)$ can be evaluated and proof of convergence to local maxima of this function can be preserved from classic EM.

2. *Constraining the M-step in the HML-EM Algorithm*: An additional challenging aspect of hybrid model learning is the wealth of system matrices that can be used to describe a linear dynamical system. There is literally an infinite number of model parameters that can describe the same dynamical system. This makes the optimization more challenging. One avenue of possible improvement would be to constrain the M-step to learn model parameters of a constrained form. For example, if all of the parameters were constrained to be in modal form, one guarantees that there is only one set of modal form model parameters that represent the same dynamical system. This can facilitate model learning. The interested reader should consult literature in the area of constrained EM methods.

3. *Extending the kMeans-EM Algorithm to Learn Discrete Model Parameters*: Currently the kMeans-EM algorithm provides a method for learning the continuous model parameters. This is because by not applying kMeans-EM to the switching case, we maintain the ability to evaluate the log-likelihood of each learned model parameters and thus have an extra criteria from which to judge algorithm performance. The kMeans-EM algorithm itself does not use log-likelihood values, it uses Euclidean distance between parameters for clustering. Therefore,

we believe that extension from its current form, to incorporating learning of the discrete model parameters as well would be a straightforward extension. However, there is a possibility for a greater rate of algorithmic divergence in the switching case, and thus future work in this area would need to address this.

4. *Providing Optimal Clusters for the kMeans-EM Method*: Because the kMeans-EM method utilizes k-means clustering for clustering the labeled model parameters, it is subject to the same limitations as the k-means algorithm. This includes sub-optimal clustering in some cases. Because the kMeans-EM algorithm relies on accurately clustered data for deriving its objective function $s(\theta)$, inaccurate clustering of the model parameters can negatively influence its performance. How to cluster model parameters as accurately as possible is a topic for future research.

5. *Active Hybrid Model Learning Using Control Inputs*: In this thesis we demonstrated an algorithm for Active Hybrid Model Learning using the notion of querying the discrete state at timesteps of high Value of Information. This algorithm has applications in areas where the discrete state of the system can be inferred visually, as in many fields of machine vision including activity recognition and tracking where a human interpreter can provide information of the discrete state upon request. An example of this could be an activity recognition scheme where the gait of a person is to be inferred from a video sequence and the different gates correspond to the discrete state. Applications where the discrete state is unobservable, as in fault detection and diagnosis, are not suited for AHML as we have derived it in this thesis. For these applications, an active learning approach where a control sequence can be designed to disambiguate the hybrid model would be better suited. These control sequences are often referred to as diagnostic signals, or auxiliary signals in the literature.

Appendix A

Appendix

In this appendix we present the derivations of the results for learning the hybrid model parameters of Linear Probabilistic Hybrid Automata. The results of this chapter were part of a joint work with Lars Blackmore and can also be found in [8, 6].

Expectation Step for Hybrid Model Learning

Our objective in the E-Step is to construct the lower bound $h\left(\theta|\theta^k\right)$. The expectation in the lower bound can now be written as an iterated expectation where the inner expectation is over the continuous mode sequence conditioned on the discrete mode sequence, observation sequence, and the current guess of model parameters, and the outer expectation is over the discrete mode sequence also conditioned on the observation sequence, and the current guess of model parameters.

$$h\left(\theta|\theta^k\right) = \sum_{x_{d0}^T}\left(p\left(x_{d0}^T|y_1^{T+1},\theta^k\right)\int p\left(x_{c0}^{T+1}|x_{d0}^T,y_1^{T+1},\theta^k\right) * \log p\left(y_1^{T+1},x_{c0}^{T+1},x_{d0}^T|\theta\right)dx_{c0}^{T+1}\right) + \mathcal{H} \tag{A.1}$$

where

$$p(y_1^{T+1},x_{c0}^{T+1},x_{d0}^T|\theta) = p(x_{c0},x_{d0}|\theta)\prod_{t=0}^{T-1}p(x_{dt+1}|x_{ct},x_{dt},\theta) \tag{A.2}$$

$$*\prod_{t=0}^{T}p(y_1^{T+1}|x_{ct+1},x_{dt},\theta)p(x_{ct+1}|x_{ct},x_{dt},\theta) \tag{A.3}$$

by the Markov properties of LPHA, and using our definition of Linear Dynamical Systems:

$$p(x_{ct+1}|x_{ct}, x_{dt}, \theta) = (2\pi)^{-\frac{n_x}{2}} |Q(x_{dt})|^{-\frac{1}{2}} \exp^{-\frac{1}{2}\delta_p' Q^{-1}(x_{dt})\delta_p}$$
$$\delta_p = x_{ct+1} - A(x_{dt})x_{ct} - B(x_{dt})u_t. \tag{A.4}$$

$$p(y_{T+1}|x_{ct+1}, x_{dt}, \theta) = (2\pi)^{-\frac{n_y}{2}} |R(x_{dt})|^{-\frac{1}{2}} e^{-\frac{1}{2}\delta_o' R^{-1}(x_{dt})\delta_o}$$
$$\delta_o = y_{T+1} - C(x_{dt})x_{ct+1} - D(x_{dt})u_t \tag{A.5}$$

The initial probability distribution $p(x_{c0}, x_{d0})$ is given by:

$$p(x_{c0}, x_{d0}|\theta) = p(x_{d0})(2\pi)^{-\frac{n_x}{2}} |V(x_{d0})|^{-\frac{1}{2}} *$$
$$\exp^{-\frac{1}{2}\left[x_{c0}-\mu(x_{d0})\right]' V^{-1}(x_{d0})\left[x_{c0}-\mu(x_{d0})\right]}. \tag{A.6}$$

The distribution $p(x_{c0}^{T+1}, x_{d0}^{T}|y_1^{T+1}, \theta^k)$ is referred to a hidden state estimation for hybrid systems. Although the continuous and discrete components of the state are interdependent as per our SLDS definition, we can view the continuous state at the next timestep $t+1$ as being fully defined once given information about the previous continuous state at time t and the previous discrete state also a time t. Therefore, given a sequence of discrete states and a sequence of observations, an exact estimate for the continuous state can be found via the Kalman Filter. For the case of state estimation, we are only interested in using observations y_0^t where t is the current timestep. However, for the problem of parameter estimation, we must take advantage of the *entire* sequence of observations and thus the E-step utilizes a Kalman Filter Smoothing procedure which performs and forwards and backwards pass through the data, yielding a distribution over the continuous state that is conditioned on the entire observation sequence and a particular discrete mode sequence, $p(x_{c0}^{T+1}|y_1^{T+1}, x_{d0}^{T})$.

The observation likelihood $p(y_1^{T+1}|x_{d0}^{T}, \theta)$ is obtained through the Kalman Filter residuals. Given this distribution, one can obtain the likelihood of a particular discrete sequence x_{d0}^{T} via an

application of Bayes' Rule where

$$p(x_{d0}^{T}|y_1^{T+1},\theta) = \frac{p(y_1^{T+1}|x_{d0}^{T},\theta)p(x_{d0}^{T}|\theta)}{\sum_{x_{d_0}^{T}} p(y_1^{T+1},x_{d0}^{T}|\theta)p(x_{d0}^{T}|\theta)} \tag{A.7}$$

Combining the continuous state distribution with the discrete state distribution, we arrive at the joint distribution over our hybrid state.

$$p(x_{c0}^{T+1},x_{d0}^{T}|y_1^{T+1},\theta^k) = p(x_{c0}^{T+1}|y_1^{T+1},x_{d0}^{T},\theta^k)p(x_{d0}^{T}|y_1^{T+1},\theta^k) \tag{A.8}$$

A.0.1 Approximate EM for Hybrid Model Learning

Approximate E-Step for Hybrid Model Learning

The approximation to the lower bound was found to be:

$$\widetilde{h}\left(\theta|\theta^k\right) = \sum_{x_{d0}^{T}\in\mathcal{S}} \left(\widetilde{p}\left(x_{d0}^{T}|y_1^{T+1},\theta^k\right)\int p\left(x_{c0}^{T+1}|x_{d0}^{T},y_1^{T+1},\theta^k\right) * \log p\left(y_1^{T+1},x_{c0}^{T+1},x_{d0}^{T}|\theta\right)dx_{c0}^{T+1}\right) + \widetilde{\mathcal{H}} \tag{A.9}$$

Approximate M-Step for Hybrid Model Learning

B. *Maximization Step for Discrete Hybrid Model Parameters*

In this section we demonstrate maximization of the lower bound in A.9 with respect to θ_d to yield the optimal discrete model parameters. We first note that the discrete model parameters are defined for each guard condition $c_i \in \mathcal{G}$. Each guard condition has a corresponding transition matrix T_i. Because the guard conditions are placed upon the continuous state estimate, we have implicitly taken into account the dependence of the discrete state transitions on the continuous state estimate. We will highlight what we deem to be the most important steps in the derivation of the optimal discrete model parameters [6, 8]. In order for our transition probability matrices to be valid, we must perform a constrained optimization using a Lagrangian Multiplier for all of the

possible discrete states $x_d \in \mathcal{X}_d$:

$$\text{Maximize over } T_i(j, \mathbf{x_d}), \;\; j = 1, \ldots, |\mathcal{X}_d| : \tag{A.10}$$

$$\tilde{h}(\theta|\theta^k)$$

$$\text{Subject to:}$$

$$\sum_{j=1}^{|\mathcal{X}_d|} T_i(j, \mathbf{x_d}) = 1. \tag{A.11}$$

This maximization involves setting the following expression to zero in order to solve for $T_i^*(j, x_d)$.

$$\frac{\partial \tilde{h}(\theta|\theta^k)}{\partial T_i(j, \mathbf{x_d})} = \frac{\partial}{\partial T_i(j, \mathbf{x_d})} \sum_{\mathbf{x_d}_0^T \in \mathcal{S}} \tilde{p}(\mathbf{x_d}_0^T | \mathbf{y}_1^{T+1}, \theta^k) *$$
$$\sum_{t=1}^{T} \int p(\mathbf{x}_{t-1} | \mathbf{y}_1^{T+1}, \mathbf{x_d}_0^T, \theta^k) \log p(\mathbf{x_d}_t | \mathbf{x_d}_{t-1}, \mathbf{x}_{t-1}, \theta) d\mathbf{x}_{t-1} \tag{A.12}$$

Evaluation of this expression is challenging because the transition probability, $p(\mathbf{x_d}_t | \mathbf{x_d}_{t-1}, \mathbf{x}_{t-1}, \theta)$, has an explicit dependence on the continuous state $\mathbf{x}_{t-1}$. The key insight that allows us to simplify this expression is that the distribution $p(\mathbf{x_d}_t | \mathbf{x_d}_{t-1}, \mathbf{x}_{t-1}, \theta)$ is actually constant for each guard condition c_i. Therefore, we can rewrite the integral over x_{t-1} as a sum over guard conditions.

$$\frac{\partial \tilde{h}(\theta|\theta^k)}{\partial T_i(j, \mathbf{x_d})} = \frac{\partial}{\partial T_i(j, \mathbf{x_d})} \sum_{\mathbf{x_d}_0^T \in \mathcal{S}} \tilde{p}(\mathbf{x_d}_0^T | \mathbf{y}_1^{T+1}, \theta^k)$$
$$* \sum_{t=1}^{T} \sum_{c_i \in \mathcal{G}} p_{c_i}(\mathbf{x_d}_0^T) \log T_i(\mathbf{x_d}_t, \mathbf{x_d}_{t-1}), \tag{A.13}$$

where $p_{c_i}(\mathbf{x_d}_0^T)$ is the probability that guard condition c_i is satisfied given the discrete mode sequence $\mathbf{x_d}_0^T$, the observation sequence $\mathbf{y}_1^{T+1}$ and the current guess of the parameters θ^k.

$$p_{c_i}(\mathbf{x_d}_0^T) = \int_{\mathcal{C}_i} p(\mathbf{x}_{t-1}|\mathbf{y}_1^{T+1}, \mathbf{x_d}_0^T, \theta^k) d\mathbf{x}_{t-1}, \tag{A.14}$$

We've used $\mathcal{C}_i$ to denote the region of x_{t-1} over which the guard c_i is satisfied. Completing the optimization we obtain the optimal value of $T_i(j, x_d)$:

$$T_i^*(j, \mathbf{x_d}) = \frac{\sum_{\mathbf{x_d}_0^T \in \mathcal{S}} \tilde{p}(\mathbf{x_d}_0^T|\mathbf{y}_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(\mathbf{x_d}_0^T)} p_{c_i}(\mathbf{x_d}_0^T)}{\sum_{\mathbf{x_d} \in \mathcal{X}_d} \left(\sum_{\mathbf{x_d}_0^T \in \mathcal{S}} \tilde{p}(\mathbf{x_d}_0^T|\mathbf{y}_1^{T+1}, \theta^k) \sum_{t \in \mathcal{F}(\mathbf{x_d}_0^T)} p_{c_i}(\mathbf{x_d}_0^T) \right)}. \tag{A.15}$$

Therefore, we have just demonstrated how to obtain the ML discrete model parameters for our hybrid system by using a constrained optimization approach involving Lagrange multipliers that also handles Autonomous Mode Transitions. This Maximum Likelihood solution can be interpreted as a weighted number of transitions from a source mode x_{t-1} to a target mode x_t for each guard condition c_i.

Bibliography

[1] Brigham Anderson and Andrew Moore. Active learning for hidden markov models: objective functions and algorithms. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 9–16, New York, NY, USA, 2005. ACM.

[2] M. Aoki. Optimization of stochastic systems. *Academic Press*, 1964.

[3] D. K. Arrowsmith and C. M. Place. *An Introduction to Dynamical Systems*. Cambridge University Press, 1990.

[4] H. Balakrishnan, I. Hwang, J. Jang, and C. Tomlin. Inference methods for autonomous stochasitc linear hybrid systems. *Lecture Notes in Computer Science*, 2993:64–79, 2004.

[5] Jeff A. Bilmes. A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, 1997.

[6] L. Blackmore, S. Gil, S. Chung, and B. Williams. Model learning for switching linear systems with autonomous mode transitions. *46th IEEE Conf. on Decision and Control*, pages 4648–4655, 2007.

[7] L. Blackmore, S. Rajamanoharan, and B. Williams. Active estimation for switching linear dynamic systems. *45th IEEE Conf. on Decision and Control*, pages 137–144, 2006.

[8] Lars Blackmore. *Robust Execution for Stochastic Hybrid Systems*. PhD thesis, Massachusetts Institute of Technology, 2007.

[9] H.A.P Blom and Y. Bar-Shalom. Interacting multiple model algorithm for systems with Markovianswitching coefficients. *IEEE Trans. on Automatic Control*, 33:780–783, 1988.

[10] Christoph Bregler. Learning and recognizing human dynamics in video sequences. pages 568–574, 1997.

[11] Sen Cheng and Philip N. Sabes. Modeling Sensorimotor Learning with Linear Dynamical Systems. *Neural Comp.*, 18(4):760–793, 2006.

[12] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[13] A. P. Dempster and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[14] A. Doucet, A. Logothetis, and V. Krishnamurthy. Stochastic sampling algorithms for state estimation of jump markov linear systems, 2000.

[15] Zoubin Ghahramani and Geoffrey E. Hinton. Parameter estimation for linear dynamical systems. Technical report, 1996.

[16] Zoubin Ghahramani and Georey E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12:963–996, 1998.

[17] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[18] Melvin Henry. Model-based estimation of probabilistic hybrid automata. Master's thesis, Massachusetts Institute of Technology, 2002.

[19] M. Hofbaur and B. Williams. Mode estimation of probabilistic hybrid systems. *Lecture Notes in Computer Science*, 2289:253–266, 2002.

[20] R. Jaulmes, J. Pineau, and D. Precup. Active learning in partially observable Markov decision processes. *Lecture Notes in Computer Science*, pages 601–607, 2005.

[21] C. J. Kim. Dynamic linear models with Markov-switching. *Journal of Econometrics*, 60(1–2):1–22, 1992.

[22] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[23] B.Juang L. Rabiner. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[24] R. McEwen. Modeling and control of a variable-length auv. Technical report, 2006.

[25] T. Minka. Expectation-maximization as lower bound maximization. 1998.

[26] Melanie Mitchell. *An introduction to genetic algorithms*. MIT Press, Cambridge, MA, USA, 1996.

[27] Kamal Paul Nigam. Using unlabeled data to improve text classification. Technical report, 2001.

[28] R. Nikoukhah and S. L. Campbell. Model identification.

[29] Sang Min Oh, James M. Rehg, Tucker Balch, and Frank Dellaert. Learning and inference in parametric switching linear dynamical systems. *iccv*, 2:1161–1168, 2005.

[30] Vladimir Pavlovic, James M. Rehg, Tat jen Cham, and Kevin P. Murphy. A dynamic bayesian network approach to figure tracking using learned dynamic models. pages 94–101, 1999.

[31] Vladimir Pavlovic, James M. Rehg, and John Maccormick. Learning switching linear models of human motion. pages 981–987, 2000.

[32] P. Peursum, S. Venkatesh, and G. West. Observation-switching linear dynamic systems for tracking humans through unexpected partial occlusions by scene objects. *IEEE 18th Intl. Conf. on Pattern Recognition*, 4:929–934, 2006.

[33] Irina Rish, Mark Brodie, Natalia Odintsova, Sheng Ma, and Genady Grabarnik. Problem diagnosis in distributed systems using active probing, 2002.

[34] Nicholas Roy and Andrew Mccallum. Toward optimal active learning through sampling estimation of error reduction. In *In Proc. 18th International Conf. on Machine Learning*, pages 441–448. Morgan Kaufmann, 2001.

[35] R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Time Series Analysis*, 3:253–264, 1982.

[36] Shuonan Song. Unsupervised learning and recognition of physical activity plans. Master's thesis, Massachusetts Institute of Technology, 2007.

[37] James C. Spall and Senior Member. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37:332–341, 1992.

[38] Simon Tong and Daphne Koller. Active learning for structure in bayesian networks. In *In International Joint Conference on Artificial Intelligence*, pages 863–869, 2001.