Inference in Hybrid Systems with Applications in Neural Prosthetics

Thesis by

Nicolas H. Hudson

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2009

(Defended September 22^{nd}, 2008)

To Jade, with much gratitude for the endless meals, love and support.

Acknowledgements

There are many people who have contributed to my Ph.D., whose auspices have been invaluable over the last five years, including but not limited to:

A big thank you to my Ph.D. Advisor, Prof. Joel Burdick, for his unwaivering support and wisdom, not to mention all of those all-expenses paid overseas conferences.

The entire Burdick Group, both current and previous members, especially Michael Wolf.

Fellow Kiwi, Prof. Jim Beck, for all of the long discussions and advice on probability theory, and members of his group, especially Alexandros Taflanidis.

Prof. Richard Anderson and his laboratory, specifically Grant Mulliken and EunJung Hwang for their advice, and Sam Musallam and Hans Scherberger for generously allowing me to work with their neural data.

My Ph.D. Committee members not already mentioned above, Prof. Richard Murray, and Prof. Pietro Perona.

All of my Thomas building friends and colleagues, SOPS and NESS members, and the Caltech community at large.

The William Pickering Fellowship.

My Wife Jade, and my Mum and Dad.

Abstract

This thesis develops new hybrid system models and associated inference algorithms to create a "supervisory decoder" for cortical neural prosthetic devices that aim to help the severely handicapped. These devices are a brain-machine interface, consisting of surgically implanted electrode arrays and associated computer decoding algorithms, that enable a human to control external electromechanical devices, such as artificial limbs, by thought alone.

Hybrid systems are characterized by discrete switching between sets of continuous dynamical activity. New hybrid models, which are flexible enough to model neurological activity, are created that incorporate both duration and dynamical state based switching paradigms. Combining generalized linear models with nonstationary and semi-Markov chains gives rise to three new hybrid systems: generalized linear hidden Markov models (GLHMM), hidden semi-Markov models (HSMM) with generalized linear model dynamics, and hidden regressor dependent Markov models (HRDMM). Bayesian inference methods, including variational Bayes and Gibbs sampling, are derived for the identification of existing and developed hybrid models. The developed inference algorithms provide advances over the current hybrid system identification literature by providing a principled way to incorporate prior knowledge and select between alternative model classes and orders, including the number of discrete system states.

Future neuroprostheses that seek to provide a facile interface for the paralyzed patient will require a supervisory decoder that classifies, in real time, the discrete cognitive, behavioral, or planning state of the brain. The developed hybrid models and inference algorithms provide a framework for supervisory decoding, where first, a hybrid-state neurological activity model is identified from data, and then used to estimate the discrete state in real time. The electrical activity of multiple neurons from a cortical area in the brain associated with motor planning (the parietal reach region), and multiple signal types, including both spike arrival times and local field potentials, are fused to give more accurate results. The model structure, including the number of discrete cognitive states, can also be estimated from the data, resulting in significantly improved decoding performance compared to existing methods.

Additional demonstrated applications include the automated segmentation of honey bee motion into discrete primitives, and generating mechanical system models for a pick-and-place machine.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis develops new methods for the identification of several classes of hybrid systems that are characterized by discrete switching between sets of continuous dynamical activity. Inspired by the characterization of biological systems into discrete sets of behaviors or modes, the developed models use a mixture of duration, time, and dynamical state-based switching paradigms. Specifically, both stationary and nonstationary Markov chains are used to govern mode switching, while generalized linear models are used to represent the set of continuous dynamics. To identify hybrid system models from data, a Bayesian framework is used, so as to facilitate incorporation of prior knowledge in a coherent way, and provide a basis for selection between a set of possible models. The inherent difficulty in identifying hybrid or switching systems from data, is a consequence of the discrete system states being "hidden", and not observed. Thus, in identifying this class of systems, simultaneous identification of the continuous dynamics and classification of the observed data into discrete model states is required. To approach the identification problem in a structured way, the variational Bayesian framework, the Gibbs sampler, and the expectation maximization algorithms are adapted for inference in developed models.

Developed methods are used for model generation in cortical neuroprosthetic medical devices that aim to help the severely handicapped. In such systems, a "supervisory decoder" is required to classify the activity of extracellular neural recordings into a discrete set of modes that model the evolution of the brain's planning process. In moving from experimental, laboratory-based prosthetic development programs to clinical applications and rehabilitation of patients, new automated methods for generating supervisory decoders and control systems are required. In the long term, the models and algorithms developed here may eventually form the backbone of an integrated prosthetic development and deployment system.

1.1 Motivation

1.1.1 Hybrid Systems, Modeling, and Identification

Hybrid dynamical systems are characterized by the interplay of discrete transitions and continuous dynamics. These systems have both discrete and continuous states: a hybrid system may jump between discrete states (or modes of operation), and while within each mode, associated dynamics determine the evolution of the continuous state. Hybrid systems naturally encompass embedded and computer-controlled systems, systems involving physical impact, and can model a range of complex systems found in nature. The primary aim of this thesis, identifying hybrid system models from data, is derived from this later class of systems. It is often convenient to impose a model with a finite set of discrete states when analyzing biological and natural systems: The walking gait of humans is naturally segmented into *stance, heel-off, swing*, and *heel-strike* modes [1]; the behaviors or movements of animals or humans is often reduced to a set of motion primitives, or movemes [2], that characterize the evolution of the system. Even complicated artificial systems such as air-traffic control, which models the interaction between the continuous flight dynamics of aircraft, a discrete set of aircraft maneuvers, and a distributed decision-making process which determines the route of the aircraft, is modeled as a hybrid system with a finite set of discrete states [3].

The key difficulty in identifying a hybrid system model from observed data is caused by the interaction of the system's continuous dynamics and discrete transitions. In most systems in nature only the state or output of the continuous dynamics are observed. The system's discrete state, or mode, is hidden from observation, and needs to be inferred from the data. Classifying the observed data into the system's discrete states can be accomplished if the continuous dynamics are known, and vice versa, if all of the observed data is classified into discrete states, the continuous dynamics associated with each mode can be identified. However, without knowing either the dynamics or the discrete states a priori, the identification process requires the simultaneous classification of observed data, and the identification of the dynamics associated with each mode [4, 5, 6, 7, 8]. The hybrid system identification problem is combinatorially hard, with the complexity increasing exponentially in the number of data points and number of modes [4].

To provide a computationally tractable approximation to the hybrid system identification problem, we use the *hidden*, *latent*, or *incomplete data* model [9] framework typically used in the machine learning community. In this class of models, latent (or hidden) variables are used to represent the classification of observed data into each system mode. The widely used hidden Markov model [10], is typically identified using this latent variable framework. Several existing algorithms, in particular the expectation maximization (EM) algorithm , variational Bayesian (VB) methods, and Monte Carlo methods such as Gibbs sampling, can be used for approximate Bayesian inference in latent variable models. The core contribution of this thesis will be in adapting these algorithms for identification

of hybrid system models.

A wide range of existing models can be fit in the hybrid system framework. By considering hybrid systems that are extensions of models for which efficient inference tools exist, tractable inference algorithms can be developed. This thesis proposes a series of hybrid system models that are suitable for both activity recognition and neurological prosthetic devices, yet are tractable to identify. The specific models and relation to existing work are reviewed in the following outline and contributions section.

1.1.2 A Supervisory Decoder for Neural Prosthetics

A "neural prosthetic" is a brain-machine interface that enables a human, via the use of surgically implanted electrode arrays and associated computer decoding algorithms, to control external electromechanical devices by pure thought alone. In this manner, some useful functions can be partially restored to patients with severe motor disorders (e.g., Lou Gehrig's disease) or with high-level spinal cord injuries. *Cognitive neural prostheses* work by "decoding", or estimating movement intent or motor plans from the recorded electrical activity of multiple neurons in brain areas (such as the posterior parietal or dorsal premotor cortices) associated with motor planning. These decoded plans can be used to drive devices such as prosthetic arms or computer interfaces [11, 12, 13]. Future practical clinical neuroprostheses that seek to provide a facile interface for the paralyzed patient will require a *supervisory decoder* whose job is to classify, in real time, the *discrete* cognitive, behavioral, or planning *state* of the brain region from which the neural signals are recorded. For example, the supervisory decoder must determine if: (1) the patient is asleep or disinterested in using the prosthetic; (2) the patient wishes to use the prosthetic; (3) the patient is planning an action that must be decoded; (4) the patient wants to execute the planned action; (5) the patient wants to scrub or change the current action. While the actual planning process in the brain is quite complex, for the purposes of supervisory decoding there are a finite number of states that model and govern the relevant high-level activities of a brain-machine interface. The knowledge of the current state in the evolution of the planning process can be used in a variety of ways. For example, depending upon the current state, different algorithms, or different parameters in the algorithm, can be applied to the decoding of movement plans. Moreover, accurate knowledge of the current cognitive state will improve the action of the prosthetic system.

The development of a supervisory decoder model and identification algorithms must ideally consider several practical aspects of neurological systems, and the realities of utilizing prosthetic devices in a clinical setting. Surgically implanting any device inter-cranially presents a severe risk to the patient, and in cortical neuroprostheses reduces the number and limits the positioning of electrodes into brain areas of interest. The current state-of-the-art clinical practice uses chronic electrode arrays to rehabilitate patients. Once implanted these arrays may produce signals with

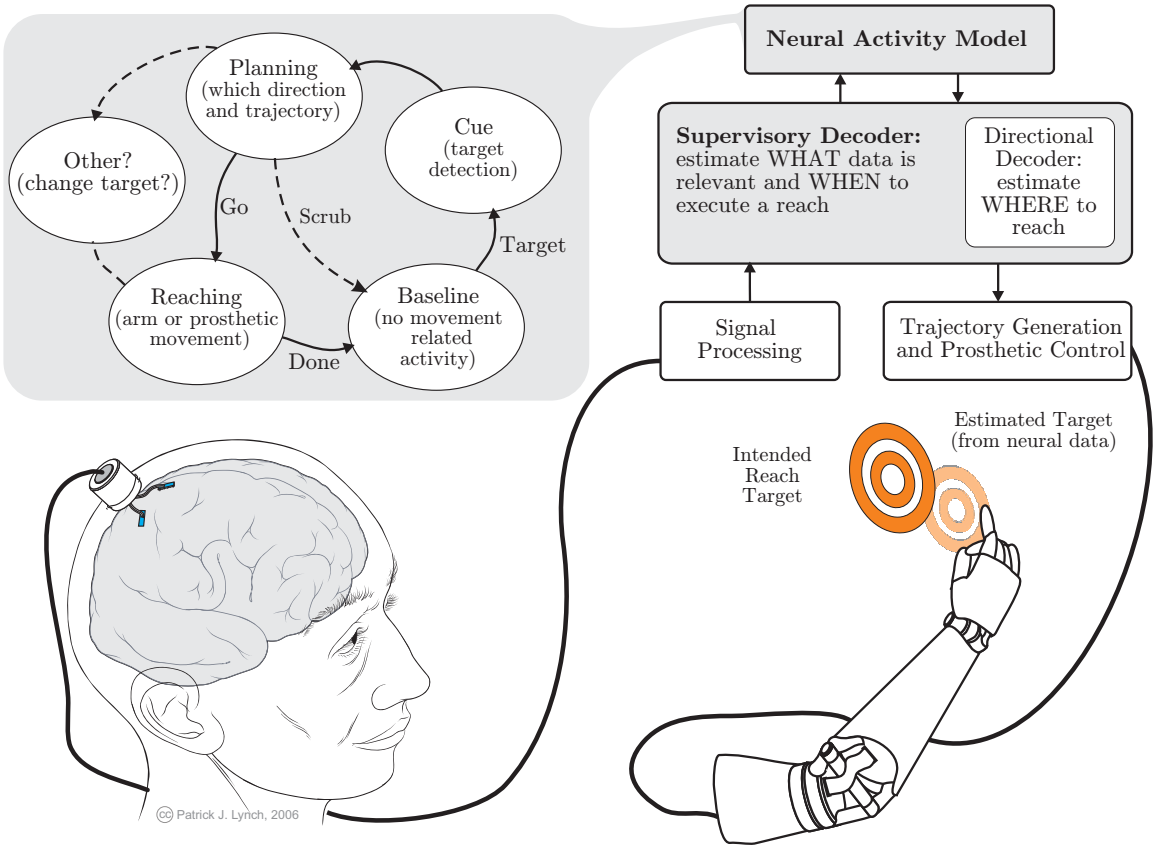

Figure 1.1: A neural prosthetic: supervisory decoding and neural models

poor signal-to-noise characteristics, and over time signal quality will degrade from reactive gliosis [14]. To make the risk of surgical implantation acceptable, the performance and expected lifetime of the prosthetic device should be maximized. By considering all available neurological signals, the practicality of implanting devices can be extended. In the cortical areas of interest here (posterior parietal or dorsal premotor), both local field potentials (LFP) signals [15], representing the aggregate extracellular activity around the implanted electrode, and single unit activity (SUA) [16], which are recordings of an individual neuron's electrical impulses, are typically utilized. Even if a signal modality, such as SUA, can provide the necessary performance in the short term, incorporating LFP signals, which are more robust to gliosis and neuron death, will prolong the life of the prosthetic device. The ideal supervisory decoder will integrate both these signals (and perhaps others) in the modeling and decoding framework. This practical requirement demands a flexible model capable of representing a range of non-Gaussian dynamical systems.

Assuming that high-quality neurological signals can be obtained from implanted electrodes, there remains the task of creating adequate models to interpret recorded neural activity. In the creation of a supervisory decoder, cortical neurological activity is abstracted into a series of discrete states or processes. This abstraction, while providing an intuitive framework to interpret the neurological activity, will introduce "noise" from the parts of the system not included in the model. Due to the incomplete understanding of massively parallel neurological processes, yet-undiscovered cortical functions, and limited recording ability, any modeling framework will need to account for highly variant, non-Gaussian process noise. Even if the same task is repeated in a consistent fashion, the observed neurological process can vary substantially trial-to-trial. Any supervisory decoder that imposes a discrete set of states to model neurological activity is making an assumption about underlying neurological processes. Even with carefully designed experiments and expert intuition, there is no "ground truth" about which discrete mode the brain is currently in, and it may even be difficult to define the optimal number of discrete states, and their correlation to neurological process. An ideal supervisory decoder should be able to automatically determine the optimal number of discrete cognitive states that best represents the observed neurological activity.

To both model the discrete transitions between cognitive states and continuous neurological dynamics, this thesis proposes using a series of hybrid system models to represent observed neural activity. Hybrid systems have become of recent interest to the neurophysiological community [17]. While hybrid systems provide a framework in which to model observed cortical neurological activity, there has been limited development of identification methods for these systems. To both facilitate inference in neurological systems, as well as other systems of interest, this thesis will focus on the identification of a class of probabilistic hybrid systems.

1.2 Thesis Outline and Contributions

The remainder of this thesis is structured as follows: Chapter 2 presents technical background material and a literature review, and includes relevant classes of hybrid systems, fundamentals of using a probabilistic inference approach, and existing learning algorithms. Hybrid system models, including Markov jump systems (MJS) [18, 19], timed automata [20, 21], and PWARX systems [4, 5, 6, 7, 8], are presented to contextualize the hybrid models developed in this thesis. Existing hybrid identification algorithms are discussed, and in conjunction with the neuroprosthetic application, motivate the development of a probabilistic inference framework. To complement the hybrid system identification literature, latent variable models, and specifically the hidden Markov model are introduced, and provide a basis for reviewing existing inference algorithms present in the machine learning community. Three inference algorithms, expectation maximization [9], variational Bayes [22, 23, 24], and the Gibbs sampler [25] are reviewed, both in general and in the context of latent variable models. A key tool, the forward-backward algorithm (a form of dynamic programming), is discussed in the context of latent variable models.

In Chapter 3 several hybrid system models are developed in tandem with the adaptation of existing EM, VB, and Gibbs sampling algorithms to create a new framework for hybrid identification. The first hybrid system model that is considered is denoted as a generalized linear hidden Markov model (GLHMM). This GLHMM is a superset of hidden Markov models [10], and combines generalized linear models (GLM) [26, 27], which represent the systems' continuous dynamics, with a discrete state Markov chain to model switching between system modes. Based on the equivalence of piecewise affine and auto-regressive models [28, 29], GLHMM inference algorithms also provide a new method to identify MJS. Both Gibbs sampler and VB algorithms are developed for inference in this defined GLHMM model class. Motivated by timed automata, where switching between discrete states is a function of the duration spent in each state, a new class of hybrid systems based on both hidden semi-Markov models (HSMM) [30, 31] and variable transition hidden Markov models (VTHMM) [32, 33, 34] are developed. The VB method, for the first time, is applied to both inference in HSMM and VTHMM models, as well as their hybrid system counterparts which utilize GLM dynamics in each mode. These GLHMM and HSMM models will form the basis for the creation of supervisory decoders in Chapter 4, and in addition can be used for inference in PWARX models, by adopting a two-step process typical in the hybrid systems community [7, 5]. Based on the switching characteristics of PWARX models, a new non-stationary, regressor-depended Markov model (HRDMM) is created, and a variational method for identification of HRDMM systems is developed. To summarize the model classes and contributions made in Chapter 3, Table 1.1 provides a concise representation of the developed models and algorithms developed in this thesis. Chapter 3 concludes with a set of estimation (or decoding) algorithms which utilize key tools developed for

Table 1.1: Model classes and identification algorithms discussed in Chapter 3: This table denotes the contributions made, by inclusion of the section number, in the context of a range of models and algorithms present in the literature. Original contributions or seminal papers on the identification of each model type are denoted for each learning algorithm: expectation maximization (EM), variational Bayes (VB), and the Gibbs sampler (GS).

	HMM	GLHMM	VTHMM & HSMM	GL-VTHMM & GL-HSMM	HRDMM
EM	[10]	AR-HMMs [10]	[30, 31, 32, 33, 34]	Sec. 3.5	Sec. 3.7
VB	[35, 36]	Sec. 3.3	Sec. 3.5	Sec. 3.5	Sec. 3.7
GS	[37]	Sec. 3.4	[38]	Sec. 3.5	

the identification algorithms.

To take full advantage of the developed model classes and identification algorithms, Chapter 4 considerers the problem of *model class selection*. In this chapter, the structure of proposed models is considered uncertain, and hence must be inferred from observed data. A detailed analysis of the information theoretic basis for model selection and model evidence decomposition are presented, both making new intuitive observations, as well as reviewing state-of-the-art model selection procedures. This information theoretic perspective provides new fundamental reasons to consider the variational approach as an alternative to the EM algorithm. In addition, estimators to calculate information theoretic quantities from posterior samples are reviewed, and equivalence between two existing estimators is proven in the case of latent variable models. This, for the first time allows an information-theoretic comparison of the VB, Gibbs sampler, and EM algorithms. In addition, several case studies are conducted that demonstrate the advantages of using the VB or Gibbs sampling approaches. In addition to model class selection, the use of automatic structure determination (ASD) priors is applied to developed models. The term ASD is applied to denote a body of work [39, 40, 36, 24, 41], that stems from the automatic relevance determination (ARD) literature [42]. To conclude Chapter 4, ASD priors newly developed for HSMMs, are utilized to automatically determine the number and structure of discrete motion primitives that represent a honey bee dance [43, 44].

Chapter 5 applies the developed modeling and identification framework to create supervisory decoders for neural prosthetics applications. This approach provides several advantages over existing supervisory decoders: First, by developing model classes capable of integrating a broad range of dynamical systems types, the inclusion of many typical neurophysiological signal types is facilitated into a supervisory decoding framework. Notably, this thesis incorporates both local field potentials and single unit activity, but can utilize most neurophysiological signal types. Second, developed methods are capable of identifying models automatically, and do not require pre-sorted neural data to initialize the identification processes. Third, in combination with the model selection tools of Chapter 4, the number, connectivity, and neurological relevance of each discrete state in the supervisory decoder is automatically determined. Fourth, the use of a HSMM allows the creation of supervisory

decoders that automatically build models which consider the duration spent in each cognitive state. Fifth, by tackling the problem from a Bayesian standpoint, the model can be updated with more data as it becomes available: over time the neurological signals will change due to cell death, plasticity, and reactive gliosis. In addition to these contributions, the developed methods are computationally efficient, and can hence be practically deployed into clinical environments. To demonstrate the characteristics of the proposed approach, several case studies are conducted on recorded neural data, and real-time decoding algorithms are used to verify the effectiveness by using the models to infer the cognitive discrete state.

Finally, Chapter 6 summarizes the contributions of the thesis, and suggests future work directions and extensions based on the presented algorithms and case studies.

Chapter 2

Hybrid Systems, Learning Algorithms, and a Review of Hidden Markov Models

2.1 Hybrid Systems

Hybrid dynamical systems are characterized by containing both continuous and discrete behavior, and are defined by a set of continuous dynamics, and discrete logic. Hybrid system can be used to model a wide variety of physical systems, such as a bouncing ball, [45] which switches between free fall and elastic impact modes. Hybrid system theory can also be used to model systems controlled by discrete logic, or embedded (discrete) computing. For example, one can imagine a thermodynamic model of an air conditioned building being split into two distinct continuous modes, one where the air conditioner is on, and one where it is off. More complicated systems, such as an aircraft collision avoidance system [3], are modeled as hybrid automata with aircraft switching between cruising and avoidance flight control maneuvers. The gait of humans [1] has been effectively modeled as a hybrid system, in which walking is broken down into *stance, heel-off, swing*, and *heel-strike* modes. The process of fault detection, where a system enters a critical state or failure mode is also a hybrid system [46].

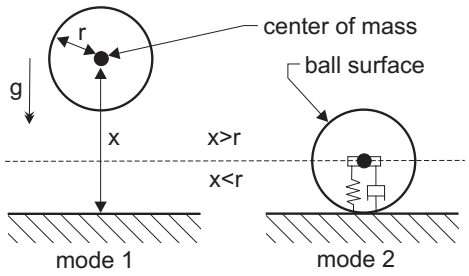

Figure 2.1: Bouncing ball: a simple hybrid system. This system is a canonical example of hybrid system, with two discrete modes: mode 1 is a free fall state and mode 2 is an elastic impact state.

This thesis studies the identification of hybrid system models from observed data. Because data collection with digital systems inherently creates discrete time observations of the system, all models in this thesis will be represented in discrete time. Due to the finite nature of data collection, only models with a finite number of discrete states will be considered. This leads to the formal definition of a discrete time finite state hybrid dynamical system (Def. 2.1), which is used to introduce terminology and the concepts behind hybrid system theory. Subsequent sections review three specific examples of hybrid systems: Markov Jump Systems in Section 2.1.1, Probabilistic Timed Automata in Section 2.1.2, and Piecewise Affine or Piecewise Auto Regressive Exogenous (PWARX) systems in Section 2.1.3.

Definition 2.1 (Discrete Time Hybrid Dynamical System)**.** *A discrete time finite state hybrid dynamical system $\mathcal{G}$ is a collection $\mathcal{G} = \{\mathcal{S}, \mathcal{X}, \mathcal{Y}, \mathcal{U}, F, H, \pi\}$, where:*

1. *$\mathcal{S} = \{S_1, S_2, ..., S_N\}$ is a set of* discrete states*, or* modes *of the system. The mode index of a system at time t_k is denoted m_k, and $m_k = i$ when S_i is the discrete state at t_k.*

2. *$\mathcal{U}$ is a set of input variables. The system input at t_k is denoted $u_k \in \mathcal{U}$.*

3. *$\mathcal{X} = \mathbb{R}^N$ is a set of continuous states, which at time t_k is denoted x_k. Associated with each mode s_i is a function $f_i(\cdot)$, which defines the evolution or flow of the continuous system state:*

$$x_k = f_i\left(x_{k-1}, u_{k-1}\right) \text{ if } m_k = i \ . \tag{2.1}$$

4. *$\mathcal{Y}$ is a set of outputs, denoted y_k, that depend on the continuous state and current discrete state. Associated with each discrete state s_i is an observation function g_i such that:*

$$y_k = g_i\left(x_k, u_k\right) \text{ if } m_k = i \ . \tag{2.2}$$

5. *H is a set of guard functions, $h_{ij}(x_k, m_k)$ for $i, j = 1, ..., N$. The system which is in mode S_i at time t_k, can transition to state S_j at t_{k+1} if the guard function h_{ij} is active:*

$$m_{k+1} = j, m_k = i \text{ if } h_{ij}(x_k, u_k) \geq 0 \ . \tag{2.3}$$

6. *F represents the set of dynamical and measurement functions f_i, g_i for $i = 1, ..., N$.*

7. *$\pi \in \mathcal{X} \times \mathcal{S}$ is a set of initial states or conditions of the system.*

$\diamond$

Remark 2.1. This definition highlights aspects of hybrid systems which are important in model creation and system identification. Another common aspect of hybrid systems that is not given in

the above definition is that of *reset* maps: If the system transitions from s_i to s_j at time t_k, then the continuous state can be reset to another point by the function $R(i, j, x_k) : S \times S \times X \to X$. If any of the functions $f(.), g(.), h(.), R(.)$ are stochastic, then the system is called a *stochastic hybrid system* [47]. ◇

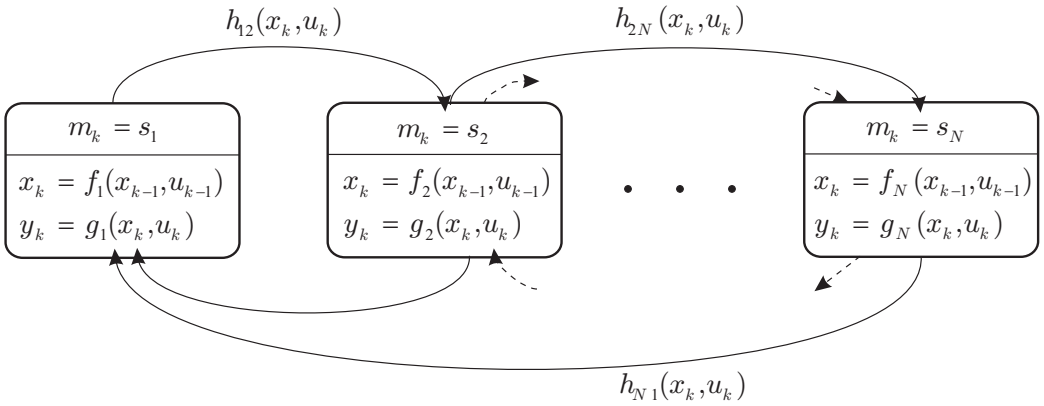

Figure 2.2: Finite graph representation of a discrete time finite state hybrid system with N discrete modes, as described in Definition 2.1

The point in utilizing a hybrid system model is that very complicated non-linear systems can often be broken down into a set of simple dynamical systems and a set of logical transition rules. This allows tractable development of control systems and verification procedures to be applied to the system. For certain classes of hybrid systems there already exist powerful toolboxes for optimal and model predictive control (HYSDEL, Hybrid Toolbox), and verification and reachability analysis (PHAVer, MATISSE), as well as many other developed toolkits.

While the problems of estimation and control of hybrid systems have been reasonably well explored, the problem of identification, or learning hybrid models from observed data, has not yet reached a point where it can be generally applied. Hybrid system identification has been a field of research for more than twenty-five years, starting with a specific class of Markov jump systems, a review of which was published by Tugnait [48] in 1982. This class of systems, where the discrete mode transitions are governed by a hidden Markov chain, is a model class that is still actively used in economics and control theory [49]. Inference in these system is still an open area of research, due to the combinatorial complexity inherent in identifying these types of systems from data. Since the year 2000, there have been several contributions to hybrid identification theory, with algorithms that identify Auto Regressive (AR) models with state-dependant mode switching. Notably in this class of systems are piecewise affine models. Timed automata are also used in the hybrid system community, and identification of similar model types exists in the machine learning and speech processing community (Section 3.5).

2.1.1 Markov-Based Probabilistic Hybrid Systems

The simplest probabilistic hybrid systems are based around a Markov transition kernel $A = \{a_{ij}\}$, with the probability of transition from mode s_i to mode s_j depending only on the current mode.

$$\mathrm{P}\big(m_{k+1} = j | m_k = i\big) = a_{ij}, \quad \text{where } \sum_{j=1}^{N} a_{ij} = 1, \text{ and } a_{ij} \geq 0, \ \forall i, j \tag{2.4}$$

This transition kernel (2.4) can be written in terms of a probabilistic guard function given in Def. 2.1, but it is considerably more convenient to treat the evolution of discrete state variable m_k as a Markov chain.

A large number of models previously treated in the probabilistic hybrid systems community incorporate stationary Markov transitions [47]. The Markov transition kernel A can be made to be a function of the current continuous state [50], giving more flexibility in the modeling process, but adds significant computational complexity even in estimation of the system state. Chapter 3 addresses a class of nonstationary Markov model which are functions of the full hybrid state (both m_k and x_k), as well as the duration spent in each mode. Identification algorithms for these types of systems are also presented, which are believed to be the only algorithms currently capable of being applied to these systems.

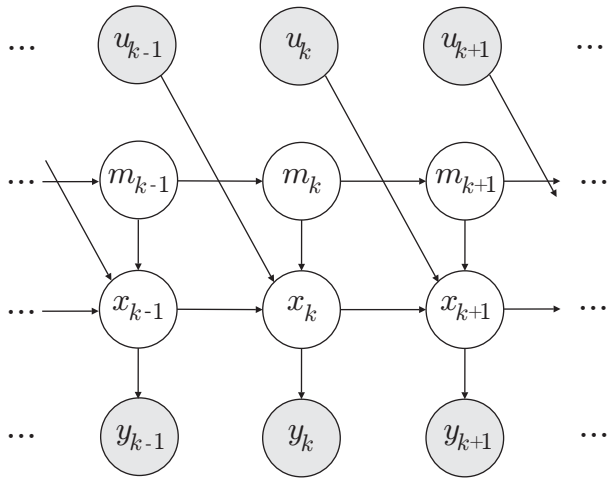

Figure 2.3: Directed acyclic graph representation of Markov Jump System, which gives the conditional dependence of system variables. Grey nodes represent observed data, in this case the system input u_k and the system output y_k. Typically the system mode m_k and the continuous state x_k are not directly observed or measured.

A Markov Jump System (MJS) is a hybrid system with a Markov transition kernel (2.4), with

linear state space dynamics associated with each mode:

$$x_k = A_{m_k} x_k + B_{m_k} u_k \tag{2.5}$$

$$y_k = C_{m_k} x_k \ , \tag{2.6}$$

where A_i, B_i, C_i are the parameters of the linear dynamics associated with the i^{th} mode. MJS are also known as switching linear systems, jump linear systems, and Markov jump linear systems. If the Markov transition kernel is relaxed to incorporate time or duration spent in each mode explicitly, these systems are known as segmental models.

MJS yield tractable estimation algorithms [51, 52] for inferring the hybrid state (x_k, m_k), and provides a framework that is tractable for developing feedback controllers [19]. Identification algorithms for these systems are derived from the field of hidden Markov models, such as the variational method by Ghahramani [53], and are utilized in the speech processing community.

Because the identification of a MJS is similar to that of hidden Markov models, the reader is referred to Section 2.3, and references within.

2.1.2 Probabilistic Timed Automata

In timed automata [20], the notion of time is captured by a finite set of real valued *clocks*. Each individual clock can be reset to zero upon certain transition of the automaton, in effect giving the duration, or time since the last reset. The transition of the timed automaton depend on constraints or guard functions of the current clock values. Probabilistic timed automata extend this formalism to allow nondeterminism in which mode to transition to and which clock is reset.

Timed automata with one or two clocks have been studied extensively [54], and have been used to verify the IEEE 802.11 Wireless protocol [55] and the IEEE 1394 root connection protocol.

Timed automata are inherently hybrid systems in that the complete system state includes information about the current discrete state as well as the clock values, which evolve in continuous time, albeit with simple deterministic behavior. Timed automata provide a more complete set of switching logic than Markov jump systems (or Markov decision processes [56]), in which durations are associated with transitions. In a discrete time setting, probabilistic timed automata with one clock are equivalent to nonstationary or semi-Markov chains (Section 3.5) [57].

Timed automata do not typically associate dynamics with each mode. The interesting aspect of timed automata is the temporal switching behavior, and this can be extended in a natural way by associating dynamics to each mode [21]. If one clock is used, this is also called a segment model [43], which is a superset of Markov jump systems. In Section 3.5, the identification of two related models is considered: nonstationary and semi-Markov models.

2.1.3 PWARX System

Piecewise autoregressive exogenous (PWARX) models are a subset of hybrid systems, and differ from MJS as the discrete mode transitions depend on the continuous system state and dynamics. This model class has been used extensively when identifying hybrid models from data. The linear dynamics in each mode are defined by an auto-regressive exogenous (ARX) process. ARX models are a powerful model for system identification as they avoid the inherent unidentifiability of state space models, caused by equivalence under unitary transforms, yet retain the same modeling fidelity [58]. A recent review paper on PWARX identification algorithms [8] gives the standard definition of this model, which is presented below in Def. 2.2.

PWARX models are equivalent to a wide range of hybrid systems, notably to piecewise affine (PWA) models [29], mixed logic dynamics (MLD) and linear complementarity (LC) frameworks [28]. This equivalence allows a wide range of developed control and verification tools to be used with PWARX models.

Definition 2.2 (Piecewise autoregressive exogenous (PWARX) system)**.** *A PWARX system takes the form:* $\mathcal{G} = \{\mathcal{S}, \mathcal{U}, \mathcal{X}, \mathcal{Y}, \Theta\}$ *where:*

1. $\mathcal{S} = \{S_1, \ldots, S_N\}$ *is a set of* N *discrete states, or modes of* $\mathcal{G}$*. The system mode* S_i *at time* t_k *is denoted by* $m_k = i$*. The hidden state sequence for all* $1 \leq k \leq T$*, is denoted* $m_{1:T} = \{m_1, m_2, ..., m_T\}$*.*

2. $\mathcal{U}$ *is a set of input variables. The system input at* t_k *is denoted* $u_k \in \mathcal{U}$*.*

3. *The regressor space* $\mathcal{X}$ *is partitioned into* N *disjoint convex polyhedral regions* $\{\chi_i\}_{i=1}^N$*, where* $\cup_{i=1}^N \chi_i = \mathcal{X}$ *and for* $i \neq j$*,* $\chi_i \cap \chi_j = 0$*. The mode of the model is determined by which region* χ_i *the regressor* x_k *is contained in. At step* k*, the model is said to be in mode* i *if*

$$m_k = i \quad \text{if } x_k \in \chi_i \ , \tag{2.7}$$

with the regressor x_k *defined as:*

$$x_k = [1, y_{k-1}, \ldots, y_{k-n_y}, u_{k-1}, \ldots, u_{k-n_u}]' \ \in \mathcal{X} \subset \mathbb{R}^{n_u+n_y+1} \ . \tag{2.8}$$

Furthermore, each polyhedral region is defined by a set of separating hyperplanes in the regressor space described by:

$$h_{ij}' x_k = 0 \ , \tag{2.9}$$

where h_{ij} *is a vector, such that for all* $x_k \in \chi_i$*,* $h_{ij}' x_k \leq 0$ *and for all* $x_k \in \chi_j$*,* $h_{ij}' x_k > 0$*. The collection of all hyperplanes defining each region* χ_i *is denoted* $H_i = \{h_{ij}\}_{j=1}^N$*.*

4. $y_k \in \mathcal{Y}$ *is the system output at* t_k*, and is given by the following piecewise ARX map, where* θ_{m_k} *are the parameters associated with the* m_k^{th} *mode:*

$$y_k = \theta_i^T x_k + e_k \quad \begin{cases} i = 1 \text{ if } x_k \in \chi_1 \\ \vdots \\ i = s \text{ if } x_k \in \chi_N \end{cases}, \tag{2.10}$$

where $e_k \sim \mathcal{N}(0, \sigma^2)$ *is the system noise.*

5. Θ *is the set of all parameters associated with the model, including all hyperplane sets* H_i *and AR parameters* θ_i*.*

$\diamond$

PWARX systems use the regressor state to create a static transition map (2.10), meaning the current system mode only depends on x_k, and not directly on the previous discrete mode m_{k-1}, as in Markov models. This transition map provides the ability to model a wide range of interesting phenomenon, such as guard rules of a bouncing ball (Fig. 2.1), power systems [59], and manufacturing processes [60]. In Section 3.7, this regressor-based switching behavior is incorporated into the Markov transition kernel, and gives greater flexibility in modeling hysteretic behavior.

There exist several different methods to identify PWARX models from data. Juloski [8] compares and summarizes four of these; a *Bayesian Approach* [7], a *Clustering Procedure* [59], the *Algebraic Geometric Approach*, [6], and the *Bounded-Error Procedure* [5]. Other approaches exist which utilize mixed-integer programming [4] and gradient-based identification routines [61]. The PWARX identification problem is summarized in Def. 2.3:

Definition 2.3 (PWARX identification problem)**.**

Given: *A system that is of PWARX form (Def. 2.2), with a known number of modes* N*, known sub-model ARX orders* n_y *and* n_u*, that are consistent throughout all sub-models, and a data set* $\{(y_k, x_k)\}_{k=1}^T$

Identify: *The parameters* $\{\theta_i\}_{i=1}^N$*, and the hyperplanes (or guardlines) that bound the polyhedral regions* $\{\chi_i\}_{i=1}^N$

$\diamond$

The identification problem in Def. 2.3 is known to be *NP-hard*, and in particular is worst case exponential in the number TN, the number of data points T times the number of modes N [4]. The only identification method that is guaranteed to find a the globally optimal solution is the mixed-integer programming approach, but is not useable for large data sets because of the complexity issue.

Two approaches are briefly outlined below: The Bayesian approach in Section 2.1.3.1 and the clustering procedure in Section 2.1.3.2. These approaches are mentioned for their ability to incorporate prior knowledge about the system, and effective performance in real data, respectively [8]. Intuition into the success of these methods will motivate the development of future algorithms in Section 3.7.

These methods do not consider the hyperplanes during in the initial identification process. A subsequent identification step of identifying the separating hyperplanes is applied after all the data is classified. While this is suboptimal, it allows the use of powerful existing techniques for guard line identification [59], such as: solving for $N(N-1)/2$ linear classifiers [24]; robust linear programming [62]; support vector machine with a linear kernel; multi-category support vector machines [63]. Another potential Bayesian method that may be more robust to outliers is relevance vector machines (see [24], and references within).

2.1.3.1 Sequential Bayesian Approach

The sequential Bayesian approach operates like a filter, classifying the the data pair (y_k, x_k) into a generating mode s_i, conditioned on the estimate of the model parameters $\{\theta_i\}_{i=1}^N$ at the current time step. After classifying the data pair, the model parameters are updated. Identification is completed in a single pass of the data, (y_k, x_k) for $k = 1, ..., T$. It is assumed that there is an informative *a priori* pdf of the model parameters θ_i, and the error e_k (2.10) is normally distributed with a known variance σ^2.

Following the notation of [7], the probability the data pair (y_k, x_k) is generated from mode s_i is expressed as a conditional density function[1]: $\mathrm{P}\big(m_k = i | y_k, x_k\big)$, and is computed using Bayes' theorem, hence the original name of the algorithm:

$$P(m_k = i | y_k, x_k) = \frac{P(y_k, x_k | m_k = i) P(m_k = i)}{\sum_{j=1}^{N} P(y_k, x_k | m_k = j) P(m_k = j)} \ . \tag{2.11}$$

The *prior* pdf $P(m_k = j)$ is equal to $\frac{1}{N}$, as it is assumed that there is no prior knowledge about the mode of the new data point. The conditional likelihood function of the data, $P(y_k, x_k | m_k = i)$ is evaluated by marginalizing over the current estimate of the parameters for mode s_i:

$$\mathrm{P}\big(y_k, x_k | m_k = i\big) = \int_{\Theta_i} \mathcal{N}(y_k - \theta_i^T x_k, \sigma^2) P_k(\theta_i) \mathrm{d}\theta_i \ , \tag{2.12}$$

where $P(y_k, x_k | \theta_i) = \mathcal{N}(\mu, \sigma^2)$ is a normal distribution with mean $\mu = y_k - \theta_i^T x_k$, and variance σ^2. This method assumes the variance is a fixed and known quantity. The probability density function of the parameters $P_k(\theta_i)$ is recomputed at every time step t_k. If a particular data point is assigned to

[1] In later section we will explicitly condition on the model class and model parameters θ_i, however the original notation used by Juloski is kept here

mode s_i, then this density is updated using Bayes' rule: $P(\theta_i|y_k, x_k)$ can be computed using Bayes Rule:

$$P_k(\theta_i|y_k, x_k) = \frac{P(y_k, x_k|\theta_i)P_{k-1}(\theta_i)}{\int_{\Theta_i} P(y_k, x_k|\theta_i)P_{k-1}(\theta_i)d\theta_i} \quad . \tag{2.13}$$

This sequential Bayesian method, which creates a estimate of the model parameters from a data set (y_k, x_k) for $k = 1, ..., T$, is summarized in Algorithm 2.1:

Algorithm 2.1 Sequential Bayesian Approach to Parameter Estimation

1: Initiate model with *a priori* parameter density functions $P_0(\theta_i)$, for $i = 1, ..., N$

2: **for** $k = 1$ to T **do**

3: use the data pair (y_k, x_k), compute the likelihood $P(m_k = i|y_k, x_k)$ using (2.11) and (2.12). The data pair is assigned to the mode with the highest likelihood:

$$m_k = \underset{i}{\operatorname{argmax}}\, P(m_k = i|y_k, x_k) \tag{2.14}$$

4: update the parameter pdf of θ_i if $m_k = i$, using (2.13). For all $\theta_j, j \neq i$, set $P_k(\theta_j) = P_{k-1}(\theta_j)$

5: **end for**

In the original method of Juloski [7], the PDFs (2.11) and (2.13) are computed using a particle filtering approach. After the parameters of each mode are estimated and the data points have been classified, the regressor space X is split into polyhedral regions using robust mixed linear programming.

This method requires two main approximations:

1. Data is assigned to a certain mode conditioned only on the observed data up to that time step.
2. Data is assigned in a maximum likelihood manner.

The effect of these assumptions gives the algorithm local convergence properties, and without accurate prior estimates of parameters algorithm may produce large errors [7]. In effect, a wrongly classified data point can never be reassigned and causes suboptimal behavior.

2.1.3.2 Clustering Procedure

The clustering technique [59] first classifies the data points into modes (*the clustering step*) and then splits the regressor state space into polyhedral regions. Optimal clustering is computationally hard so an efficient suboptimal "K-means" procedure is used.

The basis of the clustering algorithm is to group nearby (in a Euclidean sense) points into local data sets (LDs). The justification is that PWARX systems are locally linear, so small subsets of the data $\{y_k, x_k\}_{k=1}^{T}$, which are close to each other, are likely to belong to the same mode. For each

data point (y_k, x_k) a LDs designated $\mathcal{C}_k$ is constructed from the $c-1$ closest (in a Euclidean sense) neighbors, where c is an adjustable tuning parameter.

This process creates LDs of two types; *mixed* LDs, which contain points from more than one mode, and *pure* LDs, which contain data points from only a single mode.

After the LDs have been formed, a least-squares estimate of the parameters θ_k^{LS} based on the data assigned to each LDs $\mathcal{C}_k$ is calculated. Also associated with each LD is the *position* $\hat{x}_k$ of the LDs, which is the average of the regressors x_k contained in that LDs, and an empirical covariance matrices V_k of the parameters θ_k^{LS}, and covariance-like estimate Q_k of the the position.

A weighted K-means clustering algorithm is used to partition a *feature vector* $\xi_k = [\theta_k^{LS}, m_k]$, and hence assign the local data sets (and regressor points x_k) into modes. The weighting in the K-means algorithm incorporates the covariance information Q_k and V_k, with the aim that mixed LDs should have larger covariance matrices, and will then contribute less to the clustering of the feature vectors. (see [59] for more details)

The final step of the algorithm, done after the clustering step, is to estimate the subregions χ_i, using robust linear programming.

This technique utilizes the local linearity of PWARX systems. It adds structure to the identification problem by pre-grouping data points based on the prior knowledge that data points from the same mode should be close in some Euclidian sense. However it is noted that the algorithm is very sensitive to the order of the ARX submodels, and erratic behavior may occur if the order is not known exactly.

2.1.4 Motivation for Probabilistic Learning Algorithms

The literature on system identification in the hybrid systems community is limited to a small range of systems, primarily in PWARX models. Two identification algorithms were explicitly reviewed, the sequential Bayesian approach in Section 2.1.3.1 and the clustering approach in Section 2.1.3.2.

The clustering approach, a widely used and effective algorithm, gains advantage by exploiting the structure of the problem. This is achieved by an effective yet ad-hoc procedure of pre-grouping data points into local data sets. This approach is hampered however by a greedy k-means learning algorithm that is ineffective in dealing with model selection and errors in chosen model order.

The sequential Bayesian approach, seemingly less effective in practice [8], is able to introduce prior knowledge in a more structured way, but suffers from oversimplifying assumptions.

An ideal algorithm should be based on sound generalizable principles, incorporate structural knowledge about the problem, and be robust to the specified model class.

This thesis proposes that probabilistic learning algorithms, in particular Monte Carlo methods such as Gibbs sampling, and approximate deterministic methods such as variational Bayes should be used as a basis for algorithmic development. For the reader not already convinced of the advan-

tages to approaching the problem in a probabilistic framework, Section 2.2 reviews the probabilistic learning principles important for hybrid systems. In short, this class of methods is able to approach the problems of over-fitting, prior knowledge (Chapter 3) and model selection (Chapter 4) from a principled viewpoint.

This probabilistic approach has already been shown to be effective in Hybrid systems such as Markov jump systems [53], and hidden Markov models. Preliminary work [64, 65] has shown Gibbs sampling to be an natural and effective method for hybrid system identification.

2.2 Probabilistic Learning Algorithms and Latent Variables

The hybrid system identification problem of the previous section can be naturally recast as a probabilistic inference problem. Indeed the work of Juloski [7], in Section 2.1.3.1 approaches the problem in a probabilistic framework. This section introduces the foundations of probabilistic modeling for hybrid system identification, and emphasizes the key role played by latent variables in creating tractable inference algorithms. The approach used here has an inherently Bayesian viewpoint; knowledge of the model is used to create prior distributions, and these priors are updated with observed data to form posterior estimates of the model. Taking a Bayesian viewpoint in the creation of models has several advantages, including providing a rigorous approach to incorporating important prior knowledge about the system of interest in the model creation process. Additionally the use of prior density functions reduces model over-fitting, providing better generalization and predictive ability of the model. This inherent property of Bayesian inference is especially pronounced in cases where only a small amount of relevant data is available, and maximum likelihood methods can over-fit the data to such an extent that the model is unusable. In hybrid system identification, this problem can arise when identifying the transition rules between discrete modes of the system. Even in lengthy data sets, there may only be a few transitions between certain discrete modes, resulting in over-fitting when using maximum likelihood methods. Three different existing algorithms: variational Bayes, expectation maximization, and Gibbs sampling, are reviewed and compared in this section, before being utilized for hybrid system identification in Chapter 3.

All of the identification algorithms developed in this thesis will use Bayes' theorem (A.6) for inferring both model structure and parameter values. The *posterior distribution* $\mathrm{p}\big(\Theta|Y,\mathcal{M}\big)$ of the model parameters Θ, can be written as a function of the *likelihood*: $\mathrm{p}\big(Y|\Theta,\mathcal{M}\big)$, the *prior*: $\mathrm{p}\big(\Theta|\mathcal{M}\big)$ and *model evidence*: $\mathrm{p}\big(Y|\mathcal{M}\big)$ as follows:

$$\mathrm{p}\big(\Theta|Y,\mathcal{M}\big) = \frac{\mathrm{p}\big(Y|\Theta,\mathcal{M}\big)\,\mathrm{p}\big(\Theta|\mathcal{M}\big)}{\mathrm{p}\big(Y|\mathcal{M}\big)} \tag{2.15}$$

where $\mathcal{M}$ is the model class, Θ are the model parameters specified by the model class and Y is the

observed data used to update the model. The model class $\mathcal{M}$ defines the structure of the model, including the functional form, the number of parameters and any prior information associated with the model. Chapter 4 studies Bayesian model class selection in detail, giving an information theoretic interpretation the model evidence (2.15). In brief, Bayes' theorem can also used to select different model structures. Given a set of m possible model classes, $\{\mathcal{M}_1, ..., \mathcal{M}_m\}$, the posterior probability of each model can be determined by:

$$\mathrm{P}\big(\mathcal{M}_i|Y\big) = \frac{\mathrm{p}\big(Y|\mathcal{M}_i\big)\,\mathrm{P}\big(\mathcal{M}_i\big)}{\sum_{j=1}^{m}\mathrm{p}\big(Y|\mathcal{M}_j\big)\,\mathrm{P}\big(\mathcal{M}_j\big)} \ . \tag{2.16}$$

Remark 2.2. For the remainder of Chapters 2 and 3, the choice of model class $\mathcal{M}$ is not explicitly denoted, although it should be stressed that all distributions are conditioned on this information. When required, such as when choosing between different models in Chapter 4, model class conditioning will made explicit. ◇

The difficulty in applying Bayesian inference is due to the complexity of evaluating the model evidence $\mathrm{p}\big(Y|\mathcal{M}\big)$. In the hybrid system models of interest (Section 2.1), there is an additional problem of having to classify the observed data, a problem referred to as latent variable modeling.

Latent variable models or *incomplete data* models [9] treat the classification of observed data as a set of extra variables. In the hybrid system models of Section 2.1, the process of classifying the observed data $Y = \{y_k\}_{k=1}^{T}$, amounts to inferring the discrete state or mode m_k from which that data was generated. In the hybrid system case the discrete modes $\{m_k\}_{k=1}^{T}$ are referred to as latent variables.

The term "incomplete data" is derived from the fact that the likelihood of the observed data or *incomplete-data likelihood* $\mathrm{p}\big(Y|\Theta\big)$ cannot be evaluated without the latent variables. Instead, the *complete-data likelihood* $\mathrm{p}\big(Y, Z|\Theta\big)$ or the likelihood of both the observed data and the latent variables Z is typically specified by the model. The incomplete data likelihood is the marginal of the complete data likelihood:

$$\mathrm{p}\big(Y|\Theta\big) = \sum_{Z}\mathrm{p}\big(Y, Z|\Theta\big) \ . \tag{2.17}$$

This marginalization (2.17) involves summing over all possible combinations of latent variables, which results in a combinatorial explosion of the number of operations required to evaluate the incomplete data likelihood. Approximate inference schemes are therefore required to infer the model's posterior distribution.

Several algorithms have been utilized for inference in latent variable models including expectation maximization (EM) [9], Gibbs sampling (Gibbs) [25], and variational inference or variational Bayes (VB) [23, 36]. A significant body of work applies these algorithms to hidden Markov models and their extensions [38, 66, 10, 67, 32, 37]. One of the main observation in this thesis is that by considering

hybrid systems as extensions to hidden Markov models, all of these algorithms can be applied for inference in hybrid systems, as described in Section 3.

The remainder of this section gives an overview of each the EM, Gibbs and VB inference methods, presented in such a way to compare the differences and strengths of between each. The preference choosing one method should be based on the amount of data on hand, what approximations are reasonable to apply, and the amount of computational resources at hand. In brief:

- EM provides a local, efficient, point estimate to the posterior probability of model parameters, and excels when ample data creates a concentrated peaked posterior probability mass.
- Gibbs sampling provides a set of samples from the actual posterior distribution, giving more insight into the model, but it is computationally expensive with large data sets.
- Variational Bayes provides a medium between these methods, giving greater flexibility in estimating the posterior density of the model then EM, while retaining computational efficiency.

For the set of problems considered in this thesis, variational methods were found to be most suited because they are efficient enough to handle large amounts of observed data, while providing sufficiently descriptive posterior estimates to facilitate effective model selection and avoid over-fitting. Chapter 4 provides several case studies that demonstrate the advantage of the Gibbs sampling and VB approaches in calculating the model evidence and subsequent model selection problems.

While all of these methods provide different approximations to the posterior, the underlying implementation of the methods remain surprisingly similar. The next three sections will illustrate how inference in the posterior distribution of the model $\mathrm{p}(\Theta|Y)$ can be achieved by applying successive sub-model *identification* and data *classification* steps:

$$\text{Data Classification:} \qquad \mathrm{P}(Z|\Theta, Y) \tag{2.18}$$

$$\text{Identification:} \qquad \mathrm{p}(\Theta|Z, Y) \tag{2.19}$$

The next three subsections will demonstrate how each of the algorithms are used to sequentially estimate the above distributions. First the Variational Bayes method is derived in a general form and then applied specifically to approximating latent variable models. The EM algorithm will then be derived as a special case of the variational method, and includes reference back to more traditional theoretical foundations. Finally Gibbs sampling is introduced, and a qualitative comparison between the three methods is given.

2.2.1 Variational Bayesian Approximations (VB)

Variational inference or Variational Bayes (VB) is an approximate inference scheme that seeks to minimize the Kullback-Liebler (KL) divergence between a restricted family of functions of the model

parameters, and the actual posterior distribution of the model parameters [24]. There are two main attractions for using this method: the algorithm is efficient, with computation time similar to expectation maximization. As variational methods seek to fit a function of the model parameters and not just a point estimate of the parameters, over-fitting is avoided and the functions can be used for improved model selection (see Chapter 4).

Variational methods have been applied to a range of problems in Bayesian inference, including hidden Markov models [35], general graphical models with incomplete data [23], mixture models [68], Markov jump systems [53], and are also referred to as *ensemble methods*.

This section will start with a general introduction to VB, following the derivations in the text by Bishop [24], before returning to the specific application of inference in hybrid systems and latent variable models.

To establish a general variational framework, it is assumed the model is parameterized by a set Φ. In the case of hybrid systems and latent variable models this parameter set includes both the model parameters and the latent variables: $\Phi = \{\Theta, Z\}$. Here is it assumed that the selected model class defines the joint density of the parameters Φ and the observations: $\mathrm{P}\big(Y, \Phi\big)$. The probability of the data (model evidence) is then the marginal of this joint density: $p(Y) = \int_\Phi p(Y, \Phi) d\Phi$.

Proposition 2.1. [24] Given any distribution $q(\Phi)$, the following decomposition holds:

$$\log p(Y) = \mathcal{L}(q(\Phi)) + KL\big(q(\Phi)||p\left(\Phi|Y\right)\big) \ ,$$

where

$$\mathcal{L}(q(\Phi)) = \int_\Phi q(\Phi) \log \frac{p(Y, \Phi)}{q(\Phi)} d\Phi \tag{2.20}$$

$$KL(q(\Phi)||p\left(\Phi|Y\right)) = -\int_\Phi q(\Phi) \log \frac{p\left(\Phi|Y\right)}{q(\Phi)} d\Phi \ . \tag{2.21}$$

$\diamond$

Proof. By direct substitution of $p(Y, \Phi) = p\left(\Phi|Y\right) p(Y)$ into (2.20). $\square$

The variational method aims to maximize the lower bound $\mathcal{L}(q(\Phi))$ (2.20), hence minimizing the KL divergence (2.21) between the parameter posterior and the distribution $q(\Phi)$. This lower bound is the negative of the variational free energy, more typically used in physics [35]. Note that $KL(q(\Phi)||p\left(\Phi|Y\right)) \geq 0$, with the equality holding if and only if $q(\Phi) = p\left(\Phi|Y\right)$, a theorem called *Gibbs' inequality*.

The maximization of the lower bound is typically not tractable for arbitrary functions $q(\Phi)$. Instead the form of $q(\Phi)$ is restricted to a specific family of distributions, and the maximization procedure will find the member of this family for which the KL divergence is minimized. The

restriction of the form of $q(\Phi)$ is problem dependent, and should be made as general as possible to provide a good approximation. In general a factorial form of $q(\Phi)$ will be convenient to consider, where given a disjoint partition of $\Phi = \{\Phi_1, ..., \Phi_m\}$:

$$q(\Phi) = \prod_{i=1}^{M} q(\Phi_i) \ . \tag{2.22}$$

In the case of the hybrid system and latent variable models, this approximation is realized by factorizing the parameters and latent variables: $q(\Theta, Z) = q(\Theta)q(Z)$. Other than this assumption there are no additional restrictions on the distributions $q(\Phi)$.

The variational method now seeks to find the distributions $q(\Phi)$, of the form (2.22), which maximize the lower bound (2.20). Substituting (2.22) into (2.20) yields:

$$\mathcal{L}(q(\Phi)) = \int \prod_{i=1}^{M} q(\Phi_i) \left[\log p(Y, \Phi) - \log \left(\prod_{i=1}^{M} q(\Phi_i) \right) \right] d\Phi \ . \tag{2.23}$$

Considering the lower bound as a function of the factor Φ_j, and defining $\Phi_{i\neq j} = \{\Phi_1, ..., \Phi_M\} / \{\Phi_j\}$, and $q(\Phi_{i\neq j}) = \prod_{i\neq j} q(\Phi_i)$ the bound is written:

$$\mathcal{L}(q(\Phi_j)) = \int q(\Phi_j) \left[\int q(\Phi_{i\neq j}) \log p(Y, \Phi) d\Phi_{i\neq j} \right] d\Phi_j - \int \prod_{i=1}^{M} q(\Phi_i) \log \left(\prod_{i=1}^{M} q(\Phi_i) \right) d\Phi \ . \tag{2.24}$$

Proposition 2.2 ([24] Optimal Solution for Φ_j)**.** The optimal solution for $q(\Phi_j)$, which maximizes $\mathcal{L}(q(\Phi_j))$ is:

$$\log q^{\star}(\Phi_j) = \int q(\Phi_{i\neq j}) \log p(Y, \Phi) d\Phi_{i\neq j} + const \ . \tag{2.25}$$

◇

Proof. Note that the second term of (2.24) can be written as the entropy of Φ_j, plus a constant term C which is equal to the entropy of $q(\Phi_{i\neq j})$:

$$\mathcal{L}(q(\Phi_j)) = \int q(\Phi_j) \left[\int q(\Phi_{i\neq j}) \log p(\Phi, Y) d\Phi_{i\neq j} \right] d\Phi_j - \int q(\Phi_j) \log(\Phi_j) d\Phi_j + C \ .$$

This equation can then be rewritten as the negative of a KL-divergence:

$$\mathcal{L}(q(\Phi_j)) = \int q(\Phi_j) \log \frac{\exp\left[\int q(\Phi_{i\neq j}) \log p(\Phi, Y) d\Phi_{i\neq j}\right]}{q(\Phi_j)} d\Phi_j + C \ .$$

Using Gibbs' inequality, the above equation is then maximized with the proposed optimal solution (2.25). □

The optimal solution (2.25) for each $i = 1, ..., M$ provides a sequence of equations whose solution

will increase the lower bound (Algorithm 2.2). A general method to increase the lower bound starting with a initial distributions $q(\Phi_i)$ for $i = 1 : m$, is to sequentially update each Φ_i using the optimal solution. Convergence is guaranteed as the lower bound is convex in each of the factors [69].

Algorithm 2.2 General Variational Bayes Algorithm

1: Choose initial distributions: Φ_i, for $i = 1 : m$
2: **while** $\Delta\mathcal{L}(q) >$ tol **do**
3: **for** i=1:m **do**
4: $q(\Phi_i) \propto \exp\left[\int q(\Phi_{i\neq j}) \log p(Y, \Phi) d\Phi_{i\neq j}\right]$
5: **end for**
6: $\Delta\mathcal{L}(q) =$ change in lower bound $\mathcal{L}(q)$ from last step
7: **end while**

Returning to the notation of the hybrid system and latent variable models (Section 2.2), the parameter set is written $\Phi = \{\Theta, Z\}$, where Z refers to the latent variables, and Θ the system parameters. To make the latent variable problem tractable, the factorization assumption (2.22) becomes:

$$q(Z, \Theta) = q(Z)q(\Theta) \ . \tag{2.26}$$

This approximation allows the problem to be split up into classification (estimation of the latent variables), and parameter identification problems. In the case of the hybrid system and latent variable models, it was assumed that the complete data likelihood $\mathrm{p}\big(Y, Z|\Theta\big)$ is specified by the model. This requires using decomposition $\mathrm{p}\big(Y, Z, \Theta\big) = \mathrm{p}\big(Y, Z|\Theta\big)\,\mathrm{p}\big(\Theta\big)$. It then follows from (2.25) that the following recursion relations maximize the lower bound:

$$q(\Theta) = \frac{1}{C_\Theta} \exp\left[\sum_Z q(Z) \log \mathrm{p}\big(Y, Z|\Theta\big)\right] \mathrm{p}\big(\Theta\big) \tag{2.27}$$

$$q(Z) = \frac{1}{C_Z} \exp\left[\int q(\Theta) \log \mathrm{p}\big(Y, Z|\Theta\big) d\Theta\right] \ , \tag{2.28}$$

where C_Θ and C_Z are normalizing constants. Equations (2.27) and (2.28), make use of the realizations that $\sum_Z q(Z) \log \mathrm{p}\big(\Theta\big) = \log \mathrm{p}\big(\Theta\big)$, and that $\int q(\Theta) \log \mathrm{p}\big(\Theta\big) d\Theta$ is not a function of Z. The above steps are again in the form of parameter identification and data classification steps (recall equations (2.18) and (2.19)). Furthermore, in the hybrid system identification problems of interest, it is shown that (2.28) and (2.27) can be solved efficiently. The effect of the factorization assumption is discussed in Section 4.2.

2.2.2 Expectation Maximization (EM)

This section on the EM algorithm is presented to allow a comparison of EM and VB methods. The reader is assumed to be familiar with traditional derivations of EM such as in the text by McLachlan and Krishnan [70], or the original formulation by Dempster [9]. This section is brief, as all derivations follow as a special case of VB in Section 2.2.1.

Using the latent variable and hybrid system models as motivation, the same terminology of latent variables Z, parameters Θ, and observations Y is used. The major difference from the VB section is that the EM algorithm was originally designed to find the maximum of the likelihood function: $\mathrm{p}\big(Y|\Theta\big)$. Furthermore the EM method finds a maximum point estimate of Θ. As previously discussed the incomplete data likelihood is the marginal of the complete data likelihood: $\mathrm{p}\big(Y|\Theta\big) = \sum_Z \mathrm{p}\big(Y,Z|\Theta\big)$.

As in the VB section, a distribution $q(Z)$ is introduced over the latent variables. A similar decomposition holds:

$$\log p(Y) = \mathcal{L}(q(Z),\Theta) + KL\big(q||p\big) \ , \tag{2.29}$$

where

$$\mathcal{L}(q(Z),\Theta) = \sum_Z q(Z) \log \frac{\mathrm{p}\big(Y,Z|\Theta\big)}{q(Z)} \tag{2.30}$$

$$KL(q||p) = -\sum_Z q(Z) \log \frac{p\left(Z|Y,\Theta\right)}{q(Z)} \ . \tag{2.31}$$

The argument for maximizing the lower bound remains the same as in VB, and the EM algorithm now reduces to a two-stage sequential optimization technique: the *M-step* of maximizing the lower bound with respect to the parameters Θ, and *E-step* of maximizing the lower bound with respect to the latent variable distribution $q(Z)$.

The E-step in the EM algorithm then maximizes the lower bound, given a value of Θ, which we denote Θ^{old}. Using the decomposition $\log \mathrm{p}\big(Y,Z|\Theta\big) = \log \mathrm{P}\big(Z|Y,\Theta\big) + \log \mathrm{p}\big(Y|\Theta\big)$, the lower bound is written as the KL divergence:

$$\mathcal{L}(q(Z),\Theta^{old}) = \sum_Z q(Z) \log \frac{\mathrm{p}\big(Z|Y,\Theta^{old}\big)}{q(Z)} - \sum_Z q(Z)\, \mathrm{p}\big(Y|\Theta^{old}\big) \ . \tag{2.32}$$

Because the second term is a constant, the E-step is equivalent to obtaining:

$$\textbf{E-step:} \quad q(Z) = \mathrm{p}\big(Z|Y,\Theta^{old}\big). \tag{2.33}$$

The M-step in the EM algorithm maximizes the lower bound for a given function of $q(Z)$. After the

E-step substituting in $q(Z) = \mathrm{p}\big(Z|Y, \Theta^{old}\big)$ the lower bound is therefore:

$$\mathcal{L}(q(Z), \Theta) = \sum_Z \mathrm{p}\big(Z|Y, \Theta^{old}\big) \log \mathrm{p}\big(Y, Z|\Theta\big) - \sum_Z \mathrm{p}\big(Z|Y, \Theta^{old}\big) \log \mathrm{p}\big(Z|Y, \Theta^{old}\big) \ . \tag{2.34}$$

By just considering the terms in the lower bound that are a function of Θ, this is the same as maximizing $Q(\Theta, \Theta^{old})$, where:

$$Q(\Theta, \Theta^{old}) = \sum_Z \mathrm{p}\big(Z|Y, \Theta^{old}\big) \log \mathrm{p}\big(Y, Z|\Theta\big) \tag{2.35}$$

or explicity in terms of the variable θ

$$\textbf{M-step : } \Theta = \underset{\Theta}{\operatorname{argmax}} \sum_Z \mathrm{p}\big(Z|Y, \Theta^{old}\big) \log \mathrm{p}\big(Y, Z|\Theta\big) \ . \tag{2.36}$$

Note that the above function $Q(\Theta, \Theta^{old})$ is the same auxiliary function used in the original EM derivations. If maximum a posteriori (MAP) learning is undertaken using EM, the goal of this optimization procedure changes to maximizing $\mathrm{p}\big(\Theta|Y\big)$ instead of $\mathrm{p}\big(Y|\Theta\big)$. Using Bayes' rule, $\log \mathrm{p}\big(\Theta|Y\big) = \log \mathrm{p}\big(Y|\Theta\big) + \log \mathrm{p}\big(\Theta\big) - \log \mathrm{p}\big(Y\big)$, then substituting the decomposition (2.29) results in:

$$\log \mathrm{p}\big(\Theta|Y\big) = \mathcal{L}(q(Z), \Theta) + KL\big(q||p\big) + \log \mathrm{p}\big(\Theta\big) - \log \mathrm{p}\big(Y\big) \ . \tag{2.37}$$

Now instead of just maximizing the lower bound $\mathcal{L}(q(Z), \Theta)$, both the combination of the lower bound and the prior is maximized: $\mathcal{L}(q(Z), \Theta) + \log \mathrm{p}\big(\Theta\big)$. This has no effect of the E-step of the algorithm, as $\log \mathrm{p}\big(\Theta\big)$ is not a function of Z. The M-step (for MAP) is simply modified by now maximizing:

$$Q(\Theta, \Theta^{old}) = \sum_Z \mathrm{p}\big(Z|Y, \Theta^{old}\big) \log \mathrm{p}\big(Y, Z|\Theta\big) + \log \mathrm{p}\big(\Theta\big) \ . \tag{2.38}$$

In the MAP case this modifies the M step to:

$$\textbf{M-step : } \Theta = \underset{\Theta}{\operatorname{argmax}} \sum_Z \mathrm{p}\big(Z|Y, \Theta^{old}\big) \log \mathrm{p}\big(Y, Z|\Theta\big) + \log \mathrm{p}\big(\Theta\big) \tag{2.39}$$

While the EM algorithm is widely used, it is seen here to be a subcase of the VB algorithm.

2.2.3 Gibbs Sampling

This section introduces both the multi-stage Gibbs sampler, and a subcase called the two-stage Gibbs sampler. Both of these Markov chain Monte Carlo methods provide many optimality properties, but the distinction is made because the two-stage Gibbs sampler is immediately applicable in

identification of hybrid systems and latent variable models, and also provides additional theoretical convergence properties.

This section gives a brief general overview of the Gibbs sampler, and then states some convergence properties before applying the sampler to latent variable models. An excellent reference for a complete review and derivation of the Gibbs sampler properties Robert and Casella [25], in particular the chapter called 'The Two-Stage Gibbs Sampler'.

For the multi-stage Gibbs sampler, it is assumed the model is parameterized by a set Φ, which can be written as $\Phi = \{\Phi_1, ..., \Phi_m\}$, where the Φ_is are either one or multidimensional. The goal of Gibbs sampling is then to simulate from, or produce a set of samples which are distributed like, the joint distribution $\mathrm{p}\big(\Phi_1, ..., \Phi_m\big)$. A set of N samples is denoted $\{\hat{\Phi}^{(t)}\}_{t=1}^{N}$, where the t^{th} sample $\hat{\Phi}^{(t)} \triangleq \hat{\Phi}_1^{(t)}, ..., \hat{\Phi}_m^{(t)}$ from the joint distribution is denoted by:

$$\hat{\Phi}_1^{(t)}, ..., \hat{\Phi}_m^{(t)} \sim \mathrm{p}\big(\Phi_1, ..., \Phi_m\big) \ , \tag{2.40}$$

where the symbol $\sim$ is defined as *distributed as*.

Drawing samples from the joint distribution (2.40) is often intractable, and instead it is assumed that the corresponding conditional distributions, $\mathrm{p}\big(\Phi_i|\Phi_1, \Phi_2, ..., \Phi_{i-1}, \Phi_{i+1}, ..., \Phi_M\big)$ for $i = 1, ..., m$ can be sampled from.

Gibbs sampling works by utilizing the structure of the model to simplify the problem of simulating from the joint distribution $\mathrm{p}\big(\Phi_1, ..., \Phi_m\big)$. A multi-stage Gibbs sampler used to draw N samples from the joint distribution is given in Algorithm 2.3.

Algorithm 2.3 Multi-Stage Gibbs Sampler: draw N samples from $\mathrm{p}\big(\Phi\big)$

1: Start with initial samples: $\hat{\Phi}^{(0)} = \hat{\Phi}_1^{(0)}, ..., \hat{\Phi}_m^{(0)}$
2: **for** t=0,...,N **do**
3: $\quad \hat{\Phi}_1^{(t+1)} \sim \mathrm{p}\big(\Phi_1|\hat{\Phi}_2^{(t)}, ..., \hat{\Phi}_m^{(t)}\big)$
4: $\quad \hat{\Phi}_2^{(t+1)} \sim \mathrm{p}\big(\Phi_2|\hat{\Phi}_1^{(t+1)}, \hat{\Phi}_3^{(t)} ..., \hat{\Phi}_m^{(t)}\big)$
5: $\quad \vdots$
6: $\quad \hat{\Phi}_i^{(t+1)} \sim \mathrm{p}\big(\Phi_i|\hat{\Phi}_1^{(t)}, \hat{\Phi}_2^{(t)}, ..., \hat{\Phi}_{i-1}^{(t)}, \hat{\Phi}_{i+1}^{(t)}, ..., \hat{\Phi}_M^{(t)}\big)$
7: $\quad \vdots$
8: $\quad \hat{\Phi}_m^{(t+1)} \sim \mathrm{p}\big(\Phi_m|\hat{\Phi}_2^{(t+1)}, ..., \hat{\Phi}_{m-1}^{(t+1)}\big)$
9: **end for**

A significant advantage of Gibbs sampling is that only the conditional distributions are required for simulation. Since even in high-dimensional problems the individual conditional distributions are typically of low dimension, the complexity of the problem is reduced.

As mentioned, the Gibbs sampler is a Markov Chain Monte Carlo method; by construction of

Algorithm 2.3, $(\hat{\Phi}^{(t)})$ is a Markov chain. That is $(\hat{\Phi}^{(t)})$ is a function only of $(\hat{\Phi}^{(t-1)})$ as defined by Alg. 2.3.

Convergence in MCMC methods has a very specific meaning, simply that the Markov chain $(\hat{\Phi}^{(t)})$ has a unique stationary distribution which is equal to the joint distribution $\mathrm{p}\big(\Phi\big)$. In much more obvious terms, convergence implies that the average:

$$\frac{1}{N}\sum_{t=1}^{N} f(\Phi^{(t)}) \ , \tag{2.41}$$

converges to the expectation $\int f(\Phi)\,\mathrm{p}\big(\Phi\big)$ *almost surely* [25], as $N \to \infty$ for every bounded function f. Furthermore, the Markov chain can be shown to converge to the stationary distribution from any point in the domain of the distribution, meaning practically that Alg. 2.3 can be initialized with any reasonable starting point $\hat{\Phi}^{(0)}$.

Proving convergence for the Gibbs sampler is a lengthy process, but as stated in Robert and Casella [25], most decompositions (Alg. 2.3) satisfy the convergence requirements. Of more practical importance than proving convergence will occur, is diagnosing when the Markov chain has converged. Specifically, deciding when enough samples N have been drawn is not trivial. Diagnosing convergence usually involves either visually inspecting the sampler output, or running multiple Gibbs samplers (and hence chains $(\Phi^{(t)})$) and comparing the variance within each chain to the variance between chains. The specific convergence diagnostic tools used in the rest of this thesis will be presented with the specific algorithms developed for individual models.

The two-stage Gibbs sampler is a subcase of the multi-stage Gibbs sampler. It is of particular interest in the hybrid system and latent variable inference problems, as it can reduce inference of the joint density function $\mathrm{p}\big(\theta, Z|Y\big)$ into the component parts of parameter estimation and data association:

Algorithm 2.4 Two-Stage Gibbs Sampler: draw N samples from $\mathrm{p}\big(\theta, Z|Y\big)$

1: Start with initial samples: $\hat{\Theta}^{(0)}$
2: **for** t=0,...,N **do**
3: $\quad \hat{Z}^{(t+1)} \sim \mathrm{p}\big(Z|\hat{\Theta}^{(t)}, Y\big)$
4: $\quad \hat{\Theta}^{(t+1)} \sim \mathrm{p}\big(\Theta|\hat{Z}^{(t+1)}, Y\big)$
5: **end for**

Note that the samples $\Theta^{(t)}$ from Algorithm 2.4 are also distributed like the marginal probability [25]:

$$\hat{\Theta}^{(t+1)} \sim \mathrm{p}\big(\Theta|Y\big) = \sum_{Z} \mathrm{p}\big(\Theta, Z|Y\big) \ . \tag{2.42}$$

An important property of Algorithm 2.4 is that not only do the joint samples $(\hat{Z}^{(t+1)}, \hat{\Theta}^{(t+1)})$

form a Markov chain, but also each sequence $(\hat{Z}^{(t+1)})$ and $(\hat{\Theta}^{(t+1)})$ is a Markov chain. First consider the stationary Markov transition kernel produced by Algorithm (2.4) over the joint variables:

$$K\left((Z^\star,\Theta^\star)|(Z,\Theta)\right) = \mathrm{p}\big(\Theta^\star|Z^\star,Y\big)\,\mathrm{p}\big(Z^\star|\Theta,Y\big) \ . \tag{2.43}$$

As stated in [25] the kernel (2.43) has a stationary distribution of $\mathrm{p}\big(\Theta,Z|Y\big)$. Furthermore Algorithm 2.4 produces the subchain $(\hat{\Theta}^{(t)})$ that has the transition kernel:

$$K\left(\Theta^\star|\Theta\right) = \sum_Z \mathrm{p}\big(\Theta^\star|Z^\star,Y\big)\,\mathrm{p}\big(Z^\star|\Theta,Y\big) \ . \tag{2.44}$$

2.3 Hidden Markov Models (HMM)

This section gives a formal definition of the hidden Markov model (HMM), and reviews HMM inference using the EM and VB algorithms in Sections 2.3.1 and 2.3.2 . In brief, a HMM is a discrete Markov chain with an *emission* distribution associated with each state that generates the observed output. The term *hidden* is used to denote that the discrete state is not observed. HMMs have a long history in computer science, especially in speech processing [10]. The definition of a HMM (Def. 2.4) is as follows:

Definition 2.4 (Hidden Markov Model)**.** *A HMM is a system $\mathcal{G} = \{\mathcal{S},\mathcal{Y},\Theta\}$, where:*

1. *$\mathcal{S} = \{S_1,\ldots,S_N\}$ is a set of N discrete states, or modes of $\mathcal{G}$. The system mode S_i at time t_k is denoted by $m_k = i$. The hidden state sequence for all $1 \le t_k \le T$, is denoted $m_{1:T} = \{m_1, m_2, ..., m_T\}$.*

2. *The evolution of the discrete states m_k over $\mathcal{S}$ is described by a first order Markov chain, where the system state m_k at time t_k depends only on the state m_{k-1} at time t_{k-1}:*

$$\mathrm{P}\big(m_k = j|m_{k-1} = i\big) = a_{ij} \ . \tag{2.45}$$

   The state transitions are completely described by the transition kernel matrix $A = [a_{ij}]$, where:

$$a_{ij} \ge 0, \ \textit{and} \ \sum_{j=1}^{N} a_{ij} = 1 \ . \tag{2.46}$$

3. *$y_k \in \mathcal{Y}$ is the system output at t_k. The observations are independent when conditioned on the discrete state, and are completely defined by the parameterized distribution $\mathrm{p}\big(y_k|m_k,\theta_{m_k}\big)$, where θ_{m_k} are the parameters associated with the m_k^{th} mode.*

4. *The initial condition $\pi \in \mathbb{R}^N$, or the discrete state at t_0 is defined as: $\pi(i) = \mathrm{P}\big(m_0 = i\big)$.*

5. *The set of all system parameters is denoted* Θ. *This includes the transition parameters* $A = [a_{ij}]$, *the initial condition* π, *and the parameters of the conditional observation densities* $\theta_{m_k}, m_k = 1, .., N$.

$\diamond$

Remark 2.3. Note that Def. 2.4 of a HMM does not specify a specific form of the observation densities $\mathrm{p}\big(y_k|m_k, \theta_{m_k}\big)$. In the classical HMM formulation [10], the observation space $\mathcal{Y}$ consists of a finite discrete set of n symbols $V = \{v_1, \ldots, v_n\}$, where each symbol v_i represents a possible observation of the system being modeled. For instance, in a coin toss experiment, the observation symbols v_i are simply the observations the coin landing with either the head or tails side facing up. In this situation the conditional observation distribution $\mathrm{p}\big(y_k|m_k\big)$ is typically modeled as a multinominal distribution. It should be noted that the observation space $\mathcal{Y}$ can be arbitrarily defined, with the only requirement that the distribution $\mathrm{p}\big(y_k|m_k\big)$ can be specified. A common choice for modeling continuous system outputs $\mathcal{Y} = \mathbb{R}^n$ is the Gaussian distribution. The distinction between a HMM with discrete or continuous observation spaces is not typically made in the literature, so the above definition is consistent. $\diamond$

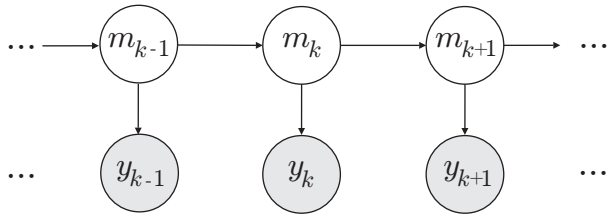

Figure 2.4: Directed acyclic graph representation of a hidden Markov model. Arrows represent the conditional dependence of system variables. Grey nodes represent observed variables and white nodes represent hidden variables.

The joint likelihood of the *complete-data likelihood*, or the likelihood of both the observed outputs $y_{1:T}$ and the hidden or latent states $m_{1:T}$ of a sequence of length T is given by:

$$\mathrm{p}\big(m_{1:T}, y_{1:T}|\Theta\big) = \mathrm{P}\big(m_1|\pi\big)\,\mathrm{p}\big(y_1|m_1, \theta_{m_1}\big)\prod_{k=2}^{T}\mathrm{P}\big(m_k|m_{k-1}, A\big)\,\mathrm{p}\big(y_k|m_k, \theta_{m_k}\big)\ . \tag{2.47}$$

The Markov chain transition kernel in a HMM implicity defines a probability distribution of state duration $\mathrm{P}\big(d\big)$. This is illustrated by the question: given that the model is in state S_i at time t_k, what is the probability that the model will stay in state S_i for d time steps? This probability can be evaluated as:

$$\mathrm{P}\big(m_{k+1:k+d-1} = i, m_{k+d} \neq i|m_k = i\big) = (a_{ii})^{d-1}\,(1 - a_{ii}) = \mathrm{P}\big(d\big)\ ,$$

which is a Geometric distribution.

For the remainder of this section we will refer to a hidden Markov model as a system with a finite discrete observation set. The probabilities $\mathrm{P}\big(y_k|m_k=i\big)$ are the *emission* probabilities for each of n symbols at each state:

$$\theta_i(v) \triangleq \mathrm{P}\big(y_k=v|m_k=i\big), \quad \text{where } \sum_{v=1}^{n}\theta_i(v)=1 \text{ and } 0\leq\theta_i(v)\leq 1 \forall i,v$$

As discussed in Section 2.2, there is considerable merit in specifying priors on a model, and estimating the posterior distributions of models. In the majority of this thesis, there is an underlying assumption that in choosing a model, the form and parameters of the prior of that model are specified. In the above definition of a hidden Markov model there is no explicit formulation for prior distributions, just as the form of the observation density is not specified.

We will now define a suitable prior distribution over the parameters of a hidden Markov model. These priors are typically used in the literature [35, 36], as they provide tractable posterior distributions:

$$\mathrm{p}\big(\Theta\big) = \mathrm{p}\big(\pi\big)\,\mathrm{p}\big(A\big)\,\mathrm{p}\big(\theta_1\big)...\,\mathrm{p}\big(\theta_N\big) \tag{2.48}$$

$$\mathrm{p}\big(\pi\big) = \mathrm{Dir}\big(\pi_1,...,\pi_N|\pi_1^0,...,\pi_N^0\big) \tag{2.49}$$

$$\mathrm{p}\big(\theta_i\big) = \mathrm{Dir}\big(\theta_i(1),...,\theta_i(n)|\theta_i^0(1),...,\theta_i^0(n)\big) \tag{2.50}$$

where $\mathrm{Dir}\big(\cdot|\cdot\big)$ denotes the Dirichlet distribution. The prior on the transition kernel A is specified independently for each row $a_{i:}=[a_{i1},...,a_{iN}]$ of A:

$$\mathrm{p}\big(A\big) = \prod_{j=1}^{N}\mathrm{p}\big(a_{i:}\big)\ , \tag{2.51}$$

where for each $a_{i:}$:

$$\mathrm{p}\big(a_{i:}\big) = \mathrm{Dir}\big(a_{i1},...,a_{iN}|a_{i1}^0,...,a_{iN}^0\big)\ . \tag{2.52}$$

Using $\mathrm{p}\big(\pi\big)$ as an example, the form of a Dirichlet prior is:

$$\mathrm{p}\big(\pi\big) = \frac{\Gamma\left(\sum_{i=1}^{N}\pi_i^0\right)}{\prod_{i=1}^{N}\Gamma\left(\pi_i^0\right)}\prod_{i=1}^{N}\pi_i^{\pi_i^0-1}\ .$$

Throughout this thesis the superscript 0 will be used to denote the parameters of the prior. The choice of these parameters and the implications of using priors is discussed in Chapter 4.

The rest of this chapter will now derive inference algorithms for estimating the HMM parameters using the EM and VB methods.

2.3.1 Expectation Maximization for HMM

The basic theory of inference in hidden Markov models was published by Baum et.al. [71] in a series of papers written in the late 1960s and early 1970s. This *Baum-Welch Algorithm* was later shown to be a special case of the expectation maximization algorithm. Perhaps the most referenced paper on EM-based inference in HMMs is the tutorial from Rabiner [10], however a recent book by Cappe et. al. [66] gives a clearer explanation in terms of deriving the EM algorithm and providing comparisons to other inference methods such as Gibbs sampling.

Using the terminology originally introduced by Dempster [9] for the EM algorithm, the aim is to find the parameters that maximize the *incomplete-data likelihood* $\mathrm{p}\big(y_{1:T}|\Theta\big)$, by instead considering the *complete-data likelihood*: $\mathrm{p}\big(m_{1:T}, y_{1:T}|\Theta\big)$.

From the derivation of the EM algorithm in Section 2.2.2, the incomplete data likelihood is maximized by iterating between the E and M step of the algorithm.

The **M-step** of the algorithm is first considered below, where it is assumed there the E-step has just computed the distribution over the hidden states: $\mathrm{P}\big(m_{1:T}|y_{1:T}, \Theta^{old}\big)$. From (2.36) the M-step is equivalent to maximizing $Q(\Theta, \Theta^{old})$ with respect to Θ, where:

$$Q(\Theta, \Theta^{old}) = \sum_{m_{1:T} \in \mathcal{S}^T} \mathrm{P}\big(m_{1:T}|y_{1:T}, \Theta^{old}\big) \log \mathrm{p}\big(y_{1:T}, m_{1:T}|\Theta\big) \ . \tag{2.53}$$

The maximization of the function (2.53) results in the following parameters updates:

$$\begin{aligned} \pi(i) &= \mathrm{P}\big(m_k = i|y_{1:T}, \Theta^{old}\big) \\ a_{i,j} &= \frac{\sum_{k=1}^{T-1} \mathrm{P}\big(m_k = i, m_{k+1} = j|y_{1:T}, \Theta^{old}\big)}{\sum_{k=1}^{T-1} \mathrm{P}\big(m_k = i|y_{1:T}, \Theta^{old}\big)} \\ \theta_i(v) &= \frac{\sum_{k=1}^{T} \delta(y_k = v) \mathrm{P}\big(m_k = i|y_{1:T}, \Theta^{old}\big)}{\sum_{k=1}^{T} \mathrm{P}\big(m_k = i|y_{1:T}, \Theta^{old}\big)} \ , \end{aligned} \tag{2.54}$$

where $\delta(\cdot)$ is the Kronecker delta function. To give an insight into the derivation of these update formulas, the formula for the transition kernel update (2.54) given above is simply thought of as the expected number of transitions from state S_i to state S_j normalized by the expected number of transitions from state S_i.

The **E-step** of the EM algorithm (2.36) calculates the PDF $\mathrm{P}\big(m_{1:T}|y_{1:T}, \Theta^{old}\big)$. However in the case of a HMM, the M-step (2.54) only requires calculation of two marginal statistics of the joint

density $\mathrm{P}\big(m_{1:T}|y_{1:T},\Theta^{old}\big)$:

$$\gamma_k(i) \triangleq \mathrm{P}\big(m_k = i|y_{1:T},\Theta\big) = \sum_{m_{1:T}\in\mathcal{S}^T} \mathrm{P}\big(m_{1:T}|y_{1:T},\Theta\big)\delta(m_k = i) \tag{2.55}$$

$$\xi_k(i,j) \triangleq \mathrm{P}\big(m_k = i, m_{k+1} = j|y_{1:T},\Theta\big) = \sum_{m_{1:T}\in\mathcal{S}^T} \mathrm{P}\big(m_{1:T}|y_{1:T},\Theta\big)\delta(m_k = i, m_{k+1} = j) \ . \tag{2.56}$$

The E-step of the EM algorithm then reduces from calculating the complete density function $\mathrm{P}\big(m_{1:T}|y_{1:T},\Theta^{old}\big)$ to the marginal statistics (2.55) and (2.56). These marginal statistics are the key to inference in HMMs, and can also be considered as a smoothing problem. As in linear systems, this problem is solved using dynamic programming, such as the Rauch-Tung-Striebel or Kalman smoother (see any optimal control text book such as [72]). This involves running a forward filter for estimating the probability of the hidden variable m_k given the observed data up to the current time step k, and then running a backwards smoother. An in depth look at the dynamic programming routing known as the forward-backward algorithm is now considered.

2.3.1.1 The Forward-Backward Algorithm for HMM

The forward-backward algorithm calculates the required marginal statistics (2.55) and (2.56) of the conditional latent mode variable density: $\mathrm{P}\big(m_{1:T}|y_{1:T},\Theta\big)$. Before getting into specific details about the forward-backward process, the form of the distribution $\mathrm{P}\big(m_{1:T}|y_{1:T},\Theta\big)$ is considered. By using the product rule:

$$\log \mathrm{P}\big(m_{1:T}|y_{1:T},\Theta\big) = \log \mathrm{P}\big(m_{1:T},y_{1:T}|\Theta\big) - \log \mathrm{P}\big(y_{1:T}|\Theta\big) \ . \tag{2.57}$$

The conditional mode distribution (2.57) can be expanded by substituting the complete data likelihood (2.47):

$$\log \mathrm{p}\big(m_{1:T}|y_{1:T},\Theta\big) = \log \mathrm{P}\big(m_1|\pi\big) + \sum_{k=1}^{T} \log \mathrm{p}\big(y_k|m_k,\theta_{m_k}\big) + \sum_{k=2}^{T} \mathrm{P}\big(m_k|m_{k-1},A\big) - \log \mathrm{P}\big(y_{1:T}|\Theta\big) \ . \tag{2.58}$$

This specific form (2.58) of the conditional mode distribution is important, and will be used later in the thesis. In particular this form appears when using the variational framework in Section 2.3.2.

The forward-backward algorithm to solve for the marginal statistics of (2.58) is now considered. The forward filter in the forward-backward algorithm [10, 66] is represented by the *forward variables*[2]:

$$\alpha_k(i) \triangleq \mathrm{P}\big(m_k = i|y_{1:k},\Theta\big) \ . \tag{2.59}$$

[2]The forward variables were originally defined as: $\mathrm{P}\big(y_{1:k}, m_k = i\big)$ [10]. This definition can cause numerical underflow, as the probability of the data $y_{1:T}$ tends to becomes very small for large data sets. While the use of the non-scaled version persists in the literature, the work of several other authors [36, 66, 73] is followed here.

The *backwards variables* are likewise used to represent the backwards smoothing operation:

$$\beta_k(i) \triangleq \mathrm{P}\big(y_{k+1:T}|m_k = i, \Theta\big) \ . \tag{2.60}$$

The forward and backward variables are used to estimate the marginal statistics $\gamma_k()$ and $\xi_k()$ required by the M-step (2.54). The marginal probability of the mode of the k^{th} data point is calculated using Bayes' theorem as follows:

$$\gamma_k(i) = \mathrm{P}\big(m_k = i|y_{1:T}, \Theta\big) \propto \mathrm{p}\big(y_{k+1:T}|m_k = i, \Theta\big)\,\mathrm{P}\big(m_k = i|y_{1:k}, \Theta\big) \tag{2.61}$$

$$\propto \alpha_k(i)\beta_k(i) \ . \tag{2.62}$$

The probability $\xi_k()$ can likewise be formulated in terms of the forward and backward variables using Bayes' theorem and then the product rule:

$$\begin{aligned}
\xi_k(i,j) &= \mathrm{P}\big(m_k = i, m_{k+1} = j|y_{1:T}, \Theta\big) \\
&\propto \mathrm{p}\big(y_{k+2:T}|m_{k+1} = j, \Theta\big)\,\mathrm{P}\big(m_k = i, m_{k+1} = j|y_{1:k+1}, \Theta\big) \\
&\propto \beta_{k+1}(j)\,\mathrm{p}\big(y_{k+1}|m_{k+1} = j, \theta_{m_k}\big)\,\mathrm{P}\big(m_{k+1} = j|m_k = i, \Theta\big)\,\mathrm{P}\big(m_k = i|y_{1:k}, \Theta\big) \\
&\propto \beta_{k+1}(j)\,\mathrm{p}\big(y_{k+1}|m_{k+1} = j, \theta_{m_k}\big)a_{ij}\alpha_k(i) \ .
\end{aligned} \tag{2.63}$$

The rest of this section derives recursive formulas for the forward (2.59) and backwards (2.60) variables. The derivations are shown in full here as similar derivations will later be used for nonstationary and variable transition hidden Markov models (Section 3.5), as well as for variational update schemes and newly derived regressor-based switching models (Section 3.7).

Proposition 2.3 (Forward variable recursion for HMMs)**.** The forward variables can be updated using:

$$\alpha_k(i) = \frac{\mathrm{p}\big(y_k|m_k = i, \theta_i\big)}{c_{y_k}} \sum_{j=1}^{N} a_{ji}\alpha_{k-1}(j) \tag{2.64}$$

where the normalizing constant is:

$$c_{y_k} = \mathrm{p}\big(y_k|y_{1:k-1}, \Theta\big) = \sum_{i=1}^{N} \mathrm{p}\big(y_k|m_k = i, \theta_i\big) \sum_{j=1}^{N} a_{ji}\alpha_{k-1}(j) \ . \tag{2.65}$$

$\diamond$

Proof.

$$\begin{aligned}
\alpha_k(i) &= \frac{\mathrm{p}\big(y_k|m_k=i,\theta_i\big)\,\mathrm{P}\big(m_k=i|y_{1:k-1},\Theta\big)}{\mathrm{p}\big(y_k|y_{1:k-1},\Theta\big)} && \text{(Bayes)}\\
&= \frac{\mathrm{p}\big(y_k|m_k=i,\theta_i\big)}{\mathrm{p}\big(y_k|y_{1:k-1},\Theta\big)}\sum_{j=1}^{N}\mathrm{P}\big(m_k=i|m_{k-1}=j,A\big)\,\mathrm{P}\big(m_{k-1}=j|y_{1:k-1},\Theta\big) && \text{(Total Prob.)}
\end{aligned}$$

By substituting definitions of a_{ij} and $\alpha_{k-1}(j)$ into the above equation, the proof of (2.64) is completed. The normalizing constant c_{y_k} (2.65) is realized by the constraint that $\sum_{i=1}^{N}\alpha_k(i)=1$. □

Remark 2.4. The computation complexity of computing the forward variables for $k=1,...,T$ for a HMM with N discrete states is $O(N^2T)$. ◇

Remark 2.5. The normalizing constant allows the calculation of the incomplete data likelihood, utilizing the product rule (A.3):

$$\mathrm{p}\big(y_{1:T}|\Theta\big) = \prod_{k=1}^{T}\mathrm{p}\big(y_k|y_{1:k-1},\Theta\big) \tag{2.66}$$

$$= \prod_{k=1}^{T} c_{y_k} \ . \tag{2.67}$$

Evaluation of the incomplete data likelihood (2.66) should be done in log-space to avoid numerical underflow: $\log\mathrm{p}\big(y_{1:T}|\Theta\big)=\sum_{k=1}^{T}\log c_{y_k}$. ◇

Proposition 2.4 (Backward variable recursion for HMM)**.** The backward variables can be updated using:

$$\beta_{k-1}(i) = \sum_{j=1}^{N}\beta_k(j)\,\mathrm{p}\big(y_k|m_k=j,\theta_j\big)a_{ij} \ . \tag{2.68}$$

◇

Proof.

$$\begin{aligned}
\beta_{k-1}(i) &= \sum_{j=1}^{N}\mathrm{p}\big(y_{k:T},m_k=j|m_{k-1}=i,\Theta\big) && \text{(Marginalization)}\\
&= \sum_{j=1}^{N}\mathrm{p}\big(y_{k+1:T}|m_k=j,\Theta\big)\,\mathrm{p}\big(y_k,m_k=j|m_{k-1}=i,\Theta\big) && \text{(Product Rule)}\\
&= \sum_{j=1}^{N}\mathrm{p}\big(y_{k+1:T}|m_k=j,\Theta\big)\,\mathrm{p}\big(y_k|m_k=j,\theta_j\big)\,\mathrm{p}\big(m_k=j|m_{k-1}=i,A\big) && \text{(Product Rule)}
\end{aligned}$$

This formula is equivalent to (2.68) by the using the definitions of a_{ij} and $\beta_k(j)$. □

Remark 2.6. The computation complexity of computing the backwards variables for $k=1,...,T$ for a HMM with N discrete states is $O(N^2T)$. ◇

A difficulty in implementing the backwards smoother (2.68), is that calculating β_k may result in numerical underflow (consider that the probability $\mathrm{p}\big(y_{1:T}|\Theta\big)$ is likely to be very small for large amounts of data). To calculate the backward variables using the relation (2.68) the variables will need to be scaled. The most common solution to this scaling problems is to use the normalizing constant c_{y_k} from the forward variables to scale the backwards ones [10]. The new backwards recursion for the scaled backwards variables $\tilde{\beta}$ becomes:

$$\tilde{\beta}_{k-1}(i) = \frac{1}{c_{y_k}} \sum_{j=1}^{N} \tilde{\beta}_k(j)\, \mathrm{p}\big(y_k|m_k = j, \theta_j\big) a_{ij} \tag{2.69}$$

and hence the scaled backwards variables $\tilde{\beta}_k(j)$ are expressed in terms of the original variables $\beta_k(j)$:

$$\tilde{\beta}_k(j) = \prod_{t=k}^{T} c_{y_t}^{-1} \beta_k(j) \ . \tag{2.70}$$

The effectiveness of using (2.69) is demonstrated by considering the product of the normalizing constants $\prod_{t=k}^{T} c_{y_t} = \mathrm{p}\big(y_{k+1:T}|y_{1:k}\big)$ used in (2.70). By decomposing the unscaled backward variable using the conditional independence of HMM and then Bayes' theorem:

$$\begin{aligned} \beta_k(i) &\triangleq \mathrm{P}\big(y_{k+1:T}|m_k = i, \Theta\big) = \mathrm{P}\big(y_{k+1:T}|m_k = i, y_{1:k}, \Theta\big) \\ &= \frac{\mathrm{P}\big(m_k = i|y_{1:T}, \Theta\big)\, \mathrm{P}\big(y_{k+1:T}|y_{1:k}, \Theta\big)}{\mathrm{P}\big(m_k = i|y_{1:k}, \Theta\big)} \ , \end{aligned} \tag{2.71}$$

we see that the term $\mathrm{P}\big(y_{k+1:T}|y_{1:k}, \Theta\big)$ in (2.71), which may cause numerical issues, will cancel with the product of the scaling constants c_{y_k}.

In summary, an algorithm to compute the forward and backward variables is given in Alg. 2.5.

Algorithm 2.5 Forward-Backward Algorithm for hidden Markov models

Initialize forward variables: $\alpha_0(i) \equiv \pi(i)$
for $k = 1$ to T **do**
  for $i = 1$ to N **do**
    $\alpha_k(i) = \frac{\mathrm{p}\left(y_k|m_k=i,\theta_i\right)}{\mathrm{p}\left(y_k|y_{1:k-1},\Theta\right)} \sum_{j=1}^{N} a_{ji}\alpha_{k-1}(j)$
    where $\mathrm{p}\left(y_k|y_{1:k-1},\Theta\right) = \sum_{i=1}^{N} \mathrm{p}\left(y_k|m_k=i,\theta_i\right) \sum_{j=1}^{N} a_{ji}\alpha_{k-1}(j)$
  end for
end for
Initialize backwards variables: $\beta_T(i) = 1$
for $k = T$ to 2 **do**
  for $i = 1$ to N **do**
    $\tilde{\beta}_{k-1}(i) = \frac{1}{\mathrm{p}\left(y_k|y_{1:k-1}\right)} \sum_{j=1}^{N} \tilde{\beta}_k(j)\, \mathrm{p}\left(y_k|m_k=j,\theta_j\right) a_{ij}$
  end for
end for

2.3.2 Variational Bayes for HMM

In this section, the Variational Bayes framework is used for inference in hidden Markov models. The developments in this section follow from an unpublished work by MacKay [35], and the thesis of Beal [36]. This derivation assumes familiarity with the variational methods described in Section 2.2.1, and will utilize the forward-backward algorithm derived in Section 2.3.1.

Following from Section 2.2.1, the aim of VB is to approximate the posterior distribution of the HMM model $\mathrm{p}\left(\Theta, m_{1:T}|y_{1:T}\right)$ with a factorized set of functions: $q(m_{1:T})q(\Theta)$. To find the best approximation, we seek to minimize the KL divergence between $\mathrm{p}\left(\Theta, m_{1:T}|y_{1:T}\right)$ and $q(m_{1:T})q(\Theta)$. This is equivalent to maximizing the lower bound (2.20) of the model evidence, and in latent variable models, is achieved by sequentially computing the following VB-E (2.28) and VB-M (2.27) steps:

$$\textbf{VB-M step} \qquad q(\Theta) = \frac{1}{C_\Theta} \exp\left[\sum_{m_{1:T}\in\mathcal{S}^T} q(m_{1:T}) \log \mathrm{p}\left(y_{1:T}, m_{1:T}|\Theta\right)\right] \mathrm{p}\left(\Theta\right) \tag{2.72}$$

$$\textbf{VB-E step} \qquad q(m_{1:T}) = \frac{1}{C_{m_{1:T}}} \exp\left[\int q(\Theta) \log \mathrm{p}\left(y_{1:T}, m_{1:T}|\Theta\right) d\Theta\right] \ . \tag{2.73}$$

The remainder of this section will provide solutions to the VB-E (2.73) and VB-M (2.72) steps, which can then be used in the variational inference algorithm (Alg. 2.2). Furthermore, an analytical expression for the variational lower bound to the model evidence is derived.

2.3.2.1 VB-M Step

Using the complete data likelihood for a HMM (2.47), the VB-M step (2.72) can be written:

$$\log q(\Theta) = \log \mathrm{p}\big(\pi\big) + \sum_{m_{1:T}\in\mathcal{S}^T} q(m_{1:T}) \log \pi_{m_1} + \log \mathrm{p}\big(A\big) + \sum_{m_{1:T}\in\mathcal{S}^T} q(m_{1:T}) \sum_{k=2}^{T} \log \mathrm{P}\big(m_k | m_{k-1}, A\big) + \sum_{i=1}^{N} \log p(\theta_i) + \sum_{m_{1:T}\in\mathcal{S}^T} q(m_{1:T}) \sum_{k=1}^{T} \log \mathrm{P}\big(y_k | m_k, \theta_{m_k}\big) - \log C_\Theta \ . \quad (2.74)$$

In expression (2.74) several HMM parameters are independent, implying without further assumption that: $q(\Theta) = q(\pi)q(A)q(\theta_1, ..., \theta_N)$. This simplifies computing the distribution $q(\Theta)$, as instead we now can consider each component independently. For example, the distribution $q(A)$, which is the variational approximation of the Markov transition matrix A, can be computed by considering all terms in (2.74) that are functions of A:

$$\log q(A) = \log \mathrm{p}\big(A\big) + \sum_{m_{1:T}\in\mathcal{S}^T} q(m_{1:T}) \sum_{k=2}^{T} \log \mathrm{P}\big(m_k | m_{k-1}, A\big) + C_A \ , \quad (2.75)$$

where C_A is a normalizing constant. Direct analysis of equation (2.75) is intractable because of the summation over all possible mode sequences $m_{1:T}$. Instead (2.75) can be written:

$$\log q(A) = \log \mathrm{p}\big(A\big) + \sum_{m_{1:T}\in\mathcal{S}^T} \sum_{k=2}^{T} \sum_{j=1}^{N} \sum_{i=1}^{N} q(m_{1:T}) \delta(m_k = j, m_{k-1} = i) \log \mathrm{P}\big(m_k = j | m_{k-1} = i, A\big) + C_A \ , \quad (2.76)$$

where $\delta(\cdot)$ is the Kronecker delta function. While (2.76) may not seem like much of an improvement over (2.75), it allows the use of the following marginal distribution:

$$\xi_k(i,j) \triangleq q(m_k = j, m_{k-1} = i) = \sum_{m_{1:T}\in\mathcal{S}^T} q(m_{1:T}) \delta(m_k = j, m_{k-1} = i) \ , \quad (2.77)$$

where (2.77) is a direct application of the Marginalization Theorem (this is the same methodology as used in the EM algorithm for HMMs (2.56)). This marginal statistic (2.77) substituted into (2.76) results in:

$$\log q(A) = \sum_{j=1}^{N} \sum_{i=1}^{N} \log a_{ij}^{a_{ij}^0 - 1} + \sum_{k=2}^{T} \sum_{j=1}^{N} \sum_{i=1}^{N} \xi_k(i,j) \log a_{ij} + C_A \ , \quad (2.78)$$

where C_A is a constant, and the prior $p(A)$ has been substituted from (2.48). The equation (2.78) is proportional to a product of Dirichlet distributions, where the i^{th} row of the transition kernel A,

is denoted $a_{i:} \triangleq [a_{i1}, a_{i2}, ..., a_{iN}]$:

$$q(A) = \prod_{i=1}^{N} q(a_{i:}) \quad \text{where } q(a_{i:}) = \mathrm{Dir}\big(a_{i1}, ..., a_{iN} | [a_{i1}^T, ..., a_{iN}^T]\big) \ , \tag{2.79}$$

where the parameters of the Dirichlet distribution $q(a_{i:})$ in (2.79) are:

$$a_{ij}^T = a_{ij}^0 + \sum_{k=2}^{T} \xi_k(i,j) \ .$$

Now that $q(A)$ has been calculated, the distributions $q(\theta_1, ..., \theta_N)$ and $q(\pi)$ are derived. First the calculation of $q(\theta_1, ..., \theta_N)$ is considered. All terms in equation (2.74) that contain $\theta_1, ..., \theta_N$ are collected to define the distribution $q(\theta_1, ..., \theta_N)$:

$$\log q(\theta_1, ..., \theta_N) = \sum_{i=1}^{N} \log p(\theta_i) + \sum_{S^T} q(m_{1:T}) \sum_{k=1}^{T} \log \mathrm{P}\big(y_k | m_k, \theta_k\big) + C_{\theta_{1:N}} \tag{2.80}$$

where $C_{\theta_{1:N}}$ is a constant. Equation (2.80) is simplified by using a second marginal distribution of $q(m_{1:T})$:

$$\begin{aligned} \log q(\theta_1, ..., \theta_N) &= \sum_{i=1}^{N} \log p(\theta_i) + \sum_{i=1}^{N} \sum_{k=1}^{T} \sum_{S^T} q(m_{1:T}) \delta(m_k = i) \log \mathrm{P}\big(y_k | m_k = i, \theta_i\big) + C \\ &= \sum_{i=1}^{N} \log p(\theta_i) + \sum_{i=1}^{N} \sum_{k=1}^{T} \gamma_k(i) \log \mathrm{P}\big(y_k | m_k = i, \theta_i\big) + C \ , \end{aligned} \tag{2.81}$$

where the marginal distribution is then defined:

$$\gamma_k(i) \triangleq q(m_k = i) = \sum_{m_{1:T} \in \mathcal{S}^T} q(m_{1:T}) \delta(m_k = i) \ , \tag{2.82}$$

and where $\delta(\cdot)$ is Kronecker delta. The distribution $q(\theta_1, ..., \theta_N)$ can be further decomposed using the form of equation (2.81) where the θ_i parameters are mutually independent, implying $q(\theta_1, ..., \theta_N) = q(\theta_1)...q(\theta_N)$. By considering all terms containing θ_i in equation (2.81):

$$\log q(\theta_i) = \log p(\theta_i) + \sum_{k=1}^{T} \gamma_k(i) \, \mathrm{P}\big(y_k | m_k = i, \theta_i\big) + C_{\theta_i} \ , \tag{2.83}$$

where C_{θ_i} is a constant. Substituting in (2.50) for the prior $\mathrm{p}\big(\theta_i\big)$, and given that $\mathrm{P}\big(y_k = v | m_k = i, \theta_i\big) = \theta_i(v)$, from Definition 2.4, equation (2.83) can be written:

$$\log q(\theta_i) = \sum_{v=1}^{n} \log \theta_i(v)^{\theta_i^0(v) - 1} + \sum_{v=1}^{n} \sum_{k=1}^{T} \gamma_k(i) \delta(y_k = v) \log \theta_i(v) + C_{\theta_i} \ . \tag{2.84}$$

Equation (2.84) is proportional to a Dirichlet distribution:

$$q(\theta_i) = \mathrm{Dir}\big(\theta_i(1), ..., \theta_i(n) | [\theta_i^T(1), ..., \theta_i^T(n)]\big) \tag{2.85}$$

where

$$\theta_i^T(v) = \theta_i^0(v) + \sum_{k=1}^{T} \gamma_k(i)\delta(y_k = v) \ .$$

The computation of $q(\pi)$ is similar to that of $q(\theta_i)$: by considering all terms containing π in equation (2.74), it can be shown that:

$$q(\pi) = \mathrm{Dir}\big(\pi(1), ..., \pi(N) | \pi^T(1), ..., \pi^T(N)\big) \ , \tag{2.86}$$

where $\pi^T(i) = \gamma_1(i) + \pi^0(i)$;

In summary, this section has provided analytical solutions for calculating the distributions $q(A)$, $q(\pi)$, and $q(\theta_1), ..., q(\theta_N)$. Furthermore, as $q(\Theta) = q(A)q(\pi)q(\theta_1), ..., q(\theta_N)$, this provides the required solution to the VB-M step (2.72).

Remark 2.7. A significant result of the VB-M step is that only the marginal distributions γ_k and ξ_k are required, instead of the joint mode distribution $q(m_{1:T})$. This will simplify the following VB-E step presented in the next section. ◇

2.3.2.2 VB-E Step

This section solves the VB-E step (2.73) of the Variational Bayes algorithm. A significant result of the VB-M step is that only the marginal statistics γ_k and ξ_k of the distribution $q(m_{1:T})$ need to be calculated (see Remark 2.7) instead of the full distribution $q(m_{1:T})$. These marginal statistics (2.77) and (2.82) will be calculated using a similar forward-backward algorithm to that used in the EM algorithm. Essentially, the distribution $q(m_{1:T})$ will be converted into the same functional form as equation (2.58), for which the forward-backward algorithm was originally devised.

The VB-E step derivation starts by substituting the HMM complete data likelihood (2.47) into the VB-E equation (2.73) resulting in:

$$\log q(m_{1:T}) =$$
$$\int_\Theta q(\Theta) \left[\log \mathrm{P}\big(m_1|\pi\big) + \sum_{k=2}^{T} \log \mathrm{P}\big(m_k|m_{k-1}, A\big) + \sum_{k=1}^{T} \log \mathrm{p}\big(y_k|m_k, \theta_{m_k}\big) \right] d\theta - \log C_{m_{1:T}} \ . \tag{2.87}$$

Utilizing the functional form of the parameters, $q(\Theta) = q(\pi)q(A)q(\theta_1)...q(\theta_N)$, found in the VB-M

step, equation (2.87) is expanded:

$$\log q(m_{1:T}) = \int_{\pi} q(\pi) \log \mathrm{P}\big(m_1|\pi\big) d\pi + \sum_{k=2}^{T} \int_{A} q(A) \log \mathrm{P}\big(m_k|m_{k-1}, A\big) dA$$
$$+ \sum_{k=1}^{T} \int_{\theta_{m_k}} q(\theta_{m_k}) \log \mathrm{p}\big(y_k|m_k, \theta_{m_k}\big) d\theta_{m_k} - \log C_{m_{1:T}} \ . \tag{2.88}$$

To simplify (2.88), the following geometric means are defined:

$$\tilde{\pi}_i = \exp \int_{\pi} q(\pi) \log \mathrm{P}\big(m_1 = i|\pi\big) d\pi \tag{2.89a}$$
$$\tilde{a}_{ij} = \exp \int_{A} q(A) \log \mathrm{P}\big(m_k = j|m_{k-1} = i, A\big) dA \tag{2.89b}$$
$$\tilde{b}_i = \exp \int_{\theta_i} q(\theta_i) \log \mathrm{p}\big(y_k|m_k = i, \theta_i\big) d\theta_i \ . \tag{2.89c}$$

Before substituting (2.89) into the the expression for $q(m_{1:T})$ (2.88), analytical expressions for the geometric means (2.89) are derived.

The calculation of the geometric mean $\tilde{a}_{ij}$ in (2.89b) is derived first. From the VB-M step we know that $q(A)$ factorizes such that the rows $a_{i:}$ of A are independent $q(A) = q(a_{1:}) \dots q(a_{1:})$, where each $q(a_{1:})$ is a Dirichlet distribution (2.79). Using this decomposition simplifies (2.89b) into:

$$\tilde{a}_{ij} = \exp \int_{a_{i:}} q(a_{i:}) \log a_{ij} da_{i:} \tag{2.90}$$

Equation (2.90) is the geometric mean of a Dirichlet distribution. A Dirichlet distribution has the following geometric mean (see (A.16)):

$$\tilde{\phi}_i = \int_{\phi} \log \phi_i \mathrm{Dir}\big(\phi_1, ..., \phi_n|u_1, ..., u_n\big) d\phi$$
$$= \psi(u_i) - \psi\left(\sum_{j=1}^{n} u_j\right) \ , \tag{2.91}$$

where ψ is is the digamma function (A.13). Note that geometric means (e.g. $\tilde{\phi}$), are sub-normalized probability distributions, such that $\sum_i \tilde{\phi}_i \leq 1$, by application of Jensen's Inequality: $E[\log(x)] \leq log(E[x])$ as log is a concave function, and $E[\cdot]$ is the expectation.

Substituting the form of the Dirichlet distribution (2.91) into the expression for the geometric mean (2.89b) results in:

$$\tilde{a}_{ij} = \psi(a_i^T j) - \psi\left(\sum_{r=1}^{N} a_{ir}^T\right) \ , \tag{2.92}$$

where a_{ij}^T are the parameters of $q(A)$ given in (2.79). The calculation of the other geometric means in (2.89) are easily computed as $q(\theta_i)$ and $q(\pi)$ are Dirichlet distributions (2.85) and (2.86), and are

also calculated using (2.91).

The geometric means (2.89) are substituted into the expression (2.88) for $q(m_{1:T})$ resulting in:

$$\log q(m_{1:T}) = \log \tilde{\pi}_{m_1} + \sum_{k=2}^{T} \log \tilde{a}_{m_k m_{k+1}} + \sum_{k=1}^{T} \log \tilde{b}_{m_k}(y_k) - \log C_{m_{1:T}} \ . \tag{2.93}$$

This distribution (2.93) is the same form as the posterior distribution $\mathrm{P}\big(m_{1:T}|y_{1:T}, \Theta\big)$ calculated by the forward-backward algorithm (2.58). The only difference between equation (2.58) and (2.93) is that the geometric means (2.89) are sub-normalized probability distributions. Sub-normalized distributions are simply unnormalized distributions that integrate to less then one, and can hence be normalized with a constant factor. The distinction between sub-normalized and unnormalized distributions is later important in proving lower bounds. This does not affect the application of the forward-backward algorithm to (2.93) as any constant terms inside the logarithms can be moved into the normalizing constant $C_{m_{1:T}}$. The forward variable recursion for (2.93) is now stated:

$$\alpha_k(i) = \frac{\tilde{b}_i(y_k)}{c_{y_k}} \sum_{j=1}^{N} \tilde{a}_{ji} \alpha_{k-1}(j) \ , \tag{2.94}$$

where:

$$c_{y_k} = \sum_{i=1}^{N} \tilde{b}_i(y_k) \sum_{j=1}^{N} \tilde{a}_{ji} \alpha_{k-1}(j) \ . \tag{2.95}$$

Note that the normalizing constants c_{y_k} in (2.95) must be smaller than if normalized distributions were used. The backwards variable recursion for (2.93) is stated:

$$\beta_{k-1}(i) = \sum_{j=1}^{N} \beta_k(j) \tilde{b}_j(y_k) \tilde{a}_{ij} \ . \tag{2.96}$$

The marginal distributions (2.61) and (2.63) are not affected if the forward and backward variables are multiplied by a constant:

$$\gamma_k(i) \propto \alpha_k(i) \beta_k(i) \tag{2.97}$$

$$\xi_k(i,j) \propto \beta_{k+1}(j) \tilde{b}_j(y_k) \tilde{a}_{ij} \alpha_k(i) \ . \tag{2.98}$$

The VB-E step then proceeds as follows: First the forward variables are initialized by $\alpha_1(i) = \pi_i \tilde{b}_i(y_1)$. Recursion (2.94) is used to calculate the forward variables for $k = 2, ..., T$. The backward variables are initialized by setting $\beta_T(i) = 1$, and then $\beta_k(i)$ is calculated using the recursion (2.96). The marginal distributions that are required for the VB-M step are calculated using (2.97) and (2.98).

2.3.2.3 Evaluation of the Lower Bound $\mathcal{L}(q)$

Evaluating the lower bound (2.20) of the variational framework is important as it allows for convergence testing (Alg. 2.2), and model selection (Chapter 4). This section will derived the lower bound for HMM following the work of Beal [36].

The first step in calculating the lower bound is to compute the normalizing constant $C_{m_{1:T}}$ from the VB-E step. As noted in [36] the forward-backward algorithm can be used to do this. Recall that using the forward-backward algorithm for the EM algorithm evaluated the marginal statistics of (2.58):

$$\log \mathrm{p}\big(m_{1:T}|y_{1:T},\Theta\big) = \log \pi_{m_k} + \sum_{k=1}^{T} \log b_{m_k}(y_k) + \sum_{k=2}^{T} a_{ij} - \log \mathrm{P}\big(y_{1:T}|\Theta\big) \ , \tag{2.99}$$

where we have used $\pi_i = \mathrm{P}\big(m_1 = i|\pi\big)$, $b_i(y_k) = \mathrm{p}\big(y_k|m_k = i, \theta_{m_k}\big)$ and $a_{ij} = \mathrm{P}\big(m_k = i|m_{k-1} = j, A\big)$. When using the forward algorithm, the product of the normalizing constants (2.66) calculates the normalizing constant: $\mathrm{P}\big(y_{1:T}|\Theta\big) = \prod_{k=1}^{T} c_{y_k}$ in (2.99). Using Variational Bayes, the forward-backward algorithm instead operates on the sequence (2.93):

$$\log q(m_{1:T}) = \log \tilde{\pi}_{m_1} + \sum_{k=2}^{T} \log \tilde{a}_{m_k m_{k+1}} + \sum_{k=1}^{T} \log \tilde{b}_{m_k}(y_k) - \log C_{m_{1:T}} \ . \tag{2.100}$$

The only difference between (2.99) and (2.100) is that the distributions $\tilde{\pi}_i$, $\tilde{a}_{ij}$, and $\tilde{b}_{m_k}(y_k)$ are sub-normalized. The product of the normalizing constants c_{y_k} in VB (2.94) still determines the normalizing constant $C_{m_{1:T}}$:

$$C_{m_{1:T}} = \prod_{k=1}^{T} c_{y_k} \ . \tag{2.101}$$

Calculating the normalizing constant $C_{m_{1:T}}$ using (2.101) directly after every VB-E step makes the computation of the lower bound far easier. The lower bound (2.20) is simplified using the factorization $q(\Theta, m_{1:T}) = q(m_{1:T})q(\Theta)$ and $q(\Theta) = q(A)q(\pi)q(\theta_1)...q(\theta_N)$ resulting in:

$$\begin{aligned}\mathcal{L}(q(m_{1:T},\Theta)) = &\int_{\Theta} q(\Theta) \log \frac{\mathrm{p}(\Theta)}{q(\Theta)} d\Theta + \int_{\Theta} \sum_{m_{1:T}\in\mathcal{S}^T} q(\Theta, m_{1:T}) \log \mathrm{p}\big(m_{1:T}, y_{1:T}|\Theta\big) d\theta \\ &- \sum_{m_{1:T}\in\mathcal{S}^T} q(m_{1:T}) \log q(m_{1:T}) \ .\end{aligned} \tag{2.102}$$

The entropy of $q(m_{1:T})$ in (2.102) can be decomposed after the VB-E step by using equation (2.73):

$$\begin{aligned}\mathrm{H}\big(q(m_{1:T})\big) &= -\sum_{m_{1:T}\in\mathcal{S}^T} q(m_{1:T})\log q(m_{1:T}) \\ &= -\sum_{m_{1:T}\in\mathcal{S}^T} q(m_{1:T})\int_\Theta q(\Theta)\log \mathrm{p}\big(m_{1:T}, y_{1:T}|\Theta\big)d\theta - \log C(m_{1:T}) \ . \end{aligned} \tag{2.103}$$

Substituting (2.103) into (2.102) results in:

$$\mathcal{L}(q(m_{1:T},\Theta)) = \int_\Theta q(\Theta)\log\frac{\mathrm{p}\big(\Theta\big)}{q(\Theta)}d\Theta + \log C(m_{1:T}) \ . \tag{2.104}$$

The KL divergence of the parameters $\int q(\Theta)\log\frac{\mathrm{p}\big(\Theta\big)}{q(\Theta)}d\Theta$ can be simplified by the factorial form of $q(\Theta)$. The lower bound (2.104) is then evaluated as:

$$\begin{aligned}&\mathcal{L}(q(m_{1:T},\Theta)) = \\ &\int_\pi q(\pi)\log\frac{\mathrm{p}\big(\pi\big)}{q(\pi)}d\Theta + \sum_{i=1}^N\int q(a_{i:})\log\frac{\mathrm{p}\big(a_{i:}\big)}{q(a_{i:})}da_{i:} + \sum_{i=1}^N\int_{\theta_i} q(\theta_i)\log\frac{\mathrm{p}\big(\theta_i\big)}{q(\theta_i)}d\theta_i + \log C(m_{1:T}) \ .\end{aligned} \tag{2.105}$$

All KL divergences in (2.105) are between two Dirichlet distributions, the solution of which is derived in Appendix B. Evaluating this lower bound (2.105) directly after the VB-E step achieves three things: First, as both the VB-E and VB-M step increase the lower bound, it provides a method for checking the correctness of the coded algorithm (that is: if the lower bound ever decreases, something is wrong). Second, monitoring the lower bound provides and indication of convergence of the VB algorithm (See Alg. 2.2). Third, this lower bound will be used for model selection in Chapters 4 and 5 of this thesis.

Chapter 3

Hidden Markov Models and Extensions as a Basis for Inference in Hybrid Systems

3.1 Introduction and Motivation

This chapter introduces a series of hybrid system models and develops inference algorithms for identification of the model parameters using observed data. Each of these models combines a discrete time dynamical system with finite state switching systems of increasing complexity. First, *generalized linear hidden Markov models* (GLHMMs), are introduced, which combine a generalization of linear dynamical systems with discrete switching determined by a stationary Markov chain. In the second model class, the discrete switching is determined by a non-stationary Markov chain. Specifically *variable transition hidden Markov models* (VTHMM) and *hidden semi-Markov Models* (HSMM) are used which explicitly model the duration spent in each discrete state of the hybrid system. In the third model class, nonstationary *regressor-dependant Markov chains* are introduced to govern the transition behavior of the system. In this case the discrete state transitions depend on the state of the system's dynamics. This new class of Markov chains leads to a class of hybrid systems termed *hidden-regressor-dependant Markov models* (HRDMM).

By considering hybrid systems as extensions to Markov models, powerful existing inference algorithms that exist for hidden Markov models (HMM) can be extended to the hybrid system identification problem. In the previous chapter, the expectation maximization (EM) algorithm, the Gibbs sampler, and variational Bayes (VB) were applied to HMMs. In this chapter these algorithms will be applied to the presented GLHMM, VTHMM, HSMM and HRDMM models.

The dynamical systems associated with each discrete mode of the hybrid system will be represented in this thesis by *generalized linear models* (GLMs). This choice of dynamical system is preferred over state space representations for two reasons: First, inference in GLMs models is more

tractable then in state space models because GLMs are extensions of auto-regressive (AR) models where the system output evolves according to a regression of previous system outputs. Inference in state space models is a more difficult problem as it requires the estimation of a hidden state. Equivalent state space representations can be created from AR models [28, 58], a subclass of GLMs. Furthermore, AR models are commonly used in the hybrid systems community, especially for the PWARX models discussed in Section 2.1.3. GLMs are also useful for modelling point processes, such as a counting process, where observations are discrete. This added flexibility is important for considered neurophysiological applications.

The hybrid system community has considered few problems in identifying hybrid systems from observed data. The identification of only two sets of models is considered: Markov jump systems[1] (see MJS in Section 2.1.1 and associated references) and the piecewise auto-regressive exogenous models (see PWARX models in Section 2.1.3 and references within). As motivation for the models explored in this thesis, consider the discrete state transition behavior associated with either of these models: The MJS transitions are governed by a stationary Markov process, implying the evolution of system's discrete state (or mode) is only a function of the previous discrete state. This type of transition rule, also used in the GLHMM, is best suited to modeling systems where the discrete state is correlated in time. Simply put, observations collected during a short time interval are more likely to be generated from the same discrete mode. This is a very reasonable assumption for many systems, especially for the applications considered in this thesis. This idea that sequentially collected data are likely to be generated by the same discrete state of the hybrid system is extended in the VTHMM and HSMM models. Secondly the PWARX type models incorporate a very different switching behavior. In simple terms, the discrete mode of a PWARX system is only a function of the system output, and is not directly dependent on the discrete state of previous time steps[2]. Typically, when identifying PWARX systems, the time correlation of data points is completely disregarded [4, 6, 59], and instead data points are clustered based on a regressor of previous system outputs alone. PWARX models are well suited to modeling a wide range of systems, and provide a class of models which can be readily identified, yet incorporate the system dynamics in the switching behavior. However, considering the applications in this thesis, where system observations are very noisy, it is found that clustering data points without using correlation in time is inadequate to form reasonable models. Instead the identification of PWARX type systems is approached in two ways. First, instead of just clustering points based on regressor values, a version of the GLHMM is used to both cluster points in time and use regressor values. Second, the developed HRDMM models provide a combination of Markov and PWARX based switching behavior.

[1] Technically only the control and estimation, not identification, of MJS is published in the hybrid system community. Instead most work in MJS is considered by researchers in machine learning and bio-informatics. See [53] and references within.

[2] Note that as the system output is a function of the discrete states, previous discrete states *indirectly* affect the current discrete state.

This chapter is organized into the the following sections:

1. Section 3.2 will first define a class of generalized linear models that are used to represent the dynamics of the continuous states, and then define the GLHMM model. Section 3.3 will derive a variational Bayes inference algorithm for identification of GLHMM systems, followed by Section 3.4 where a Gibbs sampler is developed for GLHMM identification.

2. Section 3.5 will define the HSMM and VTHMM models, and derive a class of hybrid systems based on these non-stationary Markov systems. A variational Bayesian algorithm is developed for the identification of HSMM, VTHMM and associated hybrid systems. Furthermore, an existing Gibbs sampler for VTHMM models [38] can be improved using the derived forward-backward recursions used for the variational inference algorithm.

3. Section 3.6 identifies PWARX models by relaxing the problem into a GLHMM identification problem, allowing the use of inference algorithms developed in Sections 3.2 – 3.4 . Results show that this approach generally provides equivalent performance to current PWARX identification methods, and may provide better performance in certain cases. This PWARX model identification problem is used to motivate Section 3.7 where a new HRDMM model is defined, and several HRDMM inference algorithms are developed.

4. This chapter concludes by presenting a series of causal and noncausal estimation algorithms. These estimators use components of the developed identification algorithms, and can use an identified model to infer the state sequence of new data in real time.

3.2 Generalized Linear Hidden Markov Models (GLHMMs)

This section defines a class of hybrid system called generalized linear hidden Markov models (GLHMMs). This model class is motivated by neurophysiological applications, which require flexibility in representing various neurological signals. Generalized linear models (GLMs) are capable of modeling a wide range of signals, and contain linear auto-regressive models as a special case. Consequently existing auto-regressive hidden Markov models [10, 67] are a special case of the GLHMM model specified here. Furthermore GLHMM models can represent systems with multiple output signals of mixed type: for instance, linear models may be used to represent some system outputs, and a point process can be used to model other system outputs.

Before defining a GLHMM in Definition 3.4, a review of generalized linear models is presented. This review (Section 3.2.1) will contain both a general GLM definition, as well as several specific cases that are used in this thesis. Furthermore, relevant issues pertaining to identification of these models is discussed.

3.2.1 Generalized Linear Models

Generalized linear models (GLMs) [26, 27], are an extension of linear regression that allows modeling of situations where observations of the system state are not normally distributed. In GLMs, a linear predictor is used to predict a function of the mean $\mu_k \in \mathbb{R}$ of the outcome (observed) variable:

$$g(\mu_k) = w^T x_k \ , \tag{3.1}$$

where $g(\cdot)$ is a smooth invertible *link function*, $x_k \in \mathbb{R}^d$ is a regressor vector of observed values or known inputs, and $w \in \mathbb{R}^d$ is a vector of parameters. Note that the subscript k is used to denote association with the observation y_k collected at time t_k.

The observed output of a GLM, y_k, is distributed according to a distribution $f(\mu_k)$ whose mean μ_k is described by the inverse of the link function:

$$y \sim f(\mu_k), \text{ where } \mu_k = g^{-1}(w^T x_k) \ . \tag{3.2}$$

The distribution, $f(\cdot)$, and the link function, $g(\cdot)$, are smooth and are often constrained to be pairs of compatible functions, as described in Table 3.1. In some cases $f(\cdot)$ is parameterized, such as when using a normal distribution: $f(\cdot) = \mathcal{N}(\cdot, \sigma^2)$, where σ^2 is the variance of the normal distribution. In such cases, we will denote the set of all parameters associated with the GLM as θ (e.g. $\theta = \{w, \sigma^2\}$).

GLM parameters are often identified via maximum likelihood methods [74]. To usefully identify GLMs with either Gibbs sampling or variational Bayesian methods, it must be shown that the density function $p(\theta|x_{1:T}, y_{1:T})$ is of a convenient form, which here will mean either log-concave or conjugate. In practice, the distribution $f(\mu_k)$ is often constrained to the exponential family of distributions, with an associated compatible link function $g(\mu_k)$, as shown in Table 3.1. These compatible functional forms have been proven to yield log-concave likelihoods [75], for which there exist efficient simulation tools appropriate for Gibbs sampling and can also be incorporated into the Variational Bayesian framework. Gilks [76] describes an adaptive rejection sampling technique which is based on bounding a log-concave density function with upper and lower hulls, allowing single samples to be drawn from the distribution with only a few function evaluations. Jaakkola describes local variational approximations that allow variational inference [77] in log-concave GLMs. These approximations allow for efficient inference of GLM parameters, but do require the addition of additional nuisance parameters.

While all the log-concave GLMs shown in Table 3.1, can be used in the inference algorithms developed in this section, a convenient subclass of conjugate GLMs are first considered. A conjugate model is defined such that the posterior distribution of the model parameters, after being updated

Table 3.1: Log-concave likelihood forms of $g()$ and $f()$

$f(\mu)$	$g(\mu)$
Normal	identity operator
Gamma	$g(\mu) = \log \mu$, or $g(\mu) = \mu^{\gamma}, (-1 \leq \gamma < 0)$
Poisson	$g(\mu) = \log \mu$ or $g(\mu) = \mu^{\gamma}, (-1 \leq \gamma < 0)$
Binomial	$g(\mu) = \text{logit}(\mu)$ or $g(\mu) = \Phi^{-1}(\mu)$, or $g(\mu) = log\,(-log(1-\mu))$

with observed data using Bayes theorem, is of the same form as the prior distribution. Conjugate models allow extremely efficient updating of model parameters as analytical posterior expressions can be derived, avoiding the need for adaptive rejection sampling or addition of nuisance parameters.

The following two classes of conjugate GLMs are used in subsequent examples throughout this thesis. The first example is auto-regressive models, also called an auto-regressive exogenous model if a known external input is applied:

Definition 3.1 (Auto Regressive Dynamics)**.** *An auto regressive (AR) model is a generalized linear model where the data likelihood $y_k \in \mathbb{R}$ is given by the normal distribution:*

$$y_k \sim \mathcal{N}\left(w^T x_k, (\tau)^{-1}\right) \tag{3.3}$$

with $x_k = [y_{k-1}, ..., y_{k-n}]^T$, and $\theta = \{w, \tau\}$. Typically the parameters w are refered to as the AR weights, and τ is the precision, or inverse of the variance. In this thesis a conjugate Gaussan-Gamma (A.14) *distribution is used for representing prior parameter information:*

$$p(w,\tau) = \mathrm{p}\big(w|\tau\big)p(\tau) = \mathcal{N}\left(w|w^0, (\tau\Lambda_0)^{-1}\right) Gam\big(\tau|a^0, b^0\big) \ , \tag{3.4}$$

where $w^0 \in \mathbb{R}^n, \Lambda^0 \in \mathbb{R}^{n\times n}, a^0 \in \mathbb{R}, b^0 \in \mathbb{R}$ are hyper-parameters, $\mathcal{N}\left(w|w^0, (\tau\Lambda_0)^{-1}\right)$ is the multivariate Gaussian distribution (A.7) *and $Gam(\tau|a^0, b^0)$ is the Gamma* (A.10) *distribution. Note for most applications is it sufficient to consider the class of prior precision matrices $\Lambda^0 = \lambda^0 I$, where $\lambda^0 \in \mathbb{R}$, and $I \in \mathbb{R}^{n\times n}$ is the identity matrix.* ◇

Definition 3.2 (Stationary Poisson Point Process)**.** *A stationary Poisson point process is a generalized linear model where the data likelihood for $y_k \in \mathbb{N}$ is a Poisson distribution* (A.17)

$$p\left(y_k|\lambda\right) = \frac{\lambda^{(y_k)} e^{-\lambda}}{(y_k)!} \ , \tag{3.5}$$

and $\theta = \lambda$ is termed the "firing rate", in application of this model to neural data analysis. In this thesis a conjugate Gamma distribution (A.10) *prior is used for the firing rate parameter λ:*

$$p(\lambda) = Gam\big(\lambda|a^0, b^0\big) \ . \tag{3.6}$$

◇

A nonstationary Poisson point process model is now considered, which is a log-concave (but non-conjugate) GLM. This nonstationary point process has been previously used for modeling neuronal single unit activity in GLHMMs [65, 64], and was originally proposed for modeling neural activity by Truccolo et.al. in [78].

Definition 3.3 (Nonstationary Poisson Point Process)**.** *A stationary point process is a log-concave generalized linear model where the data likelihood of* $y_k \in \mathbb{N}$ *is a Poisson distribution* (A.17)

$$p\left(y_k|x_k\right) = \frac{\lambda^{(y_k)}e^{-\lambda}}{(y_k)!} \quad , \tag{3.7}$$

and where $\lambda = w^T x_k$ *is a nonstationary rate which depends on a regressor of previous output* $x_k = [y_{k-1}, ..., y_{k-n}]^T$*. The model is parameterized by* $\theta = \lambda$*. While there is no conjugate prior available for this model, this thesis will use a Gaussian distribution to reprint prior information:*

$$p(w) = \mathcal{N}\left(w|w^0, \Sigma_0\right) \quad , \tag{3.8}$$

where $w^0 \in \mathbb{R}^n$ *and* $\Sigma^0 \in \mathbb{R}^{n\times n}$ *are hyper-parameters.* ◇

3.2.2 GLHMM Definition

The definition of generalized linear hidden Markov models combines general linear models (Section 3.2.1) and hidden Markov models (Def. 2.4). This class of models and subsequent identification algorithms has been previously presented in [64, 65].

Definition 3.4 (Generalized Linear Hidden Markov Model)**.** *A GLHMM is a system* $\mathcal{G} = \{\mathcal{S}, \mathcal{U}, \mathcal{Y}, \Theta\}$ *where:*

1. $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$ *is a set of* N *unobservable discrete states, whose evolution is governed by a first order Markov process. At each* t_k*, let* m_k *denote the mode index, i.e., at* t_k *the system is in state* S_{m_k}*. The probability of switching between modes of the system is governed by a first order Markov chain with transition matrix* $A = [a_{ij}]$*:*

$$P\left(m_k = j|m_{k-1} = i\right) = a_{ij} \quad . \tag{3.9}$$

2. $y_k \in \mathcal{Y}$ *is the set of observed system outputs measured at* t_k*. The measurement takes the form* $y_k = \left[y_k^1, \ldots, y_k^n\right]^T$*, where each output* y_k^c *is referred to as a* channel*. Each channel* y_k^c *is modeled as a generalized linear model, with parameters that depend on the discrete mode* S_{m_k}*:*

$$y_k^c \sim f_c\left({g_c}^{-1}\left(\theta_{m_k}^c x_k^c\right)\right) \quad . \tag{3.10}$$

Here f_c is assumed to be a probability distribution *from the exponential family, and g_c is a* link function*. The* linear predictor*, $\theta^c_{m_k} x^c_k$, is composed of a regressor x^c_k of n_y previous outputs and n_u previous observations of other system covariates or inputs $u_k \in \mathcal{U}$:*

$$x^c_k = \left[y^c_{k-1}, \ldots, y^c_{k-n_y}, u_{k-1}, \ldots, u_{k-n_u} \right]^T \quad , \tag{3.11}$$

and a corresponding parameter vector $\theta^c_{m_k}$. The set of regressors for all channels at t_k is denoted $x_k = \{x^1_k, ..., x^C_k\}$.

3. *Θ is the set of all model parameters, including the transition kernel matrix A, and the system parameters θ^c_i, for $i = 1, ..., N$ and $c = 1, ..., C$.*

$\diamond$

Remark 3.1. The need for several *channels* of the form (3.10) arises from the application of this work to cortical neural prosthetics (Chapter 5), where multiple implanted electrodes record the electrical activity of several cortical neurons. The independence between channels is a good assumption for models of the firing of individual neurons. In some multiple output systems, this independence assumption may be inappropriate. However this independence can be relaxed in a straightforward manner. By introducing observations from other channels in the regressor (3.11), the dependence of one channel on another can be represented. This approach is used in Section 4.4 where correlation between regressor channels is required. $\diamond$

Remark 3.2. A Dirichlet prior distribution is defined for each row of the Markovian kernel (3.9):

$$\mathrm{Dir}\big([a_{i1}, a_{i2}, ..., a_{iN}] | [a^0_{i1}, a^0_{i2}, ..., a^0_{iN}]\big) \tag{3.12}$$

where a^0_{ij} are hyper-parameters. The prior parameters for θ^c_i will depend on the specific model (3.10) chosen for that channel. See Definitions 3.1 and 3.2 for details. $\diamond$

The complete-data likelihood of a GLHMM (Def. 3.4) of a sequence of length T follows directly from that of a HMM (2.47) and is given by:

$$\mathrm{p}\big(m_{1:T}, y_{1:T} | \Theta\big) = \mathrm{P}\big(m_1 | \pi\big)\, \mathrm{p}\big(y_1 | m_1, x_1, \theta_{m_1}\big) \prod_{k=2}^{T} \mathrm{P}\big(m_k | m_{k-1}, A\big)\, \mathrm{p}\big(y_k | m_k, x_k, \theta_{m_k}\big) \quad , \tag{3.13}$$

where, due to the independence between channels:

$$\mathrm{p}\big(y_k | m_k, x_k, \theta_{m_k}\big) = \prod_{c=1}^{C} \mathrm{p}\big(y^c_k | m_k, x^c_k, \theta^c_{m_k}\big) \quad . \tag{3.14}$$

Note that the regressors $x_1, ..., x_n$ contain previous observed outputs $y_{-n}, y_{1-n}, ..., y_0$ that are not directly incorporated into the model likelihood (3.13). These observations, which are labeled with

negative subscripts, do not have associated latent variables but are instead used to initialize the model so the data likelihood $\mathrm{p}\big(y_1|m_1, x_1, \theta_{m_1}\big)$ is defined.

Gibbs sampling and variational inference algorithms for the GLHMM class are now derived. A key step for inference in both of these algorithms is the forward-backward recursion for computing the marginal smoothing statistics of the posterior mode probabilities.

3.2.3 Forward-Backward Algorithm for GLHMM

Dynamic programming is again required for smoothing the posterior probability of the latent mode variables. This section shows that essentially the same forward-backward recursions used for HMMs (Section 2.3.1) can be used for GLHMMs. The only practical difference is the conditioning on the regressor, which contains previous values of the observed system state. The following forward and backward variables are defined:

$$\text{Forward Variables:} \qquad \alpha_k(i) \triangleq \mathrm{P}\big(m_k = i|y_{-n:k}, \Theta\big) \tag{3.15}$$

$$\text{Backwards Variables:} \qquad \beta_k(i) \triangleq \mathrm{P}\big(y_{k+1:T}|m_k = i, y_{-n:k}, \Theta\big) \ . \tag{3.16}$$

The essential difference in the forward-backward variables as compared to the HMM case is the presence of the regressor variables $y_{-n:k}$ in the definition backward variables β_k. This difference is necessary to incorporate the regressor x_k in the observation densities $\mathrm{p}\big(y_k|x_k, m_k, \theta_k\big)$. HMMs that incorporated regressors into the model, such as the auto-regressive HMM [10], utilize similar forward-backward recursions to those presented here. However the definitions of the variables (3.15) and (3.16) differ slightly from this earlier work, and the recursions are shown here for completeness.

Note that the conditional mode distribution $\mathrm{P}\big(m_{1:T}|y_{-n:k}, \Theta\big)$ can be expressed using the complete data likelihood (3.13), as was done for the HMM (2.58):

$$\log \mathrm{p}\big(m_{1:T}|y_{1:T}, \Theta\big) = \log \mathrm{P}\big(m_1|\pi\big) + \sum_{k=1}^{T} \log \mathrm{p}\big(y_k|m_k, x_k, \theta_{m_k}\big) + \sum_{k=2}^{T} \mathrm{P}\big(m_k|m_{k-1}, A\big) - \log \mathrm{P}\big(y_{1:T}|\Theta\big) \ , \tag{3.17}$$

where

$$\log \mathrm{p}\big(y_k|m_k, x_k, \theta_{m_k}\big) = \sum_{c=1}^{C} \log \mathrm{p}\big(y_k^c|m_k, x_k^c, \theta_{m_k}^c\big) \ . \tag{3.18}$$

Proposition 3.1 (Forward variable recursion for GLHMMs)**.** The forward variables (3.15) can be updated using:

$$\alpha_k(i) = \frac{\mathrm{p}\big(y_k|m_k = i, x_k, \theta_i\big)}{c_{y_k}} \sum_{j=1}^{N} a_{ji} \alpha_{k-1}(j) \tag{3.19}$$

where the normalizing constant is:

$$c_{y_k} = \mathrm{p}\big(y_k|y_{1:k-1},\Theta\big) = \sum_{i=1}^{N} \mathrm{p}\big(y_k|m_k=i,x_k,\theta_i\big) \sum_{j=1}^{N} a_{ji}\alpha_{k-1}(j) \ . \tag{3.20}$$

◇

Proof. Follows directly from the proof of the forward recursion (2.64) for HMMs (Proposition 2.3). □

Remark 3.3. The forward variable recursion allows the calculation of the incomplete data likelihood by taking the product of the normalizing constants: $\mathrm{p}\big(y_{1:T}|\Theta\big) = \prod_{k=1}^{T} \mathrm{p}\big(y_k|y_{1:k-1},\Theta\big)$. ◇

Proposition 3.2 (Backward variable recursion for GLHMM)**.** The GLHMM backward variables can be updated using:

$$\beta_{k-1}(i) = \sum_{j=1}^{N} \beta_k(j)\, \mathrm{p}\big(y_k|m_k=j,x_k,\theta_j\big)a_{ij} \ . \tag{3.21}$$

◇

Proof.

$$\begin{aligned}
\beta_{k-1}(i) &= \sum_{j=1}^{N} \mathrm{p}\big(y_{k:T}, m_k=j|m_{k-1}=i, y_{-n:k-1},\Theta\big) && \text{(Marginalization)}\\
&= \sum_{j=1}^{N} \mathrm{p}\big(y_{k+1:T}|m_k=j,y_{-n:k-1},\Theta\big)\,\mathrm{p}\big(y_k,m_k=j|m_{k-1}=i,y_{-n:k-1},\Theta\big) && \text{(Product Rule)}\\
&= \sum_{j=1}^{N} \beta_k(j)\,\mathrm{p}\big(y_k|m_k=j,y_{-n:k-1},\theta_j\big)\,\mathrm{p}\big(m_k=j|m_{k-1}=i,y_{-n:k-1},A\big) && \text{(Product Rule)}
\end{aligned}$$

The above equation is equivalent to equation (3.21) by using the definition of a_{ij} and noting that $\mathrm{p}\big(y_k|m_k=j,y_{-n:k-1},\theta_j\big) = \mathrm{p}\big(y_k|m_k=j,x_k,\theta_j\big)$. □

Scaling of the backward variables is required to avoid numerical underflow. The same method is used as for HMM (2.69), scaling by $\mathrm{p}\big(y_k|y_{1:k-1},\Theta\big)$. The new backwards recursion for the scaled backwards variables $\tilde{\beta}$ becomes:

$$\tilde{\beta}_{k-1}(i) = c_{y_k}^{-1} \sum_{j=1}^{N} \tilde{\beta}_k(j)\,\mathrm{p}\big(y_k|m_k=j,x_k,\theta_j\big)a_{ij} \tag{3.22}$$

and hence the scaled backwards variables $\tilde{\beta}_k(j)$ are expressed in terms of the original variables $\beta_k(j)$:

$$\tilde{\beta}_k(j) = \prod_{t=k}^{T} c_{y_k}^{-1}\beta_k(j) \ . \tag{3.23}$$

The forward and (scaled) backwards variables are now used to generate marginal statistics of the discrete state, and are exactly the same as for the standard HMM case. The marginal probability of the mode of the k^{th} data point is calculated using Bayes theorem as follows:

$$\gamma_k(i) = \mathrm{P}\big(m_k = i | y_{-n:T}, \Theta\big) \tag{3.24}$$

$$\propto \alpha_k(i)\beta_k(i) \ , \tag{3.25}$$

and:

$$\xi_k(i,j) = \mathrm{P}\big(m_k = i, m_{k+1} = j | y_{-n:T}, \Theta\big) \tag{3.26}$$

$$\propto \beta_{k+1}(j)\, \mathrm{p}\big(y_{k+1} | m_{k+1}, x_k, \theta_{m_{k+1}}\big) a_{ij} \alpha_k(i) \ . \tag{3.27}$$

3.3 Variational Bayes for Inference in GLHMMs

This section applies the variational framework for inference in GLHMM (Def. 3.4) models. This work builds on Section 2.3.2, which applied VB to HMMs. The aim of using VB for identifying GLHMM is the same: the posterior distribution of the model $\mathrm{p}\big(\Theta, m_{1:T} | y_{1:T}\big)$ is to be approximated by a factorized set of functions $q(m_{1:T})q(\Theta)$. Finding the (locally) best approximation of the posterior amounts to sequentially solving the following VB-E and VB-M update steps:

$$\textbf{VB-M step} \qquad q(\Theta) = \frac{1}{C_\Theta} \exp\left[\sum_{m_{1:T} \in \mathcal{S}^T} q(m_{1:T}) \log \mathrm{p}\big(y_{1:T}, m_{1:T} | \Theta\big) \right] \mathrm{p}\big(\Theta\big) \tag{3.28}$$

$$\textbf{VB-E step} \qquad q(m_{1:T}) = \frac{1}{C_{m_{1:T}}} \exp\left[\int q(\Theta) \log \mathrm{p}\big(y_{1:T}, m_{1:T} | \Theta\big) d\Theta \right] \ . \tag{3.29}$$

While these VB-E and VB-M steps appear the same as for HMM, there is added difficulty in their computation: the parameter space for GLHMMs, $\Theta = \{\pi, A, \theta_i^c, i = 1, \ldots, N, c = 1, \ldots, C\}$ is larger, and furthermore, instead of associating static distributions to each mode, the GLHMM is formed around a set of dynamical systems. The remainder of this section will develop solutions to the update equations (3.28) and (3.29).

3.3.1 VB-M Step

This section derives the VB-M step (3.28) for GLHMMs. This derivation is similar to the VB-M calculations for HMMs (Section 2.3.2.1), and some aspects of the derivation, such as the variational form of the transition matrix A and the initial state π, are the same as in HMMs, since the GLHMM definition has not changed this aspect of the model. Instead, the added complexity of identifying the GLHMM model class arises from the continuous dynamics associated with each mode.

Calculating the VB-M (3.28) step proceeds by substituting the complete data likelihood (3.13) for GLHMMs into (3.28) resulting in:

$$\log q(\Theta) = \log \mathrm{p}(\pi) + \sum_{m_{1:T}\in\mathcal{S}^T} q(m_{1:T}) \log \pi_{m_1} + \log \mathrm{p}(A) + \sum_{m_{1:T}\in\mathcal{S}^T} \sum_{k=2}^{T} q(m_{1:T}) \log \mathrm{P}\big(m_k|m_{k-1}, A\big) + \sum_{i=1}^{N}\sum_{c=1}^{C} \log p(\theta_i^c) + \sum_{m_{1:T}\in\mathcal{S}^T} \sum_{k=1}^{T}\sum_{c=1}^{C} q(m_{1:T}) \log \mathrm{P}\big(y_k^c|m_k, x_k^c, \theta_{m_k}^c\big) - \log C_\Theta \ . \quad (3.30)$$

The expression (3.30) for the variational distribution $q(\Theta)$ is similar to that of a HMM (2.74) in that many of the components of Θ are independent in (3.30). The distribution $q(\Theta)$ can be factored into: $q(\Theta) = q(\pi)q(A)\prod_{c=1}^{C} q(\theta_1^c, ..., \theta_N^c)$. Thus, the distribution $q(A)$ can be derived by considering all terms in (3.30) that are functions of A:

$$\log q(A) = \log \mathrm{p}\big(A\big) + \sum_{m_{1:T}\in\mathcal{S}^T} \sum_{k=2}^{T} q(m_{1:T}) \log \mathrm{P}\big(m_k|m_{k-1}, A\big) + C_A \ , \quad (3.31)$$

where C_A is a normalizing constant. Expression (3.31) is exactly the same as for a HMM, (2.75) and hence:

$$q(A) = \prod_{i=1}^{N} q(a_{i:}) \quad \text{where } q(a_{i:}) = \mathrm{Dir}\big(a_{i1}, ..., a_{iN}|[a_{i1}^T, ..., a_{iN}^T]\big) \ . \quad (3.32)$$

The priors (3.12) for the transition matrix A are of the same form used by HMMs (2.48), and hence the parameters of the Dirichlet distribution (3.32) are again: $a_{ij}^T = a_{ij}^0 + \sum_{k=2}^{T} \xi_k(i,j)$ (see (2.79)). Note that the marginal statistic $\xi_k(i,j)$ defined in (2.77) is used. The distribution $q(\pi)$ is also identical to that of HMM (2.86).

To complete the calculation of $q(\Theta)$ in equation (3.30), the distributions $q(\theta_1^c, ..., \theta_N^c)$, for $c = 1, ..., C$ are now considered. By collecting terms of (3.30) containing θ_i^c, for $i = 1, ..., N$:

$$\begin{aligned} &\log q(\theta_1^c, ..., \theta_N^c) \quad (3.33)\\ &= \sum_{i=1}^{N} \log p(\theta_i^c) + \sum_{i=1}^{N}\sum_{k=1}^{T}\sum_{S^T} q(m_{1:T})\delta(m_k = i) \log \mathrm{P}\big(y_k^c|m_k, x_k^c, \theta_{m_k}^c\big) + C_{\theta_{1:N}^c} \\ &= \sum_{i=1}^{N} \log p(\theta_i^c) + \sum_{i=1}^{N}\sum_{k=1}^{T} \gamma_k(i) \log \mathrm{P}\big(y_k^c|m_k = i, x_k^c, \theta_{m_k}^c\big) + C_{\theta_{1:N}^c} \ , \quad (3.34)\end{aligned}$$

where the $C_{\theta_{1:N}^c}$ is a constant, and the definition of the marginal statistic is the same as (2.82):

$$\gamma_k(i) \triangleq q(m_k = i) = \sum_{m_{1:T}\in\mathcal{S}^T} q(m_{1:T})\delta(m_k = i) \ , \quad (3.35)$$

and $\delta(\cdot)$ is Kronecker delta. By considering the form of (3.34) the distribution $q(\theta_1^c, ..., \theta_N^c)$ can be

further factorized: $q(\theta_1^c, ..., \theta_N^c) = \prod_{i=1}^N q(\theta_i^c)$ where,

$$\log q(\theta_i^c) = \log p(\theta_i^c) + \sum_{k=1}^{T} \gamma_k(i) \, \mathrm{P}\big(y_k^c | m_k = i, x_k^c, \theta_{m_k}^c\big) + C_{\theta_i^c} \quad , \tag{3.36}$$

where $C_{\theta_i^c}$ is a constant. The computation of equation (3.36) is the major difference between variational inference in HMM and GLHMMs. The remainder of this section will consider the computation of (3.36) for the conjugate GLM dynamics models, in particular that for AR-models (Def. 3.1) and for stationary Poisson point processes (Def. 3.2). Beal and Ghahramani [23] have previously noted that, in general, the variational posterior of conjugate exponential models can be analytically computed, and have the same form as the prior. However, the posterior of the conjugate (AR and Poisson) models still needs to be derived, and applied to (3.36).

Remark 3.4. The non-conjugate (but log-concave) GLM forms, such as the nonstationary point process (Def. 3.3) and logistic regression models (Def 3.3), are not considered in this section. The interested reader is referred to Jaakkola [77] for a tutorial on using the variational approach for log-concave GLMs, or to Bishop [24] for the case of logistic regression models. The derivations are left out of this thesis, as they require a straightforward but lengthy explanation, and require an additional step of optimizing nuisance parameters. Furthermore, for the neurological application in Chapter 5, considering only the AR and stationary point process models is sufficient. ◇

The computation of (3.36) for an AR-model (Def. 3.1) is derived in Appendix C.3. The result is repeated here for convenience:

$$q(\theta_i^c) = q(w_i^c, \tau_i^c) = \mathcal{N}\left(w_i^c | \hat{w}_i^c, (\tau \Lambda_i^c)^{-1}\right) \mathrm{Gam}\left(\tau_i^c | \alpha_i^c, \beta_i^c\right) \quad , \tag{3.37}$$

where the parameters $\hat{w}_i^c$, Λ_i^c, α_i^c, β_i^c parameterize the posterior distribution and are computed as follows:

$$\Lambda_i^c = \sum_{k=1}^{T} \gamma_k(i) x_k^c x_k^{cT} + \Lambda^0 \tag{3.38}$$

$$\hat{w}_i^c = (\Lambda_i^c)^{-1} \left(\sum_{k=1}^{T} \gamma_k(i) y_k x_k + \Lambda^0 w^0 \right) \tag{3.39}$$

$$\alpha_i^c = a^0 + \frac{1}{2} \sum_{k=1}^{T} \gamma_k(i) \tag{3.40}$$

$$\beta_i^c = b^0 + \frac{1}{2}(w^0 - \hat{w}_i^c)^T \Lambda^0 (w^0 - \hat{w}_i^c) + \frac{1}{2} \sum_{k=1}^{T} \gamma_k(i) \big(y - (\hat{w}_i^c)^T x_k^c\big) \quad , \tag{3.41}$$

and where Λ^0, w^0, a^0,b^0 are priors defined in (3.4).

The computation of (3.36) for a stationary Poisson point process (Def. 3.2) is derived in Appendix

C.4. The result is repeated here:

$$q(\theta_i^c) = q(\lambda_i^c) = \mathrm{Gam}\big(\lambda_i^c | \alpha_i^c, \beta_i^c\big) \ , \tag{3.42}$$

where the parameters α_i^c and β_i^c are computed:

$$\alpha_i^c = a^0 + \sum_{k=1}^{T} \gamma_k(i) y_k^c \tag{3.43}$$

$$\beta_i^c = b^0 + \sum_{k=1}^{T} \gamma_k(i) \ , \tag{3.44}$$

and where a^0 and b^0 are the priors defined in (3.6).

In summary, this section has computed the VB-M step (3.28) for GLHMM. Two important aspects of this derivation are highlighted: First, the VB-M step only required the marginal statistics γ_k and ξ_k of the variational distribution $q(m_{1:T})$. Secondly, it was shown that $q(\Theta)$ is factorized, without any further assumptions, into $q(\Theta) = q(\pi)q(A)\prod_{c=1}^{C}\prod_{i=1}^{N} q(\theta_i^c)$. The VB-E step, computed in the next section, will make use of these aspects of the VB-M step.

3.3.2 VB-E Step

The VB-E step (3.29) for GLHMMs is similar to that of HMMs, and amounts to solving the integral (3.29), the logarithm of which is presented here:

$$\log q(m_{1:T}) = \int q(\Theta) \log \mathrm{p}\big(y_{1:T}, m_{1:T} | \Theta\big) d\Theta + \log C_{m_{1:T}} \ . \tag{3.45}$$

The VB-E step for GLHMM is more difficult then HMM for two reasons: the integral over the parameters space, Θ, is higher dimensional, and the complete data likelihood (3.13) is more complicated. However the integral (3.45) is still tractable, by noting that the VB-M step showed that $q(\Theta)$ is factorized as $q(\Theta) = q(\pi)q(A)\prod_{i=1}^{N}\prod_{c=1}^{C} q(\theta_i^c)$. Using this factorization, and substituting the form of the complete data likelihood (3.13), the integral (3.45) is simplified (this is essentially the same simplification used for HMMs (2.88)):

$$\log q(m_{1:T}) = \log \tilde{\pi}_{m_k} + \sum_{k=1}^{T} \log \tilde{a}_{m_k m_{k-1}} + \sum_{k=1}^{T} \log \tilde{b}_{m_k}(y_k, x_k) - \log C_{m_{1:T}} \ , \tag{3.46}$$

where the following geometric means are defined:

$$\tilde{\pi}_i = \exp \int q(\pi) \log p(m_1 = i|\pi) d\pi \tag{3.47a}$$

$$\tilde{a}_{ij} = \exp \int q(A) \log p\left(m_k = j|m_{k-1} = i, A\right) dA \tag{3.47b}$$

$$\tilde{b}_i(y_k, x_k) = \exp \int \ldots \int \left[\prod_{c=1}^{C} q(\theta_i^c)\right] \sum_{c=1}^{C} \log p\left(y_k^c|m_k = i, x_k^c, \theta_i^c\right) d\theta_i^1 ... d\theta_i^C \ . \tag{3.47c}$$

The geometric mean (3.47c) is further simplified by commuting the sum and integral terms in (3.47c):

$$\tilde{b}_i(y_k, x_k) = \prod_{c=1}^{C} \tilde{b}_i^c(y_k, x_k) \ , \tag{3.48}$$

where:

$$\tilde{b}_i^c(y_k, x_k) = \exp \int q(\theta_i^c) \log p\left(y_k^c|m_k = i, x_k^c, \theta_i^c\right) d\theta_i^c \ . \tag{3.49}$$

Remark 3.5. Note that the geometric means (3.47) produce sub-normalized[3] probability distributions, e.g. $\int \tilde{b}_i^c(y_k, x_k) dy_k < 1$, by application of Jensen's inequality: $E[\log(x)] \leq \log(E[x])$ as log is a concave function, and where $E[\cdot]$ is the expectation. $\diamond$

The problem of evaluating these geometric means (3.47a), (3.49), and (3.47b) is now addressed. The geometric mean for the Markov transition matrix (3.47b), and for the initial parameter (3.47a) are exactly the same as in the HMM case (2.91), and will not be repeated here. The remaining problem to be solved is calculation of the geometric means $\tilde{b}_i^c(y_k, x_k)$, which will depend on the specific generalized linear models used. Two important GLM cases, the auto-regressive model (Def. 3.1) and the Poisson point process model (Def. 3.2) will be considered in this section due to their extensive use in Chapters 4 and 5.

For the AR model, the geometric mean (3.49) is expressed as follows:

$$\tilde{b}_i^c(y_k^c, x_k^c) = \exp \int_0^\infty \int_{\mathbb{R}^d} q(w_i^c, \tau_i^c) \ln p\left(y_k^c|m_k = i, x_k^c, w_i^c, \tau_i^c\right) dw_i^c d\tau_i^c \ . \tag{3.50}$$

This equation (3.50) can be analytically integrated, the derivation of which is done in Appendix C.1, but repeated here for convenience:

$$\tilde{b}_i^c(y_k^c, x_k^c) = \exp\left[-\frac{1}{2} x_k^{c\,T} \Lambda_i^{c-1} x_k^c - \frac{1}{2}\left(y_k^c - (\hat{w}_c^i)^T x_k^c\right)^2 \frac{\alpha_i^c}{\beta_i^c} + \frac{1}{2}\left(\psi(\alpha_i^c) - \log \beta_i^c\right) - \frac{1}{2}\log(2\pi)\right] \tag{3.51}$$

[3] A sub-normalized distribution, discussed originally in Section 2.3.2.2 for HMM, is proportional to a normalized probability distribution (i.e. that integrates to one), and where the proportionality constant is less then one.

For the stationary Poisson point process model (Def. 3.2), the geometric mean (3.49) is expressed as follows:

$$\tilde{b}_i^c(y_k^c, x_k^c) = \exp \int q(\lambda_i^c) \log p\left(y_k^c | m_k = i, \lambda_i^c\right) d\lambda_i^c \ . \tag{3.52}$$

This equation (3.52) can be analytically integrated (see Appendix C.2) to yield:

$$\tilde{b}_i^c(y_k^c, x_k^c) = \frac{1}{y_k^c!} \exp\left(y_k^c \left(\psi(\alpha_i^c) - \log(\beta_i^c)\right) - \frac{\alpha_i^c}{\beta_i^c}\right) \ . \tag{3.53}$$

Now that the geometric means (3.47) in expression (3.46) have been computed analytically, equation (3.46) can be calculated using the forward-backward algorithm (Section 3.2.3), as was done for HMM in (2.93). The forward variables are updated using the recursion:

$$\alpha_k(i) = \frac{\tilde{b}_i(y_k, x_k)}{c_{y_k}} \sum_{j=1}^{N} \tilde{a}_{ji} \alpha_{k-1}(j) \ . \tag{3.54}$$

After completing the the forward pass, the product of the normalizing constants, c_{y_k}, in (3.46) is:

$$C_{m_{1:T}} = \prod_{k=1}^{T} c_{y_k} \ .$$

The backwards filter β_k is similarly calculated as for HMM (2.96), as well as the calculation of the marginal statistics γ_k and ξ_k (see (2.97) and (2.98) respectively).

3.3.3 Calculation of the Lower Bound

The lower bound of the variational method for GLHMMs can be calculated after the VB-E step, and is expressed as:

$$\mathcal{L}(q(m_{1:T}, \Theta)) = \int_{\Theta} q(\Theta) \log \frac{\mathrm{p}(\Theta)}{q(\Theta)} d\Theta + \log C(m_{1:T}) \tag{3.55}$$

where (3.55) is derived from (2.104). The KL divergence from the parameter prior to the posterior can be simplified by substituting the factorized form: $q(\Theta) = q(\pi)q(A) \prod_{c=1}^{C} \prod_{i=1}^{N} q(\theta_i^c)$, which was derived during the VB-M step. The lower bound for GLHMMs (3.55) can then be written:

$$\begin{aligned} &\mathcal{L}(q(m_{1:T}, \Theta)) = \\ &\int_{\pi} q(\pi) \log \frac{\mathrm{p}(\pi)}{q(\pi)} d\Theta + \sum_{i=1}^{N} \int q(a_{i:}) \log \frac{\mathrm{p}(a_{i:})}{q(a_{i:})} da_{i:} + \sum_{i=1}^{N} \sum_{c=1}^{C} \int_{\theta_i^c} q(\theta_i^c) \log \frac{\mathrm{p}(\theta_i^c)}{q(\theta_i^c)} d\theta_i^c + \log C(m_{1:T}) \ . \end{aligned} \tag{3.56}$$

To allow efficient calculation of the lower bound, analytical expression for the KL divergences in (3.56) are derived in Appendix B. For example the KL divergence of the dynamics of channel c in mode i is expressed:

$$KL\big(q(\theta_i^c)||p(\theta_i^c)\big) = -\int_{\theta_i^c} q(\theta_i^c) \log \frac{\mathrm{p}\big(\theta_i^c\big)}{q(\theta_i^c)} d\theta_i^c \ . \tag{3.57}$$

The KL divergence (3.57) for the AR models posterior (3.37) is calculated in Appendix B.6. Likewise the KL-divergence (3.57) for the stationary Poisson point process posterior (3.42) is calculated in Appendix B.4. For an information theoretic perspective of why the lower bound (3.56) is useful for model selection, see Chapter 4.

3.4 A Gibbs Sampler for Inference in GLHMMs

This section derives both two-stage and multi-stage Gibbs samplers for GLHMMs [64, 65]. The multi-stage sampler is only required for non-conjugate GLMs (see Section 3.2.1), and will be derived in Section 3.4.1.1. A two-stage Gibbs sampling method for GLHMMs follows directly from the two-stage Gibbs sampler (Alg. 2.4) in Section 2.2.3. Algorithm 3.1 draws t_{max} samples from the joint distribution $\mathrm{p}\big(\Theta, m_{1:T}|y_{1:T}\big)$ of a GLHMM. Let the t^{th} sample of a variable, Θ, be denoted: $\hat{\Theta}^{(t)}$.

Algorithm 3.1 Two-Stage Gibbs Sampler for GLHMM

1: Initial mode estimate: $\hat{m}_{1:T}^{(0)}$
2: **for** $t = 0$ to t_{max} **do**
3: $\quad \hat{\Theta}^{(t+1)} \sim \mathrm{p}\big(\Theta|\hat{m}_{1:T}^{(t)}, y_{1:T}\big)$
4: $\quad \hat{m}_{1:T}^{(t+1)} \sim \mathrm{p}\big(m_{1:T}|\hat{\Theta}^{(t+1)}, y_{1:T}\big)$
5: **end for**

The core of algorithm involves sequentially drawing samples from the two conditional distributions, shown in lines 3 and 4 of Algorithm 3.1, and can be considered identification and classification steps respectively. Sections 3.4.1 and 3.4.2 describe how to efficiently draw samples from the required conditional distributions in Algorithm 3.1.

3.4.1 Parameter Estimation Step

Drawing samples from the conditional parameter distribution $\mathrm{p}\big(\Theta|m_{1:T}, y_{1:T}\big)$ is simplified by the independence assumptions made in Definition 3.4 of a GLHMM, and the form of the complete data likelihood (3.13):

$$\mathrm{p}\big(\Theta|m_{1:T}, y_{1:T}\big) = \prod_{i=1}^{N}\prod_{c=1}^{C} \mathrm{p}\big(\theta_i^c|m_{1:T}, y_{1:T}\big)\, \mathrm{p}\big(A|m_{1:T}, y_{1:T}\big) \ . \tag{3.58}$$

The data $y_{1:T}$ and corresponding regressor $x_{1:T}$ are split into discrete sets that depend on the discrete system state $m_{1:T}$:

$$\mathcal{Y}_i^c = \{y_k^c \ : \ m_k = i, \ k = 1, \ldots, T\} \quad \text{and} \quad \mathcal{X}_i^c = \{x_k^c \ : \ m_k = i, \ k = 1, \ldots, T\} \quad . \tag{3.59}$$

The problem of sampling from (3.58) is reduced to $NC + 1$ independent sampling problems:

$$\hat{\theta}_i^c \sim \mathrm{p}\big(\theta_i^c | \mathcal{Y}_i^c, \mathcal{X}_i^c\big) \text{ for } i = 1, ..., N \text{ and } c = 1, \ldots, C \tag{3.60}$$

$$\hat{A} \sim \mathrm{p}\big(A | m_{1:T}, y_{1:T}\big) \tag{3.61}$$

whereas the individual distributions $\mathrm{p}\big(\theta_i^c | \mathcal{Y}_i^c, \mathcal{X}_i^c\big)$ can be sampled from efficiently as they are in the conjugate GLM family. For instance, both the AR models (Def. 3.1) and the stationary Poisson point process (Def. 3.2) can be sampled from analytically. For instance, the AR model posterior, derived in Appendix C.3 is:

$$\mathrm{p}\big(w_i^c, \tau_i^c | \mathcal{Y}_i^c, \mathcal{X}_i^c\big) = \mathcal{N}\left(w_i^c | \bar{w}_i^c, (\tau_i^c \Lambda_i^c)^{-1}\right) \mathrm{Gam}\left(\tau_i^c | \alpha_i^c, \beta_i^c\right) \quad , \tag{3.62}$$

where the parameters $\hat{w}_i^c$, Λ_i^c, α_i^c, β_i^c parameterize the posterior distribution and are computed using (3.59) as follows:

$$\Lambda_i^c = \sum_{x_k \in \mathcal{X}_i^c} x_k^c {x_k^c}^T + \Lambda^0 \tag{3.63}$$

$$\bar{w}_i^c = (\Lambda_i^c)^{-1} \left(\sum_{\forall k : m_k = i} y_k x_k + \Lambda^0 w^0 \right) \tag{3.64}$$

$$\alpha_i^c = a^0 + \frac{1}{2} \sum_{k=1}^{T} \delta(m_k = i) \tag{3.65}$$

$$\beta_i^c = b^0 + \frac{1}{2}(w^0 - \hat{w}_i^c)^T \Lambda^0 (w^0 - \bar{w}_i^c) + \frac{1}{2} \sum_{\forall k : m_k = i} \left(y - (\bar{w}_i^c)^T x_k^c\right) \quad , \tag{3.66}$$

and where Λ^0, w^0, a^0, b^0 are priors defined in (3.4). Sampling from (3.62) requires sampling $\hat{\tau}_i^c \sim \mathrm{Gam}\left(\tau_i^c | \alpha_i^c, \beta_i^c\right)$, and then sampling $\hat{w}_i^c \sim \mathcal{N}\left(w_i^c | \bar{w}_i^c, (\hat{\tau}_i^c \Lambda_i^c)^{-1}\right)$. For information on sampling from common distributions such as the Gaussian and Gamma distributions see [25].

Conditioned on $m_{1:T}$, each row, $a_{i(1:N)}$, of the transition kernel A can be sampled from independently [37, 66]:

$$\mathrm{p}\big(A | m_{1:T}\big) = \prod_{i=1}^{N} \mathrm{p}\big(a_{i(1:N)} | m_{1:T}\big) \quad . \tag{3.67}$$

The conditional posterior of each row $\mathrm{p}\big(a_{i(1:N)}|m_{1:T}\big)$ is a Dirichlet distribution:

$$\mathrm{p}\big(a_{i(1:N)}|m_{1:T}\big) = \mathrm{Dir}\big(a_{i1}, a_{i2}, \ldots, a_{iN}|a_{i1}^T, a_{i2}^T, \ldots, a_{iN}^T\big) \tag{3.68}$$

where the posterior parameters a_{ij} are formed using the number of transitions in $m_{1:T}$ from S_i to S_j:

$$a_{ij}^T = a_{ij}^0 + \sum_{k=2}^{T} \delta\left(m_{k-1} = i\right) \delta\left(m_k = j\right) \quad . \tag{3.69}$$

See either [37] or [66] on how to sample from a Dirichlet distribution.

3.4.1.1 Multi-Stage Sampling for Non-Conjugate Models

In the case where the GLM dynamics are of non-conjugate (but log-concave) form, adaptive rejection sampling [76] can be used to simulate posterior samples. This requires the creation of a multi-stage Gibbs sampler (Alg. 3.2), which is essentially the same as the two-stage sampler (Alg. 2.4), but now samples from the parameters Θ in several stages.

In particular each parameter set θ_i^c can no longer be updated jointly (as in (3.60)), as it does not have a conjugate form. Instead it is assumed that $\theta_i^c = [\theta_i^c(1), ..., \theta_i^c(R)]$, and each element of $\theta_i^c(r)$ will be sampled conditioned on the remainder of the set θ_i^c:

$$\theta_i^c(r) \sim \mathrm{p}\left(\theta_i^c(r)\big|m_{1:T}, \theta_i^c(r^-), \theta_i^c(r^+) y_{1:T}\right) \tag{3.70}$$

where $\theta_i^c(r^-) = \{\theta_i^c(s) : s < r\}$ and $\theta_i^c(r^+) = \{\theta_i^c(s) : s > r\}$. Note that (3.70) is log-concave (as the GLM is log-concave) and will be sampled from using adaptive rejection sampling. A multi-stage Gibbs sampler, which sequentially samples from each $\theta_i^c(r)$ is given in Algorithm 3.2:

Algorithm 3.2 Multi-Stage Gibbs Sampler for GLHMM with log-concave GLM models

Initial estimate: $\hat{m}_{1:T}^{(0)}, \hat{\Theta}_{1:T}^{(0)}$
for $t = 0$ to t_{max} **do**
  for $c = 1$ to C **do**
    for $i = 1$ to N **do**
      for $r = 1$ to R **do**
        $\hat{\theta}_i^c(r)^{(t+1)} \sim \mathrm{p}\left(\theta_i^c(r) \middle| \hat{m}_{1:T}^{(t)}, \hat{\theta}_i^c(r^-)^{(t+1)}, \hat{\theta}_i^c(r^+)^{(t)}, y_{1:T}\right)$
      end for
    end for
  end for
  $\hat{A}^{(t+1)} \sim \mathrm{p}\big(A|\hat{m}_{1:T}^{(t)}\big)$
  $\hat{m}_{1:T}^{(t+1)} \sim \mathrm{p}\big(m_{1:T}|\hat{\Theta}^{(t+1)}, y_{1:T}\big)$
end for

3.4.2 Data Classification Step

The data classification step in both the multi-stage Gibbs sampler (Alg. 3.2) and two-stage sampler (Alg 3.1) requires sampling from the conditional mode distribution $\mathrm{p}\big(m_{1:T}|\Theta, y_{1:T}\big)$. Extending previous work, [64, 65] the entire mode sequence $m_{1:T}$ will be sampled in a single block update using the forward filter (3.15) and backward Markovian sampling [66].

The forward variables $\mathrm{p}\big(m_k = i|y_{1:k}, \Theta\big)$ for GLHMMs (3.15) are first evaluated for $k = 1:T$ using the recursion (3.19) in Section 3.2.3. The Markovian backwards sampler, shown in Algorithm 3.3, is then used to simulate the entire state sequence $m_{1:T}$ from $\mathrm{p}\big(m_{1:T}|\Theta, y_{1:T}\big)$.

Algorithm 3.3 Markovian Backward Sampling

$m_T \sim p\left(m_T|y_{1:T}, \Theta\right)$
for $k = T-1$ to 1 **do**
  $m_k \sim p\left(m_k|y_{1:k}, m_{k+1}, \Theta\right)$
end for

The simulation of $m_k \sim p\left(m_k|y_{1:k}, m_{k+1}, \Theta\right)$ in Algorithm 3.3 is accomplished using the forward variables and Bayes' theorem:

$$p\left(m_k = i|y_{1:k}, m_{k+1} = j, \Theta\right) = \frac{p\left(m_k = i|y_{1:k}, \Theta\right) p\left(m_{k+1} = j|m_k = i\right)}{\sum_{i=1}^{d} p\left(m_k =_i|y_{1:k}, \Theta\right) p\left(m_{k+1} = j|m_k = i\right)} \quad . \tag{3.71}$$

Equation (3.71) is a discrete distribution, and is evaluated using the forward variables $p\left(m_k = i|y_{1:k}, \Theta\right)$, and transition parameters $a_{ij} = p\left(m_{k+1} = j|m_k = i\right)$. Simulating a m_k from (3.71) can be accom-

plished by creating a vector:

$$P_{m_k} = \left[p\left(m_k = 1 | y_{1:k}, m_{k+1} = j, \Theta\right), ..., p\left(m_k = N | y_{1:k}, m_{k+1} = j, \Theta\right) \right] \ , \tag{3.72}$$

where $\sum P_{m_k} = 1$. Drawing a sample from (3.71) is accomplished by generating a random number from a uniform distribution on $[0, 1]$, and then choosing $m_k = i$, where i is the first element of the cumulative sum of P_{m_k} that is greater or equal to the random number.

3.5 Variational Methods for HSMM and VTHMM

This section develops variational learning algorithms for the *hidden semi-Markov model* (HSMM) and *variable transition hidden Markov model* (VTHMM). These models explicitly model the duration spent in each hidden discrete state with a probability distribution. This modeling of duration is equivalent to *clocks* in timed automata (Section 2.1.2), and hence provide useful framework for identification in these systems. HSMM and VTHMM are formally described below in Definition 3.5 and Definition 3.6 respectively.

The aim of this section is twofold: first apply variational inference methods to these systems. This is a novel development, and will allow use of the variational frameworks' model selection tools, and give the added benefit of approaching the inference problem from a Bayesian perspective. Secondly, and more inline with the goals of this thesis, is to associate dynamics with each mode of the VTHMM and HSMM, and hence devise a new class of hybrid systems with timed transitions. The addition of dynamics will directly follow from the GLHMM defined in Section 3.2.

In brief, a VTHMM is a superset of HSMMs [38], and provides more flexibility in modeling allowed transitions between the models' discrete states, but contains significantly more parameters. The HSMM, while having less flexibility, allows for more efficient learning algorithms, as recently formulated by Yu and Kobayashi [73, 79]. The survey paper [80] gives a useful comparison of VTHMMs and HSMMs in a maximum likelihood EM-based inference framework.

Definition 3.5 (Hidden Semi-Markov Model)**.** *A HSMM is a system* $\mathcal{G} = \{\mathcal{S}, \mathcal{D}, \mathcal{Y}, \Theta\}$*;*

1. $\mathcal{S} = \{S_1, \ldots, S_N\}$ *is a set of of* N *discrete states, or modes of* $\mathcal{G}$*. The system is denoted to be in state* S_i *mode at time* t_k *by* $m_k = i$*. The hidden state sequence for* $1 \leq k \leq T$ *is denoted* $m_{1:T} = \{m_1, m_2, ..., m_T\}$*.*

2. $\mathcal{D} = \{1, 2, ..., D\}$ *is a discrete set of durations, or time spent in a given state* S_i*. The remaining time, or residual time, of the current state* m_k *is denoted* τ_k*.*

3. *The joint process* (m_k, τ_k) *evolves in two stages: if* $(m_k, \tau_k) = (i, d)$*, where* $d \geq 1$*, then the semi-Markov chain deterministically counts down the residual time and transitions to* $(m_{k+1}, \tau_{k+1}) =$

$(i, d-1)$. *If* $(m_k, \tau_k) = (i, 1)$ *then system probabilistically transitions to* $(m_{k+1}, \tau_{k+1}) = (j, d)$ *for some* $j \neq i$, *according to the transition kernel* $[a_{ij}]$ *and the duration probability* $p_j(d)$, *where:*

$$\mathrm{P}\big(m_{k+1} = j | m_k = i, \tau_k = 1\big) = a_{ij} \quad \textit{where} \quad a_{ii} = 0 \tag{3.73}$$

$$\mathrm{P}\big(\tau_{k+1} = d | m_{k+1} = j, \tau_k = 1\big) = p_j(d) \ , \tag{3.74}$$

are Multinomial[4] *distributions such that* $0 \leq p_i(d) \leq 1$ *and* $\sum_{d=1}^{D} p_i(d) = 1$, *and* $0 \leq a_{ij} \leq 1$ *and* $\sum_{j=1}^{N} a_{ij} = 1$. *In more intuitive terms, if the system* $\mathcal{G}$ *transitions from mode* S_i *to* S_j *at step* k, *then the system will remain in state* S_j *for duration* $t_k = d$ *with probability* $p_j(d)$.

4. *The observed system output at* t_k *is denoted* $y_k \in \mathcal{Y}$. *The output* y_k *depends only on the current mode* m_k, *and is assumed to be parameterized by* θ_{m_k}. *The observed output state sequence from* $1 \leq k \leq T$ *is denoted* $y_{1:T} = \{y_1, y_2, ..., y_T\}$.

5. *The system's initial state condition,* $\pi \in \mathbb{R}^N$, *at* t_0, *is defined as:* $\pi(i, d) = \mathrm{p}\big(m_1 = i, \tau_1 = d\big)$.

6. *The set of all system parameters is denoted* Θ. *This includes the transition parameters* $A = [a_{ij}]$, *the initial condition* π, *the parameters* θ_{m_k} *for* $m_k = 1, .., N$, *and the duration probabilities* $p_i(d)$.

$\diamond$

This HSMM model class includes the explicit-duration HMM (EDHMM) developed by Ferguson [30], and the continuously variable duration hidden Markov Models (CVDHMM) developed by Levinson [31]. HSMM are also called segment models when each discrete mode is associated to a linear state space model [81].

Variable transition hidden Markov models (VTHMMs) which are defined in Definition 3.6, are also known as inhomogeneous HMMs (IHMM) [32], and non-stationary HMMs (NSHMM) [38, 33, 34].

Definition 3.6 (Variable Transition Hidden Markov Model)**.** *A VTHMM is a system* $\mathcal{G} = \{\mathcal{S}, \mathcal{D}, \mathcal{Y}, \Theta\}$;

1. $\mathcal{S} = \{S_1, \ldots, S_N\}$ *is a set of of* N *discrete states, or modes of* $\mathcal{G}$. *The system is said to be in mode* S_i *at time* t_k *when* $m_k = i$. *The hidden state sequence for* $1 \leq k \leq T$, *is denoted* $m_{1:T} = \{m_1, m_2, ..., m_T\}$.

2. $\mathcal{D} = \{1, 2, ..., D\}$ *is a discrete set representing the duration spent in the current mode* S_i. *At time* t_k *the length of time, or duration, that the system has remained in the mode* m_k *is denoted by* τ_k.

[4] The distribution $\mathrm{P}\big(\tau_{k+1} = d | m_{k+1} = j, \tau_k = 1\big)$ can easily be changed to a parametric distribution like the Poisson distribution, which is discussed in [38].

3. *The system state m_k evolves according to a non-stationary Markov process:*

$$\mathrm{P}\big(m_{k+1} = j | m_k = i, \tau_k = d\big) = a_{ij}(d) \ . \tag{3.75}$$

The duration evolves deterministically: if $(m_k, \tau_k) = (i, d)$ and $m_{k+1} = i$ then $\tau_{k+1} = d + 1$, otherwise if $m_{k+1} \neq i$ then $\tau_{k+1} = 1$. Furthermore if $(m_k, \tau_k) = (i, D)$ and $m_{k+1} = i$ then $\tau_{k+1} = D$, i.e., the state $\tau_k = D$ is a catch-all state[5].

4. *The observed system output at t_k is denoted $y_k \in \mathcal{Y}$. The output y_k depends only on the current mode m_k, and is assumed to be parameterized by θ_{m_k}. The data likelihood is denoted $\mathrm{p}\big(y_k | m_k, \theta_{m_k}\big)$. The observed output state sequence from $1 \leq k \leq T$ is denoted $y_{1:T} = \{y_1, y_2, ..., y_T\}$.*

5. *The system initial condition π, is defined as: $\pi_i d = \mathrm{p}\big(m_1 = i, \tau_1 = d\big)$[6].*

6. *The set of all system parameters is denoted Θ. This includes the non-stationary transition parameters $A = \{a_{ij}(d)\}$, for $d = 1, ..., D$, the initial condition π, and the parameters θ_{m_k} for $m_k = 1, .., N$.*

$\diamond$

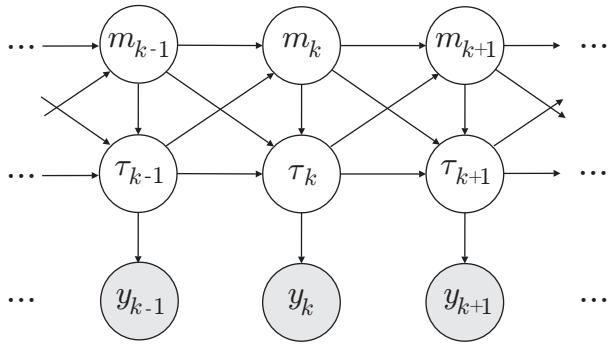

Figure 3.1: Directed acyclic graph representation of a variable transition hidden Markov model, which gives the conditional dependence of system variables. Grey nodes represent observed data y_k.

Remark 3.6. The typical definitions of VTHMM and HSMM use discrete (Multinomial) distributions for the conditional observation densities $\mathrm{p}\big(y_k | m_k, \theta_{m_k}\big)$ (see Def. 3.5, item 4, and Def 3.6, item 4). However in this thesis the interest is in associating continuous dynamics with each discrete mode,

[5] In the VTHMM definitions [32, 33] it is not explicitly defined what happens when $\tau_k = D$, and $m_{k+1} = m_k$. This is because both [32] and [33] define $D = k$, the total length of the sequence until the current time step for all calculations. However, in both papers the authors later state that the duration is constrained to a fixed $D < k$, but do not explicitly change the provided derivations. Here we will assume that if $(m_k, \tau_k) = (i, D)$ and $m_{k+1} = i$ then $\tau_{k+1} = D$.

[6] Note that in some definitions, the initial condition is forced to start with $t_1 = 1$, implying that $\pi_i d = 0$ if $d \neq 1$. While this is sensible in some situations, it is not appropriate for all circumstances and hence will not be directly enforced here. If it is known that the duration $t_1 = 1$, this fact can be enforced through a judicious choice of prior parameters

instead of a stationary discrete distribution, and so the specific form of $\mathrm{p}\big(y_k|m_k,\theta_{m_k}\big)$ is left undefined in the above definitions. Based on the GLHMM Section 3.2 of this thesis, we will use generalized linear models to represent $\mathrm{p}\big(y_k|m_k,\theta_{m_k}\big)$, and these distributions will additionally condition upon a regressor of previous observations x_k. This deviation from the original definitions (Def. 3.5 & 3.6) will be remarked upon as necessary to facilitate the inference of VTHMM and HSMM with GLM dynamics. ◇

The key difference between a HSMM and VTHMM is the complexity and flexibility of the kernel which models transitions between discrete states. The VTHMM has significantly more flexibility in the ability to model discrete transitions, but this added flexibility should not always be employed. For a 10-mode system ($N = 10$), with maximum duration of $D = 100$, the VTHMM will have $N^2D = 10000$ transition parameters to identify; where the HSMM will have $ND + N^2 = 1100$ parameters (and further compare to a HMM with $N^2 = 100$ parameters). The added compactness of the HSMM stems from the deterministic counting down in each mode. While at any duration, $\tau_k = d$, the VTHMM can transition to another discrete state, the HSMM is constrained to only transition when $\tau_k = 1$. The choice of using a VTHMM or HSMM is then application dependent and will depend on the size of the problem and amount of training data available; in the neurophysiological applications of this thesis the HSMM model framework will be used.

Because a Bayesian perspective is used for inferring model parameters, prior distributions for the VTHMM and HSMM are now defined. These priors are based on the work of Djuric and Chun [38], which is one of the few times priors have been introduced into the VTHMM and HSMM frameworks.

Definition 3.7 (Prior Distribution for HSMM)**.** *Because distributions $p_i(d)$ and a_{ij} from definition 3.5 are Multinomial, a suitable conjugate prior is the Dirichlet distribution. The prior distribution of a_{ij} is similar to that in GLHMM and HMMs, where each row of $a_{i:}$ is independent:*

$$Dir\big(a_{i1},...,a_{ii-1},a_{ii+1},...,a_{iN}|[a^0_{i1},...,a^0_{ii-1},a^0_{ii+1},...,a^0_{iN}]\big) \ , \tag{3.76}$$

but where $a_{ii} = 0$ by definition of a HSMM. The prior distribution of $p_i(:)$ is independent from other modes ($j \neq i$), where for each mode S_i the prior probability of the duration spent in that mode is modeled with a Dirichlet distribution:

$$Dir\big(p_i(1),...,p_i(D)|p^0_i(1),...,p^0_i(D)\big) \ . \tag{3.77}$$

◇

The prior distribution for VTHMM is now defined:

Definition 3.8 (Prior Distribution for VTHMM)**.** *The prior distribution for $A(d) = \{a_{ij}(d)\}$ is*

defined as a Dirichlet distribution for each row $a_{i:}(d)$ *at each time* $d = 1, ..., D$.

$$Dir\big(a_{i1}(d), ..., a_{iN}(d) | [a^0_{i1}(d), ..., a^0_{iN}(d)]\big) \ , \tag{3.78}$$

this essentially just defines a new transition kernel for each possible duration d. ◇

There already exist several inference algorithms for both the HSMM and VTHMM models. Gibbs sampling for these methods was introduced by Djuric and Chung [38]. This method is fully Bayesian and each sampling step is efficient, however convergence may be very slow [38, 66]. There also exist several approximate EM algorithms, where the E-step is replaced by generating the maximum likelihood state sequence using the Viterbi algorithm [34]. This has the advantage of segmenting the data in the corresponding E-step, and makes counting (or estimating the duration of each state) easier.

The variational methods developed here are an extension of existing EM methods already developed for these model classes. The most efficient EM algorithms for both HSMM and VTHMM are used: the EM algorithms [33] and [32] developed for VTHMM, and the recently developed methods by Yu and Kobayashi [73, 79] for HSMMs are considered. This later method provides an algorithm with computational complexity $\mathrm{O}(N^2T + NDT)$, as opposed to most methods with complexity $\mathrm{O}(N^2DT)$, as noted in the survey [80].

To develop variational methods for variable transition hidden Markov models (VTHMM) and hidden semi-Markov models (HSMM), three separate steps will be taken:

1. The VTHMM and HSMM models will be reformulated as stationary hidden Markov models in the joint space $(m_k, \tau_k) \in \mathcal{S} \times \mathcal{D}$. This will allow direct application of the variational Bayesian method developed for GLHMM, and has computational complexity $\mathrm{O}(N^2D^2T)$.

2. Specific forward-backward recursions [32] developed for the VTHMMs are utilized to improve the computational efficiency of the variational inference algorithm. This results in an inference algorithm with computational complexity $\mathrm{O}(N^2DT)$.

3. The HSMM recursions developed by Yu and Kobayashi, [73], are applied, resulting in a variational inference algorithm with computation complexity of $\mathrm{O}(N^2T + NDT)$.

In all of the above steps the lower bound for the model will be derived, demonstrating the continued usefulness of the VB method for model selection.

3.5.1 VTHMM and HSMM as Embeddings in a Stationary HMM

This section demonstrates that both the VTHMM and HSMM models, and hybrid systems based on these models, can be embedded in a stationary hidden Markov model. This is equivalent to

considering the joint process (m_k, τ_k) as a single Markov chain, and allows the variational inference algorithms presented in Section 3.3 to be utilized directly. First the stationary HMM is defined, allowing explicit mappings from the VTHMM, HSMM, and associated hybrid system models into the stationary HMM to be formulated. The resulting mappings and variational updates can then be used to perform variational Bayesian inference.

Definition 3.9 (Stationary HMM in Joint Space $\mathcal{S} \times \mathcal{D}$)**.** *Consider a stationary HMM with discrete state (m_k, τ_k) as a special case of the standard HMM (Def. 2.4). This HMM is an embedding of either a HSMM or VTHMM model, and has the following properties:*

1. *The discrete state transitions are governed by a stationary HMM chain with kernel Φ such that:*

$$\mathrm{p}\big((m_k, \tau_k) = (j, d') | (m_{k-1}, \tau_{k-1}) = (i, d)\big) = \phi_{(i,d)(j,d')} \ . \tag{3.79}$$

   Here, the state m_k is denoted the mode*, and the state τ_k is denoted the* duration*.*

2. *The observation probabilities associated with each state (m_k, τ_k) are independent of the duration[7]:*

$$\mathrm{p}\big(y_k | m_k, \tau_k, \Theta\big) = \mathrm{p}\big(y_k | m_k, \theta_{m_k}\big) \ . \tag{3.80}$$

3. *The initial condition of the system is defined as:*

$$\pi_{(j,d)} \triangleq \mathrm{P}\big((m_1, \tau_1) = (j, d)\big). \tag{3.81}$$

4. *The specific parameter set associated Θ with the stationary HMM will depend on the type of system embedded: for a HSMM model, $\Theta = \{\pi, p_j(d), a_{ij}, \theta_{m_k}\}$, and for a VTHMM model, $\Theta = \{\pi, a_{ij}(d), \theta_{m_k}\}$ for $i, j = 1, ..., N$ and $d = 1, ..., D$.*

$\diamond$

A mapping between a VTHMM and the stationary HMM (Def. 3.9) is now derived using the definition of a VTHMM (Def. 3.6):

- The transition kernel Φ of the HMM (3.79) is an injective map of the VTHMM:

$$\begin{aligned} \phi_{(i,d)(j,d')} &\triangleq \mathrm{P}\big((m_{k+1}, \tau_{k+1}) = (j, d') | (m_k, \tau_k) = (i, d)\big) \\ &= \begin{cases} a_{ij}(d) & \text{if } \{j = i, d' = d + 1\} \text{ or } \{j = i, d = D\} \text{ or } \{j \neq i, d' = 1\} \\ 0 & \text{else .} \end{cases} \end{aligned} \tag{3.82}$$

[7]The distribution $\mathrm{p}\big(y_k | m_k, \theta_{m_k}\big)$ will later be modified the form used in GLHMMs (Def 3.4), but is left ambiguous here to remain consistent with typical definitions of HSMM and VTHMM.

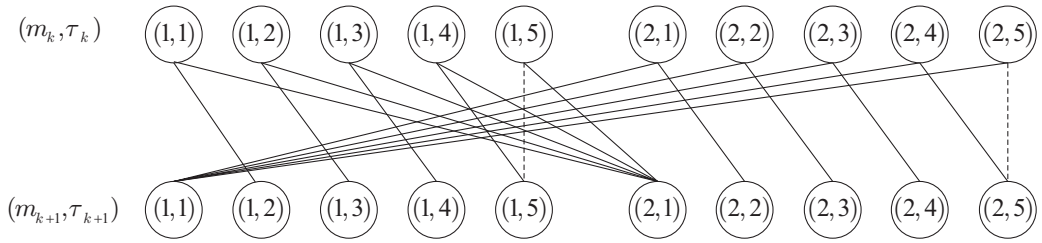

Figure 3.2: Allowable connections for a 2-state VTHMM with a maximum duration of $D = 5$. Note that the dotted connections represent the ability to transition into the same state when $\tau_k = D$. As demonstrated above, there are only 20 possible connections which can be solved with $\mathrm{O}(N^2 D)$ complexity. For the fully connected graph above there would be 100 connections, or $\mathrm{O}(N^2 D^2)$ complexity.

The prior distribution over the parameters $a_{ij}(d)$ is defined by (3.78). The zero components of the kernel Φ do not have an associated prior, as they are deterministic.

- The conditional observation densities $\mathrm{p}\big(y_k|m_k,\theta_{m_k}\big)$ in the HMM 3.80 and the initial condition π are the same as in Definition 3.6 of a VTHMM.

An example of this embedding for VTHMM systems into a stationary HMM is shown in Fig. 3.2 for a system with two modes $N = 2$ and a maximum duration of $D = 5$.

A similar mapping between a HSMM and the stationary HMM (Def 3.9) is derived using the definition (Def 3.5):

- The transition kernel Φ of the HMM (3.79) is a injective map of the HSMM:

$$\begin{aligned}\phi_{(i,d)(j,d')} &\triangleq \mathrm{P}\big((m_{k+1},\tau_{k+1}) = (j,d')|(m_k,\tau_k) = (i,d)\big)\\ &= \begin{cases} 1 & \text{if } j = i, d > 1, d' = d-1\\ a_{ij}p_j(d') & \text{if } d = 1\\ 0 & \text{else .}\end{cases}\end{aligned} \tag{3.83}$$

The prior distribution over the parameters a_{ij} and $p_j(d)$ is defined by (3.77). The constant components (taking values in $\{0,1\}$) of the kernel Φ do not have an associated prior, as they are deterministic.

- The conditional observation densities $\mathrm{p}\big(y_k|m_k,\theta_{m_k}\big)$ and initial condition π are the same in Definition 3.5 of a HSMM.

To illustrate the mapping between a VTHMM and the HMM, consider a 2-state VTHMM with a maximum duration of 5. The stationary HMM kernel is then:

$$\Phi = \left[\phi_{(i,d)(j,d')}\right] = \begin{bmatrix} 0 & a_{11}(1) & 0 & 0 & 0 & a_{12}(1) & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{11}(2) & 0 & 0 & a_{12}(2) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{11}(3) & 0 & a_{12}(3) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{11}(4) & a_{12}(4) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{11}(5) & a_{12}(5) & 0 & 0 & 0 & 0 \\ a_{21}(1) & 0 & 0 & 0 & 0 & 0 & a_{22}(1) & 0 & 0 & 0 \\ a_{21}(2) & 0 & 0 & 0 & 0 & 0 & 0 & a_{22}(2) & 0 & 0 \\ a_{21}(3) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{22}(3) & 0 \\ a_{21}(4) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{22}(4) \\ a_{21}(5) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{22}(5) \end{bmatrix},$$

which in conjunction with Figure 3.2, is both useful for visualizing the embedding, as well as demonstrates the number of deterministic (or zero probability) transitions in the stationary HMM.

The complete data likelihood of the stationary HMM (Def. 3.9) follows directly from that of a HMM (2.47):

$$\mathrm{p}\big(m_{1:T}, \tau_{1:T}, y_{1:T}|\Theta\big) = \mathrm{P}\big(m_1, \tau_1|\pi\big)\,\mathrm{p}\big(y_1|m_1, \theta_{m_1}\big) \prod_{k=2}^{T} \mathrm{P}\big(m_k, \tau_k|m_{k-1}, \tau_{k-1}, A\big)\,\mathrm{p}\big(y_k|m_k, \theta_{m_k}\big) \ . \tag{3.84}$$

To apply variational Bayes for inference in the stationary HMM, the VB-E and VB-M steps are modified from the standard GLHMM (3.29) and (3.28) by applying definition (Def. 3.9):

$$\textbf{VB-M:} \quad q(\Theta) = \frac{1}{C_\Theta} \exp\left[\sum_{(m_{1:T},\tau_{1:T})\in(\mathcal{S}\times\mathcal{D})^T} q(m_{1:T}, \tau_{1:T}) \log \mathrm{p}\big(y_{1:T}, m_{1:T}, \tau_{1:T}|\Theta\big)\right] \mathrm{p}\big(\Theta\big) \tag{3.85}$$

$$\textbf{VB-E:} \quad q(m_{1:T}, \tau_{1:T}) = \frac{1}{C_{m\tau}} \exp\left[\int q(\Theta) \log \mathrm{p}\big(y_{1:T}, m_{1:T}, \tau_{1:T}|\Theta\big) d\Theta\right] \ , \tag{3.86}$$

where the approximation that the variational posterior factorizes has been made: $q(\Theta, m_{1:T}, \tau_{1:T}) = q(\Theta)q(m_{1:T}, \tau_{1:T})$. As discussed in Section 2.2.1, sequentially solving the VB-E step (3.86) and the VB-M step (3.86) will (locally) maximize the variational lower bound. This lower bound maximization is equivalent to minimizing the KL divergence from the full posterior $\mathrm{p}\big(\Theta, m_{1:T}, \tau_{1:T}|y_{1:T}\big)$ to the factorized distribution $q(\Theta)q(m_{1:T}, \tau_{1:T})$, hence providing the "best" variational approximation possible. The remainder of this section will provide solutions to the VB-E (3.86) and VB-M (3.85) steps for the stationary joint-space HMM (Def. 3.9).

3.5.1.1 VB-M step for the Joint Space HMM

The computation of the VB-M step (3.85) is very similar to that derived for HMM (Section 2.3.2.1) and GLHMM (Section 3.3). The only effective difference in the the VB-M step (3.85) compared to the VB-M step for GLHMMs (3.28) is the additional summation over the latent variables $\tau_{1:T}$. This has little practical effect in the VB-E step, because of the invariance of the densities:

$$\mathrm{p}\big(y_k|m_k,\tau_k,\Theta\big) = \mathrm{p}\big(y_k|m_k,\theta_{m_k}\big) \ . \tag{3.87}$$

Next the PDF (3.87) must be modified to include the GLM dynamics defined Section 3.14:

$$\mathrm{p}\big(y_k|m_k,x_k,\theta_{m_k}\big) = \prod_{c=1}^{C} \mathrm{p}\big(y_k^c|m_k,x_k^c,\theta_{m_k}^c\big) \ . \tag{3.88}$$

Calculation of the VB-M step (3.85) step proceeds by substituting the complete data likelihood (3.84) into (3.85) resulting in:

$$\begin{aligned}\log q(\Theta) = \log \mathrm{p}\big(\pi\big) + \sum_{\substack{(m_{1:T},\tau_{1:T})\\ \in(\mathcal{S}\times\mathcal{D})^T}} q(m_{1:T},\tau_{1:T}) \log \pi_{m_1,\tau_1} + \\ \log \mathrm{p}\big(\Phi\big) + \sum_{\substack{(m_{1:T},\tau_{1:T})\\ \in(\mathcal{S}\times\mathcal{D})^T}} \sum_{k=2}^{T} q(m_{1:T},\tau_{1:T}) \log \mathrm{P}\big(m_k,\tau_k|m_{k-1},\tau_{k-1},\Phi\big) + \\ \sum_{i=1}^{N}\sum_{c=1}^{C} \log p(\theta_i^c) + \sum_{\substack{(m_{1:T},\tau_{1:T})\\ \in(\mathcal{S}\times\mathcal{D})^T}} \sum_{k=1}^{T}\sum_{c=1}^{C} q(m_{1:T},\tau_{1:T}) \log \mathrm{P}\big(y_k^c|m_k,x_k^c,\theta_{m_k}^c\big) - \log C_\Theta \ .\end{aligned} \tag{3.89}$$

Note that the distributions associated with the GLMs (3.88) are used. Expression (3.89) for the variational distribution $q(\Theta)$ is similar to that of a GLHMM (3.30), and again many of the model parameters Θ are independent. Without further assumption, the distribution $q(\Theta)$ is factored into: $q(\Theta) = q(\pi)q(\Phi)\prod_{c=1}^{C} q(\theta_1^c,...,\theta_N^c)$. The major difference between this variational expression (3.89) and that of a GLHMM is the added complexity of summing over $\tau_{1:T}$. While this will not have much effect on some of the factors, it does change the calculation of $q(\Phi)$ (as opposed to $q(A)$ in GLHMMs). The distribution $q(\Phi)$ can be derived by considering all terms in (3.89) that are functions of $q(\Phi)$:

$$\log q(\Phi) = \log \mathrm{p}\big(\Phi\big) + \sum_{\substack{(m_{1:T},\tau_{1:T})\\ \in(\mathcal{S}\times\mathcal{D})^T}} q(m_{1:T},\tau_{1:T}) \sum_{k=2}^{T} \log \mathrm{P}\big(m_k,\tau_k|m_{k-1},\tau_{k-1},\Phi\big) + C_\Phi \ , \tag{3.90}$$

where C_Φ is a normalizing constant. Again a marginal statistic of $q(m_{1:T},\tau_{1:T})$ is used, now defined

as:

$$\xi_k(i,d,j,d') \triangleq q\Big((m_k,\tau_k)=(j,d'),(m_{k-1},\tau_{k-1})=(i,d)\Big) = \sum_{\substack{(m_{1:T},\tau_{1:T})\\ \in(\mathcal{S}\times\mathcal{D})^T}} q(m_{1:T},\tau_{1:T})\delta\Big((m_k,\tau_k)=(j,d'),(m_{k-1},\tau_{k-1})=(i,d)\Big) \ , \quad (3.91)$$

where $\delta(\cdot)$ is the Kronecker delta function. The statistic (3.91) allows $q(\Phi)$ in (3.90) to be simplified:

$$\log q(\Phi) = \log \mathrm{p}\big(\Phi\big) + \sum_{k=2}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{d=1}^{D}\sum_{d'=1}^{D} \xi_k(i,d,j,d') \log \mathrm{P}\big(m_k,\tau_k|m_{k-1},\tau_{k-1},\Phi\big) + C_\Phi \ . \quad (3.92)$$

At this point it is more efficient to consider the HSMM and VTHMM cases separately, since the form of the transition matrix Φ is difference for each of the embeddings (3.82) and (3.83). For the VTHMM model, where $q(\Phi) = q(a_{ij}(d))$, the marginal statistic $\xi_k(i,d,j,d')$ is only non-zero for the cases $\xi_k(i,d,i,d+1)$, $\xi_k(i,D,i,D)$, and $\xi_k(i,d,j,1)$, where $j \neq i$. Hence, in the case of VTHMM (3.90) can be written:

$$\begin{aligned}\log q(\Phi) \triangleq \log q(\{a_{ij}(d)\}) = &\sum_{d=1}^{D}\sum_{j=1}^{N}\sum_{i=1}^{N} \log a_{ij}(d)^{a^0_{ij}(d)-1} \\ &+ \sum_{k=2}^{T}\sum_{i=1}^{N}\sum_{d=1}^{D} \xi_k(i,d,i,d+1)\log a_{ii}(d) + \sum_{k=2}^{T}\sum_{i=1}^{N} \xi_k(i,D,i,D)\log a_{ii}(D) \\ &+ \sum_{k=2}^{T}\sum_{j=1}^{N}\sum_{i=1}^{N}\sum_{d=1}^{D} \xi_k(i,j,d,1)\log a_{ij}(d) + C_\Phi \ , \quad (3.93)\end{aligned}$$

where the prior form of $q(a_{ij}(d))$ is taken from (3.78). Equation (3.93) is proportional to the product of Dirichlet distributions, with all rows $a_{i:}(d)$ of the transition matrix for every duration d being independent:

$$\textbf{VTHMM:} \quad q(\Phi) = q(\{a_{ij}(d)\}) = \prod_{d=1}^{D}\prod_{i=1}^{N} q(a_{i:}(d)) \ , \quad (3.94)$$

where:

$$q(a_{i:}(d)) = \mathrm{Dir}\big(a_{i1}(d),...,a_{iN}(d)|[a^T_{i1}(d),...,a^T_{iN}(d)]\big) \quad (3.95)$$

and:

$$a^T_{ij}(d) = a^0_{ij}(d) + \begin{cases} \sum_{k=2}^{T}\xi_k(i,d,j,1) & \text{if } j \neq i \\ \sum_{k=2}^{T}\xi_k(i,d,i,d+1) & \text{if } j = i \text{ and } d < D \\ \sum_{k=2}^{T}\xi_k(i,D,i,D) & \text{if } j = i \text{ and } d = D \end{cases} \ . \quad (3.96)$$

Equation (3.92) is now computed for the HSMM case, where $q(\Phi) = q(\{a_{ij}\},\{p_i(d)\})$. By using

the embedding (3.82) for HSMM, it is computed that:

$$\textbf{HSMM:}\;\; q(\Phi) = q(\{a_{ij}\},\{p_i(d)\}) = \left[\prod_{i=1}^{N} q(a_{i:})\right]\left[\prod_{j=1}^{N} q(p_j(:))\right] \;\;, \tag{3.97}$$

where the distributions in (3.97) are Dirichlet distributions:

$$q(a_{i:}) = \mathrm{Dir}\big(a_{i1},...,a_{ii-1},a_{ii+1},...,a_{iN}|a_{i1}^T,...,a_{ii-1}^T,a_{ii+1}^T,...,a_{iN}^T\big) \tag{3.98}$$

$$q(p_i(:)) = \mathrm{Dir}\big(p_i(1),...,p_i(D)|p_i^T(1),...,p_i^T(D)\big) \;\;, \tag{3.99}$$

with parameters:

$$p_j^T(d) = p_j^0(d) + \sum_{k=2}^{T}\sum_{\forall i\neq j}\xi_k(i,1,j,d) \tag{3.100}$$

$$a_{ij}^T = a_{ij}^0 + \sum_{k=2}^{T}\sum_{d=1}^{D}\xi_k(i,1,j,d) \quad \text{for} \quad j\neq i \;\;, \tag{3.101}$$

and $a_{ii}=0$ by the definition of a HSMM.

In summary, we have so far calculated the distribution $q(\Phi)$ using (3.89). It remains to calculate the distributions $q(\theta_1^c,...,\theta_N^c)$ and $q(\pi)$. First the calculation of $q(\theta_1^c,...,\theta_N^c)$ is addressed, whereby collecting all terms of θ_i^c in (3.89):

$$q(\theta_1^c,...,\theta_N^c) = \sum_{i=1}^{N}\log p(\theta_i^c) + \sum_{\substack{(m_{1:T},\tau_{1:T})\\ \in(\mathcal{S}\times\mathcal{D})^T}}\sum_{k=1}^{T} q(m_{1:T},\tau_{1:T})\log \mathrm{P}\big(y_k^c|m_k,x_k^c,\theta_{m_k}^c\big) - \log C_{\theta_{1:N}^c} \;\;. \tag{3.102}$$

Following from the GLHMM derivation (3.34), it can be shown that by defining the following marginal statistic:

$$\gamma_k(i,d) \triangleq q\big((m_k,\tau_k)=(i,d)\big) = \sum_{\substack{(m_{1:T},\tau_{1:T})\\ \in(\mathcal{S}\times\mathcal{D})^T}} q(m_{1:T},\tau_{1:T})\delta\big((m_k,\tau_k)=(i,d)\big) \;\;, \tag{3.103}$$

where $\delta(\cdot)$ is Kronecker delta, the distribution $q(\theta_1^c,...,\theta_N^c)$ in (3.102) is factorized without any further assumptions: $q(\theta_1^c,...,\theta_N^c) = \prod_{i=1}^N q(\theta_i^c)$. Furthermore, the distribution $q(\theta_i^c)$ is given by:

$$\log q(\theta_i^c) = \log p(\theta_i^c) + \sum_{k=1}^{T}\sum_{d=1}^{D}\gamma_k(i,d)\,\mathrm{P}\big(y_k^c|m_k=i,x_k^c,\theta_{m_k}^c\big) + C_{\theta_i^c} \;\;. \tag{3.104}$$

The exact form of $q(\theta_i^c)$ in (3.104) will depend on the GLM dynamics associated with each discrete mode. If we define $\gamma_k'(i) = \sum_{d=1}^D \gamma_k(i,d)$, then (3.104) is identical to the expression used in the

VB-M step of GLHMM (3.36) and in the (discrete output) HMM (2.84). Instead of repeating the final posterior expressions of (3.104), the reader is referred to the previous sections.

The calculation of $q(\pi)$ is now considered. By grouping all terms in (3.89) that contain π, the distribution $q(\pi)$ can be calculated:

$$q(\pi) = \log \mathrm{p}\big(\pi\big) + \sum_{\substack{(m_{1:T},\tau_{1:T}) \\ \in(\mathcal{S}\times\mathcal{D})^T}} q(m_{1:T},\tau_{1:T}) \log \pi_{m_1,\tau_1} \ . \tag{3.105}$$

This equation can be simplified by again using the marginal statistic (3.103):

$$q(\pi) = \log \mathrm{p}\big(\pi\big) + \sum_{k=1}^{T}\sum_{d=1}^{D} \gamma_k(i,d) \log \pi_{m_1=i,\tau_1=d} + C_\pi \ . \tag{3.106}$$

The distribution of the initial condition (3.106) is proportional to a Dirichlet distribution:

$$q(\pi) = \mathrm{Dir}\big(\pi_{11},....,\pi_{1D},\pi_{21},...,\pi_{ND} | \pi^T_{11},....,\pi^T_{1D},\pi^T_{21},...,\pi^T_{ND}\big) \ , \tag{3.107}$$

with parameters:

$$\pi^T_{id} = \pi^T_{id} + \gamma_k(i,d) \ . \tag{3.108}$$

In summary, this section has described how to compute the VB-M step (3.85) for VTHMM and HSMM, both in the case when discrete output distributions and generalized linear models representing dynamics associated to each mode are used. Two important results of this section are remarked upon. First, the distribution $q(\Theta)$ is factorized into the following forms:

$$\textbf{VTHMM:} \qquad q(\Theta) = q(\pi) \prod_{d=1}^{D}\prod_{i=1}^{N} q(a_{i:}(d)) \prod_{j=1}^{N}\prod_{c=1}^{C} q(\theta^c_j) \tag{3.109}$$

$$\textbf{HSMM:} \qquad q(\Theta) = q(\pi) \prod_{i=1}^{N} q(a_{i:}) q(p_i(:)) \prod_{j=1}^{N}\prod_{c=1}^{C} q(\theta^c_j) \ . \tag{3.110}$$

Secondly, only marginal statistics (3.91) and (3.103) of the distribution $q(m_{1:T},\tau_{1:T})$ are required:

$$\gamma_k(i,d) \triangleq \sum_{\substack{(m_{1:T},\tau_{1:T}) \\ \in(\mathcal{S}\times\mathcal{D})^T}} q(m_{1:T},\tau_{1:T}) \delta\big((m_k,\tau_k) = (i,d)\big) \tag{3.111}$$

$$\xi_k(i,d,j,d') \triangleq \sum_{\substack{(m_{1:T},\tau_{1:T}) \\ \in(\mathcal{S}\times\mathcal{D})^T}} q(m_{1:T},\tau_{1:T}) \delta((m_k,\tau_k) = (j,d'),(m_{k-1},\tau_{k-1}) = (i,d)) \ , \tag{3.112}$$

In the following VB-E section, these marginal distributions will be computed for the stationary joint space HMM (Def. 3.9).

3.5.1.2 VB-E step for the Joint Space HMM

The VB-E step (3.86) for the stationary joint HMM (Def. 3.9), and hence for the VTHMM and HSMM, can be calculated using the same forward-backward algorithm as for GLHMMs. However, this approach is inefficient and has computational complexity of $\mathrm{O}(N^2D^2T)$, as will be demonstrated in the rest of this section. Sections refsec:VTHMMinN2DT and 3.5.3 will then build on the results of this section and provide forward-backward recursions that allow the computation of the VB-E step in $\mathrm{O}(N^2DT)$ and $\mathrm{O}(N^2T + NDT)$ for VTHMM and HSMM, respectively.

The computation of the VB-E step (3.86) for the stationary joint HMM is conducted by substituting the the complete data likelihood, (3.84) into (3.86):

$$\begin{aligned}&\log q(m_{1:T}, \tau_{1:T}) = \\ &\int_\Theta q(\Theta) \left[\log \mathrm{P}\big(m_1, \tau_1|\pi\big) + \sum_{k=2}^{T} \log \mathrm{P}\big(m_k, \tau_k|m_{k-1}, \tau_{k-1}, A\big) + \sum_{k=1}^{T}\sum_{c=1}^{C} \log p\left(y_k^c|m_k = i, x_k^c, \theta_i^c\right)\right] d\Theta \\ &\qquad - \log C_{m\tau} \quad . \end{aligned} \tag{3.113}$$

Note that the GLM dynamics distribution $\prod_{c=1}^{C} p\left(y_k^c|m_k = i, x_k^c, \theta_i^c\right)$ has been used in (3.113) instead of the stationary distribution $\mathrm{p}\big(y_k|m_k, \theta_{m_k}\big)$ for generality. Using the factorization $q(\Theta) = q(\pi)q(\Phi)\prod_{i=1}^{N}\prod_{c=1}^{C} q(\theta_i^c)$ results in:

$$\log q(m_{1:T}, \tau_{1:T}) = \log \tilde{\pi}_{(m_1,\tau_1)} + \sum_{k=2}^{T} \log \tilde{\phi}_{(m_{k-1},\tau_{k-1})(m_k,\tau_k)} + \sum_{k=1}^{T} \log \tilde{b}_{m_k}(y_k, x_k) - \log C_{m\tau} \quad , \tag{3.114}$$

where from the GLHMM derivation (3.48):

$$\tilde{b}_i(y_k, x_k) = \prod_{c=1}^{C} \tilde{b}_i^c(y_k, x_k) \quad , \tag{3.115}$$

and (3.115) can be computing using either (3.51) for AR-models, (3.53) for Poisson point processes, or (2.91) for stationary Dirichlet distributions.

The geometric mean $\tilde{\pi}_{(m_1,\tau_1)}$ from (3.114) can also be calculated using (2.91) as $q(\pi)$ is a Dirichlet distribuion (3.107).

The remaining geometric mean $\tilde{\phi}_{(i,d)(j,d')}$ in (3.114), will differ for embeddings of VTHMM or HSMM models into stationary HMM. From (3.113) and (3.114) the distribution $\tilde{\phi}_{(i,d)(j,d')}$ is:

$$\tilde{\phi}_{(i,d)(j,d')} = \exp \int q(\Phi) \log \mathrm{P}\big((m_k, \tau_k) = (j, d')|(m_{k-1}, \tau_{k-1}) = (i, d), \Phi\big) d\Phi \quad , \tag{3.116}$$

where the distributions $q(\Phi)$ and $\mathrm{P}\big((m_k, \tau_k) = (j, d')|(m_{k-1}, \tau_{k-1}) = (i, d), \Phi\big)$ depend on if a VTHMM or HSMM was embedded into the stationary HMM.

In the case of a VTHMM, $q(\Phi) = \prod_{d=1}^{D}\prod_{i=1}^{N} q(a_{i:}(d))$ (see (3.94)). Using the embedding (3.82), the expression (3.116) can be written:

$$\textbf{VTHMM:} \quad \tilde{\phi}_{(i,d)(j,d')} = \begin{cases} \tilde{a}_{ij}(d) & \text{if}\{\ j = i, d' = d+1\} \text{ or } \{j = i, d = D\} \text{ or } \{j \neq i, d' = 1\} \\ 0 & \text{else} \end{cases} \tag{3.117}$$

where:

$$\tilde{a}_{ij}(d) = \exp \int q(\Phi) \log a_{ij}(d)\ d\Phi \tag{3.118}$$

$$= \exp \int q(a_{i:}(d)) \log a_{ij}(d)\ da_{i:}(d)\ . \tag{3.119}$$

In the case of a HSMM (3.97) the distribution $q(\Phi) = \prod_{i=1}^{N} q(a_{i:}) \prod_{j=1}^{N} q(p_j(:))$, and the embedding (3.83) are applied to equation (3.116). First consider the case when a HSMM has finished counting down and $\tau_k = 1$, meaning that there is no remaining time left in the current mode $m_k = i$. Equation (3.116) can then be written:

$$\tilde{\phi}_{(i,1)(j,d')} = \exp \int q(\Phi) \log a_{ij} p_j(d') d\Phi \quad \text{for} \quad j \neq i\ . \tag{3.120}$$

Recall that $a_{ii} = 0$ from the definition of a HSMM. Equation (3.120) is simplified using the properties of the logarithm:

$$\tilde{\phi}_{(i,1)(j,d')} = \exp \left[\int q(\Phi) \log a_{ij} d\Phi + \int q(\Phi) p_j(d') d\Phi \right] \tag{3.121}$$

$$= \exp \left[\int q(a_{i:}) \log a_{ij} da_{i:} \right] \exp \left[\int q(p_j(:)) p_j(d') dp_j(:) \right] \tag{3.122}$$

$$= \tilde{a}_{ij} \tilde{p}_j(d')\ , \tag{3.123}$$

where:

$$\tilde{a}_{ij} = \exp \int q(a_{i:}) \log a_{ij} da_{i:} \quad \text{for} \quad j \neq i \tag{3.124}$$

$$\tilde{p}_j(d') = \exp \int q(p_j(:)) \log p_j(d') dp_j(:) \tag{3.125}$$

are geometric means of Dirichlet distributions and can be calculated using (2.91). Note that $\tilde{a}_{ii} = 0$.

Equation (3.116) is now considered when $\tau_k \neq 1$, when from Definition 3.5, the transition to the next state is now deterministic (with probability 0 or 1). Because these transitions are deterministic, and are not functions of the parameters Φ, the transitions are invariant to the integral (3.116), i.e.,

for any constant c: $\exp \int q(\Phi) \log c \; d\Phi = c$. This implies that:

$$\textbf{HSMM:} \quad \tilde{\phi}_{(i,d)(j,d')} \triangleq \mathrm{P}\big((m_{k+1}, \tau_{k+1}) = (j, d') | (m_k, \tau_k) = (i, d)\big)$$
$$= \begin{cases} 1 & \text{if } j = i, d > 1, d' = d - 1 \\ \tilde{a}_{ij} \tilde{p}_j(d') & \text{if } d = 1, \; j \neq i \\ 0 & \text{else } . \end{cases} \tag{3.126}$$

In summary of this section so far, the VB-E step requires the calculation of the marginal statistics of $q(m_{1:T}, \tau_{1:T})$. Equation (3.114) gives the form of this distribution $q(m_{1:T}, \tau_{1:T})$, which is a function of several geometric means. The derivations following (3.114) calculate these geometric means for both the VTHMM (3.117) and HSMM (3.126).

Because the form of (3.114) is exactly the same as for GLHMMs (3.46), the same forward-backwards algorithm can be used. Because we are considering the joint space $\mathcal{S} \times \mathcal{D}$, the equivalent forward and backward variables are:

$$\alpha_k(i, d) \triangleq \mathrm{P}\big((m_k, \tau_k) = (i, d) | y_{1:k}, \Theta\big) \tag{3.127}$$
$$\beta_k(i, d) \triangleq \mathrm{P}\big(y_{k+1:T} | (m_k, \tau_k) = (i, d), \Theta\big) \; . \tag{3.128}$$

Using the forward-backward algorithm for GLHMM (Section 3.2.3) forward parameters now initialized by:

$$\alpha_1(i, d) = \frac{\pi_{id} \, \mathrm{p}\big(y_1 | m_k = i\big)}{\sum_{j=1}^{N} \sum_{d'=1}^{D} \pi_{jd'} \, \mathrm{p}\big(y_1 | m_k = j\big)} \; . \tag{3.129}$$

The forward variable recursion developed for GLHMM models (3.19) is directly applied to the joint space stationary HMM:

$$\alpha'_k(j, d') = b_i(y_k, x_k) \sum_{i=1}^{N} \sum_{d=1}^{D} \phi_{(i,d)(j,d')} \alpha_{k-1}(i, d) \tag{3.130}$$

$$c_{y_k} = \mathrm{p}\big(y_k | y_{1:k-1}, \Theta\big) = \sum_{j=1}^{N} \sum_{d'=1}^{D} \alpha'_k(j, d') \tag{3.131}$$

$$\alpha_k(j, d') = \frac{\alpha'_k(j, d')}{c_{y_k}} \; . \tag{3.132}$$

Note that the product of the constants c_{y_k} in (3.131) still calculates the data likelihood $\mathrm{p}\big(y_{1:T} | \Theta\big)$. The backward parameter recursion follows from (3.22):

$$\tilde{\beta}_{k-1}(i, d) = c_{y_k}^{-1} \sum_{j=1}^{N} \tilde{\beta}_k(j, d') \, \mathrm{p}\big(y_k | m_k = j, x_k, \theta_j\big) \phi_{(i,d)(j,d')} \; , \tag{3.133}$$

and is initialized by:

$$\tilde{\beta}_T(i,d) = c_{y_k}^{-1} \ . \tag{3.134}$$

Note that complexity of applying these forward-backward algorithms in the joint space is $\mathrm{O}(N^2D^2T)$. The marginal statistics of the joint process are also a direct application of the GLHMM E-step. From equation (3.24):

$$\gamma_k(i,d) = \mathrm{P}\big(m_k = i, \tau_k = j | y_{-n:T}, \Theta\big) \tag{3.135}$$

$$\propto \alpha_k(i,d)\tilde{\beta}_k(i,d) \ , \tag{3.136}$$

and from equation (3.26):

$$\xi_k(i,d,j,d') = \mathrm{P}\big(m_k = i, m_{k+1} = j | y_{-n:T}, \Theta\big) \tag{3.137}$$

$$= \frac{1}{c_{\xi_k}} \beta_{k+1}(j,d')\, \mathrm{p}\big(y_{k+1} | m_{k+1} = j, x_k, \theta_j\big) \phi_{(i,d)(j,d')} \alpha_k(i,d) \ , \tag{3.138}$$

where c_{ξ_k} is a normalizing constant:

$$c_{\xi_k} = \sum_i^N \sum_d^D \sum_j^N \sum_{d'}^D \beta_{k+1}(j,d')\, \mathrm{p}\big(y_{k+1} | m_{k+1} = j, x_k, \theta_j\big) \phi_{(i,d)(j,d')} \alpha_k(i,d) \ . \tag{3.139}$$

The forward-backward recursion works equally well when the distributions are sub-normalized, however now the forward recursion calculates normalizing constant of the joint process $C_{m\tau}$ in (3.113).

Sections 3.5.2 and 3.5.3 will derive more efficient forward-backward recursions from this starting point. The key idea is that because many elements of Φ are zero, they do not need to be propagated in the recursions, reducing the computational burden of calculating the marginal statistics.

3.5.1.3 Lower Bound for Variational Inference in the Joint Space HMM

The lower bound (2.20) for a joint space HMM is given by

$$\mathcal{L}(q(m_{1:T}, \tau_{1:T}, \Theta)) = \int_\Theta q(\Theta) \log \frac{\mathrm{p}(\Theta)}{q(\Theta)} d\Theta + \log C_{m\tau} \tag{3.140}$$

where equation (3.140) is derived from (2.104) and Definition 3.9. By using the factorizations (3.110) and (3.109), the lower bound (3.140) can be expressed as a sum of KL-divergences which can be efficiently calculated. Note that $\log C_{m\tau}$ is calculated after every VB-E step. The lower bound for the HSMM case is presented here, with the VTHMM case a straightforward modification[8].

By using the factorization (3.110) of $q(\Theta)$ for HSMMs, the lower bound is calculated after the

[8] For the VTHMM case, the discrete transition parameters $p_i(d)$ and a_{ij} in (3.141) are replaced with $a_{ij}(d)$. Because the posterior of these parameters is also a Dirichlet distribution, the same functional form is retained.

E-step as:

$$\mathcal{L}(q(m_{1:T},\tau_{1:T},\Theta)) = \int_{\pi} q(\pi)\log\frac{\mathrm{p}(\pi)}{q(\pi)}d\Theta + \sum_{i=1}^{N}\int q(a_{i:})\log\frac{\mathrm{p}(a_{i:})}{q(a_{i:})}da_{i:} + \sum_{i=1}^{N}\int q(p_i(:))\log\frac{\mathrm{p}(p_i(:))}{q(p_i(:))}dp_i(:) + \sum_{i=1}^{N}\sum_{c=1}^{C}\int_{\theta_i^c} q(\theta_i^c)\log\frac{\mathrm{p}(\theta_i^c)}{q(\theta_i^c)}d\theta_i^c + \log C_{m\tau} \ . \quad (3.141)$$

To allow efficient calculation of the lower bound (3.141), analytical expression for the KL-divergences are derived in Appendix B. For example the KL-divergence of the duration distribution

$$KL\big(q(p_i(:))||p(p_i(:))\big) = -\int_{p_i(:)} q(p_i(:))\log\frac{\mathrm{p}(p_i(:))}{q(p_i(:))}dp_i(:) \ , \quad (3.142)$$

is the KL-divergence between two Dirichlet distributions, which is calculated in Appendix B.7.

3.5.2 Variational VTHMM in $\mathrm{O}(N^2DT)$

The computational complexity of inference in VTHMM models can be reduced by creating a more efficient forward-backward algorithm, by not propagating the zero probability transitions in the kernel Φ. The forward-backward recursions presented here follow from the papers [33] and [32].

By considering only the non-zero entries of the Φ matrix one can obtain the following. The forward parameters can be updated recursively using the update following equations (for $1 \leq j \leq N$):

$$\alpha'_{k+1}(j,1) = \sum_{d=1}^{D}\sum_{i=1}^{N}\alpha_k(i,d)a_{ij}(d)\,\mathrm{p}\big(y_{k+1}|m_k=j\big) \quad (3.143)$$

$$\alpha'_{k+1}(j,d+1) = \alpha_k(j,d)a_{jj}\,\mathrm{p}\big(y_{k+1}|m_k=j\big) \text{ for } 1\leq d<D-2 \quad (3.144)$$

$$\alpha'_{k+1}(j,D) = \big(\alpha_k(j,D-1)+\alpha_k(j,D)\big)\,a_{jj}\,\mathrm{p}\big(y_{k+1}|m_k=j\big) \ . \quad (3.145)$$

These equations follow directly from the embedding (3.82), and are identical to those presented in both [33] and [32][9]. The forward recursion given in (3.145) allows the forward parameters $\alpha_k(i,d), i = 1:N, d = 1:D, k = 1:T$ to be computed in $\mathrm{O}(N^2DT)$. The backward recursion is then solved recursively [32] using:

$$\beta_k(j,d) = \left[\sum_{i\neq j}\beta_{k+1}(i,1)a_{ji}(d)\,\mathrm{p}\big(y_{k+1}|m_k=i\big)\right] + \beta_{k+1}(j,d+1)a_{jj}(d)\,\mathrm{p}\big(y_{k+1}|m_k=i\big) \ . \quad (3.146)$$

Remark 3.7. The backwards recursion (3.146) allows the backwards parameters $\beta_k(i,d), i = 1:N, d = 1:D, k = 1:T$ to be computed in $\mathrm{O}(N^2DT)$. ◇

[9] Neither paper explicitly states what happens if the system remains in a given state S_j for $d = D$ steps. Based on the dialogue in [32], we have added equation (3.145).

The memory requirement of this method can be reduced by storing only the $\alpha_k(i,d)$, $\beta_k(i,d)$, and $\gamma_k(i,d)$ terms with memory requirements $\mathrm{O}(NDT)$. Then instead of calculating and storing the ξ_k term, just calculate the non-zero terms in (3.93), which then has computational complexity $\mathrm{O}(N^2DT)$, as required to calculate $a_{ij}^T(D)$ in (3.96), for $i = 1, ..., N$, $j = 1, ..., N$ and $d = 1, ..., D$.

3.5.3 Variational HSMM in $\mathrm{O}(N^2T + NDT)$

This section derives efficient forward-backward recursions that can be utilized for variational inference for HSMM. These recursions are based on the work Yu and Kobayashi [73, 79], but modify their recursions to allow for the sub-normalized distributions (3.126) used in Variational Bayes, and account for the addition of GLM dynamics.

The key to deriving an efficient recursion to update the forward variables (3.130) is to consider the allowed transitions into a particular state. From Def. 3.5, the joint state $(m_k, \tau_k) = (i, d)$ can be reached in two ways: either from the same mode with decrease in the duration $(m_{k-1}, \tau_{k-1}) = (i, d+1)$, or from another mode if the duration τ is already at one, $(m_{k-1}, \tau_{k-1}) = (j, 1)$ and $j \neq i$. The forward variable recursion (3.130) can thus be written as a combination of these two possibilities:

$$\alpha'_k(i,d) = \alpha_{k-1}(i,d+1)b_i(y_k,x_k) + \left[\sum_{\substack{j=1\\j\neq i}}^{N} \alpha_{k-1}(j,1)a_{ji}\right] b_i(y_k,x_k)p_i(d) \ . \tag{3.147}$$

By evaluating the sum $\left[\sum_{\substack{j=1\\j\neq i}}^{N} \alpha_{k-1}(j,1)a_{ji}\right]$ in (3.147), then $\alpha'_k(i,d)$ can be calculated for $i = 1:N$ and $d = 1:D$ in $\mathrm{O}(ND + N^2)$, and hence the entire forward sequence for $k = 1, ..., T$ requires $\mathrm{O}(NDT + N^2T)$ computation.

The backwards sequence can likewise be computed in an efficient recursion, by considering only the non-zero transitions in Φ: The transitions from state $(m_k, \tau_k) = (i, d)$ will occur deterministically if $d > 1$ to $(m_{k+1}, \tau_{k+1}) = (i, d-1)$. If $d = 1$, then the state transitions to $(m_{k+1}, \tau_{k+1}) = (j, d')$ for some $j \in 1, ..., N$ and $d' \geq 1$. By then considering only these non-zero probability transitions, the backward recursion (3.133) is reduced to:

$$\tilde{\beta}_k(i,d) = \begin{cases} c_{y_k}^{-1} b_i(y_k,x_k)\tilde{\beta}_{k+1}(i,d-1) & \text{for } d > 1 \\ c_{y_k}^{-1} \sum_{j=1}^{N} a_{ij} b_j(y_k,x_k) \left[\sum_{d=1}^{D} p_j(d)\tilde{\beta}_{k+1}(j,d)\right] & \text{for } d = 1 \ . \end{cases} \tag{3.148}$$

Evaluating the sum $\left[\sum_{d=1}^{D} p_j(d)\tilde{\beta}_{k+1}(j,d)\right]$ only once in (3.148) for each j takes $\mathrm{O}(ND)$ operations. The calculation of $\tilde{\beta}_k(i,d)$ for $i = 1:N$ and $d = 1:D$ only then takes $\mathrm{O}(ND + N^2)$, and hence the entire forward sequence for $k = 1, ..., T$ takes $\mathrm{O}(NDT + N^2T)$.

The calculation of the marginal statistic $\xi_k(i,d,j,d')$ in (3.91) can be greatly simplified by con-

sidering the use of $\xi_k(i,d,j,d')$ in HSMM models (3.100):

$$a_{ij}^T = a_{ij}^0 + \sum_{k=2}^{T}\sum_{d=1}^{D}\xi_k(i,1,j,d) \tag{3.149}$$

$$p_j^T(d) = p_j^0(d) + \sum_{k=2}^{T}\sum_{\forall i\neq j}\xi_k(i,1,j,d) \ . \tag{3.150}$$

Instead of calculating and storing $\xi_k(i,d,j,d')$, each equation (3.149) and (3.150) is now considered separately. In equation (3.149), the marginal statistic $\xi_k(i,d,j,d')$ will be replaced by

$$\zeta_k(i,j) \triangleq \mathrm{p}\big(m_k = i, \tau_k = 1, m_{k-1} = j | y_{1:T}\big) = \sum_{d'=1}^{D}\xi_k(i,1,j,d') \ . \tag{3.151}$$

This new variable (3.151) can be calculated efficiently using only the non-zero elements Φ in equation 3.137:

$$\zeta_k(i,j) = \frac{1}{c_{\xi_k}}\alpha_{k-1}(i,1)a_{ji}b_j(y_k,x_k)\left[\sum_{d=1}^{D}p_j(d)\beta_k(j,d)\right] \ , \tag{3.152}$$

where c_{ξ_k} is the normalizing constant (3.139). Note that the summation $\left[\sum_{d=1}^{D}p_j(d)\beta_k(j,d)\right]$ is only calculated once for each j, and hence (3.152) can be calculated in $\mathrm{O}(ND+N^2)$ for each k. The normalizing constant c_{ξ_k} can also be efficiently calculated, whereby considering only the non-zero components of (3.139):

$$c_{\xi_k} = \sum_{i=1}^{N}\sum_{j=1}^{N}\left[\alpha_k(i,1)a_{ji}b_j(y_k,x_k)\sum_{d'=1}^{D}p_j(d')\beta_{k+1}(j,d')\right] \tag{3.153}$$

$$+\sum_{i=1}^{N}\left[\sum_{d=2}^{D}\alpha_k(i,d)b_i(y_k,x_k)\beta_{k+1}(i,d-1)\right] \ , \tag{3.154}$$

as using the similarity of equation (3.153) to (3.152), the normalizing constant can also be calculated[10] in $\mathrm{O}(ND+N^2)$. To evaluate (3.150) in the VB-M step, a second variable is introduced:

$$\eta_k(j,d') \triangleq \mathrm{p}\big(m_k = j, \tau_k = d', \tau_{k-1} = 1 | y_{1:T}\big) \ = \sum_{\substack{i=1\\ i\neq j}}^{N}\xi_{k-1}(i,1,j,d') \ . \tag{3.155}$$

The variable (3.151) is calculated using only the non-zero elements Φ in equation 3.137:

$$\eta_k(j,d') = \frac{1}{c_k}\left[\sum_{i=1}^{N}\alpha_{k-1}(i,1)a_{ij}\right]b_j(y_k,x_k)p_j(d')\beta_k(j,d') \ . \tag{3.156}$$

[10]It is computationally convenient to first calculate the normalizing constant c_{ξ_k} before $\zeta_k(i,j)$ and store the values calculated in (3.153) for evaluating $\zeta_k(i,j)$.

By calculating the summation $\left[\sum_{i=1}^{N} \alpha_{k-1}(i,1)a_{ij}\right]$ only once for each j, the computation of (3.156) is $\mathrm{O}(ND + N^2)$ for each k

In summary, this section has provided a method to conduct variational inference in HSMM with $\mathrm{O}(ND + N^2)$ complexity. This is the first time variational methods have been applied to HSMM models, and by modifying the recursions of Yu and Kobayashi [79, 73], the proposed method is more computationally efficient then typical EM-based HSMM inference methods with complexity $\mathrm{O}(N^2D)$ [80].

3.6 PWARX Identification using Variational Methods and Gibbs Sampling

This section utilizes the GLHMM identification framework to identify piecewise autoregressive exogenous (PWARX) models (see Section 2.1.3). A two-stage method [8, 59, 6] used in the hybrid systems community is adapted: First the PWARX problem is relaxed by removing dependance on the switching hyperplanes, allowing the system to be modeled using GLHMMs. Second, the posterior mode sequence $m_{1:T}$, in conjunction with the regressors x_k, are used to find the separating hyperplanes using multi-category robust linear programming [62].

The fundamental difference between PWARX and GLHMM models is the discrete transition characteristics of the system. PWARX models define a disjoint set of polyhedral regions $\{\chi_i\}_{i=1}^{N}$ in the regressor space. The system is said to be in mode $m_k = i$ if the regressor $x_k \in \chi_i$ (see Section 2.1.3 for details). By using the GLHMM identification process, the posterior mode probabilities $\mathrm{P}\big(m_{1:T}|y_{1:T}\big)$ are automatically recovered, which then allows identification of the hyperplanes defining each region χ_i. This two-stage process is exactly the same as used in the sequential Bayesian approach (Section 2.1.3.1) and the clustering approach (Section 2.1.3.2). By using this two-stage GLHMM approach, the model selection tools developed in Chapter 4 can be applied to the PWARX model identification problem.

Two case studies are presented. Case study 1 [65] involves a simple two-state "benchmark" problem [8] to verify the suitability of the GLHMM framework. Case study 2 studies a pick-and-place machine data set that has previously been analyzed in the hybrid system community [60].

3.6.1 Case Study 1: Benchmark Problem

To allow comparison of the GLHMM framework to existing methods of hybrid system identification, the intersecting hyperplanes PWARX system from the comparison paper on hybrid system identification [8] is considered:

The PWARX system is defined as $y_k = h(x_k) + e_k$, where h is:

$$h(x_k) = \begin{cases} \begin{bmatrix} x_k & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}^T & \text{if } x_k \in [-2.5, 0] \\ \\ \begin{bmatrix} x_k & 1 \end{bmatrix} \begin{bmatrix} -1 & 2 \end{bmatrix}^T & \text{if } x_k \in (0, 2.5] \end{cases} . \tag{3.157}$$

Data is generated by drawing the regressor from a uniform distribution $x_k \sim U[-2.5, 2.5]$, for $k = 1, \ldots, 100$. The noise, e_k, is Gaussian with variance σ^2.

The two-stage Gibbs sampler (Alg. 3.1) for GLHMM was used as following: Gaussian priors on all regressor parameters $p(\theta_i) = \mathcal{N}(0, 10^3)$ were used. Following from [8] the variance σ^2 is assumed to be known and does not need to be estimated. The regressor parameter pdfs (3.60) were explicitly derived using Bayes' theorem:

$$\mathrm{p}\big(\theta_i | \mathcal{X}^i, \mathcal{Y}^i\big) = \mathcal{N}(E(\theta_i), \Sigma_i) \text{ where}$$
$$\Sigma_i = \sigma^2 (\mathcal{X}^{i\ T} \mathcal{X}^i + \alpha^2 \mathrm{I})^{-1} \text{ and } \mathrm{E}(\theta_i) = (\mathcal{X}^{i\ T} \mathcal{X}^i + \alpha^2 \mathrm{I})^{-1} \mathcal{X}^{iT} \mathcal{Y}_i \ . \tag{3.158}$$

The Markovian transition parameters A are not needed in this example, as x_k is drawn from a uniform distribution, and there is correlation in sequential mode values m_k and m_{k+1}. The mode estimate $m_{1:T}$ in (3.71) is modified by using non-Markov switching behavior:

$$\mathrm{P}\big(m_k | y_k, x_k, \Theta\big) = \frac{\mathrm{p}\big(y_k | x_k, \theta_i, m_k = i\big)}{\sum_{j=1}^{N} \mathrm{p}\big(y_k | x_k, \theta_j, m_k = j\big)} \ , \tag{3.159}$$

as in [65].

An example of the first 1500 samples drawn by the Gibbs sampler is shown in Fig. 3.3. The last 500 samples are used to estimate parameters, shown in Fig. 3.4. Linear programming was used to infer the position of the hyperplane guard.

Following a procedure[11] in [8], the effectiveness of the Gibbs sampling procedure is tested by running the algorithm on (3.157) with a range of noise intensities σ^2. It was found that Gibbs sampling was able to identify parameters to a level of precision between the algebraic approach [6] and the clustering-based procedure [59], when the regressor length and number of modes N was known exactly.

While this PWARX "benchmark" example is useful for comparing to existing PWARX methods, it does not demonstrate any of the strengths of Bayesian inference. The GLHMM inference framework is capable of determining the AR-model orders, and number of discrete modes in an automated, information theoretic basis (see Chapter 4). As derived in Section 3.4.1, the GLHMM

[11] The metric $E[\Delta_\theta]$ from the comparison paper [8] is used.

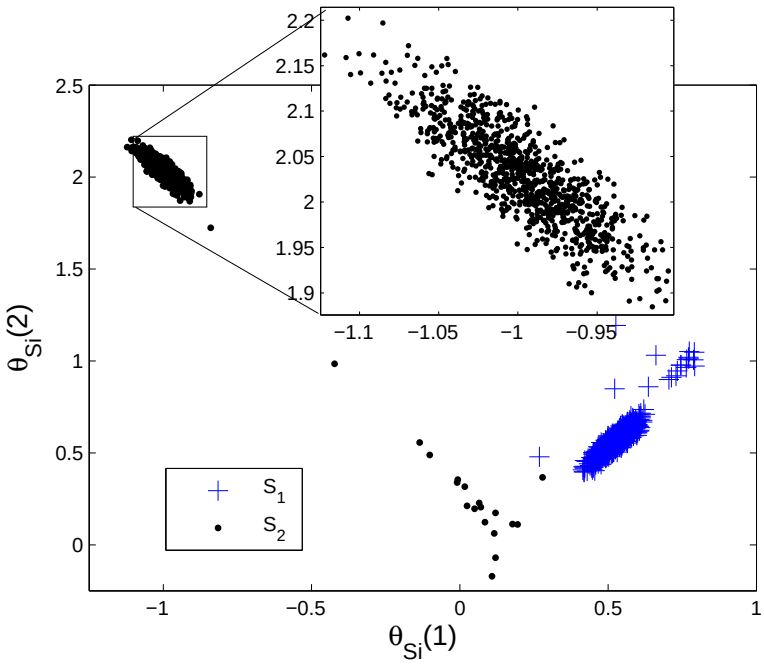

Figure 3.3: Regressor parameter samples $\hat{\theta}_1$,$\hat{\theta}_2$ from Gibbs sampler

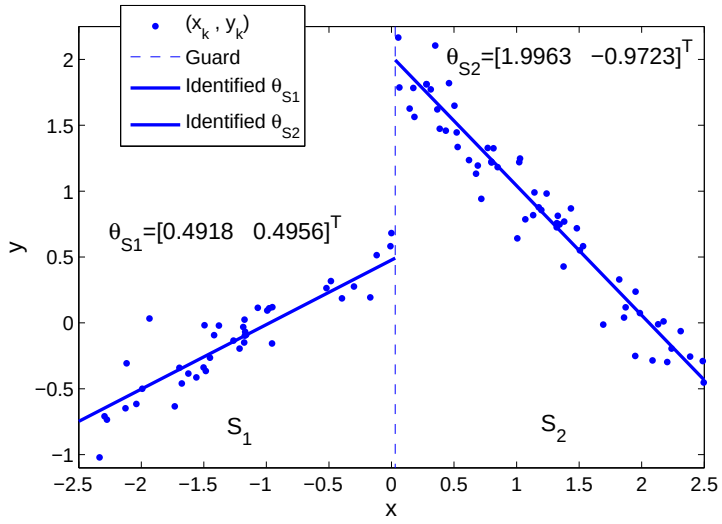

Figure 3.4: Data points (x_k, y_k) from PWARX system and identified model parameters (solid lines)

method can also automatically determine the system variance σ^2. These capabilities separate the GLHMM proposed here with the identification methods reviewed in [8].

3.6.2 Case Study 2: Pick-and-Place Machine

This section used a pick-and-place machine data set [60][12] to demonstrate PWARX identification using GLHMM models and the variational algorithm. In the pick-and-place machine experimental setup, shown schematically in Fig. 3.5, a mounting head is fixed above the impacting surface. The mounting head contains a vacuum pipette which is actuated by an electric motor and is connected to the mounting head via a spring. The position of the pipet relative to the mounting head is measured using a position sensor. The impact surface simulates the elastic properties of a printed circuit board, exhibits both linear and dry friction. The pick-and-place machine data set also contains information from a contact sensor, although in actual pick-and-place machine operation these sensors are not typically used [60]. For specific details about the experimental setup and the motivation for identifying and modeling the pick-and-place machine as a PWARX system, please refer to [60].

The identification of the pick-and-place machine was conducted as follows. A GLHMM (Def. 3.4) with two discrete states, $\mathcal{S} = \{S_1, S_2\}$, and one autoregressive channel (Def. 3.1) was used:

$$y_k = \begin{cases} \mathcal{N}(\theta_1^T x_k, \tau_1^{-1}) & \text{if } m_k = 1 \\ \mathcal{N}(\theta_2^T x_k, \tau_2^{-1}) & \text{if } m_k = 2 \end{cases}, \tag{3.160}$$

with the regressor $x_k = [y_{k-1}, y_{k-2}, u_{k-1}1]^T$. The GLHMM mode (3.160) was identified using the variational method (Section 3.3). Minimally informative priors were used during the identification process (see Definition 3.4 for details):

$$a_{ij}^0 = 1, \ \forall i, j \in \{1, 2\}. \tag{3.161}$$

$$w^0 = \mathbf{0}, \quad \lambda^0 = 0.01, \quad a^0 = 1, b^0 = 1 \ . \tag{3.162}$$

Once the GLHMM model has been identified, the hyperplane h_{12} that defines the boundary between the polyhedral regions in the regressor space is found using the posterior mode sequence $m_{1:T}$. This two-stage method is compared to a model generated using a contact sensor, which gives a "ground truth" of the actual mode sequence. This sensor, while providing very noisy measurements, provides a reasonable benchmark to compare the identified model. The posterior mode sequences of the identified GLHMM model, and the subsequently identified hyperplane of the PWARX model, are compared with those derived using the contact sensor in Figure 3.6. Results show favorable

[12] A. Juloski is greatly thanked for providing this data set.

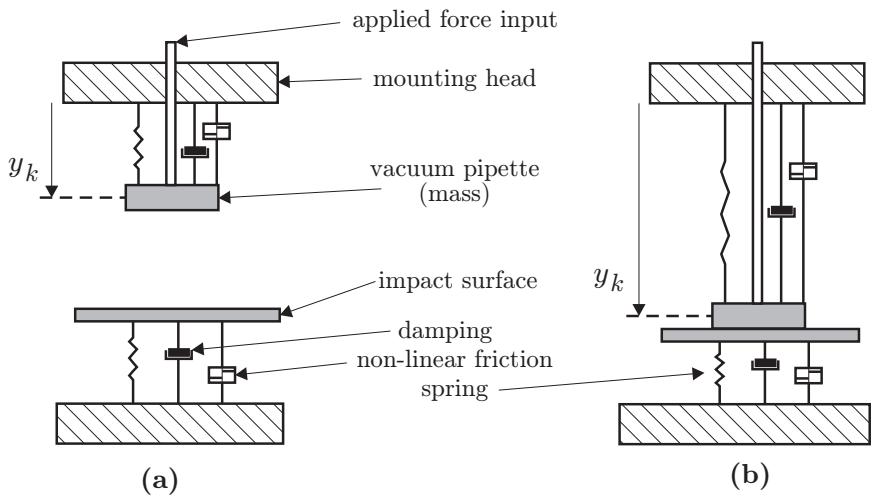

Figure 3.5: Pick-and-place machine first-principles model from [60]. The system consists of an actuated vacuum pipette (mass), which will contact an impacting surface. The impacting surface and actuator are mechanically constrained such that only movement in the vertical axis is allowed. The system exhibits both linear and dry friction. In the considered data set, there are only two discrete states or modes: (a) a *free mode*, where the mass is not in contact with the impacting surface, and (b) an *impact mode* where the mass is in contact with the impacting surface. There also exist lower and upper saturation limits imposed by the range of the actuator, but these are not observed with appropriate inputs.

comparisons of the identified model; as discussed in [60], the posterior model can confuse dry friction with the contact mode. In this case, relatively small changes in the input will cause no movement in the position of the vacuum pipette.

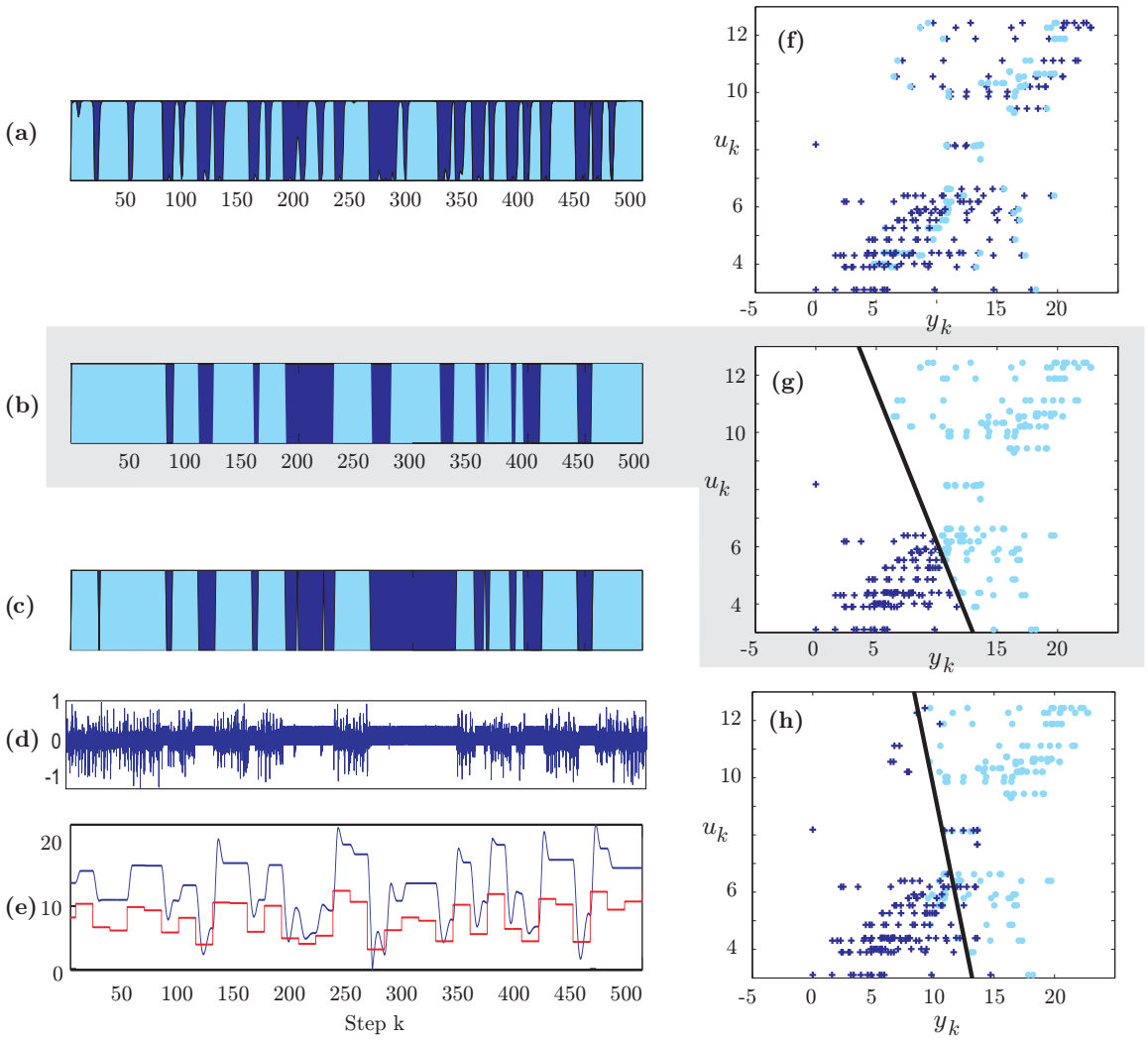

Figure 3.6: Pick-and-place machine identification results. (a) Posterior state sequence using GLHMM model. (b) Posterior state sequence using PWARX model generated from GLHMM results (see text for details). (c) A "ground truth" state sequence determined by a contact sensor. (d) Observed output from the contact sensor. (e) Observed state sequence y_k (blue) and control input u_k (red). (f) Regressor space representation of observed data and control input. The data points are color coded according to the corresponding state sequence, where each regressor point (u_k, y_k) is assigned to the most likely mode m_k. (g) Identified PWARX model segmentation of the regressor space. The identified hyperplane is depicted with a black line. (h) Using the "ground truth" mode sequence from the contact sensor, the hyperplane is found using linear programming.

3.7 Hidden Regressor Dependent Markov Models

Based on the identification of PWARX systems and generalized linear hidden Markov models, a new class of stochastic hybrid system is developed. This new model is formed around a non-stationary, regressor dependent, Markov transition kernel. This work is influenced by [50], which proposes an estimation algorithm for hybrid system models with non-stationary kernels.

In the previous section, identification of PWARX systems was approached in a two-stage process. The key to this approach, also used in [8, 6, 59], is the relaxation of the problem by removing the systems mode dependance on the polyhedral regions which ultimately determine the switching behavior. In essence, the problem is relaxed from identifying a PWARX model into a clustering problem. This approach has the advantage of allowing powerful existing classification techniques to be applied to the identification of the hyperplanes which determine the polyhedral regions in the regressor space.

In this section, the deterministic hyperplane boundaries which define the switching behavior of a PWARX system are replaced by probabilistic boundaries. Specifically, logistic regression functions (i.e., the softmax function) are used to define the probabilistic switching surfaces, or guards, of the system. The systems guards can be naturally formulated as a regressor dependant Markovian transition kernel, and motivate the creation of a hidden regressor dependent Markov model (HRDMM). The inference algorithms developed for GLHMM models in Section 3.3 can be extended for identification of this new HRDMM model class in such a way that the guard functions are directly incorporated into the identification process.

A formal definition of the HRDMM model is given in Definition 3.10. Several necessary algorithms, including a forward-backward recursion, that are needed for identification of HRDMM systems are then developed in Sections 3.7.1 to 3.7.3. Finally, Section 3.7.2 develops a variational inference algorithm for HRDMMs models.

Definition 3.10 (Hidden Regressor Dependent Markov Model (HRDMM))**.** *A HRDMM system is of the form:* $\mathcal{G} = \{\mathcal{S}, \mathcal{U}, \mathcal{Y}, \mathcal{X}, H, \Theta\}$ *where:*

1. $\mathcal{S} = \{S_1, \ldots, S_N\}$ *is a set of* N *discrete states, or modes of* $\mathcal{G}$*. The system mode* S_i *at time* t_k *is denoted by* $m_k = i$*. The hidden state sequence for all* $1 \leq k \leq T$*, is denoted* $m_{1:T} = \{m_1, m_2, ..., m_T\}$*.*

2. $\mathcal{X}$ *is the regressor space, where* x_k *is the regressor at time* t_k*:*

$$x_k = [1, y_{k-1}, \ldots, y_{k-n}, u_k^T]^T \ \in \mathcal{X} \equiv \mathbb{R}^{n+1} \ , \tag{3.163}$$

where $u_k \in \mathcal{U}$ *is a set of known input variables.*

3. *$\mathcal{X}$ contains a set of $N^2 - N$ guard regions $\{R_{ij}\}_{i\neq j=1}^{N}$, and N invariant regions $\{R_{ii}\}_{i=1}^{N}$. Each region R_{ij} is defined as* active *at t_k if $x_k \in R_{ij}$. The set of all regions associated with state S_i is defined $\mathcal{R}^i \triangleq \{R_{ij}, \forall j\}$. The guards regions R_{ij} for $j \neq i$ are treated as a probabilistic half spaces defined by a hyperplane $h_{ij} \in \mathcal{X}^*$. By convention the invariant region will have $h_{ii} \triangleq \mathbf{0}$. The probability of a region being active is a defined by the soft-max function:*

$$\mathrm{P}\big(x_k \in R_{ij} | H_i, x_k\big) = \frac{\exp(h_{ij}^T x_k)}{\sum_{l=1}^{N} \exp\left(h_{il}^T x_k\right)} \ , \tag{3.164}$$

where set of all hyperplanes associated with mode S_i is denoted $H_i \triangleq \{h_{ij}, \forall j : j \neq i\}$.The system discrete state m_k evolves according to which guard (or invariant) region is active. The discrete state evolves according the the following non-stationary Markovian kernel:

$$a_{ij}(x_k) \triangleq \mathrm{P}\big(m_k = j | m_k = i, x_k, H_i\big) = \mathrm{P}\big(x_k \in R_{ij} | H_i, x_k, m_k = i\big) \ . \tag{3.165}$$

H is defined as the set of all hyperplanes.

4. *$y_k \in \mathcal{Y}$ is the system output at t_k, and is an auto-regressive function dependent on the system mode m_k, the parameters associated with mode S_{m_k}: θ_{m_k} and $\sigma_{m_k}^2$ and the regressor x_k:*

$$\mathrm{p}\big(y_k | x_k, m_k, \theta_{m_k}, \sigma_{m_k}^2\big) = \mathcal{N}\left(\theta_{m_k}^T x_k, \sigma_{m_k}^2\right) \ . \tag{3.166}$$

5. *Θ is the set of all parameters associated with the continuous dynamics of the model and includes all AR parameters θ_i, σ_i^2.*

$\diamond$

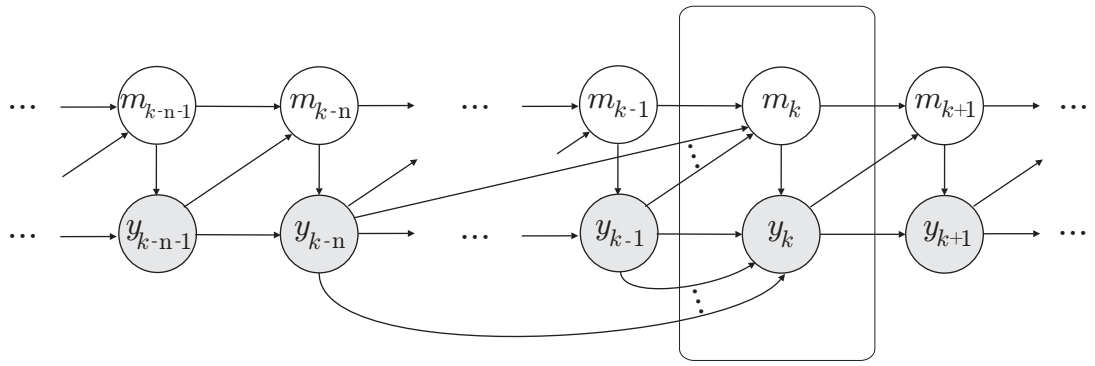

Figure 3.7: Directed acyclic graph of a HRDMM. Only the connection for the step t_k is fully shown.

Remark 3.8. The softmax decision boundary implies that the regions R_{ij} are convex and disjoint, and that only one region is active at any time. The softmax function (3.164) is defined by here by setting $h_{ii} = \mathbf{0}$. This convention is used so that the hyperplanes h_{ij} correspond to those in PWARX

models. Note however that this choice is arbitrary, because the transition function is invariant to the addition of any constant C:

$$\mathrm{P}\big(x_k \in R_{ij} | H_i, x_k\big) = \frac{\exp(h_{ij}^T x_k)}{\sum_{l=1}^{N} \exp\left(h_{il}^T x_k\right)} = \frac{\exp(h_{ij}^T x_k + C)}{\sum_{l=1}^{N} \exp\left(h_{il}^T x_k + C\right)} \quad . \tag{3.167}$$

◇

Remark 3.9. The HRDMM model can be considered a probabilistic generalization of a PWARX (Def. 2.2) system, where now the evolution of the discrete state m_k depends on the full hybrid state (m_{k-1}, x_k) instead of just x_k. This inclusion of m_{k-1} allows for the modeling of hysteretic systems. If PWARX behavior is desired, then the softmax function (3.164) can be constrained to be the same for all modes such that $h_{ij} = h_{i'j} \forall i'$. Intuition into the hysteretic modeling capabilities of the HRDMM is is given by an example of an air conditioner in Figure 3.8. ◇

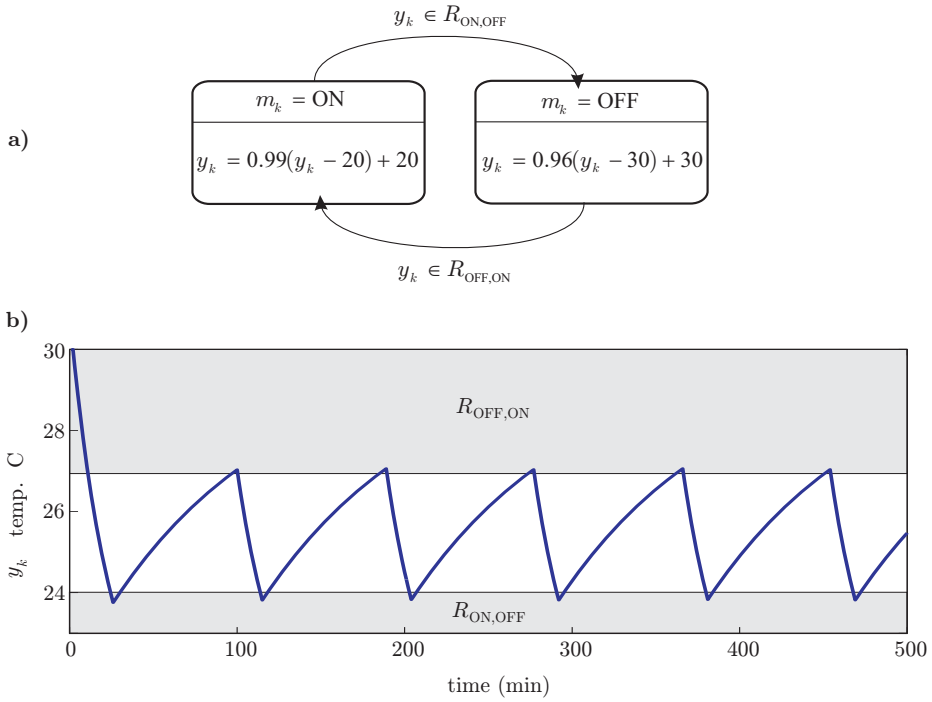

Figure 3.8: Air Conditioning System: a HRDMM. This system has two modes: Mode 1 "ON" and Mode 2 "OFF". (a) A finite state machine representation of the air conditioner. (b) Sample output of the HRDMM. Here the temperature y_k asymptotically approaches 30 if the air conditioner is off, and approaches 20 if the air conditioner is on. The system guards transition the system from on to off if $y_k < 24$ and from off to on if $y_k > 27$.

An interesting relationship between the regressor dependent Markov transition kernel (3.164) and the stationary kernel used in GLHMMs, is when the hyperplane h_{ij} is non-zero in only the first

element, and no longer depends on the regressor x_k:

$$a_{ij}(x_k) = \frac{\exp(h_{ij}(1))}{\sum_{l=1}^{N} \exp\left(h_{il}(1)\right)} \ . \tag{3.168}$$

In this case (3.168), the variables $h_{ij}(1)$ actually form a softmax basis [82] for the stationary kernel A used in HMMs.

Following from GLHMMs (3.13), the complete data likelihood of a sequence of length T for a HRDMM is given by:

$$\mathrm{p}\big(m_{1:T}, y_{-n:T}|\Theta, H\big) = \mathrm{P}\big(m_1|\pi\big)\,\mathrm{p}\big(y_1|m_1, x_1, \theta_{m_1}\big) \prod_{k=2}^{T} \mathrm{P}\big(m_k|m_{k-1}, x_k, H\big)\,\mathrm{p}\big(y_k|m_k, x_k, \theta_{m_k}\big) \ . \tag{3.169}$$

Before explicitly deriving a variational inference algorithm for HRDMMs in Section 3.7.2, forward-backward recursions for the non-stationary kernel are developed.

3.7.1 Forward-Backwards Algorithm for HRDMM

This section derives a forward-backward recursion for HRDMM models. Following from Section 3.2.3 of GLHMM, the goal of the derived algorithm is to calculate the following forward and backward variables:

$$\alpha_k(i) \triangleq \mathrm{P}\big(m_k = i|y_{-n:k}, \Theta, H\big) \tag{3.170}$$

$$\beta_k(i) \triangleq \mathrm{P}\big(y_{k+1:T}|m_k = i, y_{-n:k}, \Theta, H\big) \ , \tag{3.171}$$

and the marginal probabilities:

$$\begin{aligned} \gamma_k(i) &\triangleq \mathrm{P}\big(m_k = i|y_{-n:T}, \Theta, H\big) \\ &= \sum_{m_{1:T} \in \mathcal{S}^T} \mathrm{P}\big(m_{1:T}|y_{-n:T}, \Theta, H\big)\delta(m_k = i) \end{aligned} \tag{3.172}$$

$$\begin{aligned} \xi_k(i,j) &\triangleq \mathrm{P}\big(m_k = i, m_{k+1} = j|y_{-n:T}, \Theta, H\big) \\ &= \sum_{m_{1:T} \in \mathcal{S}^T} \mathrm{P}\big(m_{1:T}|y_{-n:T}, \Theta, H\big)\delta(m_k = i)\delta(m_{k+1} = j) \ , \end{aligned} \tag{3.173}$$

where $y_{-n:T} = \{y_{-n}, y_{1-n}, ..., y_0, y_{1:T},\}$ are the observations augmented with the initial regressor values (see Section 3.2.3). The marginal statistics (3.172) and (3.173) are again derived as functions

of the forward (3.170) and backward (3.171) variables. Using Bayes' theorem:

$$\gamma_k(i) \propto \mathrm{p}\big(y_{k+1:T}|m_k = i, y_{-n:k}, \Theta, H\big)\,\mathrm{P}\big(m_k = i|y_{-n:k}, \Theta, H\big) \tag{3.174}$$

$$=\propto \alpha_k(i)\beta_k(i)\ . \tag{3.175}$$

The marginal statistic (3.173) is likewise formulated using Bayes' theorem and then the probability axiom (A.3):

$$\xi_k(i,j) \propto \mathrm{p}\big(y_{k+2:T}|m_{k+1} = j, \Theta, H\big)\,\mathrm{P}\big(m_k = i, m_{k+1} = j|y_{-n:k+1}, \Theta, H\big) \tag{3.176}$$

$$\propto \beta_{k+1}(j)\,\mathrm{p}\big(y_{k+1}|m_{k+1}, y_{-n:k}\big)\,\mathrm{P}\big(m_{k+1} = j|m_k = i, y_{-n:k}\Theta, H\big)\,\mathrm{P}\big(m_k = i|y_{-n:k}, \Theta, H\big) \tag{3.177}$$

$$\propto \beta_{k+1}(j)\,\mathrm{p}\big(y_{k+1}|m_{k+1}, x_{k+1}\big)a_{ij}(x_k)\alpha_k(i)\ , \tag{3.178}$$

where (3.178) uses the definition of $x_k = [1, y_{k-n}, ..., y_{k-1}]$.

Proposition 3.3 (Forward variable recursion for HRDMM)**.** The forward variables can be updated using:

$$\alpha_k(i) = \frac{1}{c_{y_k}}\mathrm{p}\big(y_k|m_k = i, x_k, \theta_i\big)\sum_{j=1}^{N} a_{ji}(x_k)\alpha_{k-1}(j) \tag{3.179}$$

where the normalizing constant is:

$$c_{y_k} = \mathrm{p}\big(y_k|y_{1:k-1}, \Theta, H\big) = \sum_{i=1}^{N}\mathrm{p}\big(y_k|m_k = i, x_k, \theta_i\big)\sum_{j=1}^{N} a_{ji}(k)\alpha_{k-1}(j)\ . \tag{3.180}$$

◇

Proof.

$$\begin{aligned}\alpha_k(i) &= \frac{\mathrm{p}\big(y_k|m_k = i, y_{1:k-1}, \theta_i\big)\,\mathrm{P}\big(m_k = i|y_{1:k-1}, \Theta, H\big)}{\mathrm{p}\big(y_k|y_{1:k-1}, \Theta, H\big)} && \text{(Bayes)}\\ &= \frac{\mathrm{p}\big(y_k|m_k = i, y_{1:k-1}, \theta_i\big)}{\mathrm{p}\big(y_k|y_{1:k-1}, \Theta, H\big)}\\ &\quad\times\sum_{j=1}^{N}\mathrm{P}\big(m_k = i|m_{k-1} = j, y_{1:k-1}, \Theta, H\big)\,\mathrm{P}\big(m_{k-1} = j|y_{1:k-1}, \Theta, H\big) && \text{(Total Prob.)}\end{aligned}$$

The above equation is equivalent to (3.179) by using the definitions of $a_{ij}(x_k)$ and $\alpha_{k-1}(j)$. The normalizing constant (3.180) is realized by the constraint that $\sum_{i=1}^{N}\alpha_k(i) = 1$. □

Remark 3.10. The forward variable recursion allows the calculation of the incomplete data likelihood by taking the product of the normalizing constants: $\mathrm{p}\big(y_{1:T}|\Theta, H\big) = \prod_{k=1}^{T}\mathrm{p}\big(y_k|y_{1:k-1}, \Theta, H\big)$. ◇

Proposition 3.4 (Backward variable recursion for HRDMM)**.** The backward variables can be updated using:

$$\beta_{k-1}(i) = \sum_{j=1}^{N} \beta_k(j)\, \mathrm{p}\big(y_k | m_k = j, x_k\theta_j\big) a_{ij}(x_k) \ . \tag{3.181}$$

◇

Proof.

$$\begin{aligned}
\beta_{k-1}(i) &= \mathrm{P}\big(y_{k+1:T} | m_k = i, y_{-n:k-1}, \Theta, H\big) \\
&= \sum_{j=1}^{N} \mathrm{p}\big(y_{k:T}, m_k = j | m_{k-1} = i, y_{-n:k-1}, \Theta, H\big) && \text{(Marginalization)} \\
&= \sum_{j=1}^{N} \mathrm{p}\big(y_{k+1:T} | m_k = j, y_{-n:k}, \Theta, H\big)\, \mathrm{p}\big(y_k, m_k = j | m_{k-1} = i, y_{-n:k-1}\Theta, H\big) && \text{(Product Rule)} \\
&= \sum_{j=1}^{N} \mathrm{p}\big(y_{k+1:T} | m_k = j, y_{-n:k}\Theta\big)\, \mathrm{p}\big(y_k | m_k = j, y_{-n:k-1}, \theta_j\big) && \text{(Product Rule)} \\
&\quad \times \mathrm{p}\big(m_k = j | m_{k-1} = i, y_{-n:k-1}, A\big)
\end{aligned}$$

The above equation is equivalent to (3.181) by substitution of definitions of a_{ij} and $\beta_k(j)$. □

3.7.2 Variational Analysis for HRDMM with Model Selection

To identify HRDMMs from observed data, a variational inference algorithm is derived. This algorithm will deviate from variational inference in GLHMM (Section 3.3) in one significant way; the hyperplanes h_{ij} are treated as hyperparameters, and are updated using type II maximum likelihood. This identification scenario is chosen for two reasons. First, it allows *automatic relevance determination* (ARD) to be applied to identification of hyperplanes h_{ij}. In the case where the regressor x_k is augmented with several extra input variables, then ARD will automatically determine which regressor parameters are important and "prune" away unneeded variables. Secondly, using type II maximum likelihood is more tractable then a fully Bayesian implementation[13].

Type II maximum likelihood typically maximizes model hyperparameters (in this case H) with respect to the model evidence $\mathrm{p}\big(y_{1:T}|H\big)$. Evaluation of the model evidence requires integration over the model parameters Θ and latent variables $m_{1:T}$ (see Chapter 4 for an indepth discussion of the model evidence and relation to the model selection problem). For inference in HRDMMs, we will instead seek to maximize the model evidence times a prior over the hyperparameters[14]:

[13] Variational Bayesian inference for multi-category linear regression is an active area of research [24]. If full variational Bayesian inference is required, future HRDMM definitions could instead utilize Probit regression or the relevance vector machine [41] as a classifier.

[14] This is equivalent to viewing the identification of H as a continuous model class selection problem, where $\mathrm{p}(H)$ is then a prior over the model classes (see Chapter 4 for details on model class selection). For details on using this hyperparameter approach, see [68].

$\mathrm{p}\big(y_{1:T}|H\big)\,\mathrm{p}\big(H\big)$.

The primary goal of identification of a HRDMM model involves finding the optimal hyperplanes H that define the switching of the system. Because maximizing a function of the model evidence, $\mathrm{p}\big(y_{1:T}|H\big)\,\mathrm{p}\big(H\big)$, is intractable due to the integration over Θ, $m_{1:T}$, a variational approach is used. From Proposition 2.1 in Chapter 2, $\mathrm{p}\big(y_{1:T}|H\big)\,\mathrm{p}\big(H\big)$ can be approximated by:

$$\log[\mathrm{p}\big(y_{1:T}|H\big)\,\mathrm{p}\big(H\big)] \geq \mathcal{L}\big(q(\Theta), q(m_{1:T}), H\big) + \log \mathrm{p}\big(H\big) \tag{3.182}$$

where from equation (2.20):

$$\mathcal{L}(q(\Theta), q(m_{1:T}), H) = \sum_{m_{1:T}\in\mathcal{S}^T} \int_{\Theta} q(m_{1:T}) q(\Theta) \log \frac{\mathrm{p}\big(y_{1:T}, m_{1:T}, \Theta|H\big)}{q(m_{1:T})q(\Theta)} d\Theta \ . \tag{3.183}$$

The identification of a HRDMM model thus involves a two-step process, where (3.182) is maximized with respect to H and then the lower bound (3.183) is maximized with respect to the factorized distribution $q(\Theta)q(m_{1:T})$.

The maximization of the lower bound (3.183) is now derived, and follows directly from variational inference in GLHMMs, and again the VB-E and VB-M steps are used:

$$\textbf{VB-M step:} \qquad q(\Theta) = \frac{1}{C_\Theta} \exp\left[\sum_{m_{1:T}\in\mathcal{S}^T} q(m_{1:T}) \log \mathrm{p}\big(y_{1:T}, m_{1:T}|\Theta, H\big)\right] \mathrm{p}\big(\Theta\big) \tag{3.184}$$

$$\textbf{VB-E step:} \qquad q(m_{1:T}) = \frac{1}{C_{m_{1:T}}} \exp\left[\int q(\Theta) \log \mathrm{p}\big(y_{1:T}, m_{1:T}|\Theta, H\big) d\Theta\right] \ . \tag{3.185}$$

Using complete data likelihood (3.169) the VB-M step (3.185) becomes:

$$\begin{aligned}\log q(\Theta) = \log \mathrm{p}\big(\pi\big) + \sum_{S^T} q(m_{1:T}) \log \pi_{m_1} + \sum_{S^T} q(m_{1:T}) \sum_{k=2}^{T} \log \mathrm{P}\big(m_k|m_{k-1}, x_k, H\big) \\ + \sum_{i=1}^{N} \log p(\theta_i) + \sum_{S^T} q(m_{1:T}) \sum_{k=1}^{T} \log \mathrm{P}\big(y_k|m_k, \theta_k\big) - \log C_\Theta \ .\end{aligned} \tag{3.186}$$

By using the marginal statistics (3.172) of $q(m_{1:T})$, and using the independence in $eq:3RD_8$, $q(\Theta) = q(\pi)\prod_i q(\theta_i)$ where:

$$\log q(\theta_i) = \log p(\theta_i) + \sum_{k=1}^{T} \gamma_k(i)\, \mathrm{P}\big(y_k|m_k = i, x_k, \theta_{m_k}\big) + C_{\theta_i} \ , \tag{3.187}$$

which in the case of an AR model, is a Gaussian-Gamma distribution as derived in (3.37) for GLHMMs.

Using the complete data likelihood (3.169) and following Section 3.3.2, the VB-E step (3.184) is:

$$\log q(m_{1:T}) = \log \tilde{\pi}_{m_k} + \sum_{k=1}^{T} \log a_{m_k m_{k-1}}(x_k) + \sum_{k=1}^{T} \log \tilde{b}_{m_k}(y_k, x_k) - \log C_{m_{1:T}} \ , \tag{3.188}$$

where the geometric means $\tilde{\pi}_{m_k}$ and $\tilde{b}_{m_k}(y_k, x_k)$ have already been calculated in (3.47) for GLHMM models, and $C_{m_{1:T}}$ is a constant. Equation (3.188) can then be evaluated using the forward-backward recursion developed in Section (3.7.1), and where now the product of the normalizing constants evaluates : $C_{m_{1:T}} = \prod_{k=1}^{T} c_{y_k}$. This allows calculation of the lower bound using equation (3.55). The important result in the VB-E step (as in the GLHMM counterpart) is that the marginal statistics $\gamma_k(i)$, in (3.172) and $\xi_k(i,j)$ in (3.173) for the distribution $q(m_{1:T})$ are calculated.

The maximization of (3.182), with respect to the hyperplanes H is denoted "MAX-H":

$$\textbf{MAX-H step:} \qquad H = \underset{H}{\operatorname{argmax}} \left[\mathcal{L}\big(q(\Theta), q(m_{1:T}), H\big) + \log \mathrm{p}\big(H\big) \right] \ . \tag{3.189}$$

The MAX-H step (3.189) is solved by decomposing the lower-bound into terms just involving H:

$$\mathcal{L}(q(\Theta), q(m_{1:T}), H) = \sum_{m_{1:T} \in \mathcal{S}^T} q(m_{1:T}) \log \prod_{k=2}^{T} \mathrm{P}\big(m_k | m_{k-1}, x_k, H\big) + C_H \tag{3.190}$$

where C_H is a constant that is not a function of H, and the form of the complete data likelihood (3.169) has been used. By using the marginal statistic $\xi_k(i,j)$ in (3.173) of $q(m_{1:T})$ calculated in the VB-E step (3.185), equation (3.190) is simplified:

$$\mathcal{L}(q(\Theta), q(m_{1:T}), H) = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=2}^{T} \xi_k(i,j) \log \mathrm{P}\big(m_k = j | m_{k-1} = i, x_k, H\big) + C_H \tag{3.191}$$

as each H_i is independent in (3.191), the MAX-H step can proceed by maximizing each H_i separately. The lower bound of as a function of each H_i is written:

$$\mathcal{L}(q(\Theta), q(m_{1:T}), H_i) = \sum_{j=1}^{N} \sum_{k=2}^{T} \xi_k(i,j) \log \frac{\exp(h_{ij}^T x_k)}{\sum_{l=1}^{N} \exp\left(h_{il}^T x_k\right)} + C_{H_i} \ . \tag{3.192}$$

The MAX-H step then requires the maximization of

$$H_i = \underset{H_i}{\operatorname{argmax}} \left[\sum_{j=1}^{N} \sum_{k=2}^{T} \xi_k(i,j) \log \frac{\exp(h_{ij}^T x_k)}{\sum_{l=1}^{N} \exp\left(h_{il}^T x_k\right)} + \log \mathrm{p}\big(H_i\big) \right] \ . \tag{3.193}$$

Equation (3.193) is known as the weighted *cross entropy* cost function for the multi-class classification problem [24]. This function is convex, and many efficient optimization routines developed specifically

for this problem exist. Section 3.7.3 considers the computation of (3.193) for two specific priors $\mathrm{p}\big(H_i\big)$ that have proved advantageous in the machine learning community.

In summary, this section proposes a variational inference algorithm to identify a HRDMM from data. To clarify the required steps to implement this algorithm, Algorithm 3.4 implements a routine that will maximize the term $[\mathrm{p}\big(y_{1:T}|H\big)\,\mathrm{p}\big(H\big)]$ from (3.182):

Algorithm 3.4 Variational Algorithm for Identification of HRDMMs

1: Choose initial distributions, $q(\Theta), q(m_{1:T}),$ and initial hyperparameter set H
2: **while** $\Delta C >$tol **do**
3: MAX-H step: $H = \mathrm{argmax}_H \left[\mathcal{L}\big(q(\Theta), q(m_{1:T}), H\big) + \log \mathrm{p}\big(H\big)\right]$
4: VB-M step: $q(\Theta) = \frac{1}{C_\Theta} \exp\left[\sum_{m_{1:T}\in\mathcal{S}^T} q(m_{1:T}) \log \mathrm{p}\big(y_{1:T}, m_{1:T}|\Theta, H\big)\right] \mathrm{p}\big(\Theta\big)$
5: VB-E step: $q(m_{1:T}) = \frac{1}{C_{m_{1:T}}} \exp\left[\int q(\Theta) \log \mathrm{p}\big(y_{1:T}, m_{1:T}|\Theta, H\big) d\Theta\right]$
6: $\Delta C =$ change in $\mathcal{L}\big(q(\Theta), q(m_{1:T}), H\big) + \log \mathrm{p}\big(H\big)$ from last step
7: **end while**

3.7.3 Identification of Guard Regions

Inferring the guard regions R_{ij} defined by hyperplanes h_{ij} in (3.164) is a multi-class logistic regression problem. This section will discuss a MAP optimization algorithm for identification of hyperplanes (3.193) for two different priors $\mathrm{p}\big(H_i\big)$. To relate to existing inference algorithms[15] [83] the following notation is introduced:

$$\mathrm{p}\big(m_k = j|m_{k-1} = i, x_k, H_i\big) = a_{ij}(x_k) = \frac{\exp(\phi^j_{ki})}{\sum_{l=1}^N \exp(\phi^l_{ki})} \ , \tag{3.194}$$

where

$$\phi^j_{ki} \triangleq h^T_{ij} x_k \ . \tag{3.195}$$

Typically in multi-class logistic regression, the regressors x_k and the classifications, i.e., that $x_k \in R_{ij}$, are assumed to be known. In the case of HRDMMs, the classification of regressors is uncertain, but after each VB-E step, the marginal statistic $\xi_k(i,j)$ can be used to give a probabilistic classification of each regressor.

The goal in solving the MAX-H step (3.193), using notation (3.194) is then to maximize the following function.

$$\sum_{j=1}^N \sum_{k=2}^T \xi_k(i,j) \log a_{ij}(x_k) + \log \mathrm{p}\big(H_i\big) \tag{3.196}$$

Two priors are typically used in maximization of (3.196). The first is a simple *regularization* prior,

[15] There is a wealth of information on multi-class logistic regression. If the provided algorithms are not sufficient, try using the keywords "Multiclass Logistic Regression", or "Multinomial Logistic Regression", and for automatic relevance determination of hyperplane parameters, use the keywords "Sparse","Laplace Priors", or "LASSO".

which is simply a product of Gaussian distributions:

$$\mathrm{p}\big(H_i\big) = \prod_{j=1}^{N} \mathrm{p}\big(h_i j\big) \tag{3.197}$$

where:

$$\mathrm{p}\big(h_{ij}\big) = \mathcal{N}(\mathbf{0}, \sigma^2 I) \ . \tag{3.198}$$

This regularization prior simply stops the maximization of (3.196) from becoming unbounded if the regressors can be linearly separated [24].

A more interesting prior that automatically prunes elements of the regressor is based on L1-regularization:

$$\mathrm{p}\big(h_{ij}\big) = \alpha \sum_{j=1}^{N} \sum_{r=1}^{d} |h_{ij}(r)| \ . \tag{3.199}$$

This regularization term is known as the LASSO method in the context of ML learning, and is equivalent to a laplace prior [83], which will automatically prune out elements of the priors. The parameter α can be set to a particular value, integrated out by using Jeffery's prior ($p(\alpha) = 1/\alpha$) [83], or chosen by maximization with respect to the model evidence [84]. While all of these methods are computationally simple, the last is a form of automatic relevance determination. See [40] for theoretical aspects comparing integrating out parameters compared to evidence maximization.

Regardless of the process of choosing α, all methods use the convex cost function similar to (3.196).

3.7.4 Case Study of HRDMM: Air Conditioner

A simple two-mode case study is presented to demonstrate some key HRDMM concepts. For this case study, the air conditioner example in Figure 3.8 is used. The primary goal of this study is to identify, only using observed data, the switching logic of the air conditioner. Recall that the system has two modes: Mode S_1 "ON" and Mode S_2 "OFF". The temperature y_k asymptotically approaches 30 if the air conditioner is off, and approaches 20 if the air conditioner is on. The system guards transition the system from on to off if $y_k < 24$ and from off to on if $y_k > 27$.

In terms of a HRDMM, the system is specified as follows:

$$y_k = \begin{cases} \theta_1^T x_k + \varepsilon_k & \text{if } m_k = 1 \quad (\text{'on'}) \\ \theta_2^T x_k + \varepsilon_k & \text{if } m_k = 2 \quad (\text{'off'}) \end{cases} \ , \tag{3.200}$$

where $x_k = [y_{k-1}, 1]$, and where $\theta_1 = [0.99, 0.2]$ and $\theta_2 = [0.96, 1.2]$, and ε_k is zero mean and normally distributed noise with variance $\sigma^2 = 0.01$.

The discrete transitions of the system are specified with the following discrete logic:

$$m_{k+1} = \begin{cases} 1 \text{ if } m_k = 1 \ \& \ x_k \in R_{11} \\ 2 \text{ if } m_k = 1 \ \& \ x_k \in R_{12} \\ 1 \text{ if } m_k = 2 \ \& \ x_k \in R_{21} \\ 2 \text{ if } m_k = 2 \ \& \ x_k \in R_{22} \end{cases} , \tag{3.201}$$

where the regions R_{ij} in (3.201) are defined by $R_{11} = \{x_k > 24\}$, $R_{12} = \{x_k \leq 24\}$, $R_{22} = \{x_k < 27\}$, $R_{21} = \{x_k \geq 27\}$.

The system was identified as follows. Initially all the hyperplanes[16] h_{12} and h_{21} were set equal to $h_{ij} = [01]^T$, i.e., independent of the regressor x_k. This essentially turns the first identification step into a classification problem. The priors on the hyperplanes were set as zero-mean Gaussians, (3.198) with $\sigma^2 = 1000$. Algorithm 3.4 was run until the change $\Delta < 1e-3$. The results of this identification process are shown in Figure 3.9, and demonstrate the principles behind HRDMM model identification.

[16]Recall that, by definition of a HRDMM, $h_{ii} = [00]^T$.

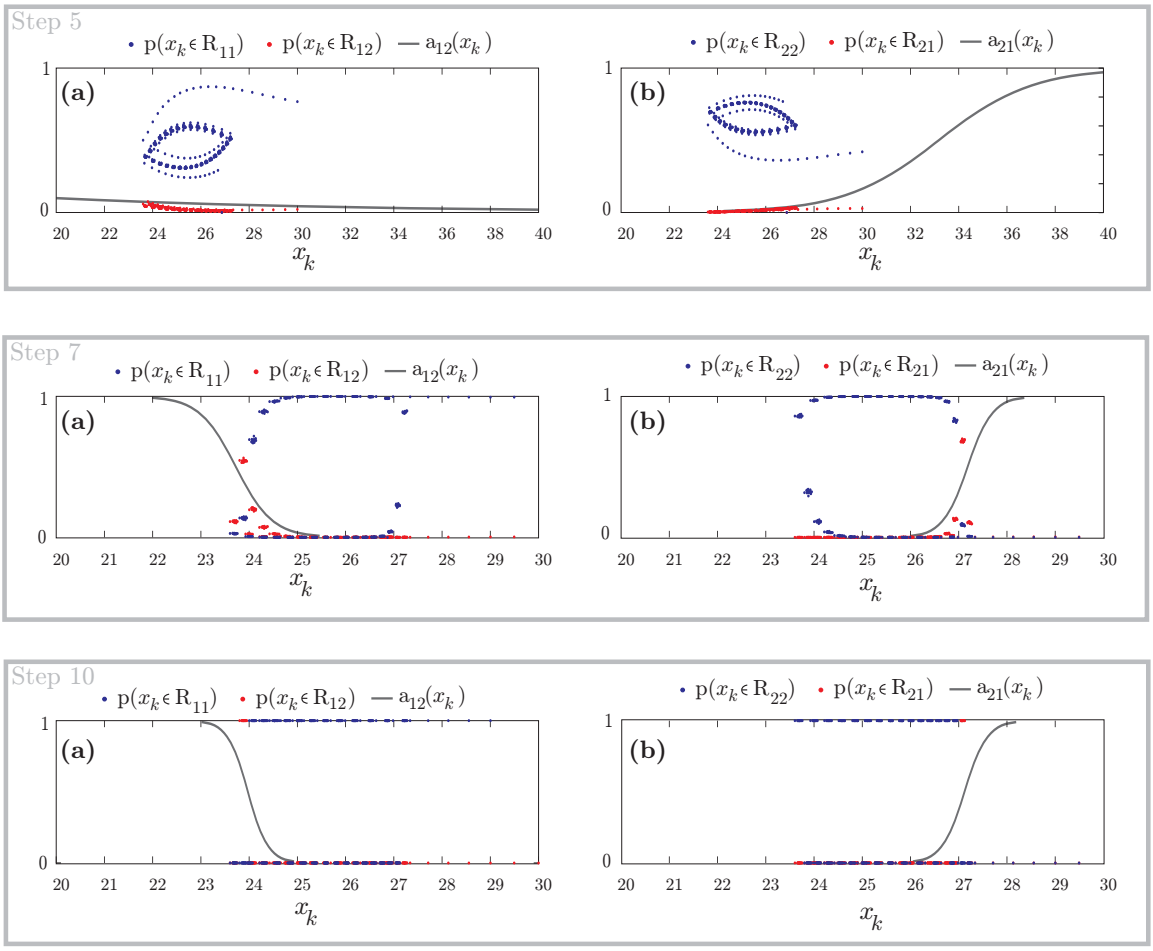

Figure 3.9: Identification of HRDMM model with Algorithm 3.4: Three steps of the algorithm are shown, along with the non-stationary transition kernel (grey line), and the regressor points x_k. The regressor points are color coded to represent which guard or invariant region R_{ij} they belong to, with the y-axis representing the probability of x_k belonging to a certain region. The transition kernel (grey line) is depicted for each algorithmic step in two separate plots. Plots denoted (a) show the transition kernel $a_{12}(x_k)$, which give the probability of transitioning from mode S_1 "ON", to mode S_2 "OFF". Plots denoted (b) show the transition kernel $a_{21}(x_k)$. By step 10, Algorithm 3.4 has converged, with the transition kernel $a_{12}(x_k)$ switching from ON to OFF when the temperature drops to around 24 degrees, and transition kernel $a_{21}(x_k)$ turning the air conditioner ON when the temperature is warmer then around 27 degrees.

3.8 Estimation and Prediction Using Identified Models

This thesis has developed several hybrid system models, including GLHMMs (Section 3.2), HSMMs and VTHMMs with GLM dynamics (Section 3.5), and HRDMMs (Section 3.7). Three algorithms: Expectation Maximization (EM), Variational Bayes (VB) and the Gibbs sampler, were applied to identify the developed hybrid system models from observed data. In this section, identified models will be used to create both non-causal and real-time implementable state estimators, where given a new data set, the goal is to infer the hidden (or latent) state sequence $m_{1:T}$. Four estimators are considered: The forward filter, the smoother, the fixed lag smoother, and the Viterbi algorithm. All of these estimators will use the developed forward-backward recursions, used in the identification process, to form a set of estimation tools.

Specifically, we will assume that a model (i.e., a GLHMMs, HSMM or HRDMM) has been identified from an observed data set $y_{1:T}$. After the identification process, estimates[17] Θ_{EST}, of the model parameters can be used to represent the identified model. The first estimate of the model parameters considered is the *maximum a posterior* value Θ_{MAP}:

$$\Theta_{\text{MAP}} = \underset{\Theta}{\operatorname{argmax}}\, \mathrm{p}\big(\Theta|y_{1:T}\big) \ . \tag{3.202}$$

The estimate Θ_{MAP} is directly calculated by EM algorithm, and is calculated by taking the analytic maximum of the factorized distribution $q(\Theta)$ when using VB. A second estimate, the expected value of the posterior density Θ_{E} can also be used:

$$\Theta_{\text{E}} = \int_{\Theta} \mathrm{p}\big(\Theta|y_{1:T}\big) d\Theta \ . \tag{3.203}$$

The estimate Θ_{E} is found using the Monte Carlo estimate (2.42) when using Gibbs sampling, and by taking the expected value of the factorized distribution $q(\Theta)$ for the VB algorithm. The choice of the two estimators (3.202) and (3.203) is somewhat arbitrary from a practical point of view; in this thesis preference is given to the expected value estimator (3.203), as it may avoid over-fitting and result in better generalization. However, if these estimators do not produce consistent results, then more complicated procedures should be used. For example, consider using the full simulated posterior of the Gibbs sampler [25], or performing smoothing with the variational algorithm using $q(\Theta)$ as in the identification process (i.e., the VB-E step) [36].

The four estimators are now defined. It is assumed that the model estimate Θ_{EST} has been formed, and that estimation takes place on a new observed data set $y_{1:T'}$ of length T'. The forward filter is now defined:

Definition 3.11 (Forward Filter)**.** *The forward filter is a casual filter suitable for processing data*

[17] In the case of HRDMMs, Θ is replaced by $\{\Theta, H\}$.

in real time, as it is observed. If k data $y_{1:k}$ have been observed, the forward filter calculates the probability of the current system mode m_k:

$$\mathrm{p}\big(m_k|y'_{1:k},\Theta_{EST}\big)\ . \tag{3.204}$$

$\diamond$

The forward filter (3.204) corresponds exactly to the forward variables in the forward-backward recursions used in the identification process (see (3.15) for GLHMMs, (3.127) for VTHMMs and HSMM where m_k is replaced by the joint state (m_k,τ_k), and (3.170) for HRDMMs). The corresponding forward recursions are then used to calculate (3.204) With each new collected data point, the previous mode estimate $\mathrm{p}\big(m_{k-1}|y'_{1:k-1},\Theta_{\mathrm{EST}}\big)$ is used, so only one forward recursion must be calculated.

Definition 3.12 (Smoother)**.** *The smoother is a non-causal filter that utilizes all available data to make an estimate of the systems' mode at t_k. The smoother calculates the marginal probability:*

$$\mathrm{p}\big(m_k|y_{1:T'},\Theta_{EST}\big)\ . \tag{3.205}$$

$\diamond$

The smoother (3.205) corresponds exactly to the marginal distribution $\gamma_k(i)$ defined by the forward-backward recursions. The calculation of m_k then involved running both the forward and backward recursions (see (3.24) for GLHMMs, (3.135) for VTHMMs and HSMM where m_k is replaced by the joint state (m_k,τ_k), and (3.172) for HRDMMs).

Definition 3.13 (Fixed Lag Smoother)**.** *A fixed lag smoother is a non-causal filter that delays the estimate of the current system mode m_k by L time steps. The fixed lag smoother calculates the marginal distribution:*

$$\mathrm{p}\big(m_k|y_{1:k+L},\Theta_{EST}\big)\ . \tag{3.206}$$

$\diamond$

In noisy systems or duration based models, waiting to collect L more data $y_{k+1:L}$ will smooth the mode estimate, and if the lag L is small enough, subsequent control or decision making may not be affected. The process of calculating (3.206) is essentially the same as for the smoother. The forward filter recursion is evaluated for $\mathrm{p}\big(m_{k+L}|y_{1:k+L},\Theta_{\mathrm{EST}}\big)$, and then the backwards recursion is run from $k+L-1$ to k.

Definition 3.14 (Viterbi Algorithm)**.** *The Viterbi algorithm [10] is a non-causal filter that calculates*

the optimal state sequence:

$$m_{1:T'} = \underset{m_{1:T'}}{\operatorname{argmax}}\, \mathrm{p}\big(m_{1:T'} | y_{1:T'}, \Theta_{EST}\big) \ . \tag{3.207}$$

$\diamond$

The Viterbi algorithm uses a two-step process to calculate the optimal state sequence, similar to the forward-backward recursion [10]. The algorithm is not provided, as it has been explicitly derived for HMMs in [10], VTHMMs in [33, 34], and HSMM in [79, 31]. The addition of GLM dynamics does not effect the Viterbi decoder, which only requires that the distribution $\mathrm{p}\big(y_k | m_k, x_k, \Theta_{\mathrm{EST}}\big)$ can be evaluated. The Viterbi algorithm is fundamentally different from the estimators in Definitions 3.11 to 3.13. The Viterbi algorithm does not provide a probabilistic estimate of the state m_k, instead choosing a maximum of $m_{1:T'}$, which can potentially reduce the intuition about the observed process. However, a benefit to the Viterbi algorithm, that has found wide use in the speech processing community, is that the estimated system sequence $m_{1:T'}$ obeys all constraints of the model. For instance, if some transition a_{ij} has zero probability, then the estimate $m_{1:T'}$ will not ever transition from S_i to S_j. The smoother (Def. 3.12) does not guarantee this. For instance if the smoother estimate m_k at each time step was chosen to be the argument maximizing (3.205), then the entire sequence $m_{1:T'}$ may not obey constraints a_{ij}.

The choice of estimation algorithm (Def 3.11 to 3.14), should be made based on the application. For many real-time control problems, the forward filter may be the only practical option available. The applications in this thesis will use all four decoding methods. For developing supervisory decoders for neural prosthetic devices in Chapter 5, a there is a preference for using the forward filter and fixed lag smoother as these estimators can give estimates of the discrete cognitive state within a reasonable time window. The fixed lag smoother is essential for use with duration based models, as the forward filter does not adequately take into account the duration constraints and time spent in each mode.

Chapter 4

Model Selection: Priors and Algorithms

Model selection is the process of selecting a specific *model class* from a proposed finite set of model classes using observed data. The model class defines the structure of the model, including the functional form, the number of model parameters, and any prior information used in the model inference process. A finite set of of m model classes is denoted $\mathcal{M} = \{\mathcal{M}_1, ..., \mathcal{M}_m\}$, where the i^{th} model class is denoted $\mathcal{M}_i$. It is convenient to think of a *model* as a particular realization from a model class, in that while the model class defines the model structure, the model itself defines the value of parameters associated with that structure.

This chapter will focus on applying model selection to the models presented in Chapter 3, using representative examples motivated by the neurophysiological problems of Chapter 5. The generalized linear hidden Markov models (Section 3.2) will be used extensively for algorithmic comparisons in this chapter, both due to their applicability to the applications of interest, and the wide range of GLHMM subclasses that are considered in practice.

Model selection algorithms are developed for the three identification algorithms considered in Chapters 2 and 3: Variational Bayes, Expectation Maximization, and Gibbs sampling. While this thesis restricts model selection to these inference algorithms, it should be noted that there are alternative model selection approaches including variable dimension Monte Carlo samplers and annealed importance sampling, which are briefly reviewed in Section 4.1.2.

This thesis will use *Bayesian model class selection* to select the "best" model from a discrete set of model classes. Given a set of m possible model classes, $\mathcal{M} = \{\mathcal{M}_1, ..., \mathcal{M}_m\}$, the posterior probability of each model class is given by Bayes' theorem:

$$\mathrm{P}\big(\mathcal{M}_i|y_{1:T}\big) = \frac{\mathrm{p}\big(y_{1:T}|\mathcal{M}_i\big)\,\mathrm{P}\big(\mathcal{M}_i\big)}{\sum_{j=1}^{m}\mathrm{p}\big(y_{1:T}|\mathcal{M}_j\big)\,\mathrm{P}\big(\mathcal{M}_j\big)} \quad , \tag{4.1}$$

where $y_{1:T}$ is the observed data, $\mathrm{P}\big(\mathcal{M}_i\big)$ is the prior probability of model class i, and the conditioning

on the set $\mathcal{M}$ is left implicit. If all models are considered equally plausible a priori, so that $\mathrm{P}(\mathcal{M})$ is a uniform distribution, then the probability of selecting a model class from set $\mathcal{M}$ is based entirely on the *model evidence*: $\mathrm{p}(y_{1:T}|\mathcal{M}_i)$, which is also referred to as the marginal data likelihood. The "best" model is then the model class with the highest model evidence. Section 4.1 will explicitly show how using the model evidence for model selection automatically penalizes complex models, and avoids over-fitting of data. If there are several model classes with similar posterior probabilities, a subset of model classes can be used for inference with Bayesian model averaging [85].

In addition to choosing between a discrete set of models with Bayesian model class selection (4.1), the process of using automatic structure determination priors (ASD) is discussed in Section 4.3. ASD priors are informative distributions that allow learning algorithms to automatically "prune" unnecessary elements from the model structure; in this thesis ASD priors are used to determine the allowed transitions between discrete system states. Using ASD priors in combination with model selection allows the number of models in the set $\mathcal{M}$ to be reduced, making the inference problem less computationally expensive.

The remainder of this chapter makes several contributions:

- The importance of model evidence for model selection is discussed in Section 4.1, and decomposition of the model evidence to demonstrate model complexity penalization is reviewed. Approximate methods of estimating the model evidence based on EM, VB, and Gibbs sampler are reviewed. Based on this review, current methods of model evidence estimation using posterior samples from the Gibbs sampler are specialized of latent variable models, and equivalence between two existing methods is proven.

- Model evidence approximations are empirically tested and compared on simulated data sets. It is shown that the VB and Gibbs sampling allow for significantly improved model selection as compared to the EM algorithm.

- The effect of automatic structure determination priors are demonstrated in conjunction with model selection, and are shown to provide an efficient means of inferring neurological models.

- The empirical results obtained by applying model selection methods to simulated data sets are considered in conjunction with derived theoretical properties of EM, VB, and Gibbs algorithms, in order to provide practical guidance on selection of the best model selection method. It is concluded that the variational method is most appropriate for the neurophysiological applications considered in this thesis, due to the computational requirements incurred from the Gibbs sampler, and lack of suitable model selection tools available for the EM algorithm.

- VB is used in conjunction with ASD priors and HSMMs to automatically determine the number of discrete states or *movement primitives* present in a *bee dance* data set. This is the first time

ASD priors have been used with HSMM, and results indicate that using little prior knowledge, the model selection tools can produce results rivaling that of a an experienced user manually segmenting the data.

4.1 Approximating the Model Evidence

This section begins with a general discussion about model complexity and decompositions of the model evidence $\mathrm{p}(y_{1:T}|\mathcal{M}_i)$. These decompositions are important as they motivate the approximations made to model evidence in Sections 4.1.1 to 4.1.3. Because we are interested in applying model selection to the hybrid system and latent variable models in Chapter 3, the following notation is assumed: An observed data sequence of length T is denoted $y_{1:T}$, with the associated latent mode variables denoted $m_{1:T}$. The model class $\mathcal{M}_i$ defines the parameter set[1] Θ, and also defines the complete data likelihood: $\mathrm{p}(y_{1:T}, m_{1:T}|\Theta, \mathcal{M}_i)$.

A simple decomposition of the model evidence is derived by first marginalizing out the latent modes $m_{1:T}$ and then using the theorem of total probability:

$$\mathrm{p}(y_{1:T}|\mathcal{M}_i) = \int \sum_{m_{1:T}\in\mathcal{S}^T} \mathrm{p}(y_{1:T}, m_{1:T}|\Theta, \mathcal{M}_i)\, \mathrm{p}(\Theta|\mathcal{M}_i) d\Theta \ . \tag{4.2}$$

This decomposition shows the difficulty in evaluating the model evidence: the integral in (4.2) is typically too complicated to analytically evaluate; Θ is typically of high dimension; and the summation over all possible discrete mode sequences implies the complexity grows combinatorially with data length T. However, the above equation (4.2) can be simplified by using a key result of the forward-backward algorithm (Alg. 2.5) which evaluates the incomplete data likelihood for a specific Θ:

$$\mathrm{p}(y_{1:T}|\Theta, \mathcal{M}_i) = \sum_{m_{1:T}\in\mathcal{S}^T} \mathrm{p}(y_{1:T}, m_{1:T}|\Theta, \mathcal{M}_i) \ . \tag{4.3}$$

Evaluating the distribution (4.3) by dynamic programming allows for simplification of the model evidence (4.2). However this still leaves (4.2) intractable because of the dimension of Θ and lack of a closed form for the incomplete data likelihood (4.3).

Let us now derive a decomposition of the model evidence which highlights the complexity penalization inherent in the use of Bayesian model class selection. Assuming that the incomplete data likelihood (4.3) can be evaluated, a decomposition derived by Muto and Beck [86] is repeated here. To start the derivation, note that the marginal posterior distribution of the parameters $\mathrm{p}(\Theta|y_{1:T}, \mathcal{M}_i)$

[1] The association of the parameter Θ to the model class model class $\mathcal{M}_i$ is left implicit, and no subscript is added to the parameter set.

is normalized (integrates to one), implying:

$$\log \mathrm{p}\big(y_{1:T}|\mathcal{M}_i\big) = \log\big[\mathrm{p}\big(y_{1:T}|\mathcal{M}_i\big)\big] \int \mathrm{p}\big(\Theta|y_{1:T},\mathcal{M}_i\big)d\Theta \ . \tag{4.4}$$

Since the model evidence is not a function of the parameters Θ, the log evidence can be brought inside the integral, and expanded using Bayes' theorem:

$$\log \mathrm{p}\big(y_{1:T}|\mathcal{M}_i\big) = \int \log\left[\frac{\mathrm{p}\big(y_{1:T}|\Theta,\mathcal{M}_i\big)\,\mathrm{p}\big(\Theta|\mathcal{M}_i\big)}{\mathrm{p}\big(\Theta|y_{1:T},\mathcal{M}_i\big)}\right] \mathrm{p}\big(\Theta|y_{1:T},\mathcal{M}_i\big)d\Theta \ . \tag{4.5}$$

Equation (4.5) is further decomposed using the properties of logarithms:

$$\begin{aligned}&\log \mathrm{p}\big(y_{1:T}|\mathcal{M}_i\big) = \\ &\quad \int \log\big[\mathrm{p}\big(y_{1:T}|\Theta,\mathcal{M}_i\big)\big]\,\mathrm{p}\big(\Theta|y_{1:T},\mathcal{M}_i\big)d\theta - \int \log\left[\frac{\mathrm{p}\big(\Theta|y_{1:T},\mathcal{M}_i\big)}{\mathrm{p}\big(\Theta|\mathcal{M}_i\big)}\right] \mathrm{p}\big(\Theta|y_{1:T},\mathcal{M}_i\big)d\theta \ .\end{aligned} \tag{4.6}$$

Decomposition (4.6) demonstrates that the model evidence is composed of two terms, a data fit term and a complexity penalization term: The first term of (4.6) is the expected value of the log-likelihood of the data, and represents the average fit of the model class. The second term of (4.6) is the KL divergence (or relative entropy) from the prior parameter distribution to the posterior, and is a measure of the information gained from the data. This second term penalizes model complexity, preventing selection of model classes that over-fit the data.

The following sections will make use of the two decompositions (4.2) and (4.6) to approximate the model evidence.

4.1.1 Information Criteria and Laplace's Asymptotic Method

In this section Akaike's information criterion (AIC), the Bayesian information criterion (BIC), and Laplace's method for asymptotic approximation are derived as approximations to the model evidence.

These derivations follow from Beck and Yuen [87], and are shown explicitly to allow comparison of these methods and to stress the implications of assumptions that are used in their derivation. Here it is assumed that the parameter set Θ is a vector of M elements.

First Laplace' method is derived by considering the integral of the the incomplete data likelihood:

$$\mathrm{p}\big(y_{1:T}|\mathcal{M}_i\big) = \int \mathrm{p}\big(y_{1:T}|\Theta,\mathcal{M}_i\big)\,\mathrm{p}\big(\Theta|\mathcal{M}_i\big)d\Theta \ . \tag{4.7}$$

Equation (4.7) comes from the model evidence (4.2), using the forward-backward algorithm to evaluate the incomplete data likelihood (4.3). The model evidence (4.7) is now approximated by defining $f(\Theta) \triangleq \mathrm{p}\big(y_{1:T}|\Theta,\mathcal{M}_i\big)\,\mathrm{p}\big(\Theta|\mathcal{M}_i\big)$, and then replacing $\log f(\Theta)$ with a 2^{nd}-order Taylor expansion at

the maximum a posteriori value Θ_{MAP}, which is a maximum of $f(\Theta)$. This Taylor series approximation is equivalent to replacing the posterior PDF for Θ with Gaussian distribution [87]. The Taylor expansion is given by

$$\log f(\Theta) \approx \log f(\Theta_{MAP}) - \frac{1}{2}(\Theta - \Theta_{MAP})^T H(\Theta_{MAP})(\Theta - \Theta_{MAP}) \tag{4.8}$$

where the Hessian takes the form $H(\Theta_{MAP}) = -\nabla\nabla \log \mathrm{p}\big(y_{1:T}|\Theta_{MAP}, \mathcal{M}_i\big)\, \mathrm{p}\big(\Theta_{MAP}|\mathcal{M}_i\big)$, and the first-order term in the Taylor expansion does not appear, since Θ_{MAP} is a local maximum of f. By substituting the Taylor expansion (4.8) into the integral expression of the model evidence (4.7), the model evidence is now expressed as a unnormalized Gaussian distribution and can be analytically integrated:

$$\begin{aligned} \mathrm{p}\big(y_{1:T}|\mathcal{M}_i\big) &\approx \int f(\Theta_{MAP}) \exp\left[\frac{1}{2}(\Theta - \Theta_{MAP})^T H(\Theta_{MAP})(\Theta - \Theta_{MAP})\right] d\Theta \\ &= f(\Theta_{MAP})(2\pi)^{m/2} |H(\Theta_{MAP})|^{-1/2} \ . \end{aligned} \tag{4.9}$$

Equation (4.9) gives Laplace's asymptotic approximation to the model evidence:

$$\log \mathrm{p}\big(y_{1:T}|\mathcal{M}_i\big) \approx \log \mathrm{p}\big(y_{1:T}|\Theta_{MAP}, \mathcal{M}_i\big) + \log \mathrm{p}\big(\Theta_{MAP}|\mathcal{M}_i\big) + \frac{M}{2}\log(2\pi) - \frac{1}{2}\ln|H(\Theta_{MAP})| \ . \tag{4.10}$$

The last three terms in Laplace's approximation (4.10) are often called the *Log Ockham Factor*, $B = \log \mathrm{p}\big(\Theta_{MAP}|\mathcal{M}_i\big) + \frac{M}{2}\log(2\pi) - \frac{1}{2}\ln|H(\Theta_{MAP})|$, which penalizes model complexity [24, 87]. However, calculating the Hessian can require considerable analytical analysis and be computationally intensive [88] and specific to each model class. Beck and Yuen [87] show that as the number of data points T becomes very large, and assuming the posterior has a single peak[2], the Ockham factor can be approximated by $B = \frac{1}{2}M\log N + R$, where M is the number of parameters, N is the number of observed data points, and R is a function that depends on the prior and is independent of N.

The Bayesian information criterion (BIC) (Schwarz [89]) is now derived by replacing the Ockham factor in (4.10) with $B = \frac{1}{2}M\log N$:

$$BIC(\mathcal{M}_i) = \log \mathrm{p}\big(y_{1:T}|\Theta_{MAP}, \mathcal{M}_i\big) - \frac{1}{2}M\log N \ . \tag{4.11}$$

Note that the approximation made in BIC (4.11) ignores the effect of the prior information in the model selection process, which may be reasonable if the priors are very broad or non-informative.

[2]Is globally identifiable [87].

Akaike's information criterion (AIC) (Akaike [90])[3] can now be viewed as a modification of BIC:

$$AIC(\mathcal{M}_i) = \log \mathrm{p}\big(y_{1:T}|\Theta_{MAP}, \mathcal{M}_i\big) - M \ . \tag{4.12}$$

It should be noted that any method, including Gibbs sampling or variational methods, can be used to estimate the MAP parameter values Θ_{MAP}, so AIC, BIC and Laplace's method can be generally applied. However, it can be the case that the posterior distribution of the parameters is not approximately Gaussian (or even uni-modal) and Laplace's method and subsequent asymptotic approximations are inappropriate. The following two sections focus on approximations to the model evidence that do not require the asymptotic and peaked posterior assumptions used in this section.

4.1.2 Model Evidence Calculations Using Posterior Samples from the Gibbs Sampler

Model selection using Markov Chain Monte Carlo methods is often approached using reversible jump algorithms that sample from variable dimension models, where the number of model parameters is treated as a uncertain. To use these methods it is necessary to specify all model classes $\mathcal{M} = \{\mathcal{M}_1, ..., \mathcal{M}_m\}$ at the outset, and require careful choice and tuning of transition rules between models to operate efficiently [25]. Also, recently developed methods such as annealed importance sampling [91] and tempered-MCMC [92], have inherent properties that allow model selection, but may require excessive computation effort [23].

This section instead seeks to approximate the model evidence $\mathrm{p}\big(y_{1:T}|\mathcal{M}_i\big)$ by using the posterior samples drawn from the two-stage Gibbs Sampler (Alg. 2.4). While MCMC methods have rapidly become mainstream for model updating and prediction, calculating the model evidence to facilitate model selection is extremely challenging. Direct Monte Carlo simulation for the model evidence (4.7) requires sampling from the prior $\mathrm{p}\big(\Theta|\mathcal{M}_i\big)$. However, this creates inefficient estimators because $\mathrm{p}\big(y_{1:T}|\Theta, \mathcal{M}_i\big)$ will have high values only in a small region of the parameter space, resulting in high variance estimators.

Two methods for estimating the model evidence from posterior samples are considered here. This thesis will refer to the methods as: the Stationarity method, proposed in a restricted way by Ritter and Tanner [93], and in generality by Cheung and Beck [94]; and the Rao-Blackwellization method proposed by Chib [95].

While the algorithm by Cheung and Beck [94] is applicable to a wide range of MCMC methods, the Stationarity method will only be considered in the context of latent variable models and Gibbs sampling. The Rao-Blackwellization method [95] is specific to Gibbs sampling, but is documented

[3] Akaike presented AIC in an equivalent form which required minimization rather then maximization and differs by a factor of two.

by Chib [96] to give better performance in estimating model evidence than the Stationarity method. In Section 4.1.2.2 the Stationarity method is specialized for use with latent variable methods and it is proven that this marginalized Stationarity method provides exactly the same estimator as [95]. Furthermore, the Rao-Blackwellization method [95] does not extend easily to general sampling algorithms, or even the multi-stage Gibbs sampler. This implies that marginalizing the Stationarity method, with its superior generalization properties, is a valuable tool for use in latent variable models. Given the wide applicability of latent variable models in economics, biology, engineering, and physics, the marginalized Stationarity method should be of interest in many fields.

As stated by Chib [96] and references within, there exist other estimators for model evidence in the literature [97, 98] but these are not considered here as they require either extensive tuning or suffer from instability.

Recall that MCMC algorithms such as the Gibbs sampler draw samples from the posterior parameter PDF. In the case of latent variable models, samples from the joint distribution $\{\hat{\Theta}^{(t)}, \hat{m}_{1:T}^{(t)}\} \sim \mathrm{p}\big(\Theta, m_{1:T}|y_{1:T}, \mathcal{M}_i\big)$ are collected. Furthermore, MCMC samples of Θ are distributed like the marginal distribution: $\{\hat{\Theta}^{(t)}\} \sim \mathrm{p}\big(\Theta|y_{1:T}, \mathcal{M}_i\big)$. With enough samples N_t, the expected value of arbitrary functions f are approximated asymptotically by the Monte Carlo estimate (2.41):

$$\frac{1}{N_t}\sum_{t=1}^{N_t} f(\hat{\Theta}^{(t)}) \rightarrow \int_{\Theta} f(\Theta)\,\mathrm{p}\big(\Theta|y_{1:T}, \mathcal{M}_i\big) \quad \text{as} \quad N_t \rightarrow \infty \ . \tag{4.13}$$

This Monte Carlo estimate of the marginal distribution (4.13) also applies to the latent variables:

$$\frac{1}{N_t}\sum_{t=1}^{N_t} f(\hat{m}_{1:T}^{(t)}) \rightarrow \sum_{m_{1:T}\in\mathcal{S}^T} f(m_{1:T})\,\mathrm{P}\big(m_{1:T}|y_{1:T}, \mathcal{M}_i\big) \quad \text{as} \quad N_t \rightarrow \infty \ . \tag{4.14}$$

4.1.2.1 Rao-Blackwellization for Estimation of the Model Evidence

Chib has introduced a method for using posterior samples from the Gibbs sampler to estimate the model evidence [95]. Using Bayes' theorem the model evidence is decomposed as:

$$\mathrm{p}\big(y_{1:T}|\mathcal{M}_i\big) = \frac{\mathrm{p}\big(y_{1:T}|\Theta, \mathcal{M}_i\big)\,\mathrm{p}\big(\Theta|\mathcal{M}_i\big)}{\mathrm{p}\big(\Theta|y_{1:T}, \mathcal{M}_i\big)} \ . \tag{4.15}$$

This identity (4.15) holds for any Θ. For any given Θ, say $\Theta^\star$, all the PDFs on the right hand side of equation (4.15) can be evaluated analytically except for the posterior parameters PDF $\mathrm{p}\big(\Theta|y_{1:T}, \mathcal{M}_i\big)$, as its normalizing constant is not known. The posterior parameter PDF is instead approximated by a Monte Carlo estimate $\hat{\pi}(\Theta^\star|y_{1:T})$, giving the following approximation to the model evidence (4.15):

$$\log \mathrm{p}\big(y_{1:T}|\mathcal{M}_i\big) \approx \log \mathrm{p}\big(y_{1:T}|\Theta^\star, \mathcal{M}_i\big) + \log \mathrm{p}\big(\Theta^\star|\mathcal{M}_i\big) - \log \hat{\pi}(\Theta^\star|y_{1:T}) \ . \tag{4.16}$$

The Monte Carlo approximation $\hat{\pi}(\Theta^{\star}|y_{1:T})$ is now derived. First, let the t^{th} posterior sample of the Gibbs sampler be $\{\hat{\Theta}^{(t)}, \hat{m}_{1:T}^{(t)}\}$. Also, recall the posterior density for the parameters Θ is evaluated using the Theorem of Total Probability:

$$\mathrm{p}\big(\Theta|y_{1:T}, \mathcal{M}_i\big) = \sum_{m_{1:T} \in \mathcal{S}^T} \mathrm{p}\big(\Theta|y_{1:T}, m_{1:T}, \mathcal{M}_i\big)\, \mathrm{P}\big(m_{1:T}|y_{1:T}, \mathcal{M}_i\big) \ . \tag{4.17}$$

$\hat{\pi}(\Theta^{\star}|y_{1:T})$ is now defined as the Monte Carlo estimate (4.14) of the posterior density (4.17):

$$\hat{\pi}(\Theta^{\star}|y_{1:T}) = \frac{1}{N_t} \sum_{t=1}^{N_t} \mathrm{p}\big(\Theta^{\star}|y_{1:T}, \hat{m}_{1:T}^{(t)}, \mathcal{M}_i\big) \ . \tag{4.18}$$

This approximation (4.18) is called Rao-Blackwellization [95], hence the name assigned to this method in this thesis. By substituting (4.18) into the model evidence expression (4.16), a principled approximation to the model evidence is derived.

Chib extends this Rao-Blackwellization method from the two-stage Gibbs sampler presented here, into the multi-stage Gibbs sampler [95], although this requires simulating from a set of new conditional distributions, with the additional amount of computation increasing linearly with the number of additional stages of the sampler. The additional simulation steps are added to any overhead already inherent with the use of a multi-stage Gibbs sampler.

Chib declares in [96] that this Rao-Blackwellization method is a better approximation then the Stationarity method (Section 4.1.2.2). It is stated in [96] that: "when Θ is high dimensional and the model contains latent variables [the Kernel method of [93]] tends to be less accurate then [the Rao-Blackwellization] method...". However, this thesis shows that by marginalizing the Stationarity method, the two estimators are equivalent in the latent variable case.

4.1.2.2 The Stationarity Condition for Estimating the Model Evidence from Posterior Samples

The Stationarity method[4] presented in this section was originally derived by Ritter and Tanner [93] in a restricted form called the Gibbs Stopper, designed for monitoring convergence of the Gibbs sampler. Cheung and Beck [94] have recently derived a more complete view of this method, facilitating model selection with both Gibbs samplers and several classes of MCMC algorithms.

This section starts by directly applying the Stationarity method from [94] to latent variable models. This method is then specialized for latent variable models by using marginalization, and its equivalence to the Rao-Blackwellization method of Chib for two-stage sampling is proven. Recall that when the two-stage Gibbs sampler (Alg. 2.4) is applied to latent variable models, one constructs

[4]This method is denoted the "Stationarity" method because of the inherent use of the the stationary property of the induced Markov chain created by the MCMC algorithms.

a Markov transition kernel $K(\Theta^\star, m^\star_{1:T}|\Theta, m_{1:T})$ with a stationary distribution $\mathrm{p}\big(\Theta, m_{1:T}|y_{1:T}, \mathcal{M}_i\big)$ where:

$$K(\Theta^\star, m^\star_{1:T}|\Theta, m_{1:T}) = \mathrm{p}\big(m^\star_{1:T}|\Theta^\star, y_{1:T}, \mathcal{M}_i\big)\, \mathrm{p}\big(\Theta^\star|m_{1:T}, y_{1:T}, \mathcal{M}_i\big) \ . \tag{4.19}$$

The key idea of the Stationarity method is that, by construction, the Markov chain corresponding to this transition kernel 4.19 has its stationary PDF as the posterior density implying:

$$\mathrm{p}\big(\Theta^\star, m^\star_{1:T}|y_{1:T}, \mathcal{M}_i\big) = \sum_{m_{1:T}\in\mathcal{S}^T} \int K(\Theta^\star, m^\star_{1:T}|\Theta, m_{1:T})\, \mathrm{p}\big(\Theta, m_{1:T}|y_{1:T}, \mathcal{M}_i\big) d\Theta \ . \tag{4.20}$$

Following directly from [94], a Monte Carlo estimate (4.13) can now be used to evaluate the joint posterior (4.20):

$$\mathrm{p}\big(\Theta^\star, m^\star_{1:T}|y_{1:T}, \mathcal{M}_i\big) \approx \frac{1}{N_t} \sum_{t=1}^{N_t} K(\Theta^\star, m^\star_{1:T}|\hat{\Theta}^{(t)}, \hat{m}^{(t)}_{1:T}) \ . \tag{4.21}$$

This estimator (4.21) is now modified for use with latent variable models, where the model evidence calculations (4.16) require an estimate of the marginal posterior: $\mathrm{p}\big(\Theta^\star|y_{1:T}, \mathcal{M}_i\big)$ instead of the joint posterior (4.21). This specialization of the Stationarity method (4.21) to latent variable models is realized by considering the form of the desired distribution:

$$\mathrm{p}\big(\Theta^\star|y_{1:T}, \mathcal{M}_i\big) = \frac{\mathrm{p}\big(\Theta^\star, m^\star_{1:T}|y_{1:T}, \mathcal{M}_i\big)}{\mathrm{p}\big(m^\star_{1:T}|\Theta^\star, y_{1:T}, \mathcal{M}_i\big)} \ , \tag{4.22}$$

where (4.22) is derived using the product rule. Equation (4.22) is significant because of the form of the kernel (4.19), which also contains the term $\mathrm{p}\big(m^\star_{1:T}|\Theta^\star, y_{1:T}, \mathcal{M}_i\big)$. This implies that the Stationarity method (4.20) can be simply modified as follows. First the kernel (4.19) is substituted into (4.20):

$$\begin{aligned}
&\mathrm{p}\big(\Theta^\star, m^\star_{1:T}|y_{1:T}, \mathcal{M}_i\big) \\
&\quad = \sum_{m_{1:T}\in\mathcal{S}^T} \int_\Theta \mathrm{p}\big(m^\star_{1:T}|\Theta^\star, y_{1:T}, \mathcal{M}_i\big)\, \mathrm{p}\big(\Theta^\star|m_{1:T}, y_{1:T}, \mathcal{M}_i\big)\, \mathrm{p}\big(\Theta, m_{1:T}|y_{1:T}, \mathcal{M}_i\big) d\Theta \ .
\end{aligned} \tag{4.23}$$

Now the decomposition (4.22) can be used to cancel out terms in (4.23), resulting in:

$$\mathrm{p}\big(\Theta^\star|y_{1:T}, \mathcal{M}_i\big) = \sum_{m_{1:T}\in\mathcal{S}^T} \int_\Theta \mathrm{p}\big(\Theta^\star|m_{1:T}, y_{1:T}, \mathcal{M}_i\big)\, \mathrm{p}\big(\Theta, m_{1:T}|y_{1:T}, \mathcal{M}_i\big) d\Theta \ . \tag{4.24}$$

By marginalizing over the variable Θ in (4.24), the Rao-Blackwellization method of Section 4.1.2.1

is shown to be a special case of the Stationarity method (4.24) for a two-stage sampler:

$$\mathrm{p}\big(\Theta^{\star}|y_{1:T},\mathcal{M}_i\big) = \sum_{m_{1:T}\in\mathcal{S}^T} \mathrm{p}\big(\Theta^{\star}|m_{1:T},y_{1:T},\mathcal{M}_i\big)\,\mathrm{P}\big(m_{1:T}|y_{1:T},\mathcal{M}_i\big)\ , \tag{4.25}$$

where the density (4.25) then has the following Monte Carlo estimate (4.14):

$$\mathrm{p}\big(\Theta^{\star}|y_{1:T},\mathcal{M}_i\big) \approx \frac{1}{N_y}\sum_{t=1}^{N_t} \mathrm{p}\big(\Theta^{\star}|\hat{m}_{1:T}^{(t)},y_{1:T},\mathcal{M}_i\big)\ . \tag{4.26}$$

The result in equation 4.26 is equivalent to the estimator derived by Chib (4.18), yet by using the Stationarity method we have added generality to the method, as described in the following remark:

Remark 4.1. The specialization of the Stationarity method (4.20) to latent variable models applies for the general sampling algorithms defined in [94]. Consider the following multi-stage Gibbs sampler kernel, where now the parameter space Θ is sampled in stages such that $\Theta = [\Theta_1^T, \Theta_2^T, ..., \Theta_R^T]^T$. A kernel for this sampler is then:

$$\begin{aligned} K(\Theta^{\star},m_{1:T}^{\star}|\Theta,m_{1:T}) = \mathrm{p}\big(m_{1:T}^{\star}|\Theta_1^{\star},\ldots,\Theta_R^{\star},y_{1:T},\mathcal{M}_i\big)\,\mathrm{p}\big(\Theta_R^{\star}|\Theta_1^{\star},\ldots,\Theta_{R-1}^{\star},m_{1:T},y_{1:T},\mathcal{M}_i\big)\ldots \\ \ldots\mathrm{p}\big(\Theta_1^{\star}|\Theta_2,\ldots,\Theta_R,m_{1:T},y_{1:T},\mathcal{M}_i\big)\ . \end{aligned} \tag{4.27}$$

Note that substituting (4.27) into the Stationarity condition (4.20) again facilitates the removal of the term $\mathrm{p}\big(m_{1:T}^{\star}|\Theta_1^{\star},\ldots,\Theta_R^{\star},y_{1:T},\mathcal{M}_i\big)$ from the integral in (4.27). An equivalent multi-stage generalization of (4.26) can then be devised. This technique should have wide applicability due to the extensive use of latent variable models in biology, economics, and engineering. ◇

Remark 4.2 (Cheung and Beck [94]). The KL divergence (or information gain) from the prior to the posterior in the decomposition (4.6) can be obtained by the expression:

$$\int \log\left[\frac{\mathrm{p}\big(\Theta|y_{1:T},\mathcal{M}_i\big)}{\mathrm{p}\big(\Theta|\mathcal{M}_i\big)}\right]\mathrm{p}\big(\Theta|y_{1:T},\mathcal{M}_i\big)d\theta \approx \frac{1}{N_t}\sum_{t=1}^{N_t}\log\mathrm{p}\big(y_{1:T}|\hat{\Theta}^{(t)},\mathcal{M}_i\big) - \log\mathrm{p}\big(y_{1:T}|\mathcal{M}_i\big)\ . \tag{4.28}$$

This approximation (4.28) will be used in Section 4.2.1 to compare against the KL divergence expression in the variational lower bound (2.20) between the prior and the factorized posterior $q(\Theta)$.

◇

4.1.3 Variational Lower Bound

Variational Bayes (Section 2.2.1) inherently creates a lower-bound approximation (2.20) to the model evidence (4.2). This lower-bound approximation has previously been used for model selection in a series of models including multi-variate auto-regressive order estimation [99], and the estimation of

the number of discrete components in mixture models [23]. In this latter reference, VB was found to have superior model selection performance to BIC, and Annealed Importance Sampling. Model selection for (discrete) hidden Markov models has been conducted on simulated data sets using VB [36], but in the context of pruning the model structure, and not directly comparing discrete model classes.

In this section the variational lower bound for latent variable models (2.20) is restated:

$$\mathcal{L}(q(m_{1:T}, \Theta)) = \int_{\Theta} \sum_{m_{1:T} \in \mathcal{S}^T} q(\Theta) q(m_{1:T}) \log \frac{\mathrm{p}\big(m_{1:T}, y_{1:T} | \Theta\big)}{q(m_{1:T})} d\theta - \int_{\Theta} q(\Theta) \log \frac{q(\Theta)}{\mathrm{p}\big(\Theta\big)} d\Theta \ , \quad (4.29)$$

to allow comparison to the model evidence decomposition (4.6).

In Sections 2.3.2.2 and 3.3.2 it was shown that the first term of (4.29) is equivalent to a sub-normalized probability distribution: $\mathrm{p}\big(y_{1:T} | \bar{\Theta}, \mathcal{M}_i\big)$, which is a lower bound to the incomplete data likelihood. The lower bound (4.29) can hence be thought of a data fit term (an approximation to the incomplete data likelihood) and a complexity penalization term, which is the KL divergence of the identified model parameters $q(\Theta)$ from the prior:

$$\mathcal{L}(q(m_{1:T}, \Theta)) = \mathrm{p}\big(y_{1:T} | \bar{\Theta}, \mathcal{M}_i\big) - \int_{\Theta} q(\Theta) \log \frac{q(\Theta)}{\mathrm{p}\big(\Theta\big)} d\Theta \ . \quad (4.30)$$

By writing the variational lower-bound as a data fit term minus a KL divergence term (information gain) in (4.30), it can be directly compared to the decomposition of the model evidence (4.6) proposed by Muto and Beck. The similarity of (4.6) and (4.30) adds insight into the effectiveness of the variational method, and is the first time the structure of the lower bound has been directly compared to the model evidence using these decompositions.

The KL divergence term in (4.30) is empirically compared to the actual KL divergence from the prior to the posterior as estimated by the Gibbs sampler (4.28) in Section 4.2.1. This appears to be the first time these two information theoretic terms have ever been directly compared.

4.2 Comparison of Model Selection Methods

The previous section (4.1) has discussed and derived several methods for approximating the model evidence. These various approximations in turn allow the use of Bayesian model class selection (4.1), to choose between a finite set of model classes.

This section introduces several simulated data sets for empirical comparison of the discussed model selection approaches. Specifically, this section compares the performance of: (1) AIC (4.12) and BIC (4.11) information critera used in conjunction with the EM algorithm; (2) the Stationarity method (4.18) using samples from the Gibbs sampler (GIBBS); and (3) the variational lower bound

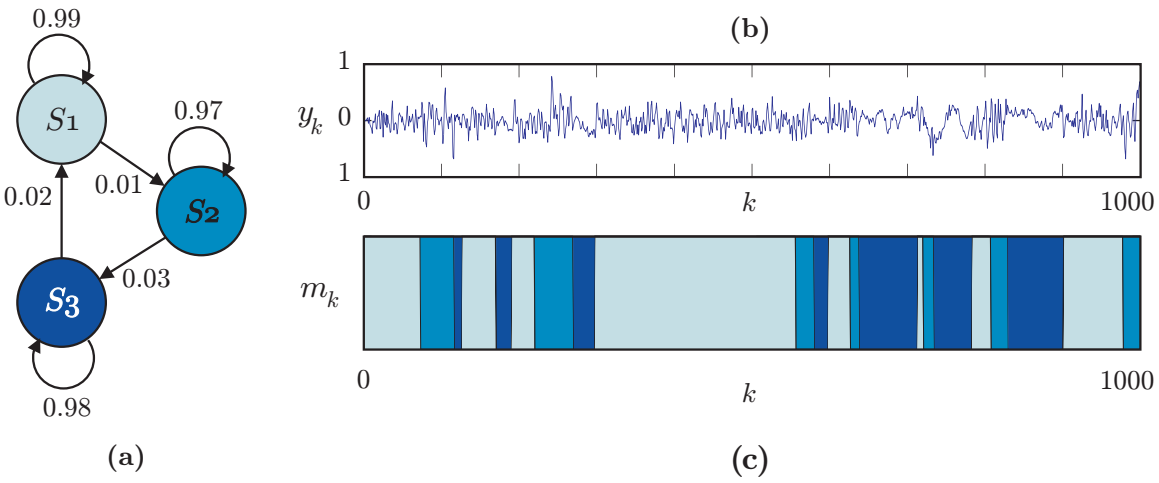

Figure 4.1: 3-state cyclic hidden Markov model: (a) Finite graph representation of discrete state and associated transitions. (b) A simulated output $y_{1:T}$ of length $T = 1000$ using the parameters defined in (4.32). (c) Corresponding simulated discrete states $m_{1:T}$.

approximation (4.29) (VB). Laplace's asymtotic method is not considered for two reasons: First, the required calculations to analytically compute the Hessian in the applications of interest are so complicated as to be impractical. Second, Laplace's method is known to be inaccurate in cases where the posterior is not well represented by a Gaussian distribution; this is exactly the case found with the sum-to-one constraints found with the Markovian transition kernel A in hidden Markov models [82]

The rest of this section is arranged as follows: First a specific GLHMM is defined to provide empirical comparisons. In Section 4.2.1 the defined GLHMM is used to compare information theoretic quantities associated with the model evidence. In this section only the "true" GLHMM mode is considered, and is designed to give intuition into how Bayesian model class selection automatically penalizes complexity. In addition, Section 4.2.1 gives direct empirical comparison of the information gain as calculated by both the variational approximation, and the information gained in the un-approximated posterior using Gibbs sampling. This analysis allows direct verification of the effectiveness of the variational approximation in applications of interest, and to the best of the authors knowledge, it is the first time these quantities have been compared. In Section 4.2.2, the model selection methods are compared by calculating the posterior probability of a set of model classes.

This section uses a three-state auto-regressive hidden Markov model (AR-HMM) to compare model selection algorithms. The AR-HMM shown in Figure 4.1 is defined by equations (4.31) and parameters (4.32) and is a special case of the GLHMM (Definition 3.4), using an auto-regressive model (Definition 3.1) of order 2 to generate dynamics in each discrete mode. The cyclic nature of the following AR-HMM simulates the repetitive nature of state transitions in the neurological

applications of Chapter 5:

$$y_k = \mathcal{N}\left(\theta_{m_k}^T x_k, \sigma_{m_k}^2\right), \quad x_k = [y_{k-1}, y_{k-2}], \ \mathrm{p}\big(m_k = j | m_{k-1} = i\big) = a_{ij} \tag{4.31}$$

where

$$A = [a_{ij}] = \begin{bmatrix} 0.99 & 0.01 & 0.00 \\ 0.00 & 0.97 & 0.03 \\ 0.02 & 0.00 & 0.98 \end{bmatrix}, \qquad \theta_1 = \begin{bmatrix} 0.5 \\ -0.3 \end{bmatrix}, \qquad \theta_2 = \begin{bmatrix} 0.7 \\ -0.4 \end{bmatrix}, \qquad \theta_3 = \begin{bmatrix} 0.8 \\ 0.1 \end{bmatrix},$$

$$\sigma_1^2 = 0.02, \qquad \sigma_2^2 = 0.04, \qquad \sigma_3^2 = 0.005 \ . \tag{4.32}$$

A Dirichlet distribution models the prior information for each row of the transition kernel A:

$$\mathrm{Dir}\big([a_{i1}, a_{i2}, ..., a_{iN}] | [a_{i1}^0, a_{i2}^0, ..., a_{iN}^0]\big) \ , \tag{4.33}$$

where the specific prior parameters $[a_{ij}^0]$ depend on the number of discrete modes N:

$$a_{ij}^0 = \begin{cases} 50 & \text{if } j = i \\ 1 & \text{if } i < N \text{ and } j = i+1 \\ 1 & \text{if } i = N, j = 1 \\ 0.01 & \text{else} \end{cases}, \qquad \text{e.g. if } N = 3\text{:} \ \ A^0 = \begin{bmatrix} 50 & 1 & 0.01 \\ 0.01 & 50 & 1 \\ 1 & 0.01 & 50 \end{bmatrix} \ . \tag{4.34}$$

The prior parameters (4.34) are informative, and bias the model identification process to choose a cyclic HMM structure. The exact effect of these prior values is demonstrated in Section 4.3.

The prior information of the AR-dynamics is modeled by the Gaussian-Gamma distribution (specified in Def. 3.1) and is the same for each mode S_i

$$p(w_i, \tau_i) = \mathcal{N}\left(w_i | w^0, (\tau_i \lambda^0 I)^{-1}\right) \mathrm{Gam}\big(\tau_i | a^0, b^0\big) \ , \tag{4.35}$$

where the prior parameters are given by:

$$w^0 = \mathbf{0} \in \mathbb{R}^n, \quad \lambda^0 = 0.001, \quad a^0 = 5 \quad b^0 = 0.1 \ . \tag{4.36}$$

The parameters 4.36 are informative, and impose a constraint on the signal variance. Figure 4.2 shows the prior PDF on the precision τ_i and its reciprocal, the variance σ_i^2. The prior on the AR parameters w is minimally informative due to the parameter λ^0.

Remark 4.3. The model class for an AR-HMM with N discrete states and n^{th}-order AR dynamics,

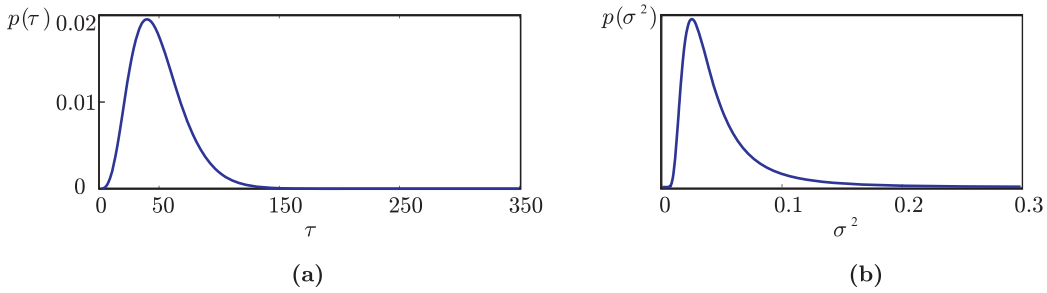

Figure 4.2: Prior distribution on system precision τ and system variance σ^2

and prior information (4.34) and (4.36) is denoted $\mathcal{M}_{(N,n)}$. ◇

4.2.1 Comparison of Information Theoretic Quantities

The decomposition of model evidence (4.6) into a model fit term and the KL divergence from the prior to the posterior provides a useful basis for algorithm comparison. This section is unique in the variational literature in that the KL divergence estimated by VB can be directly compared to the KL divergence from the prior to the posterior, as calculated by Gibbs sampling (4.28), and is useful for verifying the correctness of both approaches. This quantitative comparison gives insight into how "tight" the lower bound approximation is, and the effect of the factorization relaxation (2.22) used in latent variable models. In the AR-HMM considered here, the VB algorithm provides a very accurate lower bound, indicating that the factorization is a relatively small approximation in terms of practical effect.

Comparisons were conducted by simulating 500 data segments from the 3-state HMM (4.31), (4.32), with the length of each segment being varied between 100 and 10000 data points. 50 segments of each length:

$$T = 100, 166, 278, 464, 774, 1291, 2154, 3593, 5994, 10000 \tag{4.37}$$

were created. In this section, we are only considering computation of information theoretic quantities related to the model evidence, which inherently assumes convergence of learning algorithms. For a meaningful comparison, all algorithms used only the correct 3-state 2^{nd}-order AR-HMM model class ($\mathcal{M}_{(3,2)}$), and were initialized with the exact discrete-state sequence simulated for each data segment, guaranteeing convergence to the globally optimal solution. The next section will instead focus on choosing between several model classes. The EM and VB algorithms were run until the change in the incomplete data likelihood and lower bound, respectively, fell below a threshold of $1e-7$. The Gibbs sampler was used to draw 500 samples with no burn in period.

Results are averaged over all 50 data segments of length T. Figure 4.3 show the average model evidence $\mathrm{p}\big(y_{1:T}|\mathcal{M}_{(3,2)}\big)$ for each approximation. The KL divergence from the prior to the posterior as calculated by the Gibbs sampler (4.28) is also compared to the variational lower bound (VB) and

the KL divergence from the prior to the factorized posterior $q(\Theta)$.

The VB and Gibbs sampler based algorithms give remarkably similar estimation of all quantities. The lower bound of the VB method provides a good approximation to the more computationally intensive Gibbs sampler in this AR-HMM case. The AIC and BIC model evidence approximations are also shown in Figure 4.3 (b). Note that as AIC (4.12) is defined as the incomplete data likelihood at the MAP value $\mathrm{p}\big(y_{1:T}|\Theta_{MAP}, \mathcal{M}_{(3,2)}\big)$ minus the number of parameters $M = N(N-1) + Nn$, it appears as a constant. AIC and BIC do not provide a method to estimate the information gain (KL divergence) and are hence not shown in Fig. 4.3 (c). As seen in Fig. 4.3, the AIC and BIC methods provide poor approximations to the model evidence. The next section will give a fairer comparison by directly comparing the model selection performance of each approximation.

It should be noted that the similarity of information theoretic quantities between the Gibbs sampler and the variational method should only occur when there is a single peak in the posterior PDF. If the Gibbs sampler explores multiple peaks that are significantly disjoint (i.e. is unidentifiable), and is compared to the variational method with it's inherent local properties, then there will be an discrepancy in the information gain between methods. In this thesis it was found that with the amounts of data used and the relatively short sampling runs of the Gibbs sampler, there is usually only one posterior peak explored by the sampler. Note however, that in the case of mixture models, or HMM based models, there will always be multiple peaks in the posterior PDF due to the so called "label switching problem", where the posterior is unidentifiable because of the multiple ways of labeling the modes. In this section, the label switching problem is ignored, due to the local nature of the implemented sampling schemes. In the future, if better samplers or longer runs are conducted, then one of the many existing solutions to the label switching problem should be implemented [100, 101].

4.2.2 Model Selection of 3-State AR-HMM

This section directly compares the model selection performance of the model evidence approximations: AIC, BIC, GIBBS, and VB. Two data sets of length $T = 1000$ and $T = 5000$ were generated from the 3-state second-order AR-HMM model (Fig. 4.1 and equations (4.31) and (4.32)).

The following procedure was implemented 50 times for each of the 15 model classes $\mathcal{M}_{(N,n)}$, $N = 2, 3, 4$, $n = 1, 2, 3, 4, 5$ for both data sets $T = 1000$ and $T = 5000$:

- A random discrete state sequence $m_{1:T}^{(0)}$ of length T was generated from the uniform distribution: $\mathrm{p}\big(m_k = i\big) = \frac{1}{N}$, for $k = 1 : T$.
- The EM algorithm was initialized with AR-parameters maximizing $\mathrm{p}\big(w_i, \tau_i|m_{1:T}^{(0)}\big)$ and a transition kernel A maximizing the prior distribution (4.33). The EM algorithm was run until the MAP estimate: $\log \mathrm{p}\big(y_{1:K}|\Theta_{MAP}, \mathcal{M}_{(N,n)}\big) + \log \mathrm{p}\big(\Theta_{MAP}|\mathcal{M}_{(N,n)}\big)$ changed less then $1e-7$

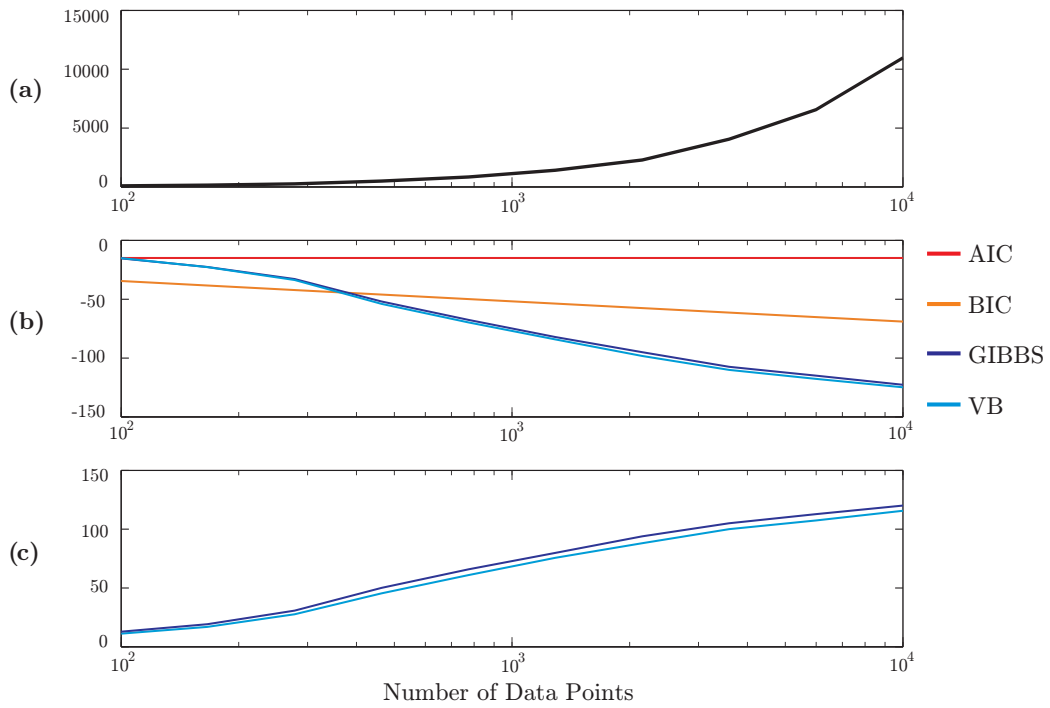

Figure 4.3: Comparison of information theoretic quantities related to the model evidence calculation: 50 data segments of length T were generated, for $T = [100, ...10000]$. All plotted values show the averaged quantities over the 50 trials. (a) The average incomplete data likelihood at the MAP value $\mathrm{p}\big(y_{1:T}|\Theta_{MAP}, \mathcal{M}_{(3,2)}\big)$ given by the EM algorithm. (b) The model evidence minus the MAP data likelihood $\mathrm{p}\big(y_{1:T}|\mathcal{M}_{(3,2)}\big)$-$\mathrm{p}\big(y_{1:T}|\Theta_{MAP}, \mathcal{M}_{(3,2)}\big)$ for the AIC, BIC, Gibbs sampler (GIBBS) and variational lower bound (VB) estimates. (d) The KL divergence from the prior to posterior parameter densities from both the variational estimate and the Gibbs sampler. The results demonstrate that VB approximation provides a very accurate lower bound to the model evidence in this AR-HMM example, as verified by the Gibbs sampler. In addition, the information gain (complexity penalization term) from the Gibbs sampler and VB are practically identical. This gives strong evidence that assumptions made in the VB algorithm are reasonable for this model class.

Table 4.1: Model selection results: The posterior probability for the model classes $\mathcal{M}_{(N,n)}$, where N is the number of discrete modes, and n is the autoregressive order, is tabulated. The results are stated as percentages (posterior probability times 100). The "true" model used to generate the data sequences was a model of class $\mathcal{M}_{(3,2)}$. Model selection results for two data sequences of $T = 1000$ and $T = 5000$ samples are shown. The variational and Gibbs sampler estimates produce the most accurate results for both data segments.

1000 data points										
	(N, n)									
	(2,2)	(2,3)	(3,2)	(3,3)	(3,4)	(3,5)	(4,2)	(4,3)	(4,4)	(4,5)
BIC	**75.64**	0.06	24.28	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AIC	0.00	0.00	**87.16**	10.58	0.67	0.02	1.43	0.09	0.01	0.00
VB	0.00	0.00	**99.78**	0.02	0.00	0.00	0.22	0.00	0.00	0.00
GIBBS	0.00	0.00	**99.52**	0.02	0.00	0.00	0.45	0.01	0.00	0.00
5000 data points										
	(N,n)									
	(2,2)	(2,3)	(3,2)	(3,3)	(3,4)	(3,5)	(4,2)	(4,3)	(4,4)	(4,5)
BIC	1.00	0.01	**98.98**	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AIC	0.00	0.00	16.32	**40.88**	1.16	0.32	2.35	32.92	0.68	5.37
VB	0.00	0.00	**99.86**	0.04	0.00	0.00	0.10	0.00	0.00	0.00
GIBBS	0.00	0.00	**99.77**	0.03	0.00	0.00	0.20	0.00	0.00	0.00

between iterations.

- The VB algorithm was initialized using the distribution $q(w_i, \tau_i) = \mathrm{p}\big(w_i, \tau_i | m^{(0)}_{1:T}\big)$ and $q(A)$ was set to the prior distribution on the A matrix (4.33). VB was run until the change in the lower bound between iterations was less then $1e-7$.

- The Gibbs algorithm was initialized with AR-parameters simulated from $\mathrm{p}\big(w_i, \tau_i | m^{(0)}_{1:T}\big)$ and an A matrix sampled from (4.33). The Gibbs sampler was run for a 500 iteration burn in period[5], after which the sampler was run for a subsequent 500 samples. For the Stationarity method for model evidence calculations, the maximum a posteriori parameter from the burn-in period was used as $\Theta^\star$ in (4.18).

- AIC and BIC quantities were calculated using the MAP likelihood: $\log \mathrm{p}\big(y_{1:K} | \Theta_{MAP}, \mathcal{M}_{(N,n)}\big)$ from the EM algorithm. The model evidence was calculated using the last 500 samples of the Gibbs sampler.

The maximum model evidence approximation generated by each method from the 50 runs is used for calculating the posterior probability of the model classes $\mathcal{M}_{(N,n)}$ for $N = 2, 3, 4$, $n = 1, 2, 3, 4, 5$. The posterior probability of the model classes with non-zero probability are shown in Table 4.1, for AIC, BIC, the Gibbs sampler (GIBBS), and VB.

[5] 500 samples were found on several test runs to give an adequate burn-in period where convergence was tested visually.

Table 4.1 shows the Gibbs sampler and variational Bayesian methods produce consistent and accurate model selection criteria for the simulated AR-HMM, outperforming the AIC and BIC information criterions by a significant margin. Here the AIC criterion is biased towards more complicated models, and the BIC criterion towards simpler models.

The variational method was found to be the fastest algorithm for identification: the slight increase in computational burden for each iteration was offset by a decrease in the number of iterations to converge compared to the EM algorithm. Gibbs sampling took considerably longer to converge due to the increased number of iterations required[6].

4.3 Automatic Model Structure Determination Priors

A efficient tool used in Bayesian inference are priors that allow automatic determination of model structure. We define these priors as ASD priors, for *automatic structure determination*[7]. ASD priors incorporate a bias towards reducing the number of model parameters or simplifying the model structure, and applied inference algorithms can then automatically prune out unnecessary model elements. These priors have previously been used for determining the number of components in a mixture model [24] and the number of modes in a (discrete) hidden Markov model [36]. There are, however, two potential downfalls to using ASD priors to automatically select the number of modes or components of a model: there is no direct incorporation of prior knowledge about the number of modes, and the method can over-prune the model structure.

This thesis advocates a combination of discrete model class selection (discussed in Section 4.2.2) used in combination with ASD priors to avoid over-pruning of model structure, allow addition of prior knowledge about the number of discrete states, yet retain the efficiency of automatic structure determination. The remainder of this section is structured as follows: First a specific Dirichlet ASD prior is defined, and the form of the distribution is analyzed. Second, an example that highlights the effect of a ASD priors used with GLHMM is presented. Third, ASD priors that are appropriate for HSMM are defined. The following Section 4.4 will present a case study of ASD priors used with HSMMs.

[6] A computation time comparison is not given, as the Gibbs algorithm was optimized and written in C, where as the other methods used a combination of Matlab and C. In practice each iteration of the Gibbs sampler, VB, and EM algorithms are similar, as essentially the same computations are required, so it is simply the number of iterations required to converge that determines the computational speed of the algorithm.

[7] This work can also be viewed as a variation of automatic relevance determination (ARD), where model structure is pruned by maximizing the model evidence with respect to hyperparameters of the prior distributions, also called type II maximum likelihood [68]. The ASD priors in this thesis instead inherently prune model structure. The evolution from ARD to ASD priors is demonstrated in the publication history of Bishop, where originally [68] ARD was used for selection of the number of components in a mixture model, with a subsequent text advocating ASD priors instead [24]. Note that Bishop does not use the term *ASD*, which is defined in this thesis for convenience, to form a distinction between priors that allow automatic pruning of model structure and those that do not. MacKay [40] provides a comparison of using ARD, or instead doing a full Bayesian analysis and integrating over all parameters. Bishop and Tipping also write a chapter in [102] called "Bayesian Regression and Classification" which shows that integrating out hyperparmeters effectively results in an ASD prior.

The automatic structure determination used in this thesis is based around the Dirichlet distribution (A.15), for modeling the prior information of the Markov transition kernel A. By choosing suitable prior parameters $a_{ij}^0 < 1$, the prior biases the inference procedure to give zero weight to transition probabilities between $S_i \rightarrow S_j$ [39, 36]. In a departure from previous methods, we advocate the use of non-uniform priors, which are especially useful in identifying cyclic structure of transition sequences, especially in cases such as the neurological applications in Chapter 5, where each discrete mode has some physical meaning connected to it. By giving each self-transition probability a_{ii} a larger prior probability $a_{ii}^0 > 1$, the inference algorithms will no longer prune away discrete states.

This approach was used in the last section (as seen in the prior parameters of A (4.34)) and shows model class selection successfully applied to cyclic AR-HMMs. This section takes a more detailed look at the specific effect of the choice of prior distributions used in this thesis. Figure 4.4 shows a Dirichlet prior (for a single row of A) with parameters $a_i^0 < 1$ for a 3-state model, compared to a prior with $a_i^0 > 1$.

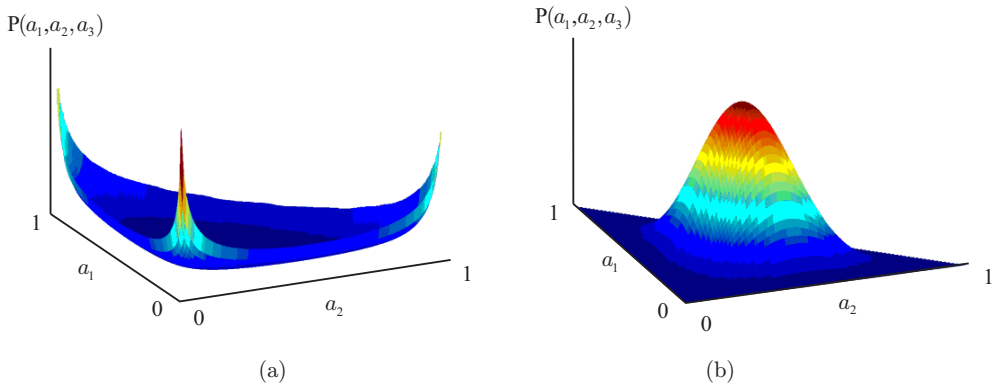

Figure 4.4: 3-state Dirichlet distribution used to model prior knowledge about a row of the transition kernel $A_{i:} = [a_1, a_2, a_3]$. Notice that only axes a_1 and a_2 are shown, as a_3 is determined by the sum-to-one constraint: $a_3 = 1 - a_1 - a_2$. (a) Prior used for automatic structure determination: $a^0 = [0.5, 0.5, 0.5]$, where the probability mass is concentrated in having only one or two connections with a low probability of all three transition probabilities being active. (b) A more typical prior with $a^0 = [5, 5, 5]$, which enforces all transition probabilities to be non-zero.

An illustrative example of the power of ASD is now presented. We define a 5-state AR-HMM (with second-order AR dynamics), with transition kernel A, shown in (4.38):

$$A = \begin{bmatrix} 0.96 & 0.04 & 0 & 0 & 0 \\ 0 & 0.960 & 0.04 & 0 & 0 \\ 0 & 0 & 0.95 & 0.05 & 0 \\ 0 & 0 & 0.025 & 0.95 & 0.025 \\ 0.04 & 0 & 0 & 0 & 0.960 \end{bmatrix}. \tag{4.38}$$

A simulated data sequence of length $T = 2500$ is generated from the 5-state AR-HMM, and identified using the VB method. Figure 4.5 shows how the inference algorithm automatically prunes out allowed transitions of the model as the number of iterations of the VB algorithm increases. The resulting identified model had states matched to the simulated model by comparing identified parameter values associated with each mode.

The parameters a_{ij}^0 of the Dirichlet prior distribution for the transition probabilities a_{ij} is given by:

$$[a_{ij}^0] = \begin{bmatrix} 5.0 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 5.0 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 5.0 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & 5.0 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.01 & 5.0 \end{bmatrix} . \tag{4.39}$$

The purpose of choosing the above prior distribution is twofold: the large self transition prior probabilities $a_{ii}^0 > 1$ enforce that all 5 states are active in the identified model, and also increase the posterior probability that self transitions are more likely. In the neural application of Chapter 5, there is considerable prior knowledge about the duration spent in each discrete state, and by choosing an appropriate prior distribution the posterior models will be biased towards this behavior. Furthermore, in the case of the neural data, we know there are few transitions between most states; a cyclic transition pattern is expected. Shown in Fig. 4.5 the number of non-zero state transitions is limited by the choice of $a_{ij}^0 < 1$. If the prior distribution instead used $a_{ij}^0 \geq 1$, then all posterior transitions a_{ij} would be non-zero.

Remark 4.4 ([36]). There is an excellent interpretation for the *strength*, or amount of information contained, in the specified Dirichlet prior distribution (4.39). We have previously used a Dirichlet distribution to represent each row $a_{i:}$, of the transition matrix:

$$\mathrm{p}\big(a_{i:}\big) = \mathrm{Dir}\big(a_{i1}, ..., a_{iN} | [a_{i1}^0, ..., a_{iN}^0]\big) \ . \tag{4.40}$$

Using this conjugate prior (4.40), it was shown in (2.79) that the posterior distribution is also a Dirichlet distribution:

$$\mathrm{p}\big(a_{i:} | y_1 : T\big) = \mathrm{Dir}\big(a_{i1}, ..., a_{iN} | [a_{i1}^T, ..., a_{iN}^T]\big) \tag{4.41}$$

where $a_{ij}^T = a_{ij}^0 + \sum_{k=1}^T \xi_k(i,j)$, and $\sum_{k=1}^T \xi_k(i,j)$ is the expected number of transitions from S_i to S_j. The prior parameter a_{ij}^0 in (4.41) can then be thought of as the number of observed transitions in an additional imaginary data set (i.e., each a_{ij}^0 essentially represents an additional number of transitions between S_i and S_j). The total strength of the prior can then be defined as the number of observations in the imaginary data set, which is simply the sum $\sum_{j=1}^N a_{ij}^0$. For instance, the sum of

the rows in (4.39) is 5.04, so this prior distribution can be thought of as equivalent to an imaginary data set with approximately 5 transitions. Because the example in Fig. 4.5 uses 2500 data points, the prior (4.39) is dominated by observed data in the posterior distribution. ◇

Remark 4.4 has severe implications for the use of Dirchlet priors in HSMM models. Because the maximum duration D is potentially large, the prior distribution $p_i^0(1:D)$ (see Def. 3.7) needs to be carefully chosen: If the maximum duration is $D = 500$, and the prior is chosen such that $p_i^0(d) = 1$, then by Remark 4.4, this is equivalent to observing 500 transitions into mode S_i. Even in large data sets, the number of transitions between modes in a HSMM may be reasonably small, and hence the prior will dominate in the posterior distribution. To avoid this problem, this thesis advocates the use of a *fixed strength prior*, where the prior is chosen such that $\sum_{d=1}^{D} p_i^0(d) = c$, where c is a constant representing the strength of the prior.

A convenient method of creating priors for HSMM durations is to utilize a Gamma distribution (A.10). The prior for the duration of each mode is then generated by:

$$p_i^0(d) = c\ d^{\alpha-1} \frac{\beta^\alpha \exp(-\beta d)}{\Gamma(\alpha)} \tag{4.42}$$

where (4.42) is a Gamma distribution, and the user can choose appropriate parameters for α, β, and c. Equation (4.42) is convenient as the Gamma distribution has mean $\frac{\alpha}{\beta}$ and variance $\frac{\alpha}{\beta^2}$, giving an intuitive method to insert prior knowledge into the identification process. Note that the value c, the strength of the prior, may need to be carefully chosen. If the parameter c is made very small, then all prior values $p_i^0(d)$ will be close to zero, resulting in the identification algorithm pruning away the majority of durations. If the value of c is large, then the prior parameters will dominate the posterior distribution. For the applications in this thesis, the strength of the prior is varied between $5 \leq c \leq 25$ depending on the amount of observed data.

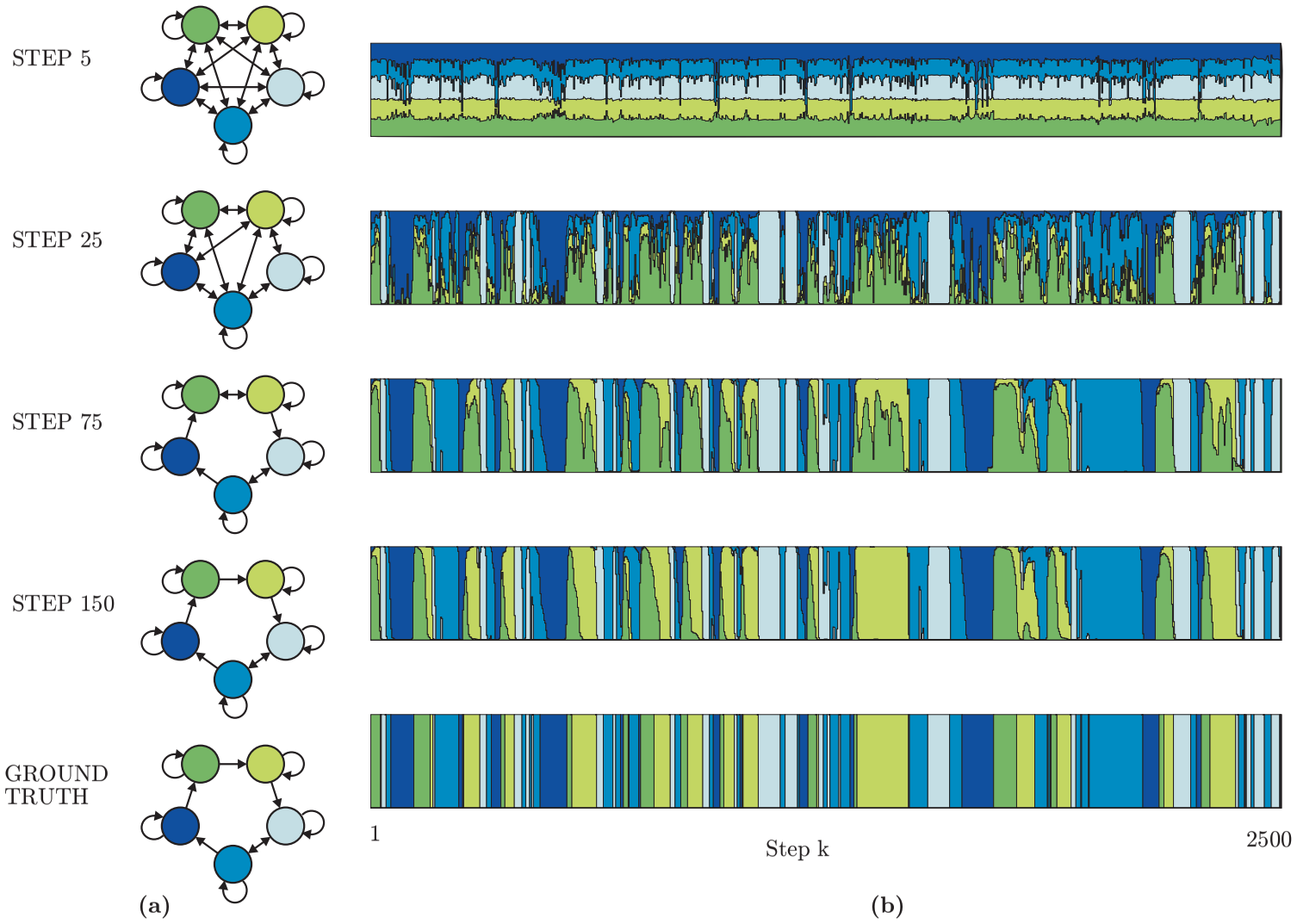

Figure 4.5: Automatic structure determination: Variational Bayes is used to infer the model parmeters. VB was run for 150 iterations, the results of which are plotted. (b) The marginal probability of the discrete state $\mathrm{p}(m_k|y_{1:2500})$ is plotted as approximated by $q(m_{1:2500})$, for several steps of the VB algorithm. (a) For each step of the VB algorithm, the transition probabilities a_{ij} between discrete states that are numerically different in the variational approximation $q(A)$ from the prior distribution are indicated with an arrow (the direction of the arrow implies into which state the system can transition). As the number of VB iterations increases, the probability of transitions $q(A)$ approaches the ground truth (4.38) .

4.4 Case Study: Oh Bee Dance Data Set

This section applies variational model selection using ASD priors and HSMM with GLM dynamics to the *dancing bee* data set [43][8]. This is the first time variational methods or ASD priors have ever been applied to a HSMM.

The goal of [43] is similar to this thesis: A maximum likelihood EM approach is taken to identify *parameterized segmented switching linear dynamical system* (PS-SLDS). In the terminology of this thesis, the PS-SLDS is equivalent to a Markov jump system (Section 2.1.1) with a semi-Markov transition kernel (Def. 3.5). The analysis in this section differs from [43] in two respects: First, [43] initialize their model using a *data-driven* technique, and have a pre-determined number of discrete modes. This method forms a prior guess about the state sequence $m_{1:T}$, by running cleverly designed pre-filters on the bees heading angle data to give a initial segmentation of discrete states[9]. This accurate initialization allows the learning process to proceed more efficiently, avoids local minima, and provides good results. This section will instead forgo the data-driven filter in [43] and use the dancing bee data to demonstrate automatic structure determination in a Bayesian HSMM with GLM dynamics using the variational algorithm. Here, the identification process will be initialized with very broad priors, without using an initial segmentation of the discrete modes, and without a fixed number of discrete states. The second difference between [43] and the model used here, is we choose to use GLM dynamics, as opposed to state space models. This section also demonstrates how the GLM can be used to represent fully observed state space models.

In brief, the data set consists of a bee worker executing a series of maneuvers called a *dance*, which communicates the location of pollen or food to other bees in the hive [44]. For this analysis, three data segments (trials) of bee dance data were used (see Fig. 4.4). The dance is conducted in a planer hive surface, allowing the bee's state to be modeled by three variables, the *x-position*, the *y-position* and the *heading angle* ϕ:

$$y_k = \begin{bmatrix} y_k^1 \\ y_k^2 \\ y_k^3 \end{bmatrix} \triangleq \begin{bmatrix} x - \text{position} \\ y - \text{position} \\ \phi - \text{heading angle} \end{bmatrix} \tag{4.43}$$

The bees motion is recorded by video camera, and the x, y, ϕ position of the bee is extracted from the video stream by a series of pre-processing steps [43]. The bees motion during the dance is typically segmented by hand into three [44, 43] distinct motion primitives or patterns: *turning left*, *turning right* and *waggling*, where the bee moves forward but rapidly oscillates its heading angle. These three canonical states are illustrated in Fig. 4.4 (g). The goal of this case study is to automatically

[8] This data is freely available online at http://www.cc.gatech.edu/∼dellaert/. The author greatly thanks Oh, Rehg and Dellart for providing this resource to the community.

[9] The exact procedure for creating the data-driven initial guess of the model is presented in the tech-report [103]

detect the number of discrete modes, segment the data into the correct mode sequence and identify representative motion model for each motion primitive.

The dynamics of the bee in each discrete state (or motion primitive) will be represented by a "stack" of GLMs:

$$y_k = \begin{bmatrix} \theta_i^1(1) & 0 & \theta_i^1(2) & 0 & 0 & 0 \\ 0 & \theta_i^2(1) & 0 & \theta_i^2(2) & 0 & 0 \\ 0 & 0 & 0 & 0 & \theta_i^3(1) & \theta_i^3(2) \end{bmatrix} \begin{bmatrix} y_{k-1}^1 \\ y_{k-1}^2 \\ \sin(y_{k-1}^3) \\ \cos(y_{k-1}^3) \\ y_{k-1}^3 \\ 1 \end{bmatrix} + \varepsilon_k \tag{4.44}$$

where the error ε_k is zero-mean and normally distributed:

$$\varepsilon_k \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \theta_i^1(3) & 0 & 0 \\ 0 & \theta_i^2(3) & 0 \\ 0 & 0 & \theta_i^3(3) \end{bmatrix} . \right) \tag{4.45}$$

Note that this model (4.44) is equivalent to a nonlinear state space model with a fully observable state. If the restricted form of covariance matrix (4.45) is not adequate, then a full multi-variate AR-model could be used instead of (4.44).

The priors on the system parameters were designed to be minimally informative. The hyper-parameters of the dynamics model were not found to be important, and a wide range of minimally informative parameters could be chosen for the Gaussian-Gamma distributions. The priors for the transition matrix and the duration models are explicitly stated here: ASD prior were used for the transition matrix A, resulting in automatic deamination of the number of discrete states in the data sequences:

$$a_{ij}^0 = 0.1 \text{ for } i \neq j \ , \tag{4.46a}$$

and where a_{ij}^0 is not defined due to the structure of a HSMM. The prior for the duration of each mode is generated by a gamma distribution:

$$p_i^0(d) = c\ d^{\alpha-1} \frac{\beta^\alpha \exp(-\beta d)}{\Gamma(\alpha)} \tag{4.46b}$$

with parameters $\alpha = 10$, $\beta = 5$, and where $c = 5$ is a constant that is equivalent to number of observations that prior represents (see Remark 4.4 in the previous section). The maximum allowed duration was $D = 250$.

The model was initialized by using an initial distribution over the joint state $m_{1:T}, \tau_{1:T}$, generated

by adding random white noise to a uniform distribution, and then re-normalized.

$$\mathrm{p}\big(m_k = i, \tau_k = d\big) \propto \frac{1}{ND} + 0.05r \ , \tag{4.47}$$

where r is a uniform random number, N is the number of discrete states, and D is the maximum allowed duration.

The identification process proceeds as follows: the model is initialized with five discrete states (N=5) using (4.47) to generate a state sequence over each dance trial. As the identification process proceeds, states are pruned away due to the bias of the ASD priors. The VB algorithm is run until convergence (when the change in the lower bound is less then $1e-7$). This procedure was repeated 5 times, and the model with the maximum posterior probability was chosen, and is depicted in Figure 4.4. Note that the method was robust to changes in the prior information, choosing the optimal 3-state model when using the range $0.01 < a_{ij}^0 < 0.5$. It was found that if the prior term a_{ij}^0 was greater then 0.5, then the optimal model would typically have more discrete modes.

The combination of the VB algorithm, HSMM model, and ASD priors provided excellent results that were consistent with data segmented by an expert user. Not only did the algorithm find the correct number of modes, but the identified mode sequences were very similar to the expert's ground truth. Furthermore, the identification process was initialized without the use of an expert or pre-filter, and only required the specification of minimally informative priors.

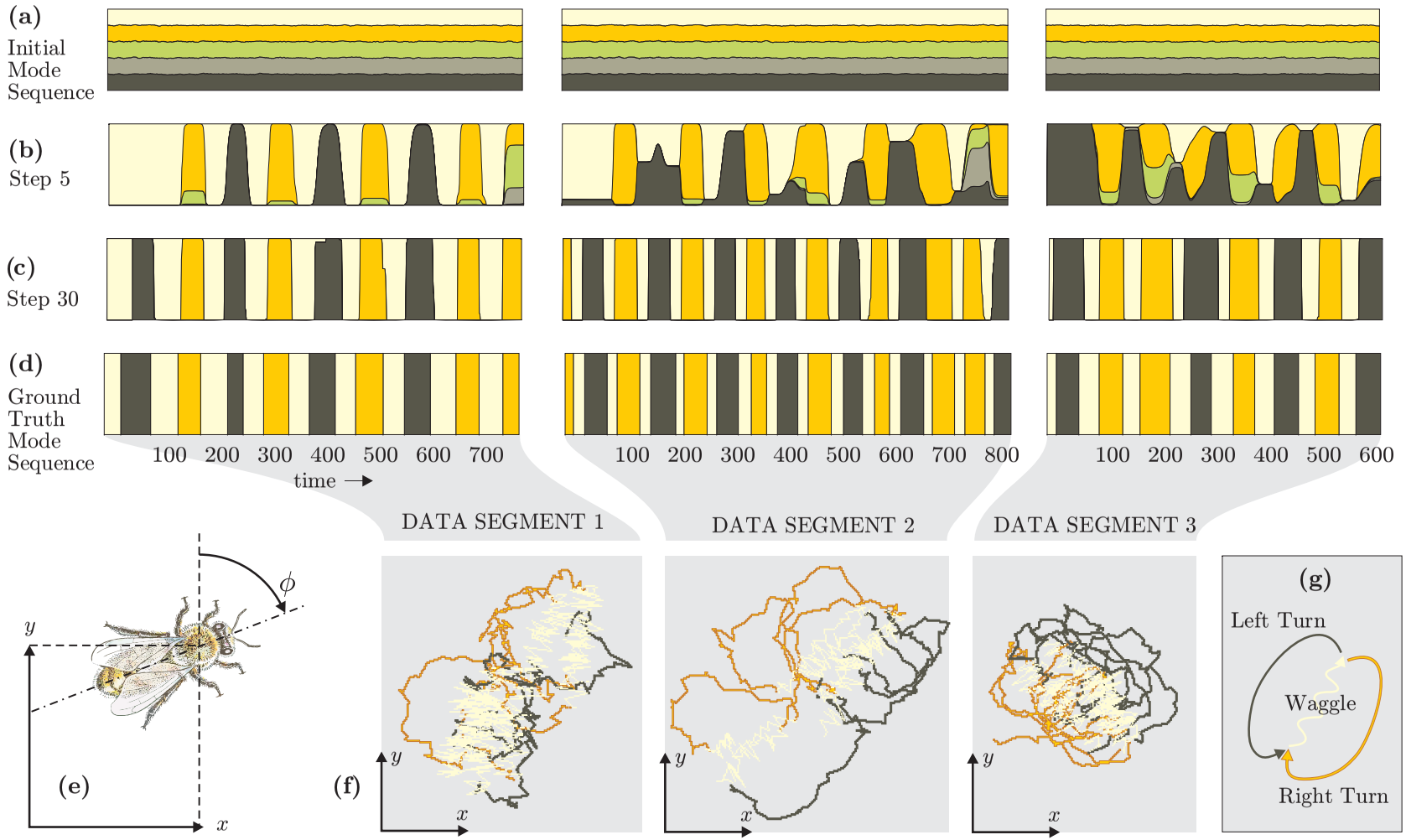

Figure 4.6: Identification of bee dance: Variational Bayes is used to identify a HSMM model with the ASD priors defined in (4.46). (a) The VB algorithm was initialized with a random state sequence (4.47) containing five modes. (b) The state sequence $q(m_1 : T)$ of the VB algorithm at iteration five. (c) VB step 30: only three discrete states remain. (d) The ground truth labels of the bee state sequence as determined by a human expert: the video data of the bee was hand segmented into *left turn*, *right turn*, and *waggle* motion primitives. (g) Legend. (f) The $x - y$ state sequence for each data segment, color coded by the expert's segmentation. *See text for details*

Chapter 5

Neural Prosthetics Application

Cortical neuroprostheses are being developed to restore motor function in individuals with high-level spinal cord injuries or severe motor disorders (e.g., Lou Gehrig's disease). Neuroprostheses work by recording the activity of multiple neurons in cortex and decoding movement intent or movement plans from this neural activity in order to generate control signals that can be used to drive devices such as prosthetic arms or computer interfaces [11, 12, 13]. Future practical clinical neuroprostheses will require a *supervisory decoder* (Fig. 1.1) whose job is to classify, in real time, the *discrete* cognitive or behavioral *state* of the brain region from which the neural signals are recorded. For example, the supervisory decoder must determine: (1) if the prosthetic patient is awake or conscious; (2) if the patient wants to use the prosthetic; (3) if the brain is currently planning a movement that should be decoded by the prosthetic; (4) if or when the movement is to be executed; (5) if the patient wants to change or scrub a plan while it is being executed; etc. The knowledge of the current state in the evolution of the planning process can be used in a variety of ways. For example, depending upon the current state, different algorithms, or different parameters in the algorithm, can be applied to the decoding of movement plans.

This chapter models the neural processes related to the brain-machine interface as a hybrid dynamical system, where the discrete states are associated to the cognitive or planning brain states, and the continuous states model the observed neural activity, such as firing rate. Thus, the design of a supervisory decoder is a two part process: (1) the identification (or learning) of the hybrid model that represents neural activity in each discrete cognitive state as well as the transition rules between cognitive states; (2) the design of an estimator which uses the identified hybrid model to classify the current neural activity into discrete cognitive or planning states.

There are a number of reasons we use the framework of hybrid systems theory to formulate our approach to the design of supervisory decoding systems. First, the supervisory decoding problem is naturally formulated in this framework. Second, the process of learning a supervisory decoding model requires both the identification of the parameters of the supervisory decoding model and the simultaneous classification of neural activity into discrete modes. These distinct computational

processes are easily handled in a hybrid system identification framework presented in Chapter 3. Furthermore, the model selection methods presented in Chapter 4 allow the identification of models when the prior knowledge about the discrete cognitive process is incomplete. Third, our formalization of the problem in hybrid systems terms allows for scaling of our method to reasonably complex hybrid supervisory decoding systems. Fourth, if neuroprostheses are to become widely used in clinical applications, a formal and automated approach to their design is necessary so that the process of adapting a prosthetic to each patient is not so labor intensive. Fifth, the hybrid system framework easily incorporates many dynamical models typically used in neural decoding, and allows for the fusion of disjoint types of signals, such as action potentials, local field potentials, and eye trackers.

The idea of using discrete state, or supervisory, decoders in neural prosthetic systems is not original to this thesis. It dates at least to the work of Shenoy et al. [13], who developed, using an ad hoc approach, a finite state machine model and decoder that classified plan activity from the parietal reach region into three discrete states; a baseline state, a plan state, and a reach state. Using off-line analysis, they showed that the imposition of a supervisory decoder on the decoding process could improve overall system accuracy. Recently, Kemere et.al. [104] have demonstrated, using signals from dorsal premotor cortex, the decoding of two different discrete states. Their work assumed a homogeneous Poisson rate model for neural firing, and used an expectation-maximization framework to find the model parameters. While their work is not directly related to the subject of this chapter, it should be noted that Wu et al. [105] have used a switching Kalman filter, a type of hybrid system, for decoding continuous arm movements. Also a recent paper by Srinivasan et. al. [17] has proposed the idea of using hybrid systems to model neural activity. Srinivasan et.al. do not propose a method to identify these systems from data, but do discuss a particle filtering approach to the estimation (state inference) problem.

This chapter will focus on identifying supervisory decoders based on neural recordings from the parietal cortex of a macaque monkey while the animal carries out tasks that simulate the operation of a neural prosthetic. Previous work [13, 15, 16] has demonstrated that the parietal reach region (PRR) in the posterior parietal cortex contains both motor planning activity as well as neural correlates of the discrete cognitive and planning states needed for a supervisory controller. The work of Snyder et.al. [13] suggests that the parietal reach region (PRR) may be well suited for generating signals useful for decoding the discrete cognitive state in prosthetics applications, as it encodes plan activity selective for arm movements, which is not dependent upon actual movement occurring. The developed supervisory decoders will combine multiple neural signal types including both local field potentials (LFP), and single unit (SUA) activity (see Section 5.1 for details on these neural signals). It is demonstrated that improved performance is achieved by utilizing more than one signal type, and has the additional benefit of potentially lengthening the use of the implanted prosthetic device.

To demonstrate the effectiveness of our proposed approach, several case studies are conducted, where a supervisory decoder is first identified from neural data, and then used to estimate key cognitive discrete states:

Case Study 1: Simulated Data Set (Section 5.2) A simulated data set consisting of a single neuron with a nonstationary firing rate is developed. The activity is modeled as a single output GLHMM, and is identified using a Gibbs sampler. Not only does the Gibbs sampler accurately identify the simulated data set, but the posterior samples are analyzed to infer the identifiability of the model.

Case Study 2: Scherberger Data Set [15] (Section 5.3) A neural data set collected by H. Scherberger [15] is analyzed, that consists of both SUA and LFP signals collected from the parietal cortex of a macaque monkey. A GLHMM model is identified from the neural data, which fuses the activity from both signal modalities, using Gibbs sampling. In this example, the number of discrete cognitive states is assumed to be known, and is determined by the experimental paradigm (see Fig. 5.3). The process by which the supervisory decoder was identified did not use any knowledge of the experimental cues or markers. The subsequent decoding results using the identified model show that the inference framework developed in Chapter 3 can be successfully applied to neural data: Key cognitive states related to prosthetic movement can be decoded with accuracy of up to 97.9%, even when the supervisory decoder is trained on a small data set using little prior information. The data used in this case study has a high signal-to-noise ratio, as each recording electrode was individually moved to a cortical location providing optimal signal quality.

Case Study 3: Musallam Data Set [16] (Section 5.4) A neural data set collected from the parietal cortex of a macaque monkey by S. Musallam [16] is analyzed. The Musallam data set consists of SUA and LFP signals recorded with a 64-electrode array. A subset of the electrodes, those with a high signal-to-noise ratio, were used in the identification and estimation process. Initially a GLHMM was identified where the number of discrete cognitive states was assumed to be known, however this resulted in a comparatively poor decode performance of 46.5%. The decode performance was achieved using the variational method, however similar results were achieved using the Gibbs sampler, and even when the training data set was pre-segmented using experimental markers and cues or expert opinion. The model selection tools developed in Chapter 4 were applied to identify the number of discrete cognitive states in the GLHMM, which resulted in a model with extra discrete states. This new supervisory decoder model was identified using variational methods, and achieved a decoding performance of 85.4%. The identified GLHMM was found to contain several new states during a *memory* period, and enter a subset of the new discrete cognitive state for only brief temporal periods, perhaps accounting

for either artifacts in the neural signals, or distraction of the monkey during the experiment. To model the duration of cognitive states explicitly, the HSMM with GLM dynamics (Def. 3.5 in Chapter 3) was used instead of the GLHMM for creating a supervisory decoder model. This new model, even when the minimum dwell time in each cognitive state is constrained, retained many of the characteristics of the identified GLHMM. The identified HSMM supervisory decoding model achieved a higher decode performance of 91.67%. These identified models effectively incorporated prior information about the experimental paradigm, and are suited to future prosthetic development work. Even when training on small data sets, with limited knowledge about the discrete temporal periods, the model identification and model selection steps can be quickly computed.

The neural case studies and theory development in Chapters 3 and 4 of this thesis contain several key contributions to the neural prosthetics community in regards to the creation of discrete state decoders and supervisory decoders:

- A new hybrid-system identification framework for supervisory decoders is proposed and developed. The utilization of more then one neural signal type when decoding, and the ability to completely automate the approach (without the pre-segmentation of some neural data), are improvements over existing methodologies.

- The proposed identification algorithms incorporate prior knowledge about the neural system into the model and constrain neural dynamics to physiological limits. The Bayesian perspective utilized in this thesis allows explicit incorporation of prior knowledge pertaining to dynamics such as neuron firing rates, and the allowable transitions and durations of discrete cognitive states.

- Developed algorithms automatically determine the transition rules between discrete cognitive states. Prior work either focuses on hand tuning a finite state machine [13] to constrain the duration spent in each cognitive state, or utilizes Markov-based switching [104] models, that do not prune away subsets of allowed transitions. Here automatic identification of both Markov and semi-Markov transition rules are considered, allowing the automatic creation of duration based transition logic.

- Automatic model selection procedures are developed which allow for the addition (or removal) of discrete cognitive and planning states. Instead of assuming that the subjects cognitive states are defined exactly by external events, the number and transitions between discrete cognitive states are automatically determined. This work may lead to increased performance in traditional prosthetic decoding work, such as determining intended reach direction, as neural temporal periods of interest can be identified in a more refined way.

5.1 Neurological Signal Models

The front end of a cortical neural prosthetic typically consists of a multi-electrode array implanted in cortical tissues [11, 12, 13, 16]. The signal recorded from each electrode contains multiple signal components that arise from different physiological origins and whose characteristics require different signal models, which are now briefly reviewed. Based on the conducted case studies, two neural signal types are considered here: local field potentials (LFP) and single unit activity (SUA). These signal classes are only a subset of all potential neural signals available for use with prosthetics [106].

5.1.1 Local Field Potentials

Cortical local field potentials (LFP) arise from the aggregate dendritic electric potentials originating from neurons in a "listening sphere" that surrounds the electrically active tip of the recording electrode [15]. Such signals average the dendritic activity of a few thousand nearby neurons. In practice, the LFP signal component is derived by amplification and band-pass filtering (usually in the range of 2–300 Hz) of the electrode signal. Historically, the LFP is modeled from the knowledge of its spectrogram [15, 107], which is optimally obtained from multitaper methods [108] which apply the Fourier transform to tapered time series obtained from the digitization of the LFP signal. Spectrograms of the LFP signal in the parietal cortex show that temporal variations of the power in certain frequency bands is correlated with intended arm reach direction, as well as changes in planning state [15, 107]. The average power in each frequency band can be modeled as a random variable with a log normal distribution.

Autoregressive (AR) (see Def. 3.1) or vector autoregressive (VAR) equations can be used for parametric spectral estimation [108], and will be used here to model the LFP signal in the time domain. AR modeling of signals is typical in electroencephalogram (EEG) recordings [109, 110, 99], and occasionally time domain representations of LFP signals are considered in the implanted electrode studies [106]. A p^{th}-order AR model, denoted AR(p), takes the form:

$$y_k = \sum_{i=1}^{p} \beta(i) y_{k-i} + \eta_k \ , \tag{5.1}$$

where $y_k \in \mathbb{R}$ is the LFP signal sampled at time t_k, and $\eta_k \sim \mathcal{N}(0, \sigma^2)$ is zero mean noise with covariance σ^2, and the model parameters are $\theta = \left\{\beta(1), ..., \beta(p), \sigma^2\right\}$. Note that the spectral density of a stationary AR(p) process (5.1) is given by [108]:

$$S(f) = \frac{\sigma^2 \Delta t}{\left|1 - \sum_{j=1}^{p} \beta(j) \exp^{-i2\pi f j \Delta t}\right|^2} \ , \tag{5.2}$$

where f is the frequency and Δt is the sampling period. While the spectrogram has been the

primary LFP modeling tool in prior work, there are two main advantages of using time domain AR models instead of frequency domain spectrogram methods. First, the real-time computation of the spectrogram is an excessive practical burden, and it additionally introduces a time lag in the response of the neural prosthetic system since, a large window size (typically 512 or 1024 msec) is needed to obtain good precision. This lag may cause undesirable psychophysical delays for the prosthetic-using patient. The AR approach effectively uses considerably smaller window sizes: the largest data window used in this thesis is from a 55^{th}-order AR model sampled at 1 kHz, resulting in an effective window width of 55 msec.

5.1.2 Single Unit Activity

Neurons generate characteristic electrical pulses called action potentials, or spikes, whose arrival times, and not waveform shape, are believed to encode information. Mathematical models used to decode neural stimuli typically focus on the firing (spiking) rate of individual neurons [111]. Numerous studies have shown that *single unit activity*[1] can be correlated to intended reach direction, as well as temporal or cognitive state in the posterior parietal cortex [15, 107]. Following standard practice, the spike arrival times are discretized into sufficiently small time bins (1 msec in our experiments) so that only one spike at most is assigned to each bin. Let the beginnings of each discretized sampling interval be denoted by the sequence of times $\{t_1, t_2, \ldots, t_k, \ldots, t_T\}$. Thus, each bin corresponds to the time interval $(t_k, t_{k+1}]$. The signal y_k is the number of spikes arriving in the interval $(t_k, t_{k+1}]$. When the bin size is sufficiently small, the spike arrival times can be modeled as a point process with a stationary Poisson distribution (see Def. 3.2):

$$f(y_k, \lambda) = \frac{\lambda^{y_k} e^{-\lambda}}{y_k!} \ , \tag{5.3}$$

where λ is the firing rate of the neuron, and is the only parameter of the model ($\theta = \lambda$). An extension of this process (5.3) can be used, the nonstationary Poisson process (see Def. 3.3) model can also be used to represent single spiking unit activity [78]. In this similar model, the nonstationary firing rate is a log linear function of the neuron's spiking history:

$$\lambda_k = \exp\left[\beta(0) + \sum_{i=1}^{p} \beta(i) y_{k-i}\right] \ . \tag{5.4}$$

As discussed in the definition of a GLHMM (Def. 3.4), the linear regressor (5.4) can also contain other system variables of interest. The next section will utilize this nonstationary model (5.4), but in

[1]The action potentials, or spikes, of more than one neuron may be recorded on a single electrode. A two-step process isolates the activity of a single neuron, or unit. First, spike waveforms are detected (in the midst of substantial background noise) in the electrical signal. The detection process also provides an estimate of the spike waveform's *arrival time*, the time at which the spike amplitude peaks. A *spike sorting* process [16] then analyzes the waveform shapes, and clusters the waveforms according to different putative neural signal sources.

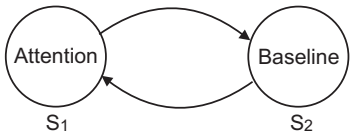

Figure 5.1: Finite state machine representation of simulated neuron behavior

subsequent case studies, the stationary firing rate model (5.3) was found to give equal performance due to the low firing rates of neurons in the analyzed data sets.

5.2 Case Study 1: Simulated Single Neuron Recording

To illustrate some key characteristics of our approach, a simulation of recorded spiking activity from a single neuron present in a higher brain cortex is created. This neuron's spiking activity is dependent on the unobservable discrete state of the surrounding cortex. For this simple example, the cortex has two discrete states: S_1, an "attention" state (i.e., the patient wants to actively use the neural prosthetic) and S_2, a "baseline" state (i.e., sleep or disinterest in using the neural prosthetic). The transition of this cortical region between the attention and baseline states is assumed to follow Markov transition probabilities. The number of spikes in successive 0.01 s time bins, for a 10 s interval is simulated.

The discrete modes are modeled by setting $m_1 = 1$ and evolving the discrete state m_k, $k = 1, \ldots, 1000$, using Markov transitions with parameters $A = [a_{i,j}]$:

$$P\left(m_{k+1} = j | m_k = i\right) = a_{ij}, \text{ where } A = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix} . \tag{5.5}$$

The neurons spiking activity in each mode is modeled with Poisson-GLMs (Def. 3.2). The firing rate, λ_k, in each mode S_i, is determined by two components: $\theta_i(1)$, the nominal firing rate of the mode, and $\theta_i(2)$, representing a change in rate depending on the spiking history. $\theta_i(2)$ can model refractory periods, a dwell period in spiking activity that is experienced immediately after spike firing:

$$\lambda_k = \begin{cases} e^{(\theta_1(1)+\theta_1(2)y_{k-1})} \text{ if } m_k = 1 \\ e^{(\theta_2(1)+\theta_2(2)y_{k-1})} \text{ if } m_k = 2 \end{cases} . \tag{5.6}$$

The following regressor parameters are used:

$$\theta_1 = \begin{bmatrix} -1 & -10 \end{bmatrix}^T, \quad \theta_2 = \begin{bmatrix} -2 & 0 \end{bmatrix}^T . \tag{5.7}$$

The parameters (5.7), correspond to a nominal firing rate of 36.78 Hz in the "attention" state, and a nominal rate of 13.53 Hz in the "baseline" state.

Table 5.1: Model parameter estimates. Expected value (E[·]) and the maximum a posteri (MAP) estimates are used, compared with actual parameter values (Model).

	Model	MAP	E[·]
A	$\begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$	$\begin{bmatrix} 0.898 & 0.102 \\ 0.102 & 0.898 \end{bmatrix}$	$\begin{bmatrix} 0.897 & 0.102 \\ 0.096 & 0.904 \end{bmatrix}$
θ_1	$\begin{bmatrix} -1 & -10 \end{bmatrix}^T$	$\begin{bmatrix} -0.964 & -2.704 \end{bmatrix}^T$	$\begin{bmatrix} -1.003 & -25.01 \end{bmatrix}^T$
θ_2	$\begin{bmatrix} -2 & 0 \end{bmatrix}^T$	$\begin{bmatrix} -2.013 & 0.201 \end{bmatrix}^T$	$\begin{bmatrix} -2.108 & -0.021 \end{bmatrix}^T$

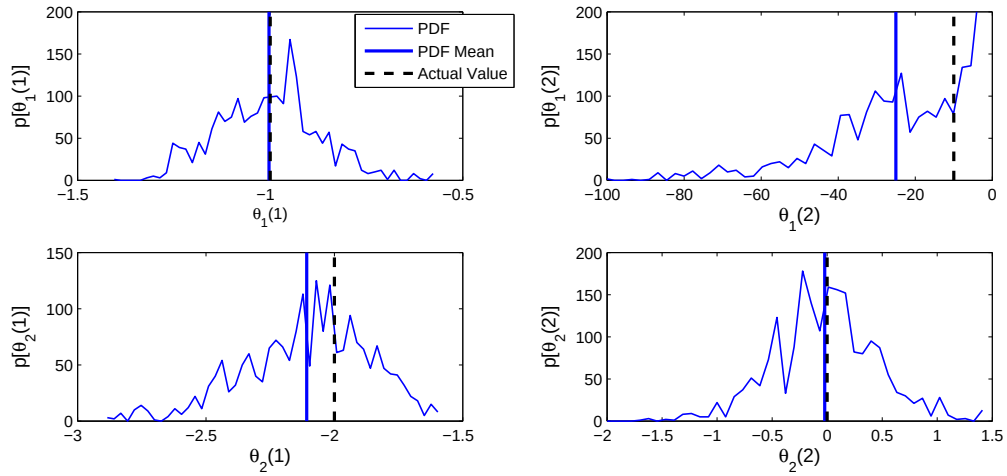

Figure 5.2: Regressor parameter posterior densities and mean estimates

The number of spikes in the current time bin are generated from a Poisson distribution with rate λ_k:

$$y_k \sim \text{Poisson}\left(\lambda_k\right) \quad . \tag{5.8}$$

An output sequence $y_k, k = 1, \ldots, 1000$ was generated from the single neuron model by using Poisson and discrete random number generators in Matlab. There were a total of 211 spike events over the simulated 10 second duration.

The multi-stage Gibbs sampling algorithm for GLHMM (see Algorithm 3.1 in Section 3.4) was run, setting $z_{max} = 5000$; the last 3000 generated samples were used for statistical analysis. Regressor parameter priors are set to dispersed normal distributions: $\theta_{i,j} = \mathcal{N}\left(0, 10^2\right)$ for $i \in \{1, 2\}$, $j \in \{1, 2\}$. Dirichlet priors are used for each row of A: $\begin{bmatrix} a_{11} & a_{12} \end{bmatrix} \sim \mathcal{D}\left(\begin{bmatrix} \alpha_1 & \alpha_2 \end{bmatrix}\right)$, $\begin{bmatrix} a_{21} & a_{22} \end{bmatrix} \sim \mathcal{D}\left(\begin{bmatrix} \alpha_2 & \alpha_1 \end{bmatrix}\right)$. Several different informative parameterizations were chosen that incorporate the assumption that sequential modes values m_k, m_{k+1} are more likely to belong to the same mode S_i:

$$\begin{bmatrix} \alpha_1 & \alpha_2 \end{bmatrix} = \begin{bmatrix} 90 & 10 \end{bmatrix}, \begin{bmatrix} 80 & 20 \end{bmatrix}, \begin{bmatrix} 70 & 30 \end{bmatrix} \quad . \tag{5.9}$$

The solution was invariant when using different informative priors (5.9), and the key parameter estimates matched the model values (see Table 5.1).

The only wide discrepancy between the MAP and expectation estimates is for the refractory parameter $\theta_1(2)$. Gibbs sampling allows analysis of the posterior densities, by constructing a histogram of the samples. The posterior density for $\theta_1(2)$, shown in Fig. 5.2, has a large support, indicating that the parameter is unidentifiable from the generated data set. This posterior distribution remains bounded, because of the proper prior distribution used by the algorithm. This unidentifiablity problem arises because the refractory physics of spike firing dictate that no sequential outputs y_k and y_{k+1} in S_2 both contain spikes. Hence the only information that can be deduced from the posterior distribution is the refractory parameter $\theta_1(2)$ significantly lowers the firing rate after a spike event has just occurred. The posterior densities thus allow the user to realize when a parameter is unidentifiable, or nearly unidentifiable.

5.3 Case Study 2: Scherberger Data Set

The two-stage Gibbs sampler (Alg. 3.1) is applied to a neural data set obtained from experiments with rhesus monkeys [15] to identify a GLHMM supervisory decoder model. This data set consists of recordings from two male rhesus (*Macaca mulatta*) monkeys, *Animal C* and *Animal D*, from electrodes placed in various positions within the parietal reach region (PRR) of the posterior parietal cortex. The neural data set contains both LFP and neural spike arrival time signals [15]. While we have analyzed data from both animals, for brevity the results presented below focus on the 96 electrode recordings from animal *D*.

The data recordings occurred while the monkeys repetitively executed a *delayed center-out reaching task*, which is commonly used to simulate the actions of a neural prosthetic. Such simulations are a necessary step in the development of this technology for eventual human use. This task is illustrated in Fig. 5.3. A task-board is placed within arm reaching distance in front of the monkey's visual field of view. Each trial proceeds as follows: A light located in the center of the task board is illuminated, and the monkey must place its reaching arm on the light to indicate that it is attending to the trial. A *target* light is flashed at one of 8 *target locations* around the task board perimeter for a short *cue period*, and then the target light is extinguished. After a random time delay (the *memory period*, during which the monkey must remember the target location and also plan its upcoming reach to the target), the center light is extinguished, cueing the monkey to reach to the remembered target location. After the monkey reaches for a target location, the original target is redisplayed. If the monkey has successfully reached for the correct target location, while also respecting the temporal structure of the sequence, it is given a juice reward.

To simulate the action of a neural prosthetic, the neural signals from the PRR are "decoded" during the memory period (when the monkey can only be planning a reach, and not executing a physical reach), to predict the monkey's subsequent physical reach, even before the reach occurs.

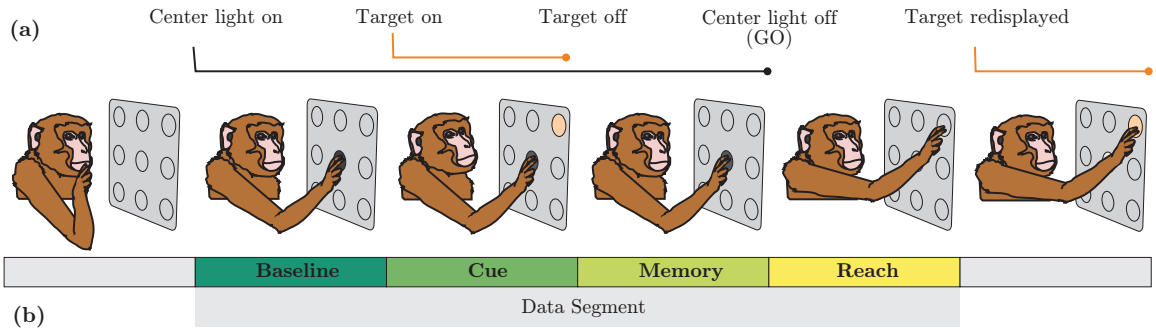

Figure 5.3: Center-out reach experiment: a) displayed experimental cues, b) corresponding cognitive or behaviorial states.

Successful prediction of the subsequent reach from the memory period signals validates the ability to decode a reaching plan from PRR neural activity. Practically, these experiments demonstrate, for example, the ability of the brain machine interface to control cursor movements on a computer screen using neural signals. Such cursor control is a basic function that would allow paralyzed patients to use a computer. In more advanced experiments that more accurately simulate a neural prosthetic, the monkey is taught to purely think about the reach to the target, and the desired cursor command is decoded from this thought [16].

We can also use these trials to simulate and validate a supervisory decoding system. The trial structure has an associated discrete number of different cognitive and planning states: (1) a *baseline state* where the monkey is idle, or starting to attend to the upcoming trial; (2) a *cue period* during which the target location is lit; (3) a *memory period* during which the location of the now extinguished target must be remembered by the monkey, and during which the monkey plans its upcoming arm movement; (4) a short "go" period (which is really a transition between memory and execution states) during which the planned movement is initiated; and (5) a *reach* or *execute* period during which the arm moves to the target location. To successfully simulate a supervisory decoder, we seek to demonstrate that the onset and duration of these different planning/cognitive periods can be correctly estimated solely from the neural signals recorded during the trial. The actual reaching behavior of the monkey is actively recorded during the task execution, providing us with a reasonably good ground truth model against which the predictions can be compared. We are particularly interested in estimating the onset of the reach state (the "go" signal). In an neural prosthetic, this signal will trigger the execution of an action associated with the decoded planning activity.

Training (identification) and testing (estimation) data sets were created by randomly choosing an n electrode subset, $\mathcal{E}_n$, from the set of available electrode signals, $\mathcal{E}$. We limited our selection to the subsets of the recorded data which included at least 7 successful reaches in each of 8 possible reach directions, and whose signal-to-noise quality exceeded a threshold [15]. From the data set $\mathcal{E}_n$, two reach trials in each of the 8 directions were randomly chosen (16 trials total) to form the training

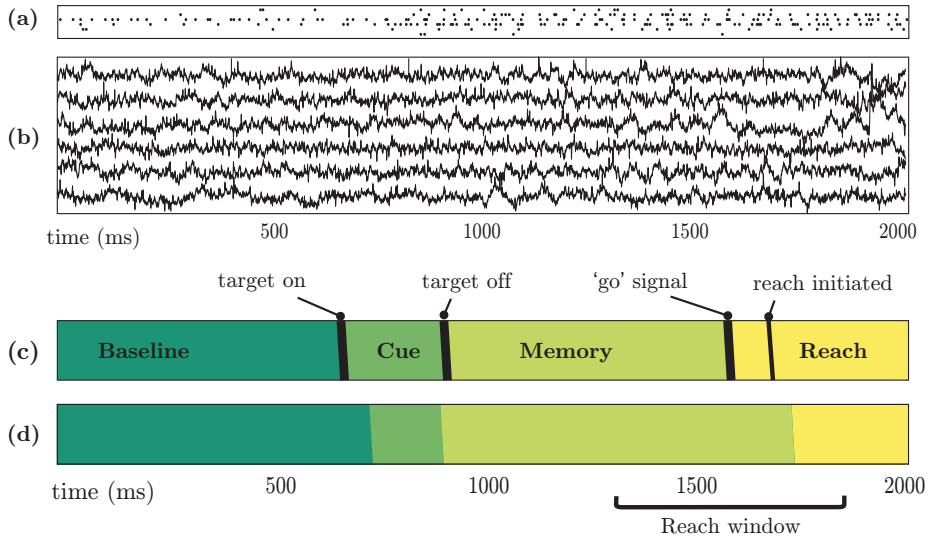

Figure 5.4: Example decode: a) Recorded neural spike arrival times for 6 electrodes; b) Local field potentials for 6 electrodes; c) Cognitive states as defined by experimental cues; d) Decoded cognitive state.

set. A testing data set was formed from the remaining 5 trials in each of the 8 directions (40 trials total). This is done 7 times for the given value of n, where n was varied from 1 to 6, resulting in 42 data sets, each containing 16 training trials and 40 testing trials.

For each of the 42 training data sets, a GLHMM was identified using the two-stage Gibbs sampler (Alg. 3.1), with prior distributions and initial conditions specified in Section 5.3.1. Estimation of the discrete cognitive state in the corresponding testing data sets is then done using the Viterbi algorithm (Def. 3.14). For each trial, a discrete state estimation, or decode, was considered *correct* when the reach state was decoded within a 300 ms window of the "go" signal, as shown in Fig. 5.4. The exact timing of the go signal is obtained experimentally by watching for the onset of the monkey's arm motion. The average percentage of correct decoding trials versus the number of electrodes n is shown in Fig. 5.5. The error bars represent standard deviations of percent correct over the seven repetitions described above.

Figure 5.5 shows that a high level of decoding performance can be achieved using a relatively small number of electrodes. This is a promising result, as the surgical complexity and risk associated with the implantation of the electrodes is proportional to the number of electrodes.

In addition to the high percent correct of decodes, the lag between the estimated onset of the reach state and the actual reach is small, 0.027s on average. From the psychophysical point of view, this is a negligible lag.

To relate the identified models back to the science conducted in [15], we can calculate the power spectral density (PSD) of the identified models, and show that we recover similar phenomenon of changing power in different frequency bands through time in LFP signals. The PSD of the AR

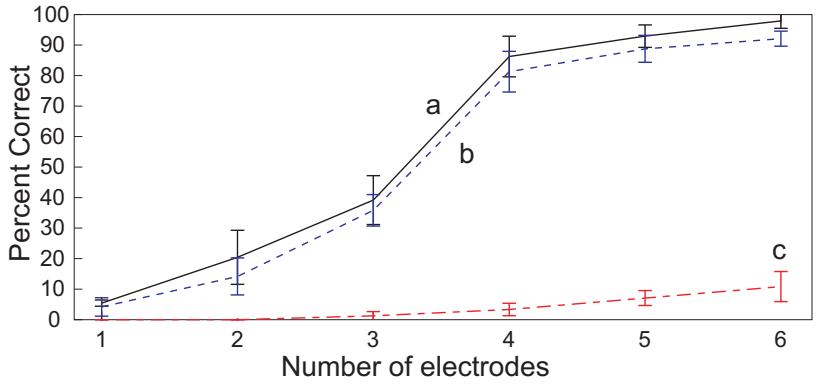

Figure 5.5: Decoding results: Percentage of correctly decoded trials, where the reach state was correctly estimated within a finite time window of the actual reach occurring; a) Decoding using both LFP and single unit activity; b) Decoding using only LFP signals; c) Decoding using only single unit activity.

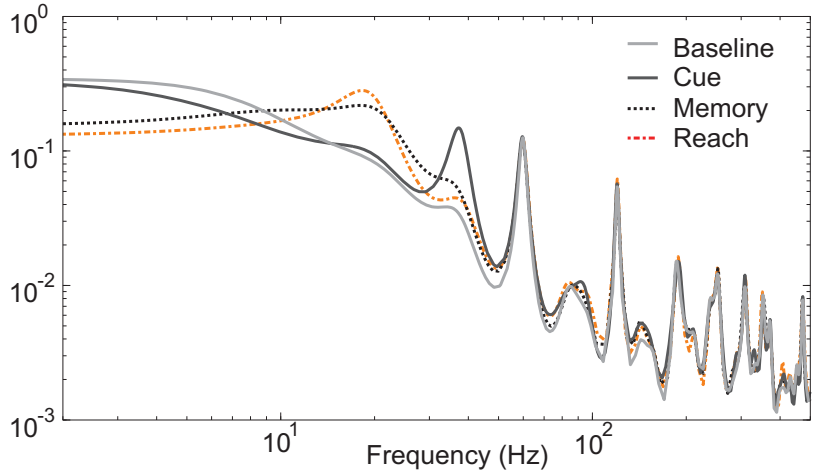

Figure 5.6: Power spectrum of the identified AR models for each discrete cognitive state for a single electrode.

models for a single electrode in each discrete state is shown in Fig. 5.6. The AR models in each discrete state show the characteristic noise peak at 60 Hz and 120 Hz, however the large discrepancy at lower frequencies is consistent with the ranges considered in other studies using spectrographic methods.

5.3.1 Prior Distributions and Initial Conditions Used for GLHMM Identification

This section presents the prior distributions and initialization procedure used in identifying the GLHMM from the Scherberger data set. The most important prior distribution specified for the GLHMM is the prior on the Markovian transition matrix A. Using the ASD prior discussed in Chapter 4, allows the the GLHMM to be biased towards identifying a left-to-right HMM. Specifically

the following prior distribution was used:

$$[a_{ij}^0] = \begin{bmatrix} 50 & 1 & 0.001 & 0.001 \\ 0.001 & 50 & 1 & 0.001 \\ 0.001 & 0.001 & 50 & 1 \\ 0.001 & 1 & 0.001 & 50 \end{bmatrix} . \tag{5.10}$$

The identified model was invariant to many of the chosen prior parameters, however it was necessary to use small (¡0.01) values for appropriate entries in (5.10) to keep the identified model's left-to-right structure. If a uniform prior distribution was used instead of (5.10), the identified model will switch quickly between discrete modes, and does not contain any relevant information.

The prior distributions for the GLM dynamics in each mode were found to be invariant to any reasonable choice of prior parameters. The large amount of neural data overwhelms any prior information in the posterior distribution. The use of prior distribution for neural dynamics still provides a useful function: If at any time during the identification process one or more discrete modes are not assigned any data, the posterior of the distribution remains proper. Furthermore it was essential to use prior distributions when modeling firing rates in the Scherberger data set. This data set is characterized by low firing rates of SUA on each electrode. It was found to be common that long periods occurred where single neurons did not fire for an extended period. If the firing behavior of these neurons was modeled using a maximum likelihood approach, the most likely firing rate parameter is zero. Due to the use of Poisson likelihood function in the GLHMM, this results in a local minima, where the model then has zero probability of that neuron ever emitting a spike. The use of proper prior distributions avoids these local minima.

The prior distribution for SUA signals was modeled as a gamma distribution in the case of a stationary Poisson process (see Def. 3.2) with parameters $a^0 = 2$ and $b^0 = 20$. This results in a mean firing rate of 40 Hz with a wide support that tapers off at 100 Hz. Model selection was found to be invariant to a large range of prior values. In the case of a nonstationary point process model (Def. 3.3) the prior distribution was defined as a zero mean Gaussian distribution with a diagonal covariance matrix $\sigma^2 I$, with $\sigma^2 = 1000$.

Gaussian-Gamma distributions are used to represent prior AR-likelihood LFP neural signals. The precision parameter (inverse of the covariance) is modeled with a Gamma distribution with parameters $a^0 = 15$ and $b^0 = 1$. This approximately corresponds to a prior on the variance with mean of 0.07, and is appropriate based on the range of the recorded LFP signals. The AR model parameters are represented with a minimally informative zero-mean distribution with $w^0 = \mathbf{0}$ and $\Lambda^0 = 0.1\mathrm{I}$.

The model is initialized by drawing a sample of the model parameters. This sample was generated for the transition matrix parameters by taking a random sample from the Dirichelt distribution

defined by parameters (5.10). The initialization of the neural dynamics parameters was achieved by sampling from a posterior distribution, where all neural data in the training set is used to form the posterior distribution. To insure convergence of the Gibbs sampler, the evolution of the parameter samples was visually monitored.

5.4 Case Study 3: Musallam Data Set

The variational Bayesian algorithm is applied to a rhesus monkey data set, [16], and used to identify both GLHMM and HSMM based supervisory decoder models. The data set consists of recordings from three male rhesus monkeys, *Monkey S*, *Monkey C*, and *Monkey O*, from a 64 electrode array implanted in the medial intraparietal area (MIP), a component of the PRR and a 32 electrode array implanted in area 5. The neural data set contains both LFP and neural spike arrival time signals. Due to larger number of collected trials, and higher performance [16], the results in this section focus on the 64 electrode array recordings from the PRR of monkey S.

The neural activity of the monkey was recorded during repetitive completion of the center-out reach task previously described in Section 5.3 and depicted in Figure 5.3. A large number of electrodes in the 64 electrode array had poor signal to noise ratios, or did not contain relevant information. Based on the performance results of Section 5.3, six LFP recordings and 12 SUA signals whose signal-to-noise ratio passed a threshold were chosen to form the presented data set. All supervisory decode models in this section were trained on the first 5 trials in each of 4 reach directions (20 training trials), and then decode performance was evaluated on the subsequent 36 reach trials in each direction (144 testing trials).

Both GLHMM and HSMM models were identified from the Musallam data set. Decoding for GLHMMs was conducted with both the forward filter (Def. 3.11), and the non-causal Viterbi algorithm (Def. 3.14). Decoding for HSMM was conducted using a fixed lag smoother (Def. 3.13) and the Viterbi algorithm. Decode performance is defined using the unique detection of the "go" signal, or onset of the reach period, as defined in Section 5.3 and depicted in Figure 5.4.

Initial results, identifying the 4-state GLHMM supervisory decoder, provided comparatively poor results. Using the variational algorithm to identify the GLHMM, only 46.53% of testing trials were correctly decoded with the Viterbi algorithm. Similar performance was achieved when the model was identified using the Gibbs sampler, or when a "hand built" model was created by pre-segmenting the training data set using experimental cues.

With extensive testing, the lower performance in the Musallam data set compared with the Scherberger data set (Section 5.3) was tentatively attributed to confusion between neural activity in subsets of the memory period, and a later segment of the reach period. This typically causes false positives, where an extra "go" signal is decoded when using the forward filter, or may cause the

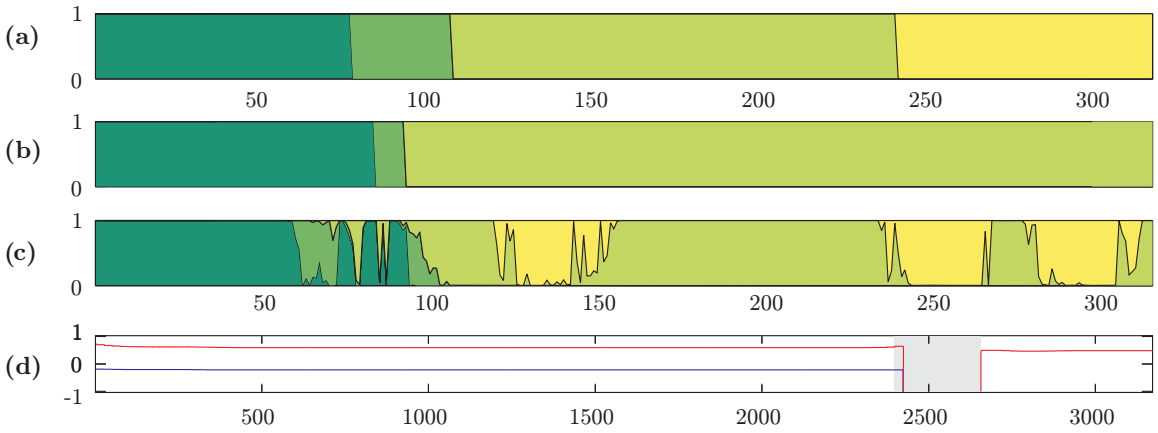

Figure 5.7: Testing trial 100/144 using identified 4-state GLHMM: (a) The discrete cognitive states as defined by the experimental cues; (b) The Viterbi algorithm is used to decode the discrete state sequence; (c) The forward filter is used to decode the discrete state sequence; (d) Recorded arm movement using a touch screen – the removal of the monkeys arm from the screen causes saturation of the recorded x-y arm position. The total arm movement period is highlighted by a grey section.

Viterbi algorithm to miss detection of the actual "go" signal completely (see Fig. 5.7). Furthermore there appear to be artifacts, or residual activity, during the baseline period, but due to the nature of neural recordings and lack of explanatory evidence, no conclusions about the baseline period of activity can be proposed.

The complete lack of a ground truth has severe implications for deriving a better supervisory decoder. While several external cues used in the center-out reach experiment can be used to propose a likely sequence of discrete cognitive states, it is impossible to confirm the accuracy of this imposed structure. Inevitably there will be many more complicated processes present in the brain cortices of interest; the only pertinent question is what effect will these unmodeled events have on the decoding of periods of interest. Instead of proposing and cross validating possible sets of discrete cognitive states, this section will use the Bayesian model class section methods derived in Chapter 4 to automatically choose the most suitable model. This methodology has the advantage of providing a rigorous model selection framework that is computationally efficient[2], allowing for effective use with prosthetic patients. In addition small training data sets can be utilized for model creation, as no training trials are required to be "thrown away" for use in a validation set. These properties of the Bayesian model class selection methodology are crucial for effective implementation in human prosthetic patients, where there are potentially a large number of discrete states of interest, and potentially a short amount of time from data collection to controller implementation.

The next two sections investigate the use of model selection for automatic creation of supervisory decoders: Section 5.4.1 applies model selection to GLHMM models and Section 5.4.2 applies model selection to HSMM based supervisory decoder models. In brief it was found that using model

[2]Bayesian model class selection is more efficient compared to using cross validation, where the identification process needs to be rerun many times on a "leave one out" data set for each covariant of interest.

Table 5.2: Supervisory decoder performance using GLHMM and HSMM models: The decode performance for the 4-state GLHMM model based on experimental cues is compared with the most-likely 8-state GLHMM and 7-state HSMM identified using Bayesian model class selection. Both the Viterbi algorithm and the forward filter are used to estimate (decode) the discrete states on the 144-trial testing data set. The forward filter, as a causal algorithm, can produce false positives where the onset of the reach state is decoded outside of actual arm movement window (see Fig. 5.4 for details). The percentage of correctly decoded trials (% Correct) refers to when the "go" signal decoded within 300 ms of the arm movement onset and no false positives occur. The percentage of trials where go signal (% Go Signal) is decoded within 300 ms of arm movement, whether or not there are false positives, is also tabulated. Because of the noncausal nature of the Viterbi algorithm there is only ever one "go" signal decoded, and hence no false positives.

	Forward Filter		Viterbi
	% Correct	% Go Signal	% Correct
4-state GLHMM	34.03	80.56	46.53
8-state GLHMM	81.25	99.31	85.42
7-state HSMM	87.5 (0)	96.53 (100)	91.67

selection, as opposed to specifying a model directly from experimental cues, drastically increased decoding performance. Table 5.2 summarizes these results.

5.4.1 GLHMM with Model Selection

To improve the performance of the supervisory decoder, the number of discrete cognitive states is now treated as uncertain, and hence required to be identified from the neural data set. This model selection process uses the methodology presented in Chapter 4: The number of discrete modes in the GLHMM are chosen using Bayesian model class selection, and the connectivity, or allowed transitions between discrete states, are determined by using ASD priors.

The identification of the number of modes in the supervisory decoder is not the only problem in model inference: The key to effectively using a supervisory decoder is knowing what each discrete mode represents. If a model with 10 modes is identified, the important task of understanding what each mode of the model represents in the actual neural system needs to be addressed. If we are interested in decoding the "go" signal, then somehow the model identification process needs to determine which discrete modes correspond to this neural process. To deal with this *association* problem, the set of potential model classes are defined by a *submode-insertion* algorithm. This submode-insertion algorithm is based on the premise that the left-to-right nature of the repetitive center-out reach task should be preserved.

The submode-insertion algorithm simply proposes a set of model classes by creating submodes, or substates associated with the original *baseline*, *cue*, *memory*, *reach* cognitive modes defined by the experiment. We do not impose any transition structure between submodes, but create prior distributions that bias towards a left-to-right evolution between submode groups. This submode concept is illustrated in Figure 5.8, where the left-to-right nature of the total model structure

Table 5.3: List of posterior probability of model classes $\mathcal{M}_c$. The optimal model class found using Bayesian model class selection is denoted $\mathcal{M}_c^\star$. Each model class is denoted by a vector denoting the number of substates associated with each experimental period (see text for details). The model defined by the experimental cues is then defined $\mathcal{M}_1 = [1111]$. The optimal model class found using model selection is $\mathcal{M}_c = [1232]$. The second most likely model class is $\mathcal{M}_c = [1132]$. The difference between the log posterior probability of the model classes $\mathrm{P}\big(\mathcal{M}_c|y_{1:T}\big)$ is known as the log odds.

$\mathcal{M}_c$	$\log \mathrm{P}\big(\mathcal{M}_c\big) - \log \mathrm{P}\big(\mathcal{M}_c^\star\big)$
[1111]	$-6.772e2$
[1132]	$-1.604e2$
[1232]	0

is depicted. In addition to preserving prior knowledge about the model structure, the submode approach gives intuition into each new discrete state's association; for instance, if a substate was generated from the original *reach* mode, then it is likely to be associated with intended movement onset.

A set of model classes were generated by inserting submodes into each original state. We use the vector $\mathcal{M}_1 = [1, 1, 1, 1]$ to denote the original left-to-right 4-mode GLHMM model class (see Fig. 5.8 (d)). The addition of a submode into the memory period is denoted $\mathcal{M}_1 = [1, 1, 2, 1]$. The optimal 8-state GLHMM model found using Bayesian model class selection is denoted: $\mathcal{M}_1 = [1, 2, 3, 2]$. This 8-state model has only one baseline state, two cue sub-states, three memory period substates and two reach substates (see Table (5.3) for the log odds of each model class).

For completeness, the original method to propose a set of model classes $\mathcal{M} = \{\mathcal{M}_1, ..., \mathcal{M}_G\}$, simply took all combinations of the vector $\mathcal{M}_c = [n_1, n_2, n_3, n_4]$, where n_i is varied between 1 to 4. This results in a very large number (256) of proposed model classes, however, using the efficient VB algorithm, this is a computationally achievable number of models to identify. This method was used to identify the optimal 8-state GLHMM. However, this method is not appropriate for general use, as even extending the number of possible substates to 5 instead of 4 results in a large increase of potential models. Furthermore it is quickly apparent that the addition of substates into some modes produces very low probability model classes. For general use, a *suboptimal substate-insertion algorithm* is suggested, where a single substate is added to the most likely model class that has been identified. By repetitively adding substates to only the most likely model, and updating the most likely model class after each new model is identified, only a subset of potential model classes need to be explored. By biasing the insertion algorithm to add substates in logical positions, the 8-state GLHMM was identified as the most likely model class and only required calculation of 27 model classes. While future work should be conducted into the general applicability of the proposed suboptimal insertion algorithm, it was found to efficiently identify models of interest in the neural examples considered here.

It was found that the optimal model $\mathcal{M}_c = [1232]$ was the most likely model. The log odds (see Table 5.3) between $\mathcal{M}_c = [1232]$ and the next most likely model $\mathcal{M}_c = [1132]$ are such that the

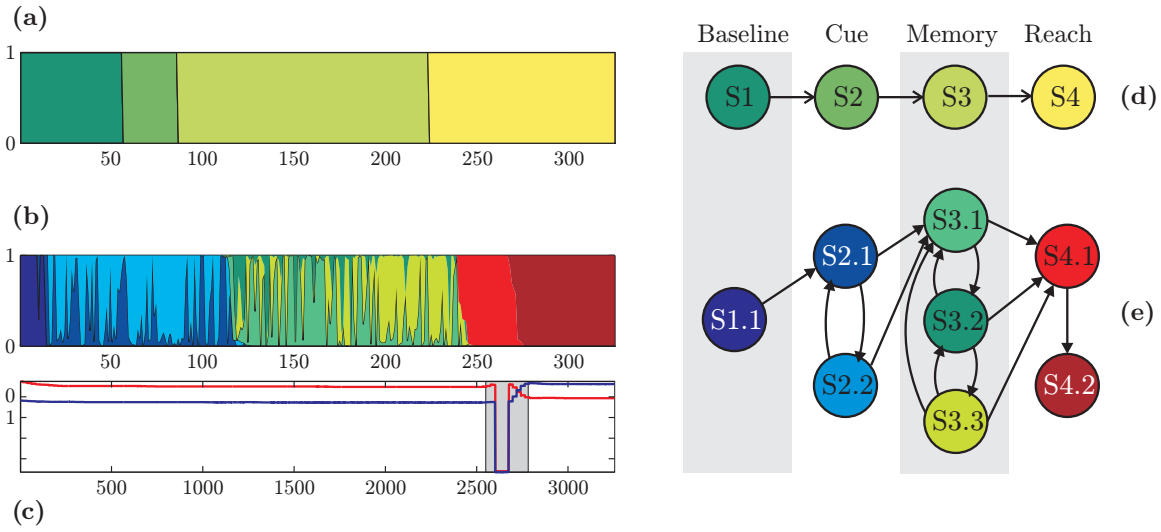

Figure 5.8: The optimal 8-state GLHMM model class identified using a combination of Bayesian model class selection and ASD priors. (a) Original left-to-right state sequence defined by experimental cues of the first trial of the training data set; the states represent *baseline*, *cue*, *memory*, and *reach*, respectively. (b) The corresponding 8-state GLHMM state sequence found during the identification process. (c) The recorded arm movement during the trial: the movement is recorded with a touch screen, and the x-y coordinates saturate to a lower bound when the monkey's hand is removed from the screen. The grey section highlights period when arm movement occurred. (d) The original left-to-right transition sequence defined by the experimental cues. (e) The 8 sub-states of the optimal GLHMM model, with allowed transitions between states depicted with an arrow.

posterior density effectively assigns all probability to the single model. In general it was found that the model class section algorithm would single out an individual model class with all other models having effectively zero posterior probability. This is a typical result when using Bayesian model class selection with reasonably large data sets.

The 8-state GLHMM provided improved supervisory decoding performance over the original 4-state GLHMM specified from the experimental markers. The forward filter decoded 81.25% of trials correctly. It should be noted that the forward filter found almost all of the actual "go" signals (99.31% of trials), however the total performance of the decode algorithm is reduced by the presence of false positives. The Viterbi algorithm was used for decoding and resulted in 85.42% of trials correctly decoded. This improved performance over the forward filter is due to the non-causality of the filter.

In view of the impressive performance gains of 8-state GLHMM, there may be concern that the model is over-learning, or "fitting" to the trial structure of the experiment, and ignoring the neural signals. While the learned 8-state model is validated on a large testing data set, it is still possible that only the sequential nature of the trial is being fit, and that the model may not generalize well to trials outside of this repetitive structure. This concern stems from the fact that that addition of extra states in a HMM can approximate modeling the duration in each mode [80]. There are three

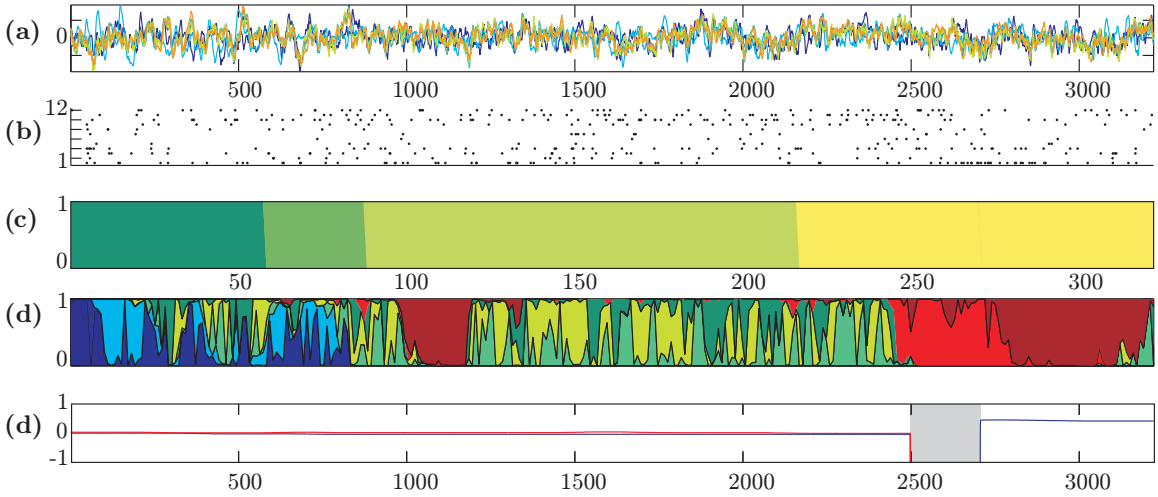

Figure 5.9: Decoding of testing trial 34/144 using 8-state GLHMM: In this example there is a false positive using the forward filter decode algorithm, represented by the dark red state in (d). This trial is presented as it demonstrates the continued confusion between the later periods of the reach state and the memory period. (a) Recorded LFP signals. (b) SUA activity of the 12 neurons used in the decode process. (c) Original left-to-right state sequence defined by experimental cues; the states represent *baseline*, *cue*, *memory*, and *reach*, respectively. (d) The recorded arm movement during the trial. The grey section highlights the period where arm movement occurs

pieces of evidence that suggest this model will generalize to other trial structures, and is capable of decoding outside of the trial structure: First, the presence of false positives means that the effect of the neural data is still present in the decoding. Second, the experiments themselves were designed to have a range of time spent in each experimental mode; notably the duration of the memory period was varied in [16] from 1.2 to 1.8 seconds. Third, and perhaps the most compelling, is the decode of testing trial 55/144. In this reach trial, the monkeys' behavior breaks from the structured center-out reach structure, and the monkey instead conducts two reaches. The first reach is to an incorrect target, and the second reach occurs after a brief delay period to the correct target. None of the training trials contained two reaches or deviated from the standard center-out reach structure. The forward filter is able to capture both reaches, negating the possibility of outfitting the model structure.

In Section 5.4.2, the effect of explicitly modeling the duration in each state is directly analyzed by replacing the Markovian transition rules of the GLHMM with the HSMM model. By considering the duration of each mode explicitly, prior information can be used to bias the model away from quickly switching between discrete states. From a neurophysiological point of view, the rapid switching between neural process found using the 8-state GLHMM may not be characteristic of expected behavior.

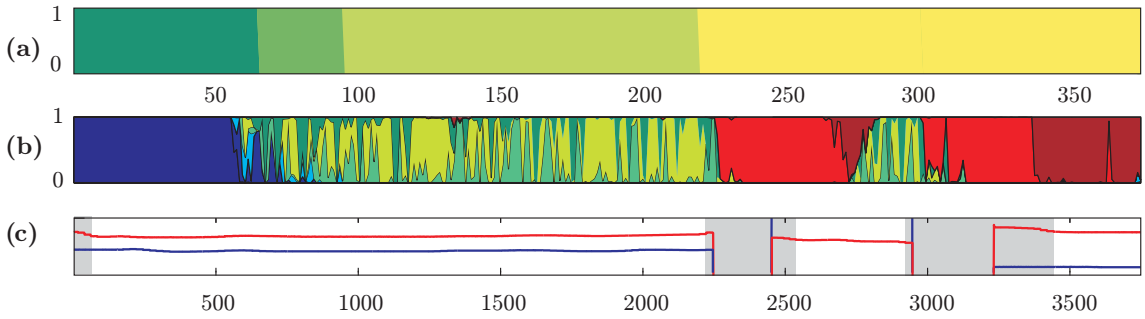

Figure 5.10: Decoding of a double reach using 8-state GLHMM (testing trial 55/144): In this trial the monkey first reaches for a target and then after a delay period conducts a second corrective reach to a second target. The forward filter is able to detect both reaches. Note that this trial is classified as a "false positive" given the performance criterion defined in Section 5.3. (a) Original left-to right state sequence defined by experimental cues. (b) Decoded state sequence using forward filter. (c) The recorded arm movement during the trial: the movement is recorded with a touch screen, and the x-y coordinates saturate to a lower bound when the monkeys hand is removed from the screen. The grey section highlights the period where arm movement occurs.

5.4.1.1 Prior Information and Initialization Used for Identification of GLHMM Models

The prior information for the transition matrix A of each model class $\mathcal{M}_c$ was defined as:

$$a_{ij}^0 = \begin{cases} 50 & \text{if } j = i \\ 1 & \text{if } j = i+1 \\ 0.1 & \text{if } S_i \text{ and } S_j \text{ are sub-states} \\ 0.001 & \text{else} \end{cases}, \tag{5.11}$$

and where the HSMM is constrained so self-transitions are not allowed. For example, the prior $\mathrm{P}(A)$ for $\mathcal{M}_c = [1231]$ is:

$$A^0 = \begin{bmatrix} \mathbf{50} & 1 & 0.001 & 0.001 & 0.001 & 0.001 & 0.001 \\ 0.001 & \mathbf{50} & \mathbf{1} & 0.1 & 0.001 & 0.001 & 0.001 \\ 0.001 & \mathbf{0.1} & \mathbf{50} & 1 & 0.001 & 0.001 & 0.001 \\ 0.001 & 0.001 & 0.001 & \mathbf{50} & \mathbf{1} & \mathbf{0.1} & 0.1 \\ 0.001 & 0.001 & 0.001 & \mathbf{0.1} & \mathbf{50} & \mathbf{1} & 0.1 \\ 0.001 & 0.001 & 0.001 & \mathbf{0.1} & \mathbf{0.1} & \mathbf{50} & 1 \\ 0.001 & 0.001 & 0.001 & 0.001 & 0.001 & 0.001 & \mathbf{50} \end{bmatrix}, \tag{5.12}$$

where groups of substates in (5.12) are denoted in bold. All other priors are retained from Section 5.3.1. The transition matrix prior distribution (5.12) strongly biases the model to choosing a left-to-right structure. This prior enforces the number of discrete states specified by the model class,

but allows automatic determination of the allowed transitions between substates (see ASD priors in Chapter 4).

The initialization of the variational algorithm for each model class was conducted using the following procedure: The experimental cues were used to divide each training trial into appropriate segments. Each segment was then randomly split into the appropriate number of sub-states defined by the model class. This initialization allows for rapid convergence of the VB algorithm, and allows the supervisory decoder to associate identified submodes to an appropriate experimental period.

5.4.2 HSMM with Model Selection

In this section a supervisory decoder based on the HSMM with GLM dynamics (Def. 3.5) is used to model the neural activity of the Musallam data set. The variational learning algorithm for HSMM developed in Section 3.5.3 is used in conjunction with the Bayesian model class selection and ASD priors defined in Chapter 4.

The suboptimal substate insertion method (described in Section 5.4.1) for enumerating a set of model classes was used. The model class with the highest posterior probability was a 7-state HSMM with: $\mathcal{M}_c = [1, 1, 3, 2]$ (see Figure 5.11). This model is remarkably similar to that identified with the GLHMM (see Section 5.4.1). The prior distributions (Section 5.4.2.1) used to model knowledge of duration in each state were minimally informative, but were biased against rapid switching between discrete states. The identified HSMM model retained the characteristic switching between several substates of the memory period, found when identifying the GLHMM model. By increasing the amount of information contained in the prior distributions, the identified models could be constrained to avoid this behavior, however this type of model class has significantly smaller posterior probability when compared to the less informative counterparts.

Supervisory decoding using the 7-state HSMM requires the use of a fixed lag smoother (Def. 3.13) instead of the forward filter used for GLHMMs. This fixed lag smoother delays estimation of the discrete cognitive state by a fixed amount of time (in this section a 0.1 second lag was used). The fixed lag smoother is required due to the nature of the HSMM. An indicative example decode using the forward filter, the fixed lag smoother and the smoother (Def. 3.12) which uses all data from the trial are compared in Figure 5.12.

The fixed lag smoother (with a 0.1 second lag) used in conjunction with the identified 7-state HSMM correctly decoded 87.5% of trials. The fixed lag smoother decodes the correct "go" signal in 96.53% of trials, but the total performance of the decode algorithm is degraded by the presence of false positives. The smoother, using all data from the trial, correctly decodes 91.67% of trials. The forward filter proved to be practically useless for decoding: 0% of trials were correctly decoded, due to the excessive number of false positives occurring in the decode process. Note that the forward filter did decode the actual "go" signal in 100% of the trials, but these "correct" decodes are practically

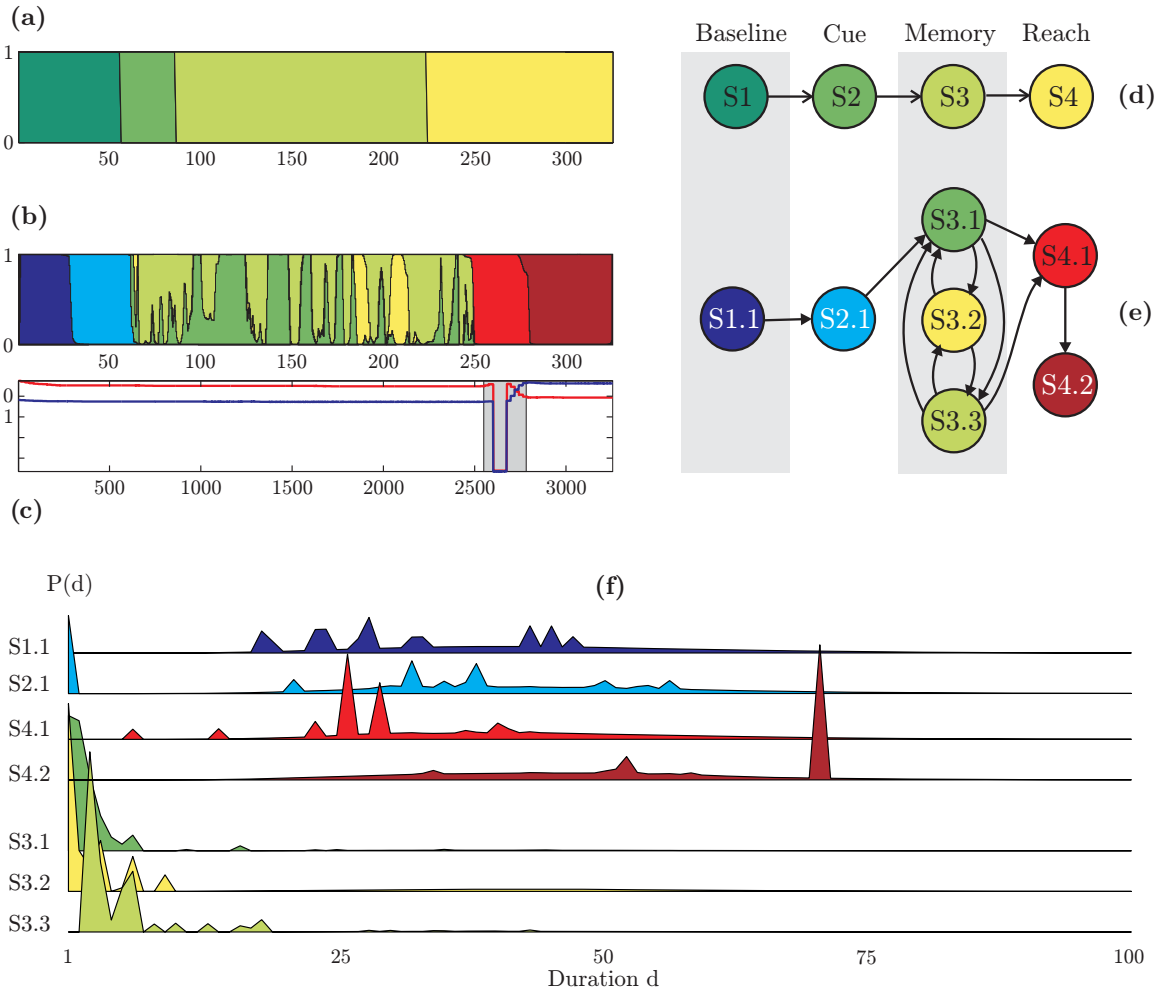

Figure 5.11: The optimal 7-state HSMM identified using a combination of Bayesian model class selection and ASD priors. (a) Original left-to-right state sequence defined by experimental cues of the first trial of the training data set; the states represent *baseline*, *cue*, *memory*, and *reach*, respectively. (b) The corresponding 7-state HSMM state sequence found during the identification process. (c) The recorded arm movement during the trial. (d) The original left-to-right transition sequence of the 4-mode model. (e) The 7 sub-states of the optimal HSMM model, with allowed transitions between states depicted with an arrow. (f) The posterior distribution of the duration spent in each discrete state. the maximum allowed duration D for any state was defined as 300, but there is no significant probability mass for durations longer then 100.

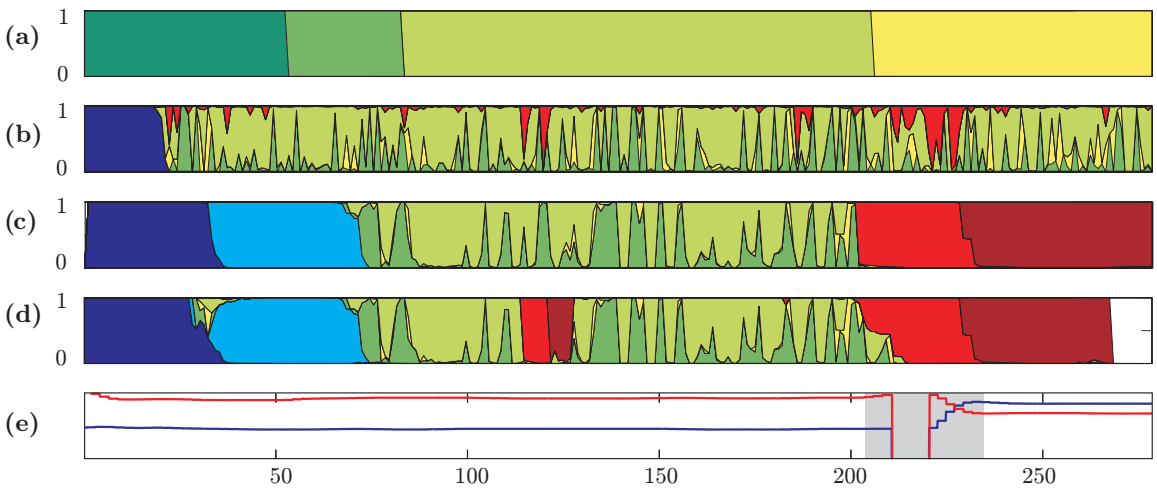

Figure 5.12: Example decode with 7-mode HSMM when using: (b) The forward filter. (c) Smoother which utilized all of the observed data. (d) Fixed lag smoother with a lag of 0.1 seconds. The decode has been shifted by 0.1 seconds to align with the smoothing results and arm movement. (a) The original left-to-right state sequence defined by experimental cues. (e) The recorded arm movement during the trial. This trial was chosen to demonstrate the improved decoding ability of the smoother over the fixed lag filter. In the majority of trials the smoother and fixed-lag smoother produce nearly identical results. In all trials the forward filter produced many low-duration false positives typically indistinguishable from the decoding of the actual "go" signal.

indistinguishable from the false positives. The forward filter proves to be a poor choice for decoding HSMM, as it does not effectively take into account the duration spent in each mode. In the speech processing community the non-causal Viterbi algorithm is typically used in conjunction with HSMM (or HMM) for decoding [10]. We found that by accepting a small lag, the fixed lag smoother could recover most of the performance of the smoother using all of the data. The introduction of a 0.1 second lag in a neurological supervisory decoder is a negligible amount of time.

5.4.2.1 Prior Data for HSMM Supervisory Decoder

This section gives specific details about the prior distribution used in identifying HSMM supervisory decoders from the Musallam data set. The prior distributions on the GLM dynamics are the same as are used in (5.3.1). The priors on the HSMM transition matrix were biased to give an overall left-to-right model, but transitions between substates were not constrained and used an ASD prior to determine connectivity:

$$a_{ij}^0 = \begin{cases} 1 & \text{if } j = i+1 \\ 0.1 & \text{if } S_i \text{ and } S_j \text{ are sub-states} \\ 0.001 & \text{else} \end{cases}, \tag{5.13}$$

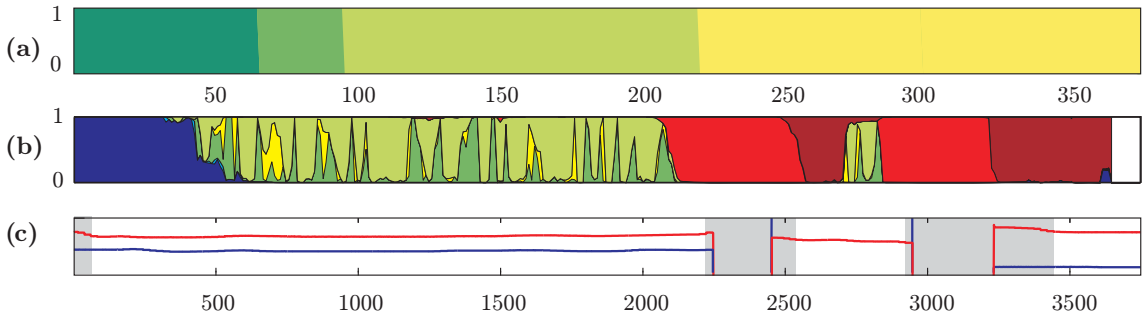

Figure 5.13: 7-state HSMM decode of double reach (trial 55/144): In this trial the monkey first reaches for a target and then after a delay period conducts a second corrective reach to a second target. (a) Original left-to-right state sequence defined by experimental cues. (b) Decoded state sequence using forward filter. (c) The recorded arm movement during the trial. The grey section highlights the period where arm movement occurs

and where the HSMM is constrained so self-transitions are not allowed. The model identification process was found to be invariant to a range of hyperparmaters used in the prior definition. The prior distribution on the duration spent in each mode is generated by a Gamma distribution:

$$p_i^0(d) = c\, d^{\alpha-1} \frac{\beta^\alpha \exp(-\beta d)}{\Gamma(\alpha)} \quad , \tag{5.14}$$

where $\alpha = 10$, $\beta = 1/5$. The value of c was set to 20, and represents the effective number of data points that the prior represents (see Remark 4.4 in Section 4.3). It was found that careful consideration of the prior distribution of the duration was required. Because of the nature of HSMM, there can be a very small number of transitions between discrete states, and if the constant c is chosen to be too high (> 100), then the prior information "swamps" the posterior distribution. if the value of c is chosen to be very small (e.g., 1) then the prior resembles a ASD prior and few posterior durations have any significant probability mass. This difficulty may be avoidable by choosing to use parameterized (e.g. Poisson or Gamma) distributions for the posterior duration model.

Chapter 6

Conclusions

6.1 Summary of Thesis Contributions

The primary contribution of this thesis is the definition of a series of hybrid system models, and the development of Bayesian inference algorithms for identification of these models from observed data. By associating continuous dynamics with both stationary and nonstationary Markov chains, a series of hybrid models capable of modeling a range of biological and engineering systems were developed. Motivating the development process is the application of supervisory decoding for neural prosthetics (Chapter 5). Here, neural activity is modeled as a hybrid system which represents both the continuous dynamics of observed extracellular neural activity, and the discrete transitions between different cognitive or planning states.

The developed models and identification methods of Chapter 3 provide novel contributions in both the fields of hybrid systems and machine learning. A series of hybrid system models based on the hidden Markov model (HMM), the hidden semi-Markov model (HSMM), and the variable transition hidden Markov model (VTHMM), were created by the addition of generalized linear model (GLM) dynamics. The resulting hybrid systems, including the generalized linear hidden Markov model (GLHMM), and its HSMM and VTHMM counterparts, were used to model both biological and mechanical systems. A key contribution in the thesis is the extension of the variational Bayesian (VB) framework to identification of the developed hybrid models; even without the addition of GLM dynamics, applying VB to HSMM and VTHMM models is a significant contribution to the machine learning literature. These models are typically used in speech processing technology, and the developed VB approach has several inherent advantages over the standard EM implementation. An additional Bayesian inference algorithm, the Gibbs sampler, is also adapted for use in GLHMM models. The GLHMM framework is applied to the identification of piecewise autoregressive exogenous (PWARX) models, a class of models that define discrete transitions based on the autoregressive state. Apart from providing a novel method of PWARX identification, this analysis motivates the development of a new class of hybrid system: the hidden regressor-dependent Markov

model (HRDMM).

When creating hybrid models of many systems, the prior intuition about the system's structure may be incomplete, and Bayesian model class selection can be used to infer the number of discrete modes, transition structures, and orders of continuous dynamics of the model. Chapter 4 shows the importance of the model evidence for Bayesian model class selection in information theoretic terms, and applies two methods to evaluate the evidence. First, the developed VB approach in Chapter 3 inherently provides an estimate of the model evidence. Second, the Stationarity method for estimating the model evidence from posterior samples is refined for use in hybrid systems and systems with latent variables. This Stationarity method allows the Gibbs sampler to be effectively used for model class selection in hybrid systems. In addition to Bayesian model class selection, automatic structure determination (ASD) priors are defined which represent a body of work that allows subsequently applied inference algorithms to "prune" out unneeded model structure. ASD priors are developed for the HSMM model, and are then demonstrated by automatically identifying the number of movement primitives in a bee dance data set.

The developed Gibbs sampler and VB inference algorithms (Chapter 3) and associated model selection tools (Chapter 4) are used to build a supervisory decoder for neural prosthetics in Chapter 5. The design of a supervisory decoder, whose job it is to classify, in real time, the discrete cognitive or behavioral state of the brain region from which the neural signals are recorded, consists of two parts: (1) the identification of the hybrid model which represents the neural activity in each discrete state, as well as the transitions between states; (2) the design of an estimator which uses the hybrid model to classify activity into the discrete cognitive or behavior states. Three important contributions are made in the new framework over existing supervisory decoders: new models which are capable of both explicitly modeling the duration spent in each cognitive state and incorporating disjoint types of recorded neural signals are developed; the developed identification process is automatic, in that it does not require recorded neural data to be pre-segmented; and, if incomplete information about either the number of cognitive modes or the underlying neural process exists, then model class selection methods can be deployed to automatically infer the optimal model structure. All of these contributions were shown to improve the performance of the supervisory decoder on recorded neural data sets.

6.2 Opportunities for Future Work

While the proceeding chapters have presented novel contributions to the machine learning, hybrid system, and neural prosthetic literature, there are three areas of research that will further the contributions of this thesis. The first two research areas relate to new applications and refinements of the designed models. The third research area proposes new experiments and algorithmic developments

which may improve the developed supervisory decoder framework and facilitate confidence in its deployment to clinical prosthetic patients.

While the main theme of this thesis considerers hybrid models with both continuous dynamics and discrete switching, the application of VB to HSMM and VTHMM models provides an opportunity to contribute to speech processing technology. Typical speech processing methodologies use discrete HMM, HSMM, or VTHMM models with a multinominal likelihood function, and the field is characterized by incremental improvements of a few percent performance on decoding recorded audio data sets. The superior generalization properties of the VB algorithm over EM may provide yet another performance gain in decoding technology when using HSMM or VTHMM models. Furthermore, the developed VB inference algorithm for HSMM has $\mathrm{O}(ND + N^2)$ computational complexity, where N is the number of discrete states, and D is the maximum duration, as opposed to most HSMM identification algorithms with $\mathrm{O}(N^2D)$ computational complexity.

The development of the HRDMM model in Chapter 3 was motivated by an application that is unexplored in this thesis. Specifically, the application of segmenting animal motion into behavioral primitives, and exploring interaction of the environment and other animals with the animals decision making processes is proposed. In Chapter 4, we demonstrated automatic segmentation of bee motion into several motion primitives. A novel research direction is to directly model the switching between these motion, or behavioral primitives, as a function of the environment. For instance, the proximity, or angle subtended in the visual field, by other nearby animals could be used as a basis for switching between discrete motion primitives. In essence, if the bee tended to stop as it neared another bee, this could be modeled in the regressor dependent Markovian transition kernel of the HRDMM. Using the provided ARD methods for HRDMMs, a large number of potential covariates could be added into the regressor, and the observed animals behavior could be used to automatically select which covariances are important in the animals decision making process. This method would prove useful in genetic experiments, where automatic phenotyping of animal behavior is required. If a new genetic strain of bee (or fly) is created in the laboratory, the 'aggressiveness' of the new animal genotype could be automatically measured as the willingness to give way to other nearby flies. Another potential system that would benefit from the application of HRDMM models is that of autonomous vehicles driving in an urban environment. A known problem in this situation is the modeling of other vehicles (or pedestrians) in the proximity of the autonomous vehicle. If nearby vehicles were modeled with a HRDMM, the transitions between basic driving primitives such as turning, stopping or accelerating could be modeled as a function of the distance to landmarks such as traffic lights, stop signs or intersections. Applying the proposed HRDMM to these applications may require some application dependent modifications: the choice of the softmax basis for the HRDMM transition kernel was based on PWARX models; this softmax function could be replaced by better classifiers, such as the relevance vector machine, or kernel that depends on a combination of temporal and state

based variables.

The primary application area, which motivated the development of proposed hybrid system models and identification algorithms, is the creation of a supervisory decoder for neural prosthetics. The developed supervisory decoder framework was successfully demonstrated on recorded neural data, but there remain many research avenues to explore the potential of the developed systems. To directly demonstrate the suitability of the developed supervisory decoder for clinical human prosthetic patients, a series of experiments and extensions should be considered. First, current data sets can be more completely analyzed and the consistency of model order selection to different training trial selections should be investigated. Second, the identification and application of the supervisory decoder to neural data sets in which a sequence of reaches are conducted is required. Third, and perhaps most difficult, is the implementation of self-paced experiments where the supervisory decoder is used to initiate actual prosthetic movement. Putting the supervisory decoder 'in the loop' will have many consequences not apparent when using pre-recorded data sets. The plasticity of the brain potentially means that the observed neural activity will change over time in response to the decoder. The proposed Bayesian inference framework is naturally updated over time with the addition of new observed data, however it is difficult to predict the effects of relearning or updating neurological models at this early stage. Finally the developed supervisory decoder framework may prove of use in understanding cortical processes in the brain. The new methods allow for an information theoretic perspective to be used in estimating the number of discrete states and transition structure between cognitive and planning activities. This added information may improve current methods of decoding intended reach directions in brain areas such as the posterior parietal or dorsal premotor cortices by refining the periods of activity used for decoding. Thus, in the long term, the advances in this thesis may provide the backbone of a practical neuroprosthetic system which incorporates recordings from higher cortical areas.

Appendix A

Probability Theorems and Distributions

This section provides a quick reference for several probability theorems and distributions which are repetitively used in the thesis. This section does not seek to provide a tutorial on probability theory.

A.1 Axioms and Theorems

Three axioms of probability are assumed throughout this thesis:

$$\mathrm{P1} : \mathrm{P}\big(b|a\big) \geq 0 \tag{A.1}$$

$$\mathrm{P2} : \mathrm{P}\big(b|a\big) + \mathrm{P}\big(\sim b|a\big) = 1 \tag{A.2}$$

$$\text{[Product Rule]} \quad \mathrm{P3} : \mathrm{P}\big(c, b|a\big) = \mathrm{P}\big(c|b, a\big)\,\mathrm{P}\big(b|a\big) \tag{A.3}$$

If proposition a states that only one of $\{b_1, ..., b_N\}$ is true, then the following probability theorems can be used; Marginalization Theorem [Sum Rule]:

$$\mathrm{P}\big(c|a\big) = \sum_{n=1}^{N} \mathrm{P}\big(c, b_n|a\big) \tag{A.4}$$

Theorem of Total Probability:

$$\mathrm{P}\big(c|a\big) = \sum_{n=1}^{N} \mathrm{P}\big(c|b_n, a\big)\,\mathrm{P}\big(b_n|a\big) \tag{A.5}$$

Bayes' Theorem is fundamental throughout the thesis:

$$\mathrm{P}\big(b_n|c, a\big) = \frac{\mathrm{P}\big(c|b_n, a\big)\,\mathrm{P}\big(b_n|a\big)}{\mathrm{P}\big(c|a\big)} \tag{A.6}$$

A.2 Probability Distributions

Distribution	**Density, Moments, Entropy, etc.**	
Multivariate normal	$\mathcal{N}(z\|w,\Sigma) = \frac{1}{(2\pi)^{d/2}\|\Sigma\|^{1/2}} \exp\left[-\frac{1}{2}(z-w)^T\Sigma^{-1}(z-w)\right]$	(A.7)
Parameters	$z \in \mathbb{R}^d,$ mean: $w \in \mathbb{R}^d$ covariance: $\Sigma \in \mathbb{R}^{d\times d}$	
First moment	$\int z^T a\mathcal{N}(z\|w,\Sigma)dz = w^T a$	(A.8)
Second moment	$\int (z^T A z^T)\mathcal{N}(z\|w,\Sigma)dz = Tr(A\Sigma) + w^T A w$	(A.9)
Gamma	$\mathrm{Gam}(\tau\|a,b) = \frac{1}{\Gamma(a)} b^a \tau^{a-1} e^{-b\tau}$	(A.10)
Parameters	$\tau > 0,$ shape: $a > 0,$ rate: $b > 0$	
Mean	$E[\tau] = \frac{a}{b}$	(A.11)
Geometric mean	$E[\log\tau] = \psi(a) - \log b$	(A.12)
	where Γ is the gamma function, and:	
	$\psi(x) = \frac{d}{dx}\ln\Gamma(x) = \frac{\Gamma'(x)}{\Gamma(x)}$	(A.13)
	is the digamma function	
Gamma-Gaussian	$\mathcal{N}(z\|w,(\tau\Lambda)^{-1})\mathrm{Gam}(\tau\|a,b) =$ $\frac{\|\tau\Lambda\|^{1/2}}{(2\pi)^{d/2}} \exp\left[-\frac{\tau}{2}(z-w)^T\Lambda(z-w)\right] \frac{1}{\Gamma(a)} b^a \tau^{a-1} e^{-b\tau}$	(A.14)
Parameters	$z \in \mathbb{R}^d, \tau > 0,$ shape: $a > 0,$ rate: $b > 0$ mean: $w \in \mathbb{R}^d, \Lambda \in \mathbb{R}^{d\times d}$	
Dirichlet	$\mathrm{Dir}(\pi\|\alpha) = \frac{\Gamma(\sum_{i=1}^N \alpha_i)}{\prod_{i=1}^N \Gamma(\alpha_i)} \prod_{i=1}^N \pi_i^{\alpha_i - 1}$ such that: $\pi_i > 0, \sum_i^N \pi_i = 1;$	(A.15)
Parameters	$\pi = \{\pi_1, ..., \pi_N\}, \alpha = \{\alpha_1, ..., \alpha_N\}$	
Geometric mean	$\langle \log\alpha_j \rangle = \psi(\alpha_j) - \psi(\sum_{i=1}^N \alpha_i)$	(A.16)
Poisson	$\mathrm{Pois}(y\|\lambda) = \frac{\lambda^y e^{-\lambda}}{y!}$	(A.17)
Parameters	$y \in \mathbb{N},$ rate: $\lambda > 0$	(A.18)

Appendix B

Cross Entropy and KL-Divergence

Approaching model selection from an information theoretic basis often requires the calculation of the information gained from the prior to the posterior of a model. This information gain is also called the *relative entropy* or the *Kullback-Leibler divergence* in information theory. To maintain consistency with the Variational learning community, this thesis will use the term Kullback-Leibler (KL) divergence. In this section the KL divergence between several distributions used throughout the thesis are calculated. This is required for two reasons: First, the KL-divergence of the Gaussian-Gamma distribution for AR models, which has a multivariate Gaussian distribution is not readily found in the literature. Second, the KL divergence of many other distributions, which are well known, are not always correctly stated[1], and may otherwise be difficult to find.

The KL divergence from distribution $q(x)$ to distribution $p(x)$ is defined:

$$KL(q||p) = \int q(x) \log \frac{q(x)}{p(x)} dx \ . \tag{B.1}$$

A useful decomposition of the KL divergence is formulated in terms of the cross entropy $H(q,p)$, and entropy $H(q)$:

$$KL(q||p) = H(q,p) - H(q) \tag{B.2}$$

where $H(q,p)$ is the *cross entropy* between two distributions $q(x)$ and $p(x)$:

$$\begin{aligned} H(q,p) &= -\int q(x) \log p(x) dx \\ &= -E_q \left[\log p(x)\right] \ , \end{aligned} \tag{B.3}$$

and $H(q)$ is the *entropy* of a distribution $p(x)$:

$$H(q) = -\int q(x) \log q(x) dx \ . \tag{B.4}$$

[1] Wikipedia for advanced mathematics is not recommended

The following sections will proceed by first calculating the cross entropy of various distributions, and then using this result to infer the KL divergence.

B.1 Cross Entropy of Gaussian Distributions

Consider two normal distributions (A.7):

$$p(x) = \mathcal{N}(x|x_p, \Sigma_p), \quad q(x) = \mathcal{N}(x|x_q, \Sigma_q) \ . \tag{B.5}$$

The cross entropy (B.3) of these distributions is calculated using the linearity of the expectation operator:

$$\begin{aligned} -H(q,p) &= E_q\left[\log p(x)\right] \\ &= \log\left(\frac{|\Sigma_p|^{-1/2}}{(2\pi)^{n/2}}\right) + E_q\left[-\frac{1}{2}x^T\Sigma_p^{-1}x\right] + E_q\left[x^T\Sigma_p^{-1}x_p\right] + E_q\left[-\frac{1}{2}x_p^T\Sigma_p^{-1}x_p\right] \ . \end{aligned} \tag{B.6}$$

These expectations (B.6) are calculated using the first and second moments (A.8),(A.9) of a multivariate normal distribution:

$$-H(q,p) = \log\left(\frac{|\Sigma_p|^{-1/2}}{(2\pi)^{n/2}}\right) - \frac{1}{2}Tr\left(\Sigma_p^{-1}\Sigma_q\right) - \frac{1}{2}x_q^T\Sigma_p^{-1}x_q + x_q^T\Sigma_p^{-1}x_p - \frac{1}{2}x_p^T\Sigma_p^{-1}x_p \ . \tag{B.7}$$

In simplified form:

$$H(q,p) = \frac{1}{2}\left[\log|\Sigma_p| + n\log(2\pi) + \mathrm{Tr}\left(\Sigma_p^{-1}\Sigma_q\right) + (x_q - x_p)^T\Sigma_p^{-1}(x_q - x_p)\right] \ . \tag{B.8}$$

B.2 KL Divergence between Gaussian Distributions

The KL divergence between two normal distributions can now be derived using the cross entropy (B.8) and noting that the entropy is a subcase of the cross entropy:

$$\begin{aligned} KL_{\mathcal{N}}\left(q||p\right) &= H(q,p) - H(q,q) \\ &= \frac{1}{2}\left(\log\left(\frac{\det(\Sigma_p)}{\det(\Sigma_q)}\right) + \mathrm{Tr}\left(\Sigma_p^{-1}\Sigma_q\right) - \mathrm{Tr}(I_n) + (x_q - x_p)^T\Sigma_p^{-1}(x_q - x_p)\right) \ , \end{aligned} \tag{B.9}$$

where n is the dimension of $x \in \mathbb{R}^n$.

B.3 Cross Entropy of Gamma Distributions

Consider two Gamma distributions (A.10):

$$p(\tau) = \text{Gam}(\tau|a_p, b_p), \quad q(\tau) = \text{Gam}(\tau|a_q, b_q) \ . \tag{B.10}$$

The cross entropy (B.3) of these distributions can be evaluated by expanding the $\log p(\tau)$ term using the definition (A.10), and using the linearity of the expectation operator:

$$\begin{aligned} -H(q,p) &= E_q[\log p(\tau)] \\ &= \log\left(\frac{b_p^{a_p}}{\Gamma(a_p)}\right) + E_q\left[(a_p - 1)\log\tau\right] + E_q[-b_p\tau] \ . \end{aligned} \tag{B.11}$$

The last two terms of (B.11) are evaluated using the mean (A.11) and geometric mean (A.12) of the Gamma distribution:

$$H(q,p) = -\log b_p^{a_p} + \log\Gamma(a_p) - (a_p - 1)\left(\psi(a_q) - \log(b_q)\right) + b_p\frac{a_q}{b_q} \ . \tag{B.12}$$

B.4 KL Divergence between Gamma Distributions

The KL divergence between two Gamma distributions can now be derived using the cross entropy (B.12):

$$\begin{aligned} KL_{\text{Gam}}\left(q||p\right) = &= H(q,p) - H(q,q) \\ &= \log\frac{\Gamma(a_p)b_q^{a_q}}{\Gamma(a_q)b_p^{a_p}} + (a_q - a_p)(\psi(a_q) - \log(b_q)) + (b_p - b_q)\frac{a_q}{b_q} \ . \end{aligned} \tag{B.13}$$

B.5 Cross Entropy of Gaussian-Gamma Distributions

Consider two Gaussian-Gamma distributions (A.14) :

$$p(x,\tau) = \mathcal{N}_p \ \text{Gam}_p, \quad q(x,\tau) = \mathcal{N}_q \ \text{Gam}_q \ , \tag{B.14}$$

where the following notation is used for compactness:

$$\mathcal{N}_p = \mathcal{N}(x|x_p, (\tau A_p)^{-1}), \quad \text{Gam}_p = \text{Gam}(\tau|a_p, a_p) \ . \tag{B.15}$$

The cross entropy (B.3) is now stated explicitly terms of the above distributions:

$$H(q,p) = -\int\int \mathcal{N}_q \ \text{Gam}_q \log\left[\mathcal{N}_p \ \text{Gam}_p\right] dxd\tau \ . \tag{B.16}$$

Using the independence of x of the distribution Gam_p, and splitting the log term of (B.16) results in:

$$H(q,p) = -\int \mathrm{Gam}_q \left[\int \mathcal{N}_q \log \mathcal{N}_p dx \right] d\tau + \int \mathcal{N}_q \left[\int \mathrm{Gam}_q \log \mathrm{Gam}_p d\tau \right] dx \ . \tag{B.17}$$

The second term in the RHS of (B.17) is simplified by recognizing the inner integral of τ is the cross entropy of the Gamma distributions (B.12) and the outer integral over x is of a normalized Gaussian and simply integrates to one. The first term in (B.17) is left as an expectation of the cross entropy of normal distributions:

$$H(q,p) = E_{\mathrm{Gam}_q}\left[H(\mathcal{N}_q, \mathcal{N}_p)\right] + H(\mathrm{Gam}_q, \mathrm{Gam}_p) \tag{B.18}$$

B.6 KL Divergence between Gaussian–Gamma Distributions

The KL divergence of two Gaussian-Gamma distributions is derived using the Cross entropy (B.18):

$$\begin{aligned} KL_{\mathcal{N}\mathrm{Gam}}\left(q||p\right) &= H(q,p) - H(q,q) \\ &= E_{\mathrm{Gam}_q}\left[H(\mathcal{N}_q, \mathcal{N}_p) - H(\mathcal{N}_q, \mathcal{N}_q)\right] + H(\mathrm{Gam}_q, \mathrm{Gam}_p) - H(\mathrm{Gam}_q, \mathrm{Gam}_q) \quad \text{(B.19)} \\ &= E_{\mathrm{Gam}_q}\left[KL_{\mathcal{N}}\left(\mathcal{N}_q||\mathcal{N}_p\right)\right] + KL_{\mathrm{Gam}}\left(\mathrm{Gam}_q||\mathrm{Gam}_p\right) \quad \text{(B.20)} \end{aligned}$$

The second term of (B.20) is evaluated using the KL divergence between Gamma distribution (B.13). The first term in (B.20) can be calculated by noting that if $\Sigma_p = (\tau A_p)^{-1}$ and $\Sigma_q = (\tau A_q)^{-1}$ are substituted in to the KL divergence expression (B.9), then the result is linear in τ. This allows the expectation operator, a linear operator, to be moved inside the KL divergence and τ to be replaced with its expected value:

$$\begin{aligned} &KL_{\mathcal{N}\mathrm{Gam}}\left(q||p\right) = \\ &\qquad KL_{\mathcal{N}}\left(\mathcal{N}(x|x_q, (\langle\tau\rangle A_q)^{-1})||\mathcal{N}(x|x_p, (\langle\tau\rangle A_p)^{-1})\right) + KL_{\mathrm{Gam}}\left(\mathrm{Gam}_q||\mathrm{Gam}_p\right) \ , \end{aligned} \tag{B.21}$$

where $\langle\tau\rangle = \frac{a_q}{b_q}$ is the expectation of τ with respect to the distribution Gam_q.

B.7 KL Divergence between Dirichlet Distributions

The KL divergence between two Dirichlet distributions:

$$p(a) = \mathrm{Dir}(a|\alpha_T), \quad q(a) = \mathrm{Dir}(a|\alpha_0) \tag{B.22}$$

is [36]:

$$KL_{\mathrm{Dir}}\left(q||p\right) =$$
$$\log\frac{\Gamma\left(\sum_{i=1}^{n}\alpha_T(i)\right)}{\Gamma\left(\sum_{i=1}^{n}\alpha_0(i)\right)}+\sum_{i=1}^{n}\log\frac{\Gamma\left(\alpha_0(i)\right)}{\Gamma\left(\alpha_T(i)\right)}+\sum_{i=1}^{n}\left(\alpha_T(i)-\alpha_0(i)\right)\left(\psi(\alpha_T(i))-\psi\left(\sum_{i=1}^{n}\alpha_0(i)\right)\right) \quad \text{(B.23)}$$

where n is the length of α.

Appendix C

Posteriors and Integrals for AR and Poisson Models

C.1 Geometric Mean of AR Likelihood with a Gaussian-Gamma Distribution Parameter Model

This section calculates the geometric mean of AR model likelihood under the Gaussian-Gamma distribution:

$$b_k(y,x) = \exp \int_0^\infty \int_{\mathbb{R}^d} q(w,\tau) \ln p\left(y,x|w,\tau\right) dw d\tau \tag{C.1}$$

where the data likelihood for a single observation (x,y) is:

$$p\left(y,x|w,\tau\right) = \left(\frac{\tau}{2\pi}\right)^{1/2} \exp\left(-\frac{\tau}{2}\left(y - w^T x\right)^2\right) \quad , \tag{C.2}$$

and the Gaussian-Gamma distribution of the model parameters is defined as:

$$q(w,\tau) = \mathcal{N}\left(w;\hat{w},(\tau\Lambda)^{-1}\right) \mathrm{Gam}\left(\tau|a_T,b_T\right) \quad . \tag{C.3}$$

Here Λ is a positive definite precision matrix, and $w \in \mathbb{R}^d$. Because the Gaussian-Gamma distribution (C.3) defines the distribution of τ independently of w, the geometric mean (C.1) is simplified by explicitly substituting of (C.3):

$$b_k(y,x) = \exp \int_\tau \mathrm{Gam}\left(\tau|a_T,b_T\right) \left[\int_w \log\ p\left(y,x|w,\tau\right) \mathcal{N}\left(w;\hat{w},(\tau\Lambda)^{-1}\right) dw\right] d\tau \tag{C.4}$$

This integral (C.4) is analytically integrated by first considering the inner integral of w, which is defined as $F(\tau)$, and can be evaluated by expanding the error $\left(y - w^T x\right)^2$ term of the likelihood

(C.2):

$$F(\tau) = \int_w \log\ p\left(y, x|w, \tau\right) \mathcal{N}\left(w; \hat{w}, (\tau\Lambda)^{-1}\right) dw$$

$$= \int_w \log\left(\frac{\tau}{2\pi}\right)^{1/2} \mathcal{N}\left(w; \hat{w}, (\tau\Lambda)^{-1}\right) dw - \int_w \frac{\tau}{2}(y^2)\mathcal{N}\left(w; \hat{w}, (\tau\Lambda)^{-1}\right) dw \tag{C.5}$$

$$+ \int_w \frac{\tau}{2}(2w^T xy)\mathcal{N}\left(w; \hat{w}, (\tau\Lambda)^{-1}\right) dw - \int_w \frac{\tau}{2}(w^T xx^T w)\mathcal{N}\left(w; \hat{w}, (\tau\Lambda)^{-1}\right)\ .dw \tag{C.6}$$

To solve the integrals in (C.5) and (C.6), the *first* (A.8) *and second* (A.9) *moments* of the multivariate normal distribution are used, resulting in:

$$F(\tau) = \log\left(\frac{\tau}{2\pi}\right)^{1/2} - \frac{\tau}{2}\left(y^2\right) + \frac{\tau}{2}\left(2yx^T\hat{w}\right) - \frac{\tau}{2}\left(\hat{w}^T xx^T\hat{w} + Tr\left(xx^T(\tau\Lambda)^{-1}\right)\right)\ . \tag{C.7}$$

The Trace expression in (C.7) can be simplified by using the commutative property: $Tr(AB) = Tr(BA)$, implying:

$$Tr\left(xx^T(\tau\Lambda)^{-1}\right) = Tr\left(x^T(\tau\Lambda)^{-1}x\right) = x^T(\tau\Lambda)^{-1}x\ . \tag{C.8}$$

Using the complete square $(y^2 - 2yx^T\hat{w} + \hat{w}^T xx^T\hat{w}) = (y - \hat{w}^T x)^2$ and substituting the Trace expression (C.8) further simplifies (C.7):

$$F(\tau) = \log\left(\frac{\tau}{2\pi}\right)^{1/2} - \frac{\tau}{2}\left(y - \hat{w}^T x\right)^2 - \frac{1}{2}x^T\Lambda^{-1}x\ . \tag{C.9}$$

The analytic expression (C.9) can now be substituted into the geometric mean expression (C.1):

$$b_k(y, x) = \exp\int_\tau \mathrm{Gam}\left(\tau|a_T, b_T\right) F(\tau)d\tau \tag{C.10}$$

$$= \exp\int_\tau \log\left(\frac{\tau}{2\pi}\right)^{1/2} \mathrm{Gam}\left(\tau|a_T, b_T\right) d\tau \tag{C.11}$$

$$\times \exp\int_\tau -\frac{\tau}{2}\left(y - \hat{w}^T x\right) \mathrm{Gam}\left(\tau|a_T, b_T\right) d\tau \tag{C.12}$$

$$\times \exp\int_\tau -\frac{1}{2}x^T\Lambda^{-1}x\ \mathrm{Gam}\left(\tau|a_T, b_T\right) d\tau\ . \tag{C.13}$$

The integrals (C.11) to (C.13) are now analytically integrated. By noting that $\mathrm{Gam}\left(\tau|a_T, b_T\right)$ is normalized, implying that it integrates to 1, the integral (C.13) simplifies to:

$$\exp\int_\tau -\frac{1}{2}x^T\Lambda^{-1}x\mathrm{Gam}\left[\tau|a_T, b_T\right] d\tau = \exp\left(-\frac{1}{2}x^T\Lambda^{-1}x\right)\ . \tag{C.14}$$

Integral (C.12) is simplified by calculating the the mean of the Gamma distribution (A.11):

$$\exp\left[-\frac{1}{2}\left(y - \hat{w}^T x\right)^2 \int_\tau \tau\mathrm{Gam}\left(\tau|a_T, b_T\right) d\tau\right] = \exp\left[-\frac{1}{2}\left(y - \hat{w}^T x\right)^2 \frac{a_T}{b_T}\right] \tag{C.15}$$

Integral (C.11) is solved by rearranging terms to form the geometric mean of the Gamma distribution (A.12):

$$\begin{aligned}
&\exp \int_\tau \frac{1}{2} \left(\log \tau - \log(2\pi)\right) \mathrm{Gam}\left(\tau | a_T, b_T\right) d\tau \\
&= \exp \left[\int_\tau \frac{1}{2} \log \tau \mathrm{Gam}\left(\tau | a_T, b_T\right) d\tau - \int_\tau \frac{1}{2} \log(2\pi) \mathrm{Gam}\left(\tau | a_T, b_T\right) d\tau \right] \\
&= \exp \left[\frac{1}{2} \big(\psi(a_T) - \log b_T \big) - \frac{1}{2} \log(2\pi) \right] \qquad \text{(C.16)}
\end{aligned}$$

The Gaussian-Gamma geometric mean $b_k(x, y)$ is now calculated by substituting equations (C.14), (C.15) and (C.16) into (C.10):

$$b_k(y, x) = \exp \left[-\frac{1}{2} x^T \Lambda^{-1} x - \frac{1}{2} \left(y - \hat{w}^T x \right)^2 \frac{a_T}{b_T} + \frac{1}{2} \big(\psi(a_T) - \log b_T \big) - \frac{1}{2} \log(2\pi) \right] \ . \qquad \text{(C.17)}$$

C.2 Geometric Mean of Poisson Likelihood with a Gamma Firing Rate Model

This section calculates the geometric mean of a Poisson observation likelihood with a Gamma firing rate distribution:

$$b(y) = \exp \int q(\lambda) \log p\left(y | \lambda\right) d\lambda \ , \qquad \text{(C.18)}$$

where the firing rate λ is represented by a Gamma (A.10) distribution $q(\lambda) = \mathrm{Gam}(\lambda | a, b)$ and the observation y likelihood is a Poisson distribution (A.17). The integral (C.18) is then:

$$b(y) = \exp \int \left(y \log(\lambda) - \lambda - \log(y!) \right) \mathrm{Gam}(\lambda | a, b) d\lambda \ . \qquad \text{(C.19)}$$

The geometric mean (C.19) is simplified by using the mean (A.11) and geometric mean (A.12) of a Gamma distribution, resulting in the analytic expression:

$$b(y) = \frac{1}{y!} \exp \Big(y \left(\psi(a) - \log(b) \right) - \frac{a}{b} \Big) \ . \qquad \text{(C.20)}$$

C.3 Gaussian-Gamma Conjugate Posterior Update: Weighted Regression for AR Models

In this section the posterior $q(w, \tau)$ of the AR model (Def 3.1) parameters $w \in \mathbb{R}^d$ and precision $\tau > 0 \in \mathbb{R}$ are calculated using a Gaussian-Gamma (A.14) prior: $\mathrm{p}\big(w|\tau\big)p(\tau)$ and conjugate Gaussian likelihood: $p\left(y_k, x_k | w, \tau\right)$. The likelihoods are weighted, with each data point y_k having associated

weight r_k. The form of the AR model log posterior is given by:

$$\log q(w,\tau) = \log p\left(w|\tau\right) + \log p(\tau) + \sum_{k=1}^{T} r_k \log p\left(y_k, x_k|w,\tau\right) + C \tag{C.21}$$

For generality the prior will be assumed to have parameters $w_0 \in \mathbb{R}^d, A_0 \in \mathbb{R}^{d\times d}, a_0 \in \mathbb{R}, b_0 \in \mathbb{R}$:

$$p(w,\tau) = \mathcal{N}\left(w|w_0, (\tau A_0)^{-1}\right) \mathrm{Gam}\left(\tau|a_0, b_0\right) \ , \tag{C.22}$$

furthermore A_0 is positive definite precision matrix and has the Schur decomposition $A_0 = U_0^T U_0$. For completeness the (Gaussian) form of the data likelihood is recalled to be:

$$p\left(y_k, x_k|w,\tau\right) = \left(\frac{\tau}{2\pi}\right)^{1/2} \exp\left(-\frac{\tau}{2}(w^T x_k - y_k)^2\right) \ , \tag{C.23}$$

where $y_k \in \mathbb{R}$ has associated regressor $x_k \in \mathbb{R}^d$. The following matrices are now defined for $k = 1, ..., T$:

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_T^T \end{bmatrix} \in \mathbb{R}^{T\times d}, \quad \begin{bmatrix} y_1 \\ \vdots \\ y_T \end{bmatrix} \in \mathbb{R}^{T\times 1}, \quad R = \begin{bmatrix} r_1 & & \\ & \ddots & \\ & & r_T \end{bmatrix} \in \mathbb{R}^{T\times T} \ . \tag{C.24}$$

The posterior (C.21) is reformulated in terms of the defined matrices (C.24):

$$\log q(w,\tau) = \frac{d}{2}\log\tau - \frac{\tau}{2}(w-w_0)^T A_0 (w-w_0) + (a_0 - 1)\log\tau - b_0\tau \tag{C.25}$$

$$+ \frac{1}{2}\sum_{k=1}^{T} r_k \log\tau - \frac{\tau}{2}(Y - Xw)^T R (Y - Xw) + C \ , \tag{C.26}$$

where the terms in line (C.25) are derived from the prior (C.22), the terms in (C.26) are from the likelihood (C.23), and C is a constant. All terms in the log posterior (C.26) and (C.25) that depend on the regressor parameters w are grouped and the following identity is noted:

$$-\frac{\tau}{2}(w-w_0)^T A_0 (w-w_0) - \frac{\tau}{2}(Y - Xw)^T R (Y - Xw) = -\frac{\tau}{2}(v - Lw)^T (v - Lw) \ , \tag{C.27}$$

where the following matrices have been defined:

$$v = \begin{bmatrix} R^{\frac{1}{2}} y \\ U_0 w_0 \end{bmatrix}, \quad L = \begin{bmatrix} R^{\frac{1}{2}} X \\ U_0 \end{bmatrix}, \quad R^{\frac{1}{2}} = \begin{bmatrix} \sqrt{r_1} & & \\ & \ddots & \\ & & \sqrt{r_T} \end{bmatrix} \ . \tag{C.28}$$

Furthermore, the following completed-square identity is proposed:

$$(v - Lw)^T(v - Lw) = (v - L\hat{w})^T(v - L\hat{w}) + (w - \hat{w})^T L^T L(w - \hat{w}) \tag{C.29}$$

where $\hat{w}$ is the pseudo inverse:

$$\hat{w} = (L^T L)^{-1} L^T v \tag{C.30}$$

Equation (C.29) can be verified by expanding the RHS of (C.29), and substituting $\hat{w}^T L^T L\hat{w} = v^T L\hat{w}$, which is a direct result of the pseudo inverse definition (C.30). The posterior (C.21) is then simplified using the relations (C.27), (C.29) and matrices (C.28):

$$\log q(w, \tau) = \frac{d}{2}\log\tau + (a_0 - 1)\log\tau - b_0\tau + \left(\frac{1}{2}\sum_{k=1}^{T} r_k\right)\log\tau \tag{C.31}$$

$$- \frac{\tau}{2}(w - \hat{w})^T L^T L(w - \hat{w}) + -\frac{\tau}{2}(v - L\hat{w})^T(v - L\hat{w}) + C \ . \tag{C.32}$$

The posterior (C.31) can be verified to be the following Gaussian-Gamma distribution by equating terms:

$$q(w, \tau) = \mathcal{N}\left(w|\hat{w}, (\tau A_T)^{-1}\right) \mathrm{Gam}\left(\tau|a_T, b_T\right) \ , \tag{C.33}$$

such that:

$$\hat{w} = (L^T L)^{-1} L^T v \qquad = (X^T RX + A_0)^{-1}(X^T RY + A_0 w_0) \tag{C.34}$$

$$A_T = L^T L \qquad = X^T RX + A_0 \tag{C.35}$$

$$a_T = a_0 + \frac{1}{2}\sum_{k=1}^{T} r_k \tag{C.36}$$

$$b_T = b_0 + \frac{1}{2}(v - L\hat{w})^T(v - L\hat{w}) \qquad = \begin{array}{l} b_0 + \frac{1}{2}(w_0 - \hat{w})^T A_0(w_0 - \hat{w}) \\ + \frac{1}{2}(y - X\hat{w})^T R(y - X\hat{w}) \end{array} \tag{C.37}$$

C.4 Poisson Point Process Conjugate Posterior Update

In this section the posterior $q(\lambda)$ of the stationary Poisson point process model (Def. 3.2) with firing rate $\lambda \geq 0$ is calculated for a Gamma prior $\mathrm{p}(\lambda)$. The form of the point process posterior is:

$$\log q(\lambda) = \sum_{k=1}^{T} r_k \log p\left(y_k|\lambda\right) + \log p(\lambda) + C \ , \tag{C.38}$$

where C is a constant, and the likelihood is a Poisson distribution:

$$p\left(y_k|\lambda\right) = \frac{\lambda^{y_k} e^{-\lambda}}{y_k!} \ . \tag{C.39}$$

Given that the prior of the firing rate λ is a gamma distribution $\mathrm{Gam}(\lambda_i|a_0, b_0)$ equation (C.38) can be written:

$$\log q^{\star}(\lambda_i) = \sum_{k=1}^{T} r_k \log \frac{\lambda^{y_k} e^{-\lambda}}{y_k!} + (a_0 - 1)\log\lambda - b_0\lambda + C \ . \tag{C.40}$$

Equation (C.40) is simplified:

$$\log q(\lambda) = \left(a_0 + \sum_{k=1}^{T} y_k r_k - 1\right)\log\lambda - \left(b_0 + \sum_{k=1}^{T} r_k\right)\lambda \ , \tag{C.41}$$

and now (C.41) is the same form as a Gamma distribution: $\mathrm{Gam}\big(\lambda|a_T, b_T\big)$ where:

$$a_T = a_0 + \sum_{k=1}^{T} y_k r_k \tag{C.42}$$

$$b_T = b_0 + \sum_{k=1}^{T} r_k \ . \tag{C.43}$$

Bibliography

[1] I. Pappas, M. Popovic, T. Keller, V. Dietz, and M. Morari, "A reliable gait phase detection system," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 9, pp. 113–125, June 2001.

[2] L. Goncalves, E. D. Bernardo, and P. Perona, *Seeing, Thinking and Knowing*, ch. Movemes for Modeling Biological Motion Perception, pp. 143–170. Springer Netherlands, 2004.

[3] C. Tomlin, J. Lygeros, and S. Sastry, "Synthesizing controllers for nonlinear hybrid systems," *Lecture Notes in Computer Science*, vol. 1386, p. 360, 1998.

[4] J. Roll, A. Bemporad, and L. Ljung, "Identification of piecewise affine systems via mixed-integer programming," *Automatica*, vol. 40, pp. 37–50, January 2004.

[5] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino, "A greedy approach to identification of piecewise affine models," *Lecture Notes in Computer Science*, vol. 2623, p. 97, 2003.

[6] R. Vidal, S. Soatto, Y. Ma, and S. Sastry, "An algebraic geometric approach to the identification of a class of linear hybrid systems," in *Proc. of IEEE Conference on Decision and Control*, 2003.

[7] A. Juloski, S. Weiland, and W. Heemels, "A Bayesian approach to identification of hybrid systems," *Proceedings of the IEEE Conference on Decision and Control*, vol. 1, pp. 13–19, 2004.

[8] A. Juloski, G. Ferrari-Trecate, R. Vidal, S. Paoletti, and J. Niessen, "Comparison of four procedures for the identification of hybrid systems," *Lecture Notes in Computer Science*, vol. 3414, pp. 354–369, 2005.

[9] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Royal Statistical Soc. B*, vol. 39, pp. 185–197, 1977.

[10] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, 1989.

[11] B. Pesaran, S. Musallam, and R. Andersen, "Cognitive neural prosthetics," *Current Biology*, vol. 16, pp. 77–80, 2006.

[12] M. A. L. Nicolelis, "Brainmachine interfaces to restore motor function and probe neural circuit," *Nature Reviews Neuroscience*, vol. 4, pp. 417–422, 2003.

[13] K. Shenoy, D. Meeker, S. Cao, S. Kureshi, B. Pesaran, C. Buneo, A. Batista, P. Mitra, J. Burdick, and R. Andersen, "Neural prosthetic control signals from plan activity," *Neuroreport*, vol. 14, no. 4, pp. 591–596, 2003.

[14] J. N. Turner, W. Shain, D. H. Szarowski, M. Andersen, S. Martins, M. Isaacson, and H. Craighead, "Cerebral astrocyte response to micromachined silicon implants," *Experimental Neurology*, vol. 156, no. 1, pp. 33–49, 1999.

[15] H. Scherberger, M. Jarvis, and R. Andersen, "Cortical local field potential encodes movement intentions in the posterior parietal cortex," *Neuron*, vol. 46, p. 347354, 2005.

[16] S. Musallam, B. D. Corneil, B. Greger, H. Scherberger, and R. A. Andersen, "Cognitive control signals for neural prosthetics," *Science*, vol. 305, pp. 258–262, 2004.

[17] L. Srinivasan, U. T. Eden, S. K. Mitter, and E. N. Brown, "General-purpose filter design for neural prosthetic devices," *Journal Neurophysiology*, vol. 98, pp. 2456–2475, 2007.

[18] O. Costa, M. Fragoso, and R. Marques, "Discrete-time Markov jump linear systems," in *Probability and its Applications*, Springer, 2005.

[19] M. W. Hofbaur and B. C. Williams, "Mode estimation of probabilistic hybrid systems," *Lecture Notes in Computer Science : Hybrid Systems: Computation and Control: 5th International Workshop, HSCC 2002, Stanford, CA, USA, March 25-27, 2002. Proceedings*, pp. 253–, 2002.

[20] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[21] M. Archer and C. Heitmeyer, "Verifying hybrid systems modeled as timed automata: A case study," in *Proceedings of the International Workshop on Hybrid and Real-Time Systems (HART'97)* (O. Maler, ed.), vol. 1201, (Grenoble, France), pp. 171–185, Springer-Verlag, 1997.

[22] T. Jaakkola and M. Jordan, "Bayesian parameter estimation through variational methods," *Statistics and Computing*, vol. 10, pp. 25–37, 2000.

[23] M. J. Beal and Z. Ghahramani, "The variational bayesian em algorithm for incomplete data: with application to scoring graphical model structures," *Bayesian Statistics*, vol. 7, 2002.

[24] C. Bishop, *Pattern Recognition and Machine Learning*. Information Science and Statistics, Springer, 2006.

[25] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer, 2004.

[26] C. E. McCulloch, "Generalized linear models," *Journal of the American Statistical Association*, vol. 95, pp. 1320–1324, Dec. 2000.

[27] J. P. Hoffmann, *Generalized Linear Models: an Applied Approach*. Pearson Education, 2004.

[28] W. Heemels, B. D. Schutter, and A. Bemporad, "Equivalence of hybrid dynamical models," *Automatica*, vol. 37, pp. 1085–1091, 2001.

[29] S. Weiland, A. L. Juloski, and B. Vet, "On the equivalence of switched affine models and switched arx models," in *Proceedings of the 45th IEEE Conference on Decision & Control*, 2006.

[30] J. Ferguson, "Variable duration models for speech," *In Proc. of the Symposium on the Application of Hidden Markov Models to Text and Speech*, pp. 143–179, 1980.

[31] S. Levinson, "Continuously variable duration hidden Markov models for speech analysis," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 11, pp. 1241–1244, Apr 1986.

[32] P. Ramesh and J. Wilpon, "Modeling state durations in hidden Markov models for automatic speech recognition," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 381–384, Mar 1992.

[33] B. Sin and J. H. Kim, "Nonstationary hidden Markov model," *Signal Processing*, vol. 46, pp. 31–46, 1995.

[34] S. Vaseghi, "State duration modelling in hidden Markov models," *Signal Processing*, vol. 41, pp. 31–41, 1995.

[35] D. MacKay, "Ensemble learning for hidden Markov models," 1997. `http://www.inference.phy.cam.ac.uk/mackay/abstracts/ensemblePaper.html`.

[36] M. Beal, *Variational algorithms for approximate bayesian inference*. PhD thesis, The Gatsby Computational Neuroscience Unit, University College London, 2003.

[37] C. Robert, G. Celeux, and J. Diebolt, "Bayesian estimation of hidden Markov chains: A stochastic implementation," *Statistics & Probability Letters*, vol. 16, pp. 77–83, January 1993.

[38] P. M. Djuric and J.-H. Chun, "An MCMC sampling approach to estimation of nonstationary hidden Markov models," *IEEE Transactions on Signal Processing*, vol. 50, pp. 1113–1123, May 2002.

[39] D. MacKay and L. Peto, "A hierarchical Dirichlet language model," *Natural Language Engineering*, vol. 1, no. 3, pp. 1–19, 1995.

[40] D. MacKay, "Comparison of approximate methods for handling hyperparameters," *Neural Computation*, vol. 11, no. 5, pp. 1035–1068, 1999.

[41] C. Bishop and M. E. Tipping, *Advances in Learning Theory: Methods, Models and Applications*, ch. Bayesian Regression and Classification. IOS Press, 2003.

[42] D. J. C. MacKay, "Probable networks and plausible predictions a review of practical bayesian methods for supervised neural networks," *Network: Computation in Neural Systems*, vol. 6, pp. 469–505, 1995.

[43] S. M. Oh, J. M. Rehg, and F. Dellaert, "Parameterized duration modeling for switching linear dynamic systems," *IEEE International Conference on Computer Vision and Pattern Recognition*, 2006.

[44] J. L. Gould and C. G. Gould, *The Honey Bee*. Scientific American Library, 1988.

[45] S. N. Simic, K. H. Johansson, S. Sastry, and J. Lygeros, "Towards a geometric theory of hybrid systems," in *Hybrid Systems: Computation and Control*, pp. 421–436, 2000.

[46] F. Zhao, X. Koutsoukos, H. Haussecker, J. Reich, and P. Cheung, "Distributed monitoring of hybrid systems: A model-directed approach," in *IJCAI*, 2001.

[47] H. Blom and J. Lygeros, eds., *Stochastic Hybrid Systems: Theory and Safety Critical Applications (Lecture Notes in Control and Information Sciences)*. Springer-Verlag, 2006.

[48] J. K. Tugnait, "Detection and estimation for abruptly changing systems," *Automatica*, vol. 18, pp. 607–615, September 1982.

[49] A. Blake, B. North, and M. Isard, "Learning multi-class dynamics," in *Advances in neural information processing systems II*, (Cambridge, MA, USA), pp. 389–395, MIT Press, 1999.

[50] H. A. Blom and E. A. Bloem, "Exact bayesian and particle filtering of stochastic hybrid systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, pp. 55–70, 2007.

[51] A.-V. Rosti and M. Gales, "Rao-blackwellised gibbs sampling for switching linear dynamical systems," in *ICASSP*, vol. 1, pp. 809–812, 2004.

[52] A. Doucet, A. Logothetis, and V. Krishnamurthy, "Stochastic sampling algorithms for state estimation of jump Markov linear systems," *IEEE Transactions On Automatic Control*, vol. 45, pp. 188–202, 2000.

[53] Z. Ghahramani and G. E. Hinton, "Variational learning for switching state-space models," *Neural Computation*, vol. 12, no. 4, pp. 831–864, 2000.

[54] F. Laroussinie, N. Markey, , and P. Schnoebelen, "Model checking timed automata with one or two clocks," in *Concurrency Theory (CONCUR04)*, vol. 3170 of LNCS, pp. 387–401, 2004.

[55] M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston, "Performance analysis of probabilistic timed automata using digital clocks," *Form. Methods Syst. Des.*, vol. 29, no. 1, pp. 33–78, 2006.

[56] L. de Alfaro, "Temporal logics for the specification of performance and reliability," in *14th An. Symp. on Theor. Aspects of Comp. Sci. (STACS97),*, pp. 165–176, Springer, 1997.

[57] G. I. Lopez, H. Hermanns, , and J.-P. Katoen, "Beyond memoryless distributions: Model checking semi-Markov chains," *Proceedings of Process Algebra and Probabilistic Methods. Performance Modeling and Verification. Joint International Workshop (PAPM-PROBMIV 2001)*, vol. 2165, pp. 57–70, 2001.

[58] L. Ljung, *System Identification, Theory for the User*. Prentice Hall, second edition ed., 1999.

[59] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari, "A clustering technique for the identification of piecewise affine systems," *Automatica*, vol. 39, pp. 205–217, February 2003.

[60] A. Juloski, W. Heemels, and G. Ferrari-Trecate, "Data-based hybrid modelling of the component placement process in pick-and-place machines," *Control Engineering Practice*, vol. 12, pp. 1241–1252, 2004.

[61] E. Munz, T. Hodrus, and V. Krebs, "Gradient-based identification of hybrid systems," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 10, pp. 25–40, 2004.

[62] K. Bennett and O. L. Mangasarian, "Multicategory discrimination via linear programming," *Optimization Methods and Software*, vol. 3, pp. 27–39, 1993.

[63] E. J. Bredensteiner and K. P. Bennett, "Multicategory classification by support vector machines," *Comput. Optim. Appl.*, vol. 12, no. 1-3, pp. 53–79, 1999.

[64] N. Hudson and J. Burdick, "Learning hybrid system models for supervisory decoding of discrete state, with applications to the parietal reach region," in *3rd International IEEE/EMBS Conference on Neural Engineering*, pp. 587–592, May 2007.

[65] N. Hudson and J. Burdick, "A stochastic framework for hybrid system identification with application to neurophysiological systems," *Lecture Notes in Computer Science : Hybrid Systems: Computation and Control*, pp. 568–582, 2007.

[66] O. Capp, E. Moulines, and T. Ryden, *Inference in Hidden Markov Models*. Springer Series in Statistics, 2005.

[67] M. Cassidy and P. Brown, "Hidden Markov based autoregressive analysis of stationary and non-stationary electrophysiological signals for functional coupling studies," *J Neurosci Methods*, vol. 116, pp. 35–53, 2002.

[68] A. Corduneanu and C. M. Bishop, "Variational bayesian model selection for mixture distributions," in *Artificial intelligence and statistics*, pp. 27–34, 2001.

[69] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[70] G. McLachlan and T. Krishnan, *The EM algorithm and its Extensions*. Wiley, 1997.

[71] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *The Annals of Mathematical Statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[72] A. E. Bryson, *Applied Linear Optimal Control Examples and Algorithms*. Cambridge University Press, 2002.

[73] S.-Z. Yu and H. Kobayashi, "Practical implementation of an efficient forward-backward algorithm for an explicit-duration hidden Markov model," *IEEE Transactions on Signal Processing*, vol. 54, pp. 1947–1951, May 2006.

[74] C. E. McCulloch, "Maximum likelihood algorithms for generalized linear mixed models," *Journal of the American Statistical Association*, vol. 92, pp. 162–170, Mar. 1997.

[75] P. Dellaportas and A. F. M. Smith, "Bayesian inference for generalized linear and proportional hazards models via gibbs sampling," *Applied Statistics*, vol. 42, no. 3, pp. 443–459, 1993.

[76] W. R. Gilks, N. G. Best, and K. K. C. Tan, "Adaptive rejection metropolis sampling within gibbs sampling," *Applied Statistics*, vol. 44, no. 4, pp. 455–472, 1995.

[77] T. S. Jaakkola, "Tutorial on variational approximation methods," in *Advanced Mean Field Methods: Theory and Practice*, pp. 129–159, MIT Press, 2000.

[78] W. Truccolo, U. T. Eden, M. R. Fellows, J. P. Donoghue, and E. N. Brown, "A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects," *J Neurophysiol*, vol. 93, no. 2, pp. 1074–1089, 2005.

[79] S.-Z. Yu and H. Kobayashi, "An efficient forward-backward algorithm for an explicit-duration hidden Markov model.," *IEEE Signal Processing Letters*, vol. 10, pp. 11–14, 2003.

[80] M. T. Johnson, "Capacity and complexity of HMM duration modeling techniques," *IEEE Signal Processing Letters*, vol. 12, pp. 407–410, 2005.

[81] K. P. Murphy, "Hidden semi-Markov models (HSMMs)." November 2002.

[82] D. MacKay, "Choice of basis for laplace approximation," *Machine Learning*, vol. 33, pp. 77–86, 1998.

[83] G. C. Cawley, N. L. Talbot, and M. Girolami, "Sparse multinomial logistic regression via bayesian l1 regularisation," in *Advances in Neural Information Processing Systems 19* (B. Schölkopf, J. Platt, and T. Hoffman, eds.), pp. 209–216, Cambridge, MA: MIT Press, 2007.

[84] D. MacKay, "The evidence framework applied to classification networks," *Neural Computation*, vol. 4, pp. 720–736, 1992.

[85] J. Hoeting, D. Madigan, A. Raftery, and C. Volinsky, "Bayesian model averaging," *Statistical Science*, vol. 14, pp. 382–401, 1999.

[86] M. Muto and J. L. Beck, "Bayesian updating and model class selection for hysteretic structural models using stochastic simulation," *Journal of Vibration and Control*, vol. 14, pp. 7–34, 2008.

[87] J. L. Beck and K. Yuen, "Model selection using response measurements: Bayesian probabilitic approach," *Journal of Engieering Mechanics*, vol. 130, pp. 192–203, 2004.

[88] M. T. Wolf, *Target Tracking Using Clustered Measurements, with Applications to Autonomous Brain-Machine Interfaces*. PhD thesis, California Institute of Technology, 2008.

[89] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, pp. 461–464, 1978.

[90] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, pp. 716–723, 1974.

[91] R. M. Neal, "Annealed importance sampling," *Statistics and Computing*, vol. 11, pp. 125–139, April 2001.

[92] J. L. Beck and S.-K. Au, "Bayesian updating of structural models and reliability using Markov chain monte carlo simulation," *Journal of engineering mechanics*, vol. 128, pp. 380–391, 2002.

[93] C. Ritter and M. A. Tanner, "Facilitating the gibbs sampler: The gibbs stopper and the griddy-gibbs sampler," *Journal of the American Statistical Association*, vol. 87, no. 419, pp. 861–868, 1992.

[94] S. Cheung and J. Beck, "On using posterior samples for model selection for structural identification," in *Asian-Pacific Symposium on Structural Reliability and its Applications*, (Hong Kong), June 2008.

[95] S. Chib, "Marginal likelihood from the gibbs output," *Journal of the American Statistical Association*, vol. 90, no. 432, pp. 1313–1321, 1995.

[96] S. Chib and I. Jeliazkov, "Marginal likelihood from the metropolis-hastings output," *Journal of the American Statistical Association*, vol. 96, no. 453, pp. 270–281, 2001.

[97] A. E. Gelfand and D. K. Dey, "Bayesian model choice: Asymptotics and exact calculations," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 56, no. 3, pp. 501–514, 1994.

[98] M. A. Newton and A. E. Raftery, "Approximate bayesian inference with the weighted likelihood bootstrap," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 56, no. 1, pp. 3–48, 1994.

[99] W. Penny and S. Roberts, "Bayesian multivariate autoregressive models with structured priors," *Vision, Image and Signal Processing, IEE Proceedings*, vol. 149, pp. 33–41, 2002.

[100] A. Jasra, C. C. Holmes, and D. A. Stephens, "Markov chain monte carlo methods and the label switching problem in bayesian mixture modeling," *Statistical Science*, vol. 20, pp. 50–67, 2005.

[101] L. Spezia, "Reversible jump and the label switching problem in hidden Markov models," *Journal of Statistical Planning and Inference*, vol. In Press, 2008. In Press.

[102] J. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, eds., *Advances in Learning Theory: Methods, Model and Applications*. IOS Press, 2003.

[103] S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert, "Learning and inferring motion patterns using parametric segmental switching linear dynamic systems," tech. rep., GVU Center, College of Computing, Georgia Institute of Technology, Atlanta, GA, U.S.A., 2006.

[104] C. Kemere, B. Yu, G. Santhanam, S. Ryu, A. Afshar, T. Meng, and K. Shenoy, "Hidden Markov models for spatial and temporal estimation for prosthetic control," in *Society for Neuroscience, Abstract Viewer / Itinerary Planner*, (Atlanta, GA), 2006. In press.

[105] W. Wu, M. Black, D. Mumford, Y. Gao, E. Bienenstock, and J. Donoghue, "Modeling and decoding motor cortical activity using a switching kalman filter," *IEEE Transactions on Biomedical Engineering*, vol. 51, pp. 933–942, June 2004.

[106] E. Stark and M. Abeles, "Predicting movement from multiunit activity," *The Journal of Neuroscience*, vol. 27, pp. 8387–8394, 2007.

[107] B. Pesaran, J. S. Pezaris, M. Sahani, P. P. Mitra, and R. A. Andersen, "Temporal structure in neuronal activity during working memory in macaque parietal cortex," *Nature Neuroscience*, vol. 5, pp. 805 – 811, 2002.

[108] D. B. Percival and A. T. Walden, *Spectral analysis for physical applications: multitaper and conventional univariate techniques*. Cambridge University Press, 1993.

[109] K. Blinowska, L. Czerwosz, W. Drabik, P. Franaszczuk, and H. Ekiert, "Eeg data reduction by means of autoregressive representation and discriminant analysis procedures," *Electroencephalography and Clinical Neurophysiology*, vol. 6, pp. 650–658, 1981.

[110] P. Madhavan, B. Stephens, D. Klingberg, and S. Morzorati, "Analysis of rat eeg using autoregressive power spectra," *Journal of Neuroscience Methods*, vol. 40, pp. 91–100, 1991.

[111] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press, 2001.