Change Point Modeling and Regime Learning in Markov Switching Systems

Modeling the Effects of Students' Interactions with Immersive Simulations

A thesis presented

by

Nicholas S. Hoernle

to

Institute for Applied Computational Sciences

in partial fulfillment of the requirements

for the degree of

Master of Engineering

in the subject of

Computational Science and Engineering

Harvard University

Cambridge, Massachusetts

May 2018

Thesis Advisors:
Dr. Pavlos Protopapas
Prof. Barbara Grosz
External Advisor:
Dr. Kobi Gal

Author:
Nicholas S. Hoernle

Change Point Modeling and Regime Learning in Markov Switching Systems

Abstract

Simulations that combine real world components with interactive digital media provide a rich setting for students with the potential to assist knowledge building and understanding of complex physical processes. This thesis addresses the problem of modeling the effects of multiple students' simultaneous interactions on the complex and exploratory environments such simulations provide. We work towards assisting educators with the difficult task of interpreting student exploration and we direct this research by considering the specific case of Connected Worlds, an innovative mixed reality simulation.

We introduce switch based models and propose the switching state space model as a good candidate for decomposing the time series state log data that result from a session with Connected Worlds. The model breaks the time series into periods that individually are easily interpreted. This work presents an inference algorithm that learns the transition points between successive periods in the time series as well as the internal dynamics that govern each period. The algorithm differs from other switch based inference schemes in that its primary goal is to decompose the time series in a way that is human interpretable.

The model was applied to data that was obtained from Connected Worlds and it was used to segment sessions into multiple periods. Inferred parameters associated with the

individual periods were used to generate text-based descriptions of the system dynamics. Two experimental visualizations of the model output were designed. These visualizations included the generated text-based descriptions and a movie representation of the system dynamics. We designed two experiments to test the interpretability of the inferred periods and the text descriptions of the dynamics that were present in the associated video clip. Our results indicate that the switch based model finds relevant boundaries between salient periods and it generates representative text that describes the period dynamics.

Contents

List of Figures

Acknowledgments

Thank you to the Oppenheimer Memorial Trust, the Harvard IACS, the NSF and my family for helping to make this study period financially feasible.

To my parents, for always being supportive of my endeavours no matter how far they may take me from home. Thank you for your comments, input and the encouragement you have given in reading my work and throughout my time at Harvard. Doug, Niki and Ali, thank you for the constant support and for the interest that you have shown in the work that I do.

Barbara, thank you for the time and effort that you have committed to this project, it has been a privilege to work with you. To Andee and Leilah, thank you for accepting me to work on the ESSIL project. Your insights and passion for these education tasks are inspiring.

To Kobi, for all of the time that you have spent mentoring me over this past year. I appreciate your input and I am excited about having the opportunity to continue working with you at Edinburgh.

Pavlos, you have been an inspiration throughout my time at Harvard. Thank you for being a thoughtful mentor as well as a supportive friend. I look forward to following the progress of the CSE Program as it continues to thrive under your leadership.

Introduction

Complex systems simulations are becoming increasingly common in formal and informal STEM learning environments (Smørdal *et al.*, 2012). These simulations present scientific phenomena in a manner that connects pedagogical learning outcomes to the firsthand experience of real-world outcomes. However, the open-ended and exploratory nature of these simulations presents challenges to teachers' understanding of students' learning. Students' actions have both immediate and long-term effects on a simulation, leading to a rich array of potential outcomes. Teachers may wish to lead discussions on students' interactions to highlight salient learning opportunities, but if there are too many 'moving parts' to the simulation, this becomes a challenging ideal.

This thesis presents an investigation into and a case study for using artificial intelligence techniques to extract useful information from the log files of exploratory learning environments. We study an exploratory learning environment that simulates an ecosystem and aims to teach students about systems, systems thinking and how their actions can have possible long term and spatially removed effects on the system. The system states that result from students' interactions with the simulation are logged and form a time series of data. This time series represents the dynamical responses of the complex system to the group specific interactions. Our hypothesis is: *complex time series log data can be decomposed into periods that individually are readily understood*. Together these periods explain the dynamics that ensued, but individually they might explain salient fragments or examples of the larger causal cycles at play.

We study methods for extracting salient periods from the log files that are generated by complex exploratory learning environments. Switching state space models (Ghahramani and Hinton, 2000) are a class of model for time series data where the parameters controlling a linear dynamic system switch according to a discrete latent process. These models have been used in a wide variety of domains including control (Ikoma *et al.*, 2002), statistics (Cappé *et al.*, 2009), econometrics (Giordani *et al.*, 2007) and signal processing (Kim *et al.*, 1999). They combine hidden Markov and state space models to capture *regime* switching in non-linear time series data (Whiteley *et al.*, 2010). The intuition is that a system evolves over time but may undergo a regime change that results in an intrinsic shift in the system's characteristics. Allowing for discrete points in time where the dynamics change enables simple linear models to capture more complicated and non-linear dynamics. We propose that regime switching models also help to increase the *interpretability* of large and complex systems by automatically segmenting a time series into regions of approximately uniform dynamics. The result is that a complex session is broken into smaller periods that are more readily understood when one reflects upon the session.

The switch based model formalizes a procedure for decomposing a time series into coherent periods broken by the change points between those periods. However, inference in these models is extremely challenging. The tightly coupled nature of the regime parameters and the locations of switch points presents a computationally intractable problem. We design a novel inference algorithm for learning the parameters that are associated with the unknown individual regimes. Given the regime specific parameters, inference over the change points can be conducted using standard techniques from time series modeling.

Our eventual goal is to automatically generate relevant summaries of the system dynamics such that teachers can effectively engage students in discussions that stem from their own experiences with the simulation. To this end, we investigate the human interpretability of the resulting periods that are identified by the discussed inference algorithm. We use regime specific parameters to generate a short description of the dynamics that were present

in a given period. We conduct two validation experiments to evaluate the extent to which the inferred switch points and the associated regime descriptions are comprehensible to external human validators.

The thesis proceeds as follows. In Chapter 1, we introduce Connected Worlds, the exploratory learning environment that is used as a case study for this investigation. Chapter 2 gives a detailed description of the switching state space model. This chapter elaborates on hidden Markov and state space models and presents the switch models that combine these two baselines. In addition, Chapter 2 discusses the past work for performing maximum likelihood estimation of the parameters that are associated with the switching state space model. Lastly, this chapter describes the inference algorithm that we use for extracting salient periods from the Connected Worlds log data. Chapter 3 describes the evaluation of the inference algorithm. We generate synthetic data and test the algorithm's ability to correctly detect change points in a given time series. We then design two experiments that aim to test the coherence and interpretability of the model's output. Finally, Chapter 4 presents two avenues for future research. Work in Bayesian non-parametrics offer exciting possibilities for applications in these domains as the number of periods is learned by the model. Moreover, future work will need to design and evaluate tools that incorporate such information and are appropriate for classroom implementation.

Chapter 1

Connected Worlds

1.1 Description of Connected Worlds

1.1.1 Introduction to Connected Worlds

Connected Worlds[1] (CW) is a multi-person ecology simulation with the goal of teaching students about complex systems and systems thinking. It consists of an immersive environment comprising four interconnected biomes and a central flow of water that is fed by a waterfall. Students plant trees which flourish or die, animals arrive or depart, and rain clouds form, move through the sky and deposit rain into the waterfall. All the while, students direct the water stream to different areas in the simulation to provide enough water to sustain plant and animal life. The simulation exhibits large scale feedback loops and presents the opportunity for participants to experience how their actions can have (often unintended) effects that are significantly removed in time and/or space.

Students interact with CW by positioning logs to control the direction of water that flows in the simulation and by planting trees in the different biomes. Water can be directed

[1] https://nysci.org/home/exhibits/connected-worlds/

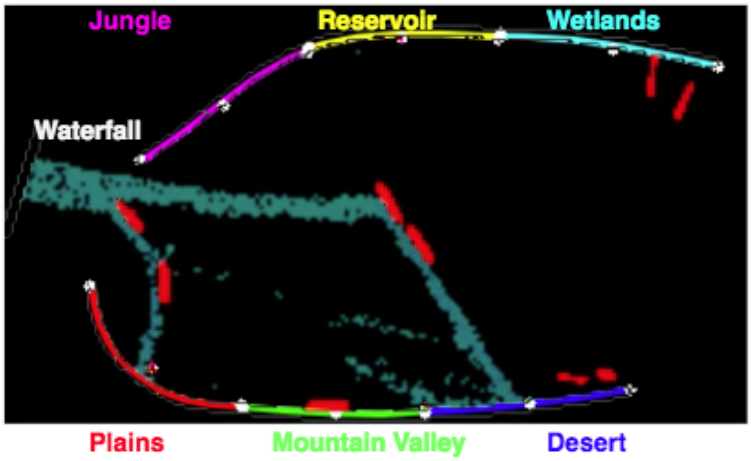


Figure 1.1: *Bird's eye view of the CW simulation. Biomes are labeled on the perimeter and logs appear as thick red lines. Water enters via the waterfall and in this image it is mainly flowing from left to right toward the Desert and the Plains.*

to each of the four biomes (Desert, Plains, Jungle and Wetlands) and the distribution of flowing water depends on the placement of the logs. Water on the floor of the simulation (which represents a flood plain) is a usable resource as it can be directed to the biomes. Certain sources of water are under the students' direct control. Students can actively release water into the system from the water that is stored in the Reservoir or from surplus water that is present in a biome. Rainfall events are out of the students' control and these release water into the Waterfall (to replenish the primary source of water) and into the individual biomes. The Mountain Valley can also receive rain and it forms a river source when this happens. Figure 1.1 is a snapshot showing a bird's eye view of the state of logs and water in the CW simulation. Not seen in this snapshot are the plant and animal counts or levels in the biomes. The volume of water that is stored in each biome is also not shown.

1.1.2 Water Cycle Simulation

Every biome in CW shares the common water resource. The amount of water for a given simulation is pre-defined, thus it forms a *limited resource* that students are required to manage carefully. CW simulates a water cycle where water is directed to biomes and used

to sustain plants and animals. Evaporation, which is dependent on the numbers and levels of plants, and rainfall, which depends on the amount of evaporation, brings water back to the water sources where it once again becomes a usable resource.

Figure 1.2 displays a schematic of the water cycle that is present in CW. Water is available from three sources: the Waterfall, the Mountain Valley and the Reservoir. The Waterfall flows whenever water is available and thus the amount of input water from the Waterfall is out of the students' control. The Mountain Valley is a secondary source of water and it only flows when it rains in this area[2]. Students choose when to release water from the Reservoir and thus, apart from overflow events, this source of water is directly under their control. Overflow events result when it rains in the Reservoir and the water 'spills' out onto the simulation floor when the Reservoir is at full capacity. Students position logs on the floor of the simulation (see Figure 1.1) to direct some portion of water to some subset of the four biomes. Once a biome has water, the students can plant trees. Higher level trees require the presence of lower level trees. The number and level of trees affects the amount of water that a biome requires. Lastly, when water is present in a biome, students can pipe water out of the biome by placing a log at that biome's wall (one of the purple, aqua, red or blue lines in Figure 1.1).

1.1.3 Plant-Animal Relations

Students plant trees and observe the arrival of animals that use the trees to support their habitat. Mimicking a real-world scenario, the fauna for a biome depend on the flora, and the flora are unique for a specific biome. In other words, each biome can support different plants and each type of plant attracts a certain type of animal. Larger and higher level plants require the presence of smaller and lower level plants. Often the animals that are supported by the top level (level 4) plants are the most aesthetically pleasing, and part of the goal for a

[2]System parameters allow the rain in the Mountain Valley to be turned off, stopping the flow from this area. The rain in the Mountain Valley has been disabled for the experiments presented in this study.

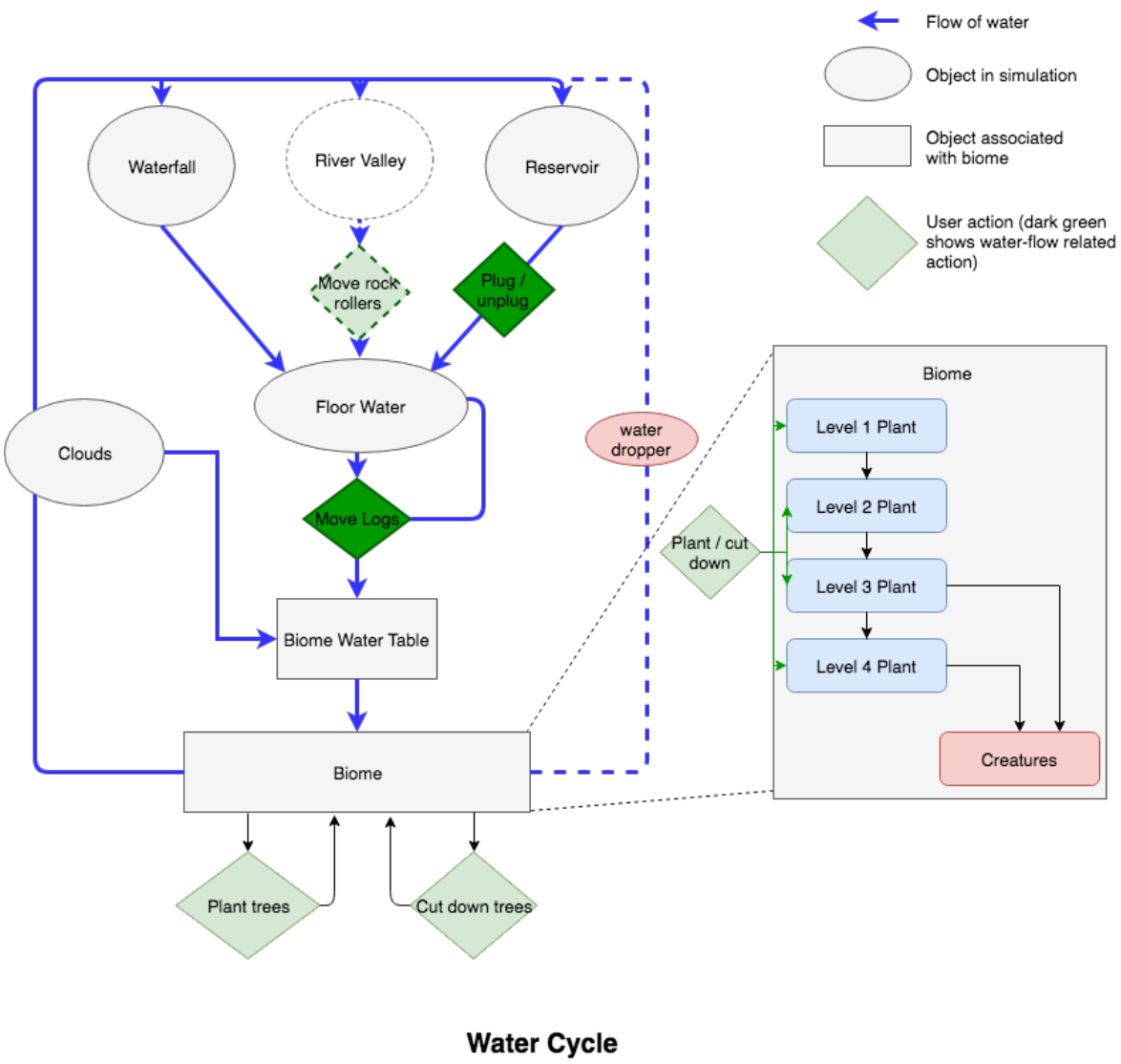


Figure 1.2: *Flow chart showing the water cycles that are present in CW. Blue solid lines represent main water flows, dotted lines represent other possible flows of water that often are not present or are negligible. Green, diamond boxes represent the actions that the students can take that will change the water cycles that are present. Gray circles are regions or objects in the simulation and gray boxes relate to one of the four biomes.*

given simulation might be to attract these animals. Students are thus required to manage their resources to support lower level plants and animals before achieving the top level in a specific biome. The plant and animal relationships are summarized in Appendix A.

1.2 Need for Assistive Technology

The nature of the simulation is complex on a variety of dimensions. The simulation involves a large number of students simultaneously executing actions that change the state of the environment. No one person - including the teacher or interpreter - can possibly follow all student actions and the resulting system changes, even in a relatively short simulation. Each participant may have a different view of what transpired, depending on the actions he or she took and the state changes that he or she noticed. Thus we have been developing tools to support teachers' understanding of the effects of students' interactions in complex exploratory learning environments such as CW. In Chapter 2 we present a model that aims to decompose a session with CW into periods that might assist a teacher in identifying salient learning opportunities from a class's specific session with CW.

Chapter 2

The Switching State Space Model

We introduce switch based models as a tool for segmenting a time series into periods that can individually be described by simpler dynamics. In this chapter, we describe switching models and related work that leads to the construction of these models. Thereafter, we propose a novel inference algorithm for learning the parameters that are associated with the switches and the resulting periods.

2.1 Switch Based Models

The switching state-space Model (SSSM) (Ghahramani and Hinton, 2000), or switching linear dynamical system (SLDS) (Fox *et al.*, 2009), captures nonlinear dynamical phenomena by considering discrete points in time where a system switches among conditionally linear dynamical modes. Our primary reason for exploring this model is to perform inference over the assignment of data to specific linear dynamical modes (or regimes), thereby characterizing periods that exhibit dynamics controlled by the regimes. The ease of interpretability of the conditionally linear regimes is an attractive merit of this model, while the presence of the discrete switch points allows the model to capture a wide variety of non-linear dynamics.

The SSSM has been used for prediction tasks (Fox *et al.*, 2007; Li and Jilkov, 2003) and also for inference tasks (Fox *et al.*, 2009; Jonsen *et al.*, 2007; Pavlovic *et al.*, 2001) where the primary goal was to discover latent structure in time series data. Oh *et al.* (2008) use the term 'learning' to describe the segmentation of the motion of honey bees into different behavioral regimes. They describe the identification of the parameters that characterize the motion within a regime as 'quantification'. Similarly, we aim to (1) 'learn' the periods that define salient regimes in the CW simulation and (2) 'quantify' these regimes for the purpose of generating a useful description of the system dynamics.

This work is broadly related to the field of mining information from time series data (Esling and Agon, 2012; Horst and Abraham, 2004). Alternative time series mining approaches are concerned with finding small sections or events that might be of relevance within the larger time series. We rather focus on decomposing the *entire* time series into periods that are coherent to a human observer. In contrast to the switching systems that we discuss, these other approaches have aimed to cluster short segments from time series data to identify certain patterns that might arise in a database (Vlachos *et al.*, 2003; Tanaka *et al.*, 2005; Patel *et al.*, 2002). This involves windowing a time series and defining wavelets, motifs (Patel *et al.*, 2002) or segments that are commonly found throughout the database. Preston *et al.* (2009) propose an approach to define and mine for events in a time series database.

The SSSM has been extensively applied to niche domains. We present a new avenue of study in which SSSMs are used to describe complex time series in a way that can be easily interpreted by people. Ghahramani and Hinton (2000) introduce and give a detailed presentation of the SSSM. Pavlovic *et al.* (2001) and Giordani *et al.* (2007) use switching models to capture non-linear behavior in a time series, applied to model human motion and econometrics respectively. SSSMs have also been applied in object tracking domains where it is necessary to predict the trajectory of multiple objects (Fox *et al.*, 2007). Whiteley *et al.* (2010) introduce a sequential Monte Carlo algorithm for inference over SSSMs using discrete particle filters. We propose an algorithm to perform posterior inference over the

latent variables in this model with the primary focus of interpretability of the model.

Before describing the SSSM in Section 2.3, we first provide a brief introduction to the linear state-space model for *continuous* time series data and the hidden Markov model for *discrete* time series data. The SSSM couples these two models and thus this background work is a necessary step toward constructing the SSSM.

2.2 Background Information

2.2.1 State Space Models and Hidden Markov Models

Hidden Markov models (HMM) and state-space models (SSM) define a joint probability density over a sequential collection of hidden state ($\mathbf{X}$) and observed ($\mathbf{y}$) random vectors (Ghahramani, 2001; Shumway and Stoffer, 2000). The HMM refers to the case when the states and observations have discrete values. For example at a time step t the hidden state $\mathbf{X_t}$ is represented by a categorical variable that can take one of M possible discrete values, where a value indexes a state. The associated observations $\mathbf{y_t}$ are discrete symbols where the observation probabilities $P(\mathbf{y_t} \mid \mathbf{X_t})$ can be fully specified by an $M \times K$ observation matrix, with K being the number of possible observation states. Given the hidden state at time t in an HMM, the associated observation is independent of all other observations. Moreover, the hidden states obey the Markov independence property: state $\mathbf{X_t}$ is conditionally independent from the previous states ($\mathbf{X_k}$ for $k \in 1, 2, \ldots, t-2$) and future states ($\mathbf{X_k}$ for $k \in t+2, \ldots, T$) given the value of state $\mathbf{X_{t-1}}$ and $\mathbf{X_{t+1}}$ respectively. The joint probability for the sequence of states and observations can therefore be factored as[3]:

$$P(\mathbf{X_{1:T}}, \mathbf{y_{1:T}}) = P(\mathbf{X_1})P(\mathbf{y_1} \mid \mathbf{X_1}) \prod_{t=2}^{T} P(\mathbf{X_t} \mid \mathbf{X_{t-1}})P(\mathbf{y_t} \mid \mathbf{X_t}) \tag{2.1}$$

[3]Here we use the notation $X_{1:t}$ to refer to all states $X_k, k \in [1, 2, \ldots, t]$

When the model is changed to allow for real-valued state and observation vectors, it is termed a state-space model. The simplest and most commonly used models that follow this structure assume that the transition and output functions are linear and time invariant and the distributions of the state and observation variables follow multivariate Gaussian distributions (Ghahramani and Hinton, 2000). The linear Gaussian SSM is defined by the tuple (A, C, Q, R) in:

$$\begin{aligned} \mathbf{X_t} &= A\mathbf{X_{t-1}} + w_t \\ \mathbf{y_t} &= C\mathbf{X_t} + v_t \end{aligned} \tag{2.2}$$

where A is the state transition matrix, $w_t \sim \mathcal{N}(0, Q)$ is the state transition noise, C is the output matrix and $v_t \sim \mathcal{N}(0, R)$ is the observation noise.

The problem of inference or state estimation for a SSM with known parameters consists of estimating the posterior probabilities of the hidden variables given a sequence of observed values. This inference problem can be broken into *filtering*, *smoothing* and *prediction* (Shumway and Stoffer, 2000). The goal of filtering is to use all the data up to time t to calculate the probability of the hidden state X_t. Smoothing, aims to use all of the data available from time $1 \dots T$ (with $T > t$) to calculate the probability of X_t. Lastly, prediction is calculating the probability of the future states X_{t+1} given all the data from time $1 \dots t$. A minor terminology note is that when we perform filtering and smoothing in the context of linear Gaussian state space models, the algorithms are termed Kalman filtering and smoothing respectively.

2.2.2 Forward Algorithm

The forward algorithm, also termed filtering, aims to calculate the probability that a certain state X_t adopts a specific value using only data that is available up to that point in time ($y_{1:t}$). The joint probability of the latent state and the previous data is:

$$
\begin{aligned}
P(X_t, y_{1:t}) &= \sum_{X_{t-1}} P(X_t, X_{t-1}, y_{1:t}) \\
&= P(y_t \mid X_t) \sum_{X_{t-1}} P(X_t \mid X_{t-1}) P(X_{t-1}, y_{1:t-1})
\end{aligned}
\tag{2.3}
$$

We have applied the Markov independence property where:

$$
\begin{aligned}
y_t \mid X_t &\perp\!\!\!\perp y_{1:t-1},\ X_{1:t-1} \\
X_t \mid X_{t-1} &\perp\!\!\!\perp y_{1:t-1},\ X_{1:t-2}
\end{aligned}
\tag{2.4}
$$

Note that the joint probability $P(X_{t-1}, y_{1:t-1})$ can be expressed in the same form as equation 2.3 but dependent on $P(X_{t-2}, y_{1:t-2})$. This defines an efficient recursive routine for calculating the probabilities for all states up to time t using the past data $y_{1:t}$. Note that $P(y_t \mid X_t)$ and $P(X_t \mid X_{t-1})$ are given by the transition and emission probabilities (or the emission likelihood in the SSM) and this recursion can be efficiently implemented using a dynamic programming implementation.

2.2.3 Backward Algorithm

When the data are available in an off-line setting, it is advantageous to use *future* values to help calculate the probability of the state at time t. The backward algorithm, also termed smoothing, achieves this by calculating the probability of X_t by using all available data $y_{1:T}$, with $t < T$. The backward recursion depends on the value of the forward filter and a new term that can be factored similarly to equation 2.3.

$$
\begin{aligned}
P(X_t, y_{1:T}) &= P(y_{1:t}, y_{t+1:T}, X_t) = P(y_{1:t}, y_{t+1:T} \mid X_t) P(X_t) \\
&= P(y_{t+1:T} \mid X_t)[P(y_{1:t} \mid X_t) P(X_t)]
\end{aligned}
\tag{2.5}
$$

$P(y_{t+1:T} \mid X_t)$ corresponds to calculating the probability of observing the future data given the current state (called backward values). $P(y_{1:t} \mid X_t)P(X_t) = P(y_{1:t}, X_t)$ is the probability from the forward algorithm in equation 2.3. We are now able to factor the backward term, again using the Markov independence property:

$$\begin{aligned} P(y_{t+1:T} \mid X_t) &= \sum_{X_{t+1}} P(y_{t+1:T}, X_{t+1} \mid X_t) \\ &= \sum_{X_{t+1}} P(y_{t+1:T} \mid X_{t+1}) P(X_{t+1} | X_t) \end{aligned} \tag{2.6}$$

We can calculate $P(y_{t:T} \mid X_t)$ in terms of $P(y_{t+1:T} \mid X_{t+1})$ thereby defining the set of backward recursions that are needed to calculate fully the joint probability in equation 2.5. This recursion is given by:

$$\begin{aligned} P(y_{t:T} \mid X_t) &= \sum_{X_{t+1}} P(y_t, y_{t+1:T}, X_{t+1} \mid X_t) \\ &= P(y_t \mid X_t) \sum_{X_{t+1}} P(y_{t+1:T} \mid X_{t+1}) P(X_{t+1} \mid X_t) \end{aligned} \tag{2.7}$$

One would therefore start the backward recursion from time T to calculate successively the joint probabilities of the states $X_{t:T}$. From equation 2.5, it can be seen that the forward and backward recursions are both used to calculate the probability of X_t given all of the data $y_{1:T}$. In practice, the forward algorithm is run for one pass of the data from $t = 1$ to $t = T$ and then the backward algorithm is run from $t = T$ to $t = 1$. Note that we have referred to the HMM in the above equations (the marginalizing steps are summations) but appropriate adjustments (to integrals) will present the SSM filtering and smoothing recursions.

2.2.4 Viterbi Algorithm

The Viterbi algorithm (Forney, 1973) for HMMs recursively finds the most probable path of states that generated the observed data (Nasrabadi, 2007). We first define the partial

probability $\pi_{t,i}$ to be the probability of the best path for states $X_{1:t}$ that leads to state $X_t = i$ as the terminating state. Intuitively, the best path leading to state $X_t = i$ at time step t must include the best path to X_{t-1} (assuming the same ending state i). We can therefore expect a recursive structure to an algorithm that decodes the best state sequences. In the following, we denote X_i as all possible terminating states that can be adopted at the time step t.

$$\pi_{t,i} = \max_{\forall X_i} \left\{ \pi_{t-1,j} P(y_t \mid X_i) P(X_i \mid X_{t-1}) \right\} \tag{2.8}$$

The algorithm uses the maximum probability from the previous state along with the observation probabilities for decoding the most likely present state.

2.3 Switching State Space Models

A SSSM includes M latent continuous state-space models and a discrete switching variable. Each of the models, referred to as a regime, has its own dynamics. At each point in time, the switching variable selects one of the individual state-space models to generate an observation vector.

The SSSM model is formalized as:

$$\begin{aligned} \mathbf{X_t^{(m)}} &= \Phi^{(m)} \mathbf{X}_{t-1}^{(m)} + w_t^{(m)} \\ \mathbf{Y_t} &= S_t A^{(m)} \mathbf{X}_t^{(m)} + v_t \end{aligned} \tag{2.9}$$

Here, $X_t^{(m)}$ denotes the latent continuous valued state for process m at time t. S_t is a discrete valued switching variable that selects the m^{th} regime such that regime m at time t is active. The real-valued state $X_t^{(m)}$ and the parameters associated with the m^{th} regime produce the real-valued observation vector Y_t. The states' evolution over time depends on the regime dependent transition matrix $\Phi^{(m)}$ and the regime dependent noise $w_t^{(m)}$. When

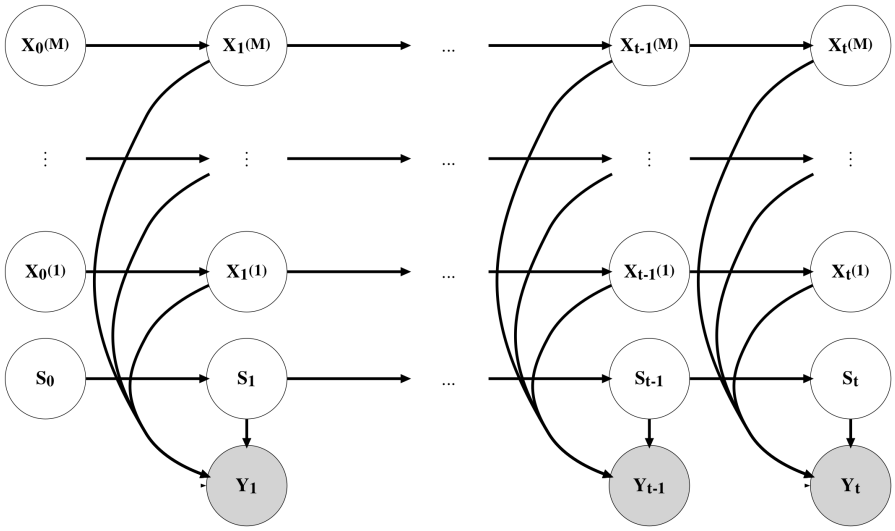


Figure 2.1: *Graphical model for the switching-state space model. A latent discrete switching variable (S_t) selects an active, real-valued state space model ($X_t^{(m)}$). The observation vector (Y_t) depends on the active regime at time t.*

the states and observations are assumed to follow Gaussian distributions, each regime can be viewed as a linear Gaussian SSM as discussed in section 2.2.1.

Figure 2.1 presents a graphical representation of a SSSM. Edges between nodes represent conditional dependencies, gray nodes are observed variables and white nodes are latent variables. Not shown in the figure are the regime dependent transition noise $w_t^{(m)}$ and the observation noise v_t which have the same interpretation as the noise models in the SSM. $A^{(m)}$ is the output matrix in the state space formulation, which can be regime dependent but for the applications that follow, it is taken to be the identity matrix.

2.3.1 Interpretability of the SSSM

We illustrate the SSSM interpretability by presenting an example of how it can describe the effects of students' interactions in CW. Y_t represents the observed water level in the different areas of the simulation at time t. $X_t^{(m)}$ describes the expected levels of water under regime

m at time t. $\Phi^{(m)}$ controls the water flow in the simulation according to the transitions in regime m. S_t selects which of the regimes to use to describe the water levels Y_t.

A single regime is insufficient for modeling the effects of students' interactions with CW, because students' actions have complex effects on the system dynamics. For example, when students choose to direct water to the Desert and Plains and plant many trees in the Desert, the system dynamics are entirely different from the case when water is directed towards the Jungle and the Desert and the Plains are left to dry. We therefore need to define multiple regimes, where each regime describes a series of events that can be (stochastically) explained by the regime dynamics. A regime is active for a duration of time in CW. We call this duration a period. In our example, in one period water mainly flows to the Plains and to the Desert. In the next period, students move logs to re-route water flow to the Jungle (potentially because plant life is dying there). The students may also release water from the Reservoir to direct to the Desert. These temporal dynamics are captured by the second period's regime parameters. The output of the model is a number of regimes, each active for the corresponding periods, that together capture the system dynamics.

It is worth highlighting that for the SSSM in general, and indeed for the inference that we target in CW, the model is highly under-specified. The points in time that define the changes between regimes are unknown, the duration of the periods are unknown and the regime specific parameters are unknown. Moreover, the number of regimes is unknown. The goal of Section 2.4.3 is to introduce an efficient unsupervised algorithm for performing inference over these unknown variables. For this implementation, we define a reasonable number of regimes M but in Section 4.1.1 we discuss the related work in Bayesian non-parametrics which allow us to remain agnostic about the number of regimes that are present.

2.3.2 Connected Worlds as an Autoregressive System

In Section 2.3.1, we have described a system where the direct observation of the underlying process is available. This is termed a vector autoregressive model (VAR) with switching (Fox

et al., 2009; Shumway and Stoffer, 2000; Kim, 1994). An order r switching VAR process with observations $\mathbf{y_t}$ is defined by:

$$\mathbf{y_t} = \sum_{i=1}^{r} A^{(m)} \mathbf{y_{t-i}} + w_t^{(m)} \qquad w_t^{(m)} \sim \mathcal{N}(0, Q^{(m)}) \tag{2.10}$$

Equation 2.10 shows that the observations at time t depend linearly on the previous r observation vectors. In CW, we use a $VAR(1)$ process. Since the SSSM generalizes the switching VAR process, we refer to the SSSM throughout this study but recognize that the models also apply to the VAR specific case. Any $VAR(r)$ process can be written in a SSM format as follows:

$$\mathbf{X_t} = \begin{bmatrix} A_1 & A_2 & \dots & A_r \\ I & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & I & 0 \end{bmatrix} \mathbf{X_{t-1}} + \begin{bmatrix} I \\ 0 \\ \vdots \\ 0 \end{bmatrix} w_t \tag{2.11}$$

$$\mathbf{y_t} = \begin{bmatrix} I & 0 & \dots & 0 \end{bmatrix} \mathbf{X_t} \tag{2.12}$$

2.4 Inference in Switching State Space Models

We aim to perform inference over the latent states $X_t^{(m)}$, the regime parameters $\Phi^{(m)}$, and latent switching variable S_t in the SSSM defined in equation 2.9. Computing posterior distributions for SSSMs is computationally intractable (Murphy and Russell, 2002; Kim, 1994). Figure 2.1 displays a graph that consists of M state space models that are marginally independent. These models become conditionally dependent when Y_t is observed (as is the case in this figure and in general for the models that we deal with). The result is that $X_t^{(m)}$ is conditionally dependent on the value of all of the other latent states and switching variables

for times 1 through T and regimes 1 through M (Murphy and Russell, 2002; Ghahramani and Hinton, 2000).

Previous approaches address the inference intractability by using approximation methods such as variational inference (Ghahramani and Hinton, 2000) or a 'merging of Gaussians' (Kim *et al.*, 1999; Murphy and Russell, 2002). Variational inference approximations transform an intractable Bayesian expectation problem into an optimization problem by minimizing the Kullback-Leibler divergence between a simpler family of approximating distributions and the unknown, intractable posterior (Attias, 2000; Saul and Jordan, 1996; Saul *et al.*, 1996; Blei *et al.*, 2003). The merging of Gaussians, discussed in Section 2.4.1, uses a single Gaussian to approximate the mixture of M Gaussians at each time step. While these methods have seen success in previous work, they cannot be applied in our domain, because they allow the system to move back and forth between regimes. Such switching results in frequent regime changes which can hinder the interpretability of the model. This work takes a different approach. It imposes structure on the model to address inference and interpretability challenges together. Another avenue for inference is the sticky hierarchical Dirichlet process hidden Markov model (sticky HDP-HMM) (Fox *et al.*, 2009, 2007), an extension of the hierarchical Dirichlet process (HDP) (Teh *et al.*, 2005). We discuss this model in further detail in Section 4.1.1 and present it as a promising avenue for future work.

2.4.1 Approximate Maximum Likelihood Estimation for Switching State Space Models

Shumway and Stoffer (1991) assume known regime parameters and use the state space specific observation probabilities to perform inference over the switching variables in the SSSM (using the Viterbi algorithm presented in 2.2.4). Their implementation is extended by Kim (1994); Kim *et al.* (1999) and Bar-Shalom and Li (1993) to handle the case where the regime parameters are unknown. The maximum likelihood estimate (MLE) presented by these authors employs a 'Gaussian merging' assumption (Ghahramani and Hinton, 2000) to

make the inference tractable. Kim (1994) presents a basic filtering and smoothing algorithm for the SSSM, combined with a MLE of the unknown regime parameters.

In the standard state space filtering formulation, we wish to calculate the expected value of the latent states X_t, given the observations $y_{1:t}$. This corresponds to calculating:

$$X_{t|t} = E[X_t \mid y_{1:t}] \tag{2.13}$$

Following state space notation for the expectation, $X_{t|t}$ refers to the expected value of the latent state at time t given the previous observations. Similarly, $P_{t|t}$ refers to the covariance of the estimate for $X_{t|t}$ conditioned on the data $y_{1:t}$:

$$P_{t|t} = E[(X_t - X_{t|t})(X_t - X_{t|t})' \mid y_{1:t}] \tag{2.14}$$

With the introduction of the switching variable S_t, we require $X_{t|t}^{(i,j)}$ which is the expected value for X_t, conditioned on the past data $y_{1:t}$ and the value of $S_t = j$ and $S_{t-1} = i$. The expected values for the states and the covariances associated with these estimators are:

$$\begin{aligned} X_{t|t}^{(i,j)} &= E[X_t \mid y_{1:t}, S_{t-1} = i, S_t = j] \\ P_{t|t}^{(i,j)} &= E[(X_t - X_{t|t})(X_t - X_{t|t})' \mid y_{1:t}, S_{t-1} = i, S_t = j] \end{aligned} \tag{2.15}$$

In computing the Viterbi algorithm, we would require the expectations and covariances in equations 2.15 for every possible value for i and j. If there are M regimes in the model, this results in M^2 terms that are calculated at every time step. The resulting Kalman filter experiences a growth of terms that is exponential in T, a problem that is also encountered in Section 2.4.3. Kim (1994) proposes an algorithm for reducing the $(M \times M)$ posteriors for $X_{t|t}^{(i,j)}$ into just M posteriors by merging the Kalman filter to a single term at each time step.

The approximation that Kim (1994) employs is given by:

$$X_{t|t}^{(j)} = \frac{\sum_{i=1}^{M} P[S_{t-1} = i, S_t = j \mid y_{1:t}] X_{t|t}^{(i,j)}}{P[S_t = j \mid y_{1:t}]} \tag{2.16}$$

Ghahramani and Hinton (2000) call this the 'Gaussian merging' approach because in equation 2.16, $X_{t|t}^{(j)}$ represents the re-normalized mixture of Gaussians in the numerator. $X_{t|t}^{(j)}$ is the $E[X_t \mid S_t = j, y_{1:t}]$ with the S_{t-1} dependency marginalized out. The covariance of this estimator $P_{t|t}^{(j)}$ is calculated in a similar fashion and can also be interpreted as the posterior from a weighted mixture of normals[4].

Finally, we can use numerical optimization to perform parameter estimation. The Gaussian merging is used to construct a forward filter to approximate the likelihood of the regime parameters and the associated switches. This likelihood is maximized to obtain an approximate MLE for the parameters. It is worth noting that the optimization is highly susceptible to local maxima due to the tightly coupled nature of the regime parameters and the locations of the switch points. Thus, the numerical optimization might terminate early at these local optima leading to incorrect inferences. This is seen in Section 3.3.1.

2.4.2 Background on Inference for Probabilistic Models

The SSSM is a probabilistic graphical model and thus Bayesian inference techniques are available for approximating the posterior distribution over the latent parameters in the model. Bayesian inference is appealing in this domain as prior knowledge can be built into the model and inference algorithm to assist with the inference task. Here, we review techniques for approximating the posterior distribution in a complicated graphical model and motivate Markov Chain Monte Carlo as an appropriate inference tool.

Introduction to Bayesian Inference

Bayesian models lend themselves to domains where the interpretability of the model is of importance (Gelman *et al.*, 2014): the model structure that is imposed on a domain is

[4]See the derivation in Kim (1994) for further details.

considered part of the scientists prior knowledge. Due to this structured (often hierarchical) model, inference about the latent variables has a strong interpretability within the model's domain.

Bayesian statistics is derived from Bayes rule:

$$P(\theta \mid X) = \frac{P(X \mid \theta)P(\theta)}{\int P(X \mid \theta)P(\theta)d\theta} \tag{2.17}$$

Evaluating equation 2.17 can occasionally be done analytically when the conjugate structure in certain problems allows for the direct computation within known families of distributions. This structure is often not available, or conjugacy imposes undesirable restrictions given the problem setting. Therefore, other means for *approximating* the posterior distribution are often required when performing inference in Bayesian models. Variational inference (Attias, 2000; Saul and Jordan, 1996; Saul *et al.*, 1996; Blei *et al.*, 2003) and Monte Carlo (MC) are two techniques commonly used to approximate the posterior distribution. Variational inference can give misleading results when dealing with highly correlated and co-dependent data, such as data present in time series models (Turner and Sahani, 2011). MC rather attempts to draw independent samples from the unknown distribution to use the finite set of samples as an approximation to the original distribution. MC techniques are appealing due to their strong convergence properties albeit at a high computation cost (MacKay, 1998).

MC sampling is a popular approach for performing posterior inference in Bayesian statistics as it presents a technique for evaluating expectations under an unknown target distribution (MacKay, 1998). MC techniques use the fact that given independent samples $x^{(r)}$ from a target distribution $P(X)$, the samples can provide a robust estimator of expectations under the distribution defined by P. The strong convergence properties of MC methods state that the error of the estimator tends to 0 as the number of samples tends to ∞. Concretely, the expectation $\mathbf{\Phi}$ of a function ϕ under the distribution P can be approximated

by the MC estimator $\hat{\Phi} = \frac{1}{R}\sum_r \phi(x^{(r)})$. Here $\mathbf{\Phi} = \int P(x)\phi(x)dx$ is the true value for the expectation (MacKay, 1998). Examples of MC sampling include rejection, importance and slice sampling (Neal, 2003) but these techniques are shown to scale poorly to high dimensional and tightly correlated distributions.

Markov Chain Monte Carlo

Rather than attempt to draw entirely independent samples from the target distribution, we can use current samples to propose new, dependent samples in the form of a Markov chain. Given a valid transition function in the Markov chain, the stationary distribution of the samples converge to the true target distribution (MacKay, 1998). The subset of MC techniques that use current samples to propose new, dependent samples is termed Markov Chain Monte Carlo (MCMC). When explicit structure of the domain allows the derivation of complete conditionals[5] (MacKay, 1998), Gibbs sampling presents an attractive MCMC technique. More generally, the Metropolis method is used when complete conditionals are not available.

Metropolis is an algorithm for sampling from an unknown distribution with density $\frac{1}{Z}f(x)$. Here, $f(x)$ is an un-normalized probability density and $\frac{1}{Z}$ is the normalizing constant. The algorithm uses the current sample x to propose a new sample x^* from a given proposal distribution (i.e., $g(x^* \mid x)$). The candidate sample x^* is evaluated according to an acceptance probability $a(x^*, x)$, and if the proposal state is accepted, we set the new sample $x' = x^*$. Alternatively, the new sample is set to the previous sample $x' = x$. The acceptance probability for Metropolis is given by:

$$a(x^*, x) = min\left[1, \frac{g(x \mid x^*)}{g(x^* \mid x)}\frac{f(x^*)}{f(x)}\right] \tag{2.18}$$

[5]The *complete conditionals* are the conditional distributions of a subset $T \subset \{1, 2, \ldots n\}$ of the parameters given all other parameters not in T.

It has been shown that the update from x to x' leaves f invariant (MacKay, 1998; Gelman *et al.*, 2014) and therefore as the Markov chain is run indefinitely, the Metropolis algorithm provably converges to drawing samples from the stationary distribution that is defined by density $f(x)$. Metropolis provides a general structure for defining a Markov chain that will converge to drawing samples from the target distribution. However, the algorithm is highly dependent on the proposal distribution g, and if g is poorly chosen, the algorithm can reject many samples resulting in a slow convergence of the chain. Similarly, a poor choice of g can result in samples that are highly correlated and thus the new samples do not explore the posterior distribution sufficiently for a given *limited number* of samples, leading to a high bias in the estimator. Variants of the Metropolis algorithm have been proposed to assist the exploration of the target distribution. The state of the art is the Hamiltonian Monte Carlo (HMC) method and the No U-Turn Sampler (NUTS) which modifies HMC by tuning parameters automatically.

Hamiltonian Monte Carlo and the No U-Turn Sampler

Hamiltonian Monte Carlo (Neal *et al.*, 2011) was introduced as another MCMC technique to combat the inadequacies of Metropolis. In particular, HMC introduces *a structured and deterministic* search of the probability space before making a proposal sample. The aim of HMC is to efficiently *explore* the target distribution and thus draw pseudo-independent samples while still using the convergence guarantees of MCMC.

HMC is inspired by statistical mechanics where particles in space are known to interact and follow trajectories that are defined by Hamiltonian mechanics (Neal *et al.*, 2011). Hamiltonian equations define the energy of the particle over momentum (p) and position (q) phase space. HMC is an auxiliary model where a momentum variable (p) is introduced that allows us to simulate a deterministic dynamic through (p, q) space with the analogy that the probability corresponds to the distribution of the position (q) of a particle. Hamiltonian dynamics have three properties that make this simulation useful for the MCMC use-case.

The Hamiltonian (1) preserves energy, (2) has reversible dynamics and (3) conserves volume in phase space. Properties (1) and (3) allow the simulation of Hamiltonian dynamics along some probability distribution contour such that a new proposal is found that has the same energy as the current sample. Property (2) is important for conserving *detailed balance*, a property of MCMC that ensures the target distribution is left invariant by the simulation (i.e., does the sampler actually converge to drawing samples from the target distribution).

For HMC, a numerical integration scheme is needed to simulate the differential equations that are defined by the Hamiltonian (Neal *et al.*, 2011). This numerical integration scheme is known as a symplectic integrator as is required to conserve energy throughout the course of the simulation, corresponding to maintaining property (1) of the Hamiltonian. The leapfrog method makes a small adjustment to Euler's numerical integration scheme by alternating between the momentum and position updates with a half-step spacing the discretization updates. The integration requires two user-defined parameters: the step size ϵ and the number of steps used to simulate the dynamics L. Tuning L and ϵ is shown to be challenging in practice and can have dramatic effects on the success of the HMC algorithm for exploring the entire typical set of the target distribution (Hoffman and Gelman, 2014). Note that choosing ϵ too small will result in unnecessary computation to simulate Hamiltonian trajectories. Choosing L too large will result in Hamiltonian trajectories returning to their starting point, defeating the goal of exploring the target probability space to find *independent* samples.

Hoffman and Gelman (2014) introduce the No U-Turn Sampler (NUTS) to automatically tune these two parameters for the given problem. In essence the algorithm constructs a binary tree simulating the HMC dynamics for a number of steps that doubles with each level in the tree. The algorithm terminates when the simulated dynamics experience a momentum that points back toward the starting point of the simulation. The imprecise nature of the numerical integration scheme requires that the proposed state be accepted

with an acceptance probability from the Metropolis update[6]. NUTS is implemented in a number of probabilistic programming languages to enable automatic inference in complex user-defined models. We use the Stan-MC[7] (Carpenter *et al.*, 2016) implementation of NUTS for performing inference in the model defined in Section 2.4.3.

2.4.3 Algorithm for Posterior Inference of Switching State Space Models

Computing the posterior distributions over the latent variables in an SSSM corresponds to approximating the joint distributions over $X_t^{(m)}$, S_t and $\Phi^{(m)}$, given the observation vector $\mathbf{Y}$. To perform inference over the regime parameters and the switch assignments of the SSSM, we make two assumptions, which arise from the need to create human interpretable descriptions of the complex system behavior. **Assumption 1:** the system advances through a series of regimes, each regime is active for a period, and then switches to an entirely new regime, one that has not been used before. **Assumption 2:** the regime remains active for the maximum possible time for which it can be used to describe the period.

Without making these assumptions there are M possible assignments of regimes for each time step, making a total of M^T combinations of possible assignments, which is exponential in the number of time steps. Moreover, in the worst case, the number of possible periods is bounded by T with a switch at every time step. In contrast, under our assumptions, there are only two possible assignments of regimes for each time step (i.e., the choice is to stay in the current regime or to progress to the next regime), making for a total of 2^M combinations of possible assignments, where M is constant. The number of possible periods under this methodology is bounded by M.

[6]If the Hamiltonian dynamics were simulated perfectly, there would be no need for this acceptance check. But, because there is error in the numerical integration, we need to accept the proposed sample in the same way that Metropolis accepts proposed samples.

[7]`http://mc-stan.org/`

Non-identifiability and Label Switching

A well known problem with MCMC inference in mixture models is that of identifiability (Jasra *et al.*, 2005). Models are non-identifiable when two sets of parameters can explain the observed data equally well. This problem is also referred to as label switching as the labels given to classes are unrelated to the class specific parameters. MCMC samplers experience a random permutation of labels during sampling and the resulting posterior distribution is not class specific but rather multi-modal and identical to the marginal posteriors for the individual classes.

This phenomenon can be seen in a simple Gaussian mixture model with means μ_0, μ_1 and covariances Σ_0, Σ_1. The marginal posterior distributions of the parameters are identical. Figure 2.2 shows the posterior sample traces for the means from a one dimensional Gaussian mixture model with true means $\mu_1 = -1, \mu_2 = 1$ and standard deviations $\sigma_1 = \sigma_2 = 0.75$. The label switch can be clearly seen at samples indexed 1000 and 2000 respectively. The smeared and bi-modal posterior distributions for the mean parameters are also clearly evident. If the chain was not truncated (and if other MCMC chains were run), the posterior distributions for μ_0 and μ_1 would be identical. A solution to the identifiability problem is to add constraints on the model prior (e.g., enforcing $\mu_0 > \mu_1$) which makes the prior not symmetrical. However, defining such constraints in higher-dimensional domains is non-trivial.

Another possible solution for the identifiability problem is to provide labels for part of the data. This is termed semi-supervised learning and we incorporate this solution into our model. In the context of the CW domain, we can label observations as belonging to one regime or another. Let $S_{t,t+1,\ldots,t-1+K,t+K}$ be a consecutive set of K state variables such that S_t and S_{t+K} have known value assignments (regime m and regime $m+1$, respectively). The values for the state variables $S_{t+1,\ldots,t-1+K}$ are unknown. By Assumption 1, the switch between regimes m and $m+1$ occurs at some S_l where $t \leq l \leq t+K$. Therefore, the value

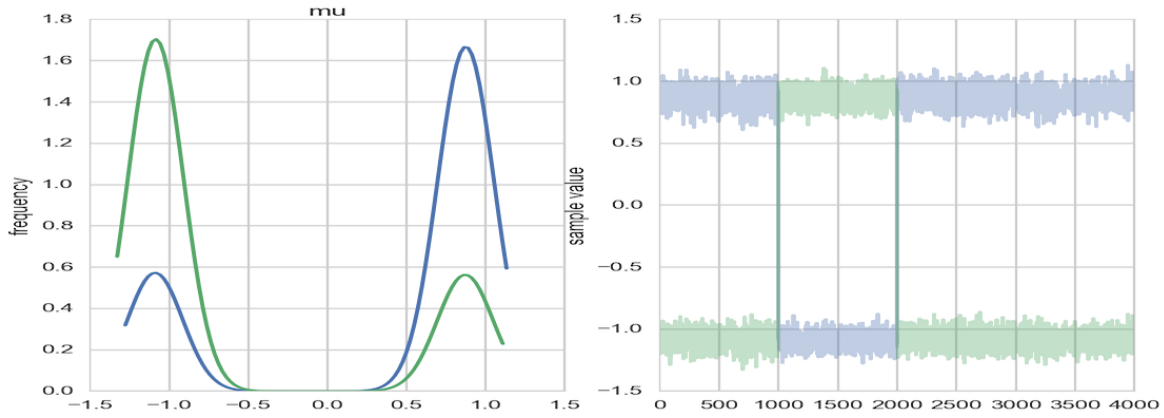


Figure 2.2: *Posterior samples and associated density plot from the posterior of a Gaussian mixture model with true parameters $\mu_1 = -1, \mu_2 = 1, \sigma_1 = \sigma_2 = 0.75$. Left shows the density plot for the posterior of the mean parameters. Right shows the posterior sample trace. The label switches can be seen at samples 1000 and 2000 resulting in the multi-modal posterior on the left.*

of S_l determines the values for all of the unknown states in this period as S_t is assigned to regime m for $t < l$ and it is assigned to regime $m + 1$ for $t \geq l$.

Algorithm Sketch

We provide a sketch of this process in Algorithm 1. **Step 1** initializes the M supervised switch variables, one per regime. The labeled switch variables are spaced uniformly in time and are assigned to regimes in increasing order according to Assumption 1. This uniform method for initialization can be justified by Assumption 2, in that any set of regimes that provides an interpretable model is sufficient. The number of expected time steps in each period is $K = T/M$, and there are $K - 2$ unlabeled switch variables between each pair of switch variables assigned to regimes.

Step 2 performs MCMC sampling to approximate the posterior of the model. For the case when the value of the switch variable is known, the posterior of $X_t^{(m)}$ can be directly sampled by following the structure of a state space model. In the case where the switch variable is unknown, we have a marginalization problem over the two possible values of S_t. For the hidden Markov model (HMM) structure this can be efficiently computed with the

forward algorithm (Shumway and Stoffer, 2000). To formulate the HMM forward algorithm, we use the observation probabilities from the individual state space models in place of the emission probabilities of a standard HMM. Here, π_{S_i} refers to the belief of the state of the switching variable given the evidence up to that point in time.

Step 3 uses the regime specific parameters $\Phi^{(m)}$ to make a maximum likelihood assignment of an observation to a regime using the Viterbi algorithm, thereby specifying the value of $S_t \forall t \in [1:T]$.

Algorithm 1: Posterior inference algorithm

Input: M (number of regimes), $\mathbf{Y}$ (vector of observations for T time steps).

1 Initialization: Label one datapoint per regime, leaving $T-(M+1)$ unlabeled datapoints.

2 MCMC Inference: Draw samples for $X_t^{(m)}$ from the posterior distribution defined by the structured probability model:

for Y_t *in* $\mathbf{Y}$ **do**

  if $S_t = m$ *is known* **then**

    sample from $P(X_t^{(m)}, \Phi^{(m)} \mid X_{t-1}, S_t = m, Y_t)$

  else

    marginalize over S_t. Sample from $\sum_{i=m-1}^{m} \pi_{S_i} P(X_t^{(i)}, \Phi^{(i)} \mid X_{t-1}, S_t = i, Y_t)$

3 Posterior Inference: Use the posterior for regime parameters ($\Phi^{(m)}$) to run a Viterbi pass on the data $\mathbf{Y}$ to make a maximum likelihood assignment of the value of S_t to regime m (thereby learning the switching variables S_t).

Output: S_t (assignments to regimes), $\Phi^{(m)}$ (regime specific posterior distributions).

Algorithm 1 can be computed on any SSSM where Assumptions 1 and 2 are appropriate. Such a model is shown in Figure 2.3. The model depicts a subset of the time series with K time steps from time t to time $t+K$. There are two supervised labels at the boundaries of the subset with the variable S_t assigned to regime m and variable S_{t+K} assigned to regime $m+1$. The unknown $K-2$ states in between are marginalized over such that the regime specific posteriors can still be approximated. This model is repeated for the $M-1$ switches in the data. The setup is flexible in that informative priors for the model noise and transition matrices can be specified as required by domain knowledge.

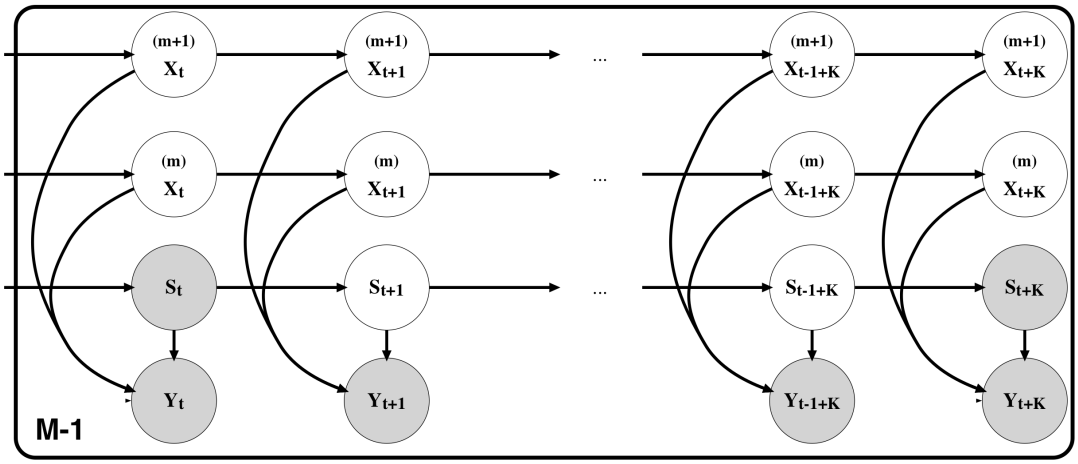


Figure 2.3: *Updated graphical model showing the semi-supervised switching labels, along with the choice of only two chains between two semi-supervised points. This representation is repeated $M-1$ times to describe the $M-1$ switches between the M regimes. Note that the labeled switch variables at the boundaries (S_t and S_{t+K}) are shared between successive regimes.*

Chapter 3

User Study

3.1 Empirical Validation

We evaluate two aspects of Algorithm 1. First, we show that it finds the true regime labels in a synthetic dataset. Thereafter, we use data that were collected from Connected Worlds to implement two experiments to test whether the inferred periods are interpretable to independent human validators. The first of these experiments aims to verify the salience of the inferred change points within the context of CW. The second experiment is to verify the intelligibility of the automatically generated descriptions for the inferred periods. Due to the tightly coupled relationship between the period boundaries and the regime specific parameters, separating these two factors for comparison is exceedingly difficult. The two user studies present two perspectives on the interpretability of the periods and the associated regime parameters. The significant results suggest that the SSSM produces periods and regimes descriptions that do have interpretabilty in the CW domain.

3.2 Verification of Regime Label Inference

$$\begin{aligned} \mathbf{X_t^{(1)}} &= 0.99\,\mathbf{X_{t-1}} + w_t^{(1)} \quad w_t^{(1)} \sim \mathcal{N}(0,1) \\ \mathbf{X_t^{(2)}} &= 0.9\,\mathbf{X_{t-1}} + w_t^{(2)} \quad w_t^{(2)} \sim \mathcal{N}(0,10) \\ \mathbf{Y_t} &= S_t\mathbf{X_t} + v_t \quad v_t \sim \mathcal{N}(0,0.1) \end{aligned} \tag{3.1}$$

Equation 3.1, an adapted model from Ghahramani and Hinton (2000), describes a SSSM with two regimes and a continuous state space[8]. The transition parameters and noise are regime dependent with $A^{(1)} = 0.99$, $A^{(2)} = 0.9$, $Q^{(1)} = 1$ and $Q^{(2)} = 10$. S_t is the categorical switching variable that chooses between the two regimes at every time step. The prior probability[9] of each of the regimes is 0.5 and the transition probabilities are $\Phi_{1,1} = \Phi_{2,2} = 0.95$ and $\Phi_{1,2} = \Phi_{2,1} = 0.05$. This model was used to generate 1000 time series, each with 200 observations.

Figure 3.1 shows an example of a generated time series. The switch points are shown according to the true model (top), the inferred labels from Algorithm 1 (middle) and the Gaussian merging baseline (bottom). Each period is represented by a sequence of black and gray colored circles. As shown in Figure 3.1, the periods inferred by both Algorithm 1 and the baseline overlap to some extent with the true periods. However, there is substantially more noise (higher frequency of switches) in the inferred periods of the baseline.

We ran Algorithm 1 to learn the switch points on this data, setting the number of regimes to nine. The percent agreement between the inferred regime labels and the known labels presents the accuracy for each algorithm for each time series. Figure 3.2 shows a histogram of the algorithm accuracy according to Algorithm 1 (*a*) and the baseline Gaussian merging (*b*). The bi-modal and long tailed distribution for the baseline approach demonstrates its

[8]The Ghahramani and Hinton (2000) model describes a state space that is disjoint at regime switches. We choose to make the state space continuous at the switch points as this more accurately mimics the scenario that is present in CW.

[9]$P(S_1 = 1) = p_1 = P(S_1 = 2) = p_2 = 0.5$

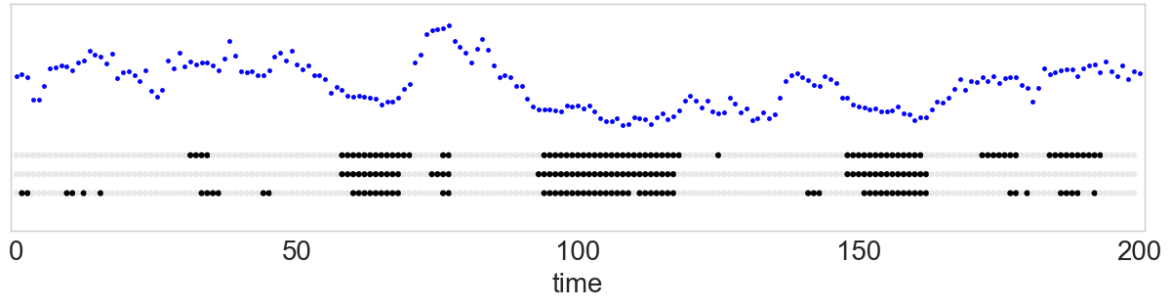


Figure 3.1: *An example of a generated time series from the SSSM model of equation 3.1. The x axis represents time, and the y axis shows the observations. Regime labels are shown as black and gray dots representing the two label options. True labels (top) are compared to the inferred labels from Algorithm 1 (middle) and the Gaussian merging (bottom).*

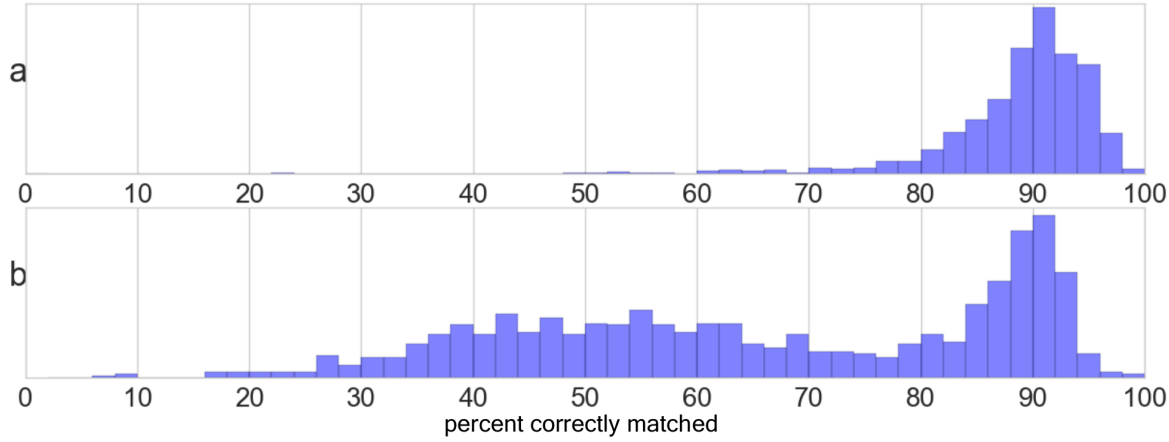


Figure 3.2: *Histogram of the percent of correctly inferred labels for the observed output. The structured sampling Algorithm 1 (a) learns the regime labels more accurately than the uninitialized Gaussian merging algorithm (b).*

susceptibility to local optima. When this algorithm converges toward the correct labeling, it finds a similar structure to that inferred by Algorithm 1. However, if the initialization of the optimization (chosen randomly) is poor, this algorithm can easily result in labeling all of the data as belonging to one regime or the other. The mean accuracy of Algorithm 1 was 89%, materially higher than the 66% achieved by the Gaussian merging approach.

The superior performance of Algorithm 1 can be directly attributed to the switching behavior that is enforced by Assumptions 1 and 2, which was not assumed by the baseline model. Although Algorithm 1's model structure encourages the discovery of switches, the

uniformly spaced initialization of labels should not be seen as an advantage as no prior knowledge of the actual switches is used in performing this step. The justification for enforcing a static number of switches is from Assumption 2 where the goal is to find more stable regimes and thus a lower frequency of switches between regimes. An implementation note is that although Algorithm 1 searches for 9 regimes in the data, we know that only two are present. The presence of two regimes can be specified in the hierarchical structure of the model such that each of the 9 regimes draws its parameters from only two options. The number of regimes for the Gaussian merging approach is merely set to two.

3.3 Validation of Model Interpretability

The inference from Algorithm 1 aims to assist teachers when leading their students through a review discussion of a particular simulation session. Thus we aim to validate that the inferred switch points are interpretable to a human seeking to understand the "story" of the simulation. The relative magnitudes of the inferred regime parameters are used to generate a short description for the water flows for each period. The CW system provides a video representation of the log positions and the resulting water flow in the simulation. See Figure 1.1 for one frame from the video visualization that shows the biome positions, a distribution of logs on the floor of the simulation and the resulting flow of water at that time step. This video representation was presented to human validators to verify the change points (presented in Section 3.3.1) that are found by Algorithm 1 and to verify the intelligibility (presented in Section 3.3.3) of the automatically generated descriptions for the inferred periods.

3.3.1 Experiment 1

We designed an experiment[10] that asked evaluators to select one of three possible switch points between every pair of consecutive periods. Evaluators saw a composite of 1) the movie of the two periods; 2) a description of the dynamics of each of the two periods (e.g., "water flows to the Desert and Plains") and 3) a set of three possible switch points between the periods. The evaluator's task was to choose the switch point that best matched the change in dynamics between the two periods. One of the three switch points was that inferred by Algorithm 1; the other two were random times sampled uniformly from the beginning of the first period to the end of the second period with the constraint that any two presented times cannot be within $10s$ of one another.

Evaluators worked with five sessions, each of which included 8 to 12 periods of system dynamics. Selecting the correct switch point is not a trivial task: it requires distinguishing between the changes in the system that indicate different dynamic regimes from the changes that are noise within the same dynamic regime. We see an evaluator's ability to choose a switch point, based on the movie and a description of the two contiguous periods, as evidence that the inferred periods are potentially useful to a teacher who wants to guide students in constructing a causal description of their experience with the simulation.

3.3.2 Experiment 1 Results

Figure 3.3 presents the results of the validation using four evaluators with knowledge of the CW domain. The five sessions are shown along the x-axis; the fraction of correctly selected switch points is shown by the bin heights. The dashed line represents a random baseline in which the selected switch probability corresponds to $\frac{1}{3}$. Under the null hypothesis, the performance of an evaluator would not be significantly different than the random baseline.

[10] Available at `http://essil-validation.s3-website-us-east-1.amazonaws.com/`

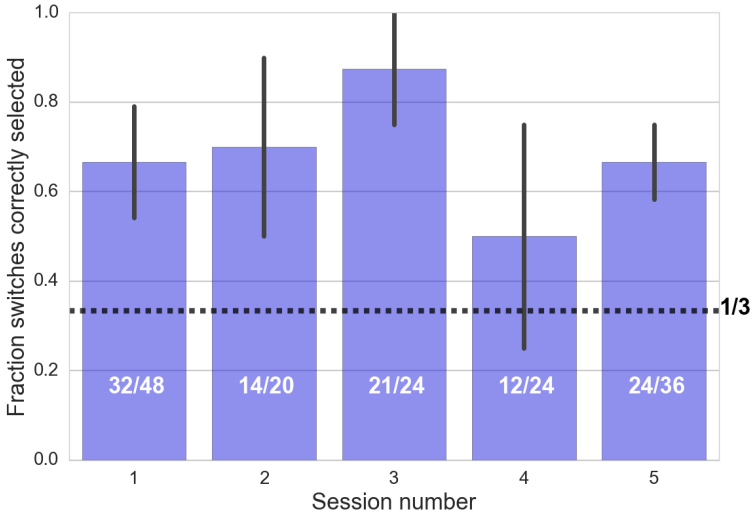


Figure 3.3: *Expert validation of five different test files from sessions with CW. The bar plot shows the fraction of correctly identified switches between automatically identified periods.*

The results indicate that the evaluators chose the switch point identified by Algorithm 1 significantly more often than the random baseline ($p < 1 \times 10^{-4}$). This suggests that the inferred switch points were interpretable as meaningful changes to the system dynamics. The differences in interpretability seen in Figure 3.3 (e.g. Session 4 was more difficult to interpret than Session 3) can provide a direction for further investigation in how to support teachers and students in making sense of their experiences in CW. For example Appendix B gives a brief investigation into the file and periods specific difficulties. In the Appendix we discuss how the session complexity might require an adaptive choice for the number of inferred regimes. Session 4 is dramatically more complex than Session 3 and thus we may wish to introduce more periods to describe this additional complexity. We are also able to investigate the factors that lead to poor regime descriptions. These are largely caused by factors that are not considered by the model, such as rain events in the biomes. We recommend that the system logs these events for the disambiguation of causes of system changes.

3.3.3 Experiment 2

Experiment 2 aims to test the interpretability of the automatically generated regime descriptions when presented alongside the associated period clip from the video representation of the students' work. Validators are asked to choose one of three plausible descriptions for the dynamics that are displayed in the video clip of the period. Experiment 2 further introduces a control condition where the change points in the time series are pre-defined and the descriptions are generated, conditioned on the pre-defined change points.

Figure 3.4 shows a screen-shot of the visualization[11] that the users are given. Label (1) shows the video clip; this clip is played for the duration of an inferred period. Label (2) indicates one of three descriptions that might be associated with the video clip. Validators are asked to watch the video and select the description that most accurately describes the dynamics in the video for the period. The dummy descriptions for each period are randomly generated.

The control condition in this experiment is to compare the algorithmically inferred period boundaries with boundaries that are uniformly spaced over the duration of the time series. We return to the hypothesis that deconstructing the time series into smaller periods will improve the period interpretability as the complexity of the period must decrease for shorter period lengths. Allowing Algorithm 1 to define the period boundaries should assist in finding periods that are more coherent for the associated descriptions. The algorithmically inferred periods should therefore be easier to identify than the periods that are defined uniformly across the time series.

The visualization (represented by the screen-shot in Figure 3.4) was shown to 40 independent validators. Each participant had little or no prior knowledge about CW. A short tutorial of 14 slides introduces the objective for validation and explains the key aspects of

[11] Available at `https://essil-validation.herokuapp.com/`

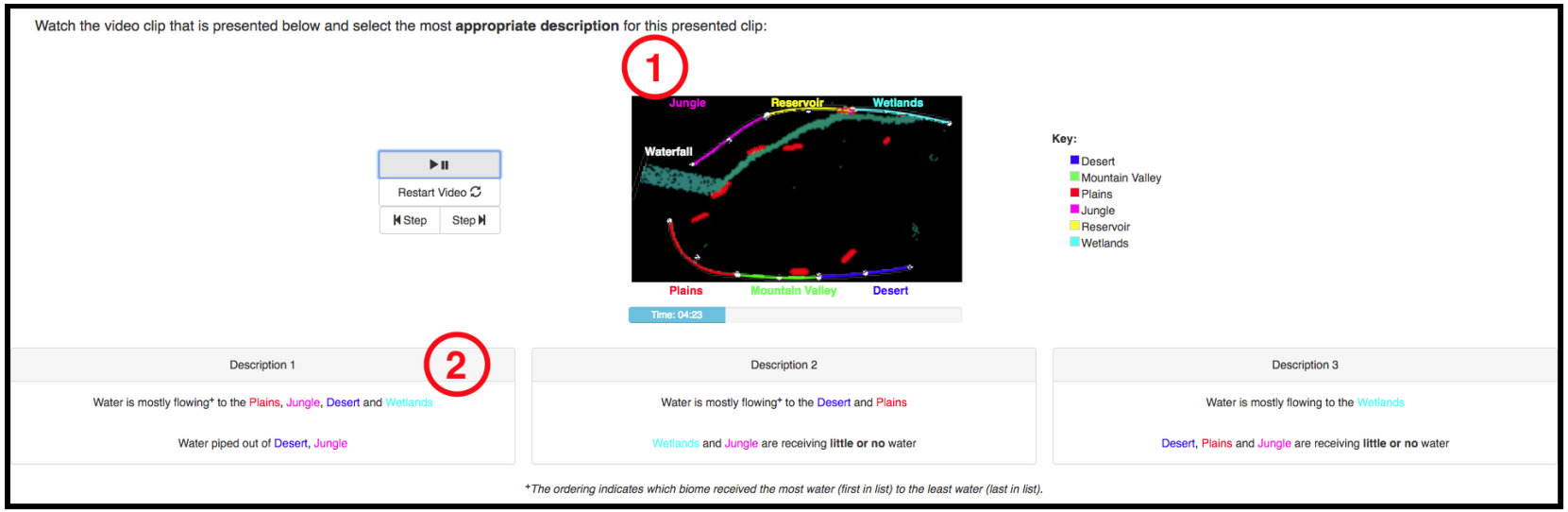


Figure 3.4: *Screen-shot of the user interface designed to evaluate the interpretability of Algorithm 1. The video representation at (1) is played for the duration of the period. Three plausible descriptions (of which (2) is one of these) are presented and the validator is asked to select the description that best describes the dynamics shown in the video. In this case, Description 3 is the correct solution.*

the video. The participants are shown 8 randomly selected periods from the control and the test groups. The validators are required to watch the video associated with the period and thereafter to select the description that most accurately describes the water dynamics. The results from Experiment 2 are given in Section 3.3.4.

3.3.4 Experiment 2 Results

40 validators independently labeled 296 periods. The periods were generated from the same 5 session files that were presented in Section 3.3.1. The five files respectively had 12,8,8,8 and 10 inferred periods from Algorithm 1. The control condition was implemented with a uniform period spacing with 8 periods and again with 5 periods in all 5 test files. Figure 3.5 shows the raw accuracy that was achieved for each file, under each algorithm. It can be seen that increasing the number of periods increases the ability of the validators to select the algorithmically generated periods. As the lengths of the periods decrease, there are fewer dynamics in the corresponding video clip and therefore the period description should be more recognizable to the validator.

The same general shape of the bar plots in Figures 3.5 and 3.3 suggests that the file appears to affect the interpretabilty of the periods in a similar manner for both experiments. This is intuitively correct as sessions where the system dynamics are more stable (possibly due to students executing fewer actions or having a simpler goal) might make for simpler and more easily understood descriptions. However, in cases where there are possibly many conflicting student agendas, the resulting dynamics might be complex and hard to interpret.

3.3.5 Analysis of Results for Experiment 2

Noting that the interpretability of the periods is file specific, we design a logistic regression analysis of the results. Specifically, a hierarchical logistic regression was conducted to

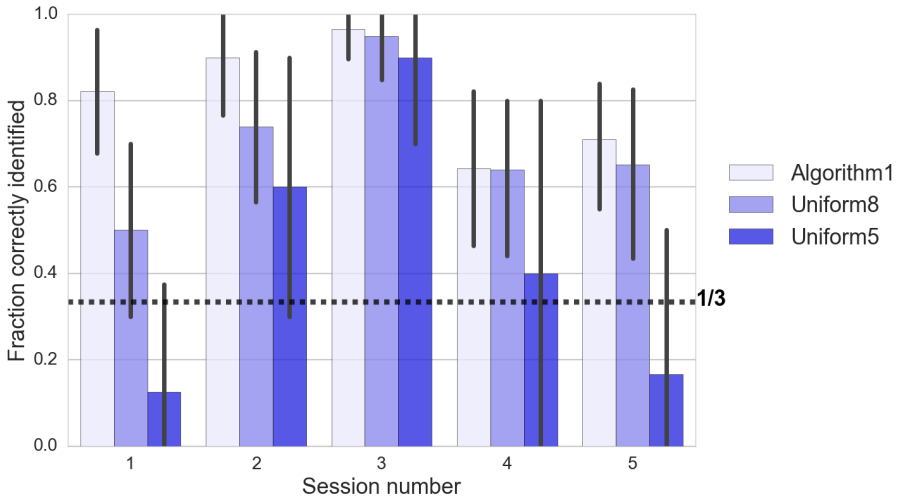


Figure 3.5: *Bar plot of the results from Experiment 2. There are 5 files that were tested. The fraction of correctly selected period descriptions from Algorithm 1 is compared to that from the control conditions. The control conditions include uniformly spaced periods for* 8 *and* 5 *periods respectively.*

determine the effect of the different algorithms on the probability that a validator successfully chooses the given algorithm's generated description. We compare the effects of seeing periods and descriptions from Algorithm 1 and the control with 8 periods (Uniform8) to the baseline of the control with 5 periods (Uniform5).

The graphical model for the hierarchical logistic regression is presented in Figure 3.6. μ_f refers to a file specific 'ease of labeling'. The Algorithm 1 and Uniform8 'interpretability effects' are added to the file specific parameter. Period specific parameters are drawn from a normal prior distribution with mean defined by the file parameter added to the effect of the algorithm used. Each period specific parameter p_i represents the log-odds of a successful outcome in a Bernoulli trial. Note that the period specific parameters p_i depend on the mean defined by the file and the algorithm that was used to generate the period. The variance around the file-algorithm mean is defined by the parameter σ, shared among all periods. σ is an unknown model parameter and thus we place a weakly informative *half-*$\mathcal{N}(0,1)$ prior on this parameter. We conduct inference over the algorithm interpretability effect parameters (α_a) and the period specific log-odds parameters (presented in Appendix B).

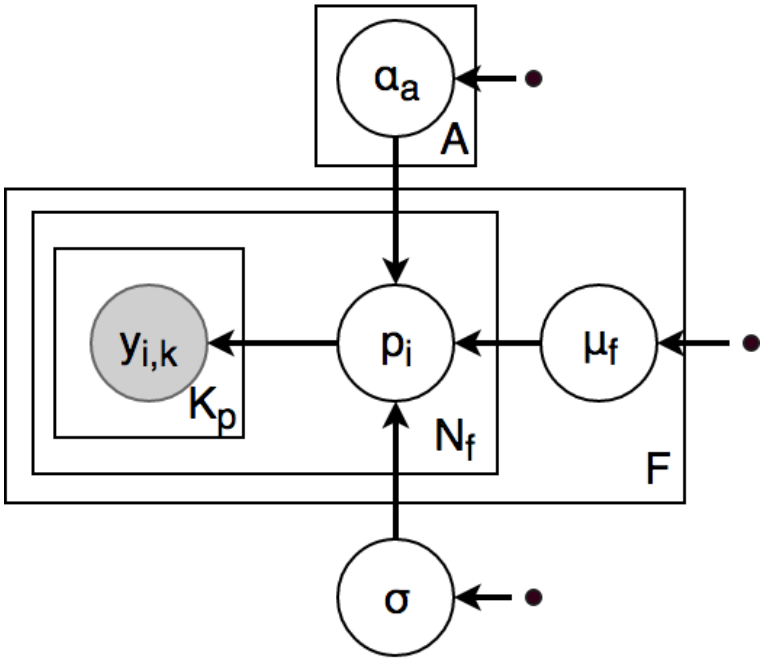


Figure 3.6: *Graphical model for the hierarchical logistic regression that is conducted to determine the interpretability effect of Algorithm 1 and Uniform8 over the base Uniform5 condition. The observed data y corresponds to a Bernoulli trial where a description is selected to match a video clip or not. A given trial has a period specific probability of being chosen correctly p_i. p_i, in turn, depends on the file's ease of labeling and the algorithm effect. The α_a parameters represent the effect of the algorithm on the probability of correctly selecting the appropriate description. The periods depend on the file and algorithm means by a global variance parameter σ.*

$$
\begin{aligned}
y_{i,k} \mid p_i &\sim Bern(logit^{-1}(p_i)) \\
p_i \mid \mu_f, \alpha_a, \sigma &\sim \mathcal{N}(\mu_f + \mathbb{1}\{a\}\alpha_a, \sigma) \\
\mu_f &\sim \mathcal{N}(0,1) \\
\sigma &\sim \textit{half-}\mathcal{N}(0,1)
\end{aligned}
\tag{3.2}
$$

The generative sampling distributions are given in equation 3.2. The inverse logit function is used to map the real valued log-odds parameter p_i to a probability between 0 and 1. α_a refers to the algorithm specific interpretability effect, with α_1, the parameter associated with using Algorithm 1 and α_2, the parameter associated with using Uniform8 to generate the period. $\mathbb{1}\{a\}$ is the indicator that algorithm a was used to produce the description and define the duration for period i.

The mean posterior value for $\alpha_1 = 2.03$ with a Bayesian 95% posterior confidence interval of $[1.18, 2.96]$. The value is quoted in log-odds and corresponds approximately to being ~ 8 times more likely that a validator will select a period correctly under Algorithm 1 than under the baseline Uniform5. This is a highly significant result suggesting that Algorithm 1 improves on the baseline Uniform5. The control group with 8 uniformly spaced periods has a posterior mean $\alpha_2 = 1.28$ (multiplicative effect of ~ 4 times more likely to select the period correctly over Uniform5) with a posterior confidence interval of $[0.43, 2.21]$. This too does not include 0, the expected value for no discernible improvement, and therefore suggests that merely increasing the number of periods does help to make the periods more interpretable. However, Algorithm 1 can be compared to Uniform8 by evaluating the probability $P(\alpha_1 > \alpha_2)$. In other words, we are interested in the probability that Algorithm 1 is associated with a greater probability of choosing the generated description. This posterior probability is significant with a p-value of $p = 0.04$. The results suggest that the presented Algorithm 1 significantly improves the interpretability of the generated descriptions.

The structure of the inference executed in section 3.3.4 allows us to consider the posterior

probability of p_i for each of the presented periods. This discussion is given in Appendix B.

Chapter 4

Future Work and Conclusion

4.1 Future Work

There are two areas to be considered for future work. Firstly, Algorithm 1 assumes a known number of regimes. This is problematic in that sessions of different lengths or complexities would naturally have a different number of periods. It is not clear how to choose this number for a given session yet, as was seen in Section 3.3.3, it can have a major effect on the algorithm output. Bayesian non-parametrics presents a technique for remaining agnostic about the number of regimes to search for. Section 4.1.1 reviews related work in Bayesian non-parametrics for HMMs and we motivate why these models are promising for this domain. In Section 4.1.2 we discuss the future work that builds on this thesis with the aim of working towards our original goal of presenting a tool that assists teachers when reviewing sessions from Connected Worlds.

4.1.1 Bayesian Non-parametric Learning for the SSSM

Introduction to Hierarchical Dirichlet Process

The Dirichlet process (DP) (Ferguson, 1973) defines a distribution over probability measures on a parameter space Θ. It is parameterized by G_0, a base distribution, and α, a concentration parameter. The DP consists of discrete atoms that are distributed on the base measure G_0 with mass that depends on α.

$$\begin{aligned} G \mid G_0, \alpha &= \sum_{k=1}^{\infty} \beta_k \delta_{\theta_k} \\ \theta_k &\sim G_0 \end{aligned} \tag{4.1}$$

δ in equation 4.1, refers to the Kronecker delta function and represents the atoms of G that are distributed according to $\theta_k \sim G_0$. The atoms have mass β_k, where $\beta \sim GEM(\alpha)$ (Neal, 2000). Note that the Dirichlet process consists of infinitely many atoms; the term Bayesian non-parametrics stems from this countably infinite number of parameters. The DP can be used to model a countably infinite number of mixtures in a mixture model. Here each θ_k describes the parameters associated with mixture component k and this component has a mixture weight given by β_k.

The Chinese Restaurant Process (CRP) (Neal, 2000; Gershman and Blei, 2012), is an abstraction of the DP. It marginalizes over the probability associated with an atom and instead presents the DP in terms of the clusters that are formed. The CRP imagines a restaurant with an infinite number of tables, tables being the components in the mixture model. Customers (data) arrive at the restaurant and choose a table with probability proportional to the number of customers already at that table, or choose a new table with probability proportional to α. As more customers enter the restaurant, more tables are

chosen with:

$$E[N_t] = \alpha \log(N_c) \tag{4.2}$$

N_t and N_c refer to the number of tables and customers respectively (Gershman and Blei, 2012). The number of components is random and grows as new data are observed. It is important to understand that while conceptually, and for the generative model, there are an infinite number of components, in practice a finite dataset exhibits a finite number of clusters (only a finite number of tables can have customers seated at them) (Blei *et al.*, 2006).

The hierarchical Dirichlet process (HDP) extends equation 4.1 by placing a Dirichlet process prior on many group specific Dirichlet processes (Teh *et al.*, 2005). The associated abstraction is the Chinese Restaurant Franchise (CRF) where restaurants may have menus that offer the same dish, but may also have dishes that are restaurant specific. This allows restaurants to assign different mass to table clusters where the customers still have the same dish. Now customers choose a restaurant in a franchise and choose a dish from the restaurant of choice. The HDP draws G_0 from a Dirichlet process $DP(\gamma, H)$, and it draws group (restaurant) specific distributions $G_j \sim DP(\alpha, G_0)$. The base measure G_0 acts as the expected value for encoding the frequency of each global, shared parameter (Fox *et al.*, 2007).

$$E[G_j \mid G_0] = G_0 \tag{4.3}$$

Hierarchical Dirichlet Process for Hidden Markov Models

We may wish to use the HDP as the clustering prior to infer the parameters and transition probabilities in a HMM. We assume an unknown number of regimes and thus model this with a DP prior. Simply using a DP prior is insufficient for modeling HMM dynamics as the DP would place a static probability on observing the next state $X_t \mid X_{t-1}$ for all possible X_{t-1} which is clearly not the case for the HMM. The transition to state X_t from X_{t-1} must depend on state specific probabilities $\pi_{X_{t-1}}$ and not some global partition prior π. The

HMM therefore involves a set of mixture models that each depend on a specific state. The state indexes a row of the transition matrix, where the probabilities in this row correspond to the mixing proportions for the choice of the next state. We therefore encode this state (regime) dependent transition by using the HDP which still encourages shared structure between the individual transitions. Now each regime m might have its specific transition probabilities π_m but the different regimes might share the affinity to transition to certain 'dominant' regimes.

Fox *et al.* (2009, 2007) discuss problems with the HDP approach. The HDP-HMM inadequately models the temporal persistence of states. Each state is allowed to have a unique transition mixture, with mass shared among states for certain transitions that are more probable. However, it is impossible to encourage self transitions with simply the base hierarchical parameter H. The result is that the HDP-HMM exhibits a rapid inferred switching from one state to the next (Fox *et al.*, 2007). Rather, if we introduce a higher probability of a self transition, we encourage the HMM to have an affinity for remaining in any given regime for a greater length of time. This directly ties to Assumption 2 in Section 2.4.

The adjustment to the HDP-HMM that Fox *et al.* (2009, 2007) propose is to add a self-transition affinity parameter κ. The resulting model is termed the sticky hierarchical Dirichlet process for hidden Markov models (sticky HDP HMM). Inference is performed using a modified Gibbs sampler for the HDP (Teh *et al.*, 2005). This model presents an attractive alternative to Algorithm 1 as the number of regimes is not pre-defined and more importantly, the model specification allows the growth of the number of regimes with the length and/or complexity of the data. This captures the intuitive reality of the simulation more accurately in that we would want more regimes to describe longer and more complex sessions.

An avenue for the extension of the Fox *et al.* (2009) model is to encourage the linear growth of the number of regimes with the length of a given session. The Pitman-Yor

process (Pitman and Yor, 1997) extends the DP for linear growth of clusters (not logarithmic as presented above). Blunsom and Cohn (2011) have applied this concept to develop the hierarchical Pitman-Yor process HMM. It seems natural to extend these models to include the regime affinity parameter κ from Fox *et al.* (2007, 2009).

4.1.2 The SSSM as an Assistive Classroom Tool

With the introduction of rich and complex learning environments, we should remain aware that these simulations pose challenges for teachers who wish to structure learning around a class's session. These teachers may require additional tools to assist them with the domain specific challenges that arise when designing lesson plans around sessions with these simulations. In Section 1.2, we discussed how it is hard for a participant or observer to track the state of the CW simulation. It is therefore also challenging to identify the salient learning opportunities that arise from the students' interactions with the simulation. In addition to this, standard evaluation metrics might not be available for activities engendered by these complex simulations. The investigation of how to integrate these tools into the class is therefore of paramount importance.

Section 3.1 presents a study that demonstrates how the SSSM is used to decompose a large session from CW into small periods that individually are interpretable. This thesis has focused mainly on the time series data itself and has presented techniques for modeling the effects of students' actions on the system state. Future work will investigate the application of these models for producing an assistive system for implementation in the classroom. An investigation into the techniques for presenting a meaningful, holistic picture of the session remains for future studies.

The design of classroom assistive tools should focus on what information they present to the students as well as how they present the information. We propose two principles for choosing information that is relevant to present to students and teachers:

- **Personal salience**: includes scenarios from the simulation experience that are likely to be memorable for the students.
- **Explanatory coherence**: includes a subset of the simulation's causal chains that enables students' discussion of an aspect of the underlying explanatory model.

The work in this thesis has focused solely on the *explanatory coherence* topics. A full assistive model will not only include information that describes system dynamics and changes to the system state, but it will also highlight key elements from the simulation that will be important to the individual students. The proposed future work involves designing, implementing and testing this classroom tool.

Another exciting avenue for future research involves exploring the trade-off that is made between the predictive power of a model and the explanatory coherence that the model achieves. Wu *et al.* (2017) have suggested a method for regularizing deep learning models to facilitate people's understanding of their predictions. This is an important balance to review and one that we intend to consider in educational settings.

4.2 Conclusion

This thesis has made three contributions. Firstly, we have recognized that complex exploratory learning environments might require assistive tools to help teachers and students review meaningful information from a given interaction session. This novel research has studied the possibilities for extracting information from the log files of an exploratory learning environment. We used the Connected World simulation as a test case and represented the log data as a time series. It was our hypothesis that a time series can be decomposed into shorter periods that are individually more interpretable and manageable than the session as a whole.

Secondly, we have applied switching state space models to the task of decomposing the time series into these shorter and individually coherent periods. Our work has built upon previous time series analysis tools and presents an algorithm for learning the change points and regime parameters that are associated with the switching state space model. We have further conducted a survey of possible future work in Bayesian non-parametrics for allowing the data to influence the number of regimes that are inferred in a given session. A flexible number of regimes is especially appealing for a setting such as Connected Worlds where the sessions vary dramatically in length and complexity.

Lastly, we have designed two user-based experiments that test the intelligibility and change point relevance of the model output. The human interpretability of the model was the ultimate goal of this investigation. However, evaluating the different aspects of the model ('learning' vs 'quantification') was a challenging task. Our experiments suggest that the model not only finds a good representation of the periods that might be present in Connected Worlds, but that it also summarizes each period (of about 30 seconds) into a brief two to three lines of text. The text conceptually captures much of the dynamics of an associated video representation of the period. Between the two studies, we show that it is possible to simplify a complex time series into periods of activity that are human interpretable.

We have left the design and implementation of a classroom tool as future work. This tool should aim to interactively support teachers and students for post session reviews. This work presents exciting new possibilities for classroom artificial intelligence tools that can support learning in these rich and immersive exploration environments.

References

Attias, H. (2000). A variational baysian framework for graphical models. In *Advances in neural information processing systems*, pp. 209–215.

Bar-Shalom, Y. and Li, X.-R. (1993). Estimation and tracking- principles, techniques, and software. *Norwood, MA: Artech House, Inc, 1993.*

Blei, D. M., Jordan, M. I. *et al.* (2006). Variational inference for dirichlet process mixtures. *Bayesian analysis*, **1** (1), 121–143.

—, Ng, A. Y. and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, **3** (Jan), 993–1022.

Blunsom, P. and Cohn, T. (2011). A hierarchical pitman-yor process hmm for unsupervised part of speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, Association for Computational Linguistics, pp. 865–874.

Cappé, O., Moulines, E. and Rydén, T. (2009). Inference in hidden markov models. In *Proceedings of EUSFLAT Conference*, pp. 14–16.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P., Riddell, A. *et al.* (2016). Stan: A probabilistic programming language. *Journal of Statistical Software*, **20** (2), 1–37.

Esling, P. and Agon, C. (2012). Time-series data mining. *ACM Computing Surveys (CSUR)*, **45** (1), 12.

Ferguson, T. S. (1973). A bayesian analysis of some nonparametric problems. *The annals of statistics*, pp. 209–230.

Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, **61** (3), 268–278.

Fox, E., Sudderth, E. B., Jordan, M. I. and Willsky, A. S. (2009). Nonparametric bayesian learning of switching linear dynamical systems. In *Advances in Neural Information Processing Systems*, pp. 457–464.

Fox, E. B., Sudderth, E. B. and Willsky, A. S. (2007). Hierarchical dirichlet processes for tracking maneuvering targets. In *Information Fusion, 2007 10th International Conference on*, IEEE, pp. 1–8.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A. and Rubin, D. B. (2014). *Bayesian data analysis*, vol. 2. CRC press Boca Raton, FL.

Gershman, S. J. and Blei, D. M. (2012). A tutorial on bayesian nonparametric models. *Journal of Mathematical Psychology*, **56** (1), 1–12.

Ghahramani, Z. (2001). An introduction to hidden markov models and bayesian networks. *International journal of pattern recognition and artificial intelligence*, **15** (01), 9–42.

— and Hinton, G. E. (2000). Variational learning for switching state-space models. *Neural computation*, **12** (4), 831–864.

Giordani, P., Kohn, R. and van Dijk, D. (2007). A unified approach to nonlinearity, structural change, and outliers. *Journal of Econometrics*, **137** (1), 112–133.

Hoffman, M. D. and Gelman, A. (2014). The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, **15** (1), 1593–1623.

Horst, B. and Abraham, K. (2004). *Data mining in time series databases*, vol. 57. World scientific.

Ikoma, N., Higuchi, T. and Maeda, H. (2002). Tracking of maneuvering target by using switching structure and heavy-tailed distribution with particle filter method. In *Control Applications, 2002. Proceedings of the 2002 International Conference on*, IEEE, vol. 2, pp. 1282–1287.

Jasra, A., Holmes, C. C. and Stephens, D. A. (2005). Markov chain monte carlo methods and the label switching problem in bayesian mixture modeling. *Statistical Science*, pp. 50–67.

Jonsen, I. D., Myers, R. A. and James, M. C. (2007). Identifying leatherback turtle foraging behaviour from satellite telemetry using a switching state-space model. *Marine Ecology Progress Series*, **337**, 255–264.

Kim, C.-J. (1994). Dynamic linear models with markov-switching. *Journal of Econometrics*, **60** (1-2), 1–22.

—, Nelson, C. R. *et al.* (1999). State-space models with regime switching: classical and gibbs-sampling approaches with applications. *MIT Press Books*, **1**.

Li, X. R. and Jilkov, V. P. (2003). Survey of maneuvering target tracking. part i. dynamic models. *IEEE Transactions on aerospace and electronic systems*, **39** (4), 1333–1364.

MacKay, D. J. (1998). Introduction to monte carlo methods. In *Learning in graphical models*, Springer, pp. 175–204.

Murphy, K. P. and Russell, S. (2002). Dynamic bayesian networks: representation, inference and learning.

Nasrabadi, N. M. (2007). Pattern recognition and machine learning. *Journal of electronic imaging*, **16** (4), 049901.

Neal, R. M. (2000). Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, **9** (2), 249–265.

— (2003). Slice sampling. *Annals of statistics*, pp. 705–741.

— *et al.* (2011). Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, **2** (11).

Oh, S. M., Rehg, J. M., Balch, T. and Dellaert, F. (2008). Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *International Journal of Computer Vision*, **77** (1-3), 103–124.

Patel, P., Keogh, E., Lin, J. and Lonardi, S. (2002). Mining motifs in massive time series databases. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, IEEE, pp. 370–377.

Pavlovic, V., Rehg, J. M. and MacCormick, J. (2001). Learning switching linear models of human motion. In *Advances in neural information processing systems*, pp. 981–987.

Pitman, J. and Yor, M. (1997). The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, pp. 855–900.

Preston, D., Protopapas, P. and Brodley, C. (2009). Event discovery in time series. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, SIAM, pp. 61–72.

Saul, L. K., Jaakkola, T. and Jordan, M. I. (1996). Mean field theory for sigmoid belief networks. *Journal of artificial intelligence research*, **4**, 61–76.

— and Jordan, M. I. (1996). Exploiting tractable substructures in intractable networks. In *Advances in neural information processing systems*, pp. 486–492.

Shumway, R. H. and Stoffer, D. S. (1991). Dynamic linear models with switching. *Journal of the American Statistical Association*, **86** (415), 763–769.

— and — (2000). Time series analysis and its applications. *Studies In Informatics And Control*, **9** (4), 375–376.

Smørdal, O., Slotta, J., Moher, T., Lui, M. and Jornet, A. (2012). Hybrid spaces for science learning: New demands and opportunities for research. In *International Conference of the Learning Sciences. Sydney, Australia. http://www. uv. uio. no/intermedia/english/research/projects/miracle/news/pdf/ICLS 2012 Symposium Hybrid_Spaces. pdf*.

Tanaka, Y., Iwamoto, K. and Uehara, K. (2005). Discovery of time-series motif from multi-dimensional data based on mdl principle. *Machine Learning*, **58** (2-3), 269–300.

Teh, Y. W., Jordan, M. I., Beal, M. J. and Blei, D. M. (2005). Sharing clusters among related groups: Hierarchical dirichlet processes. In *Advances in neural information processing systems*, pp. 1385–1392.

Turner, R. E. and Sahani, M. (2011). Two problems with variational expectation maximisation for time-series models. *Bayesian Time series models*, **1** (3.1), 3–1.

Vlachos, M., Lin, J., Keogh, E. and Gunopulos, D. (2003). A wavelet-based anytime algorithm for k-means clustering of time series. In *In Proc. Workshop on Clustering High Dimensionality Data and Its Applications*, Citeseer.

Whiteley, N., Andrieu, C. and Doucet, A. (2010). Efficient bayesian inference for switching state-space models using discrete particle markov chain monte carlo methods. *arXiv preprint arXiv:1011.2437*.

Wu, M., Hughes, M. C., Parbhoo, S., Zazzi, M., Roth, V. and Doshi-Velez, F. (2017). Beyond sparsity: Tree regularization of deep models for interpretability. *arXiv preprint arXiv:1711.06178*.

Appendix A

Plant and Animal Relations

For completeness, we have included a map to highlight the relationships between the plants and animals in Connected Worlds. Students plant trees and observe the arrival of animals that use the trees to support their habitat. Mimicking a real-world scenario, the fauna for a biome depend on the flora, and the flora are unique for a specific biome. Larger and higher level plants require the presence of smaller and lower level plants. Figure A.1 shows a schematic that displays the plant and animal relations of Connected Worlds.

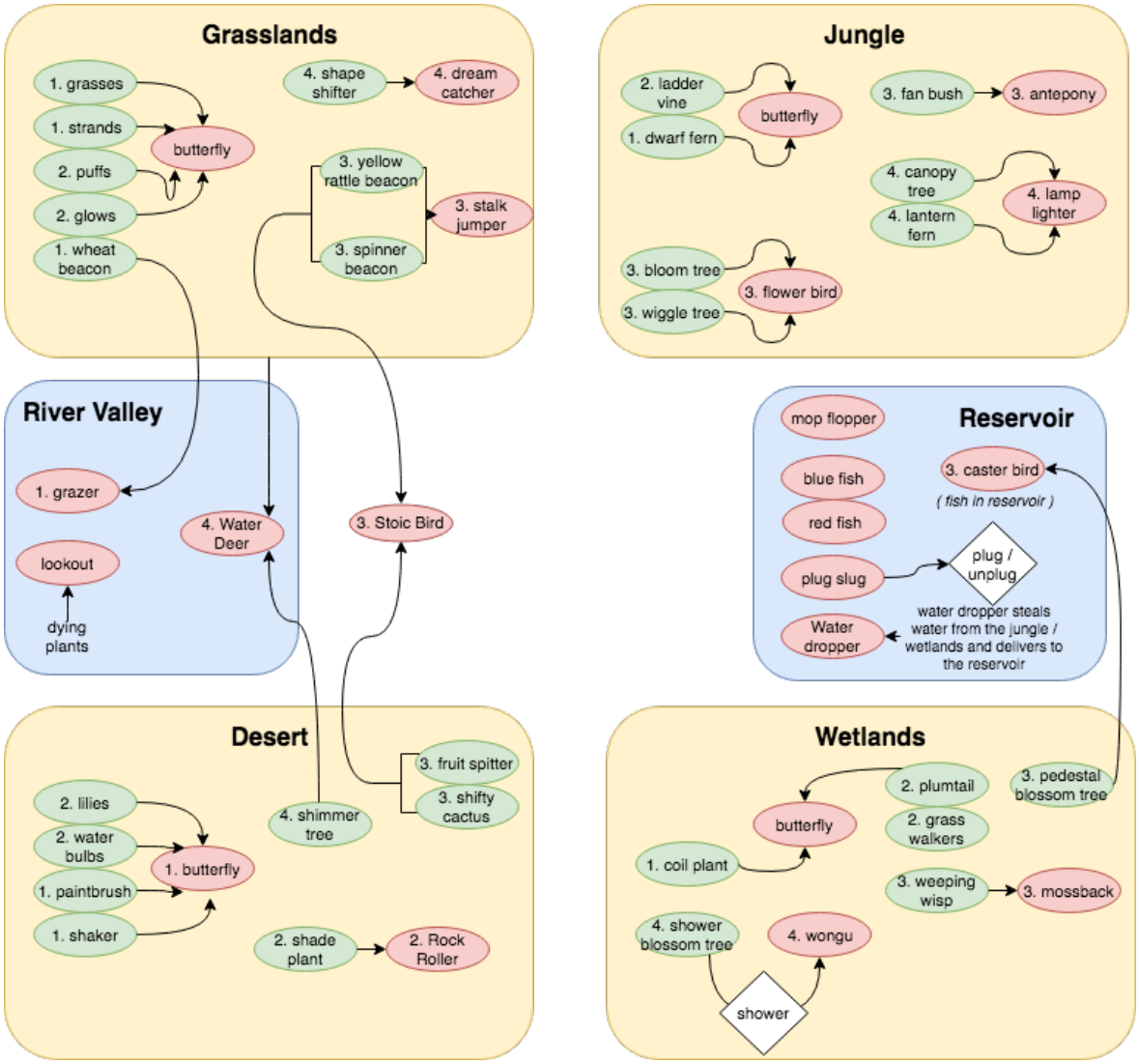


Figure A.1: *Map of the relationships between plants, animals and biomes in the CW environment. The biomes are shown in yellow. The blue boxes correspond to areas of the simulation that do support animals but do not have plants that grow there. The fauna-flora specific relations can clearly be seen by the arrows that link the plants (green ovals) and animals (red ovals). The plants and animals are endemic to their biome.*

Appendix B

Posterior Probability of Correct Period Selection

Figure B.1 presents a visualization of the period boundaries for the 5 sessions that were studied in the text. The x-axis corresponds to time and the y-axis shows the algorithm that was used to generate the boundaries. The vertical yellow lines indicate the inferred period boundaries; a period exists between two yellow boundaries. The periods are labeled sequentially for reference in this discussion. Finally, the color of the period indicates the posterior probability that a human validator will select the algorithmically generated description for that period. Light blue corresponds to a low probability of the algorithmic description being correctly chosen and dark blue or black corresponds to a high probability.

In general we note that the Algorithm 1 periods are darker than the periods of Uniform8; the Uniform8 periods are darker than the periods for Uniform5. We also note that when the period boundaries line up entirely (between different algorithms), the period colors tend to be very similar (refer to *Session 1, Algorithm 1, period 12* and *Session 1, Uniform8, period 8* for one such example). The file difficulty variability is seen with Session 3 being the darkest

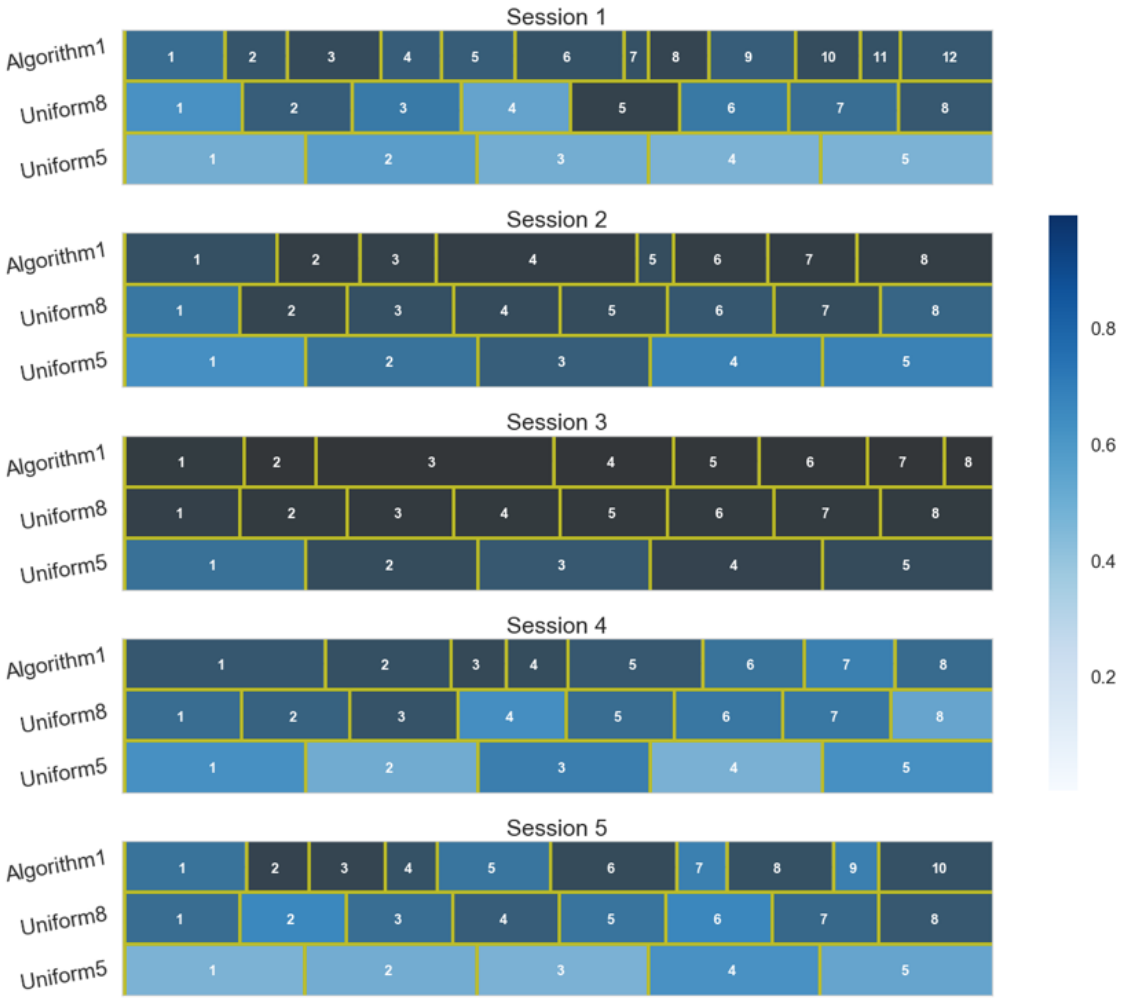
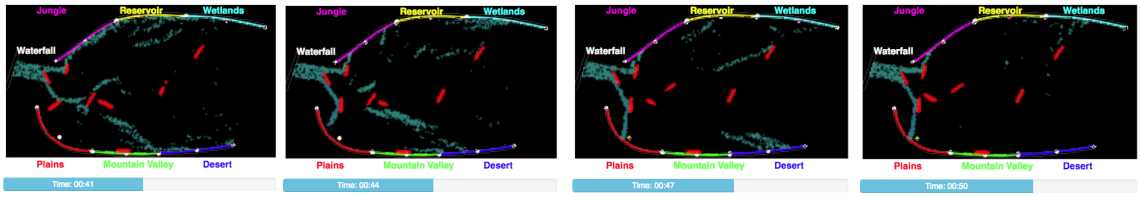


Figure B.1: *Diagram to represent the periods that are inferred by Algorithm 1 in comparison to those defined by Uniform5 and Uniform8. The vertical yellow lines denote the period boundaries and the color of the period presents the posterior probability that an external validator will select the text description that is generated by the algorithm. Dark colors correspond to high probability for being selected and light colors correspond to a low probability.*



(a) *frame 1 (41s)* **(b)** *frame 2 (44s)* **(c)** *frame 3 (47s)* **(d)** *frame 4 (50s)*

Figure B.2: *Sequence of* 4 *snapshots from the video given to validators (each frame is 3 seconds apart). The logs are moved in frames 1 and 2 such that the water is mostly flowing to the Plains and Jungle in frames 3 and 4 and afterwards.*

(easiest) and Sessions 4 and 5 being the lightest on average (i.e., for these two sessions, it was the most difficult to select the algorithm's generated description).

It is interesting to investigate some specific aspects of this visualization. Starting with Session 1, Algorithm 1 has a slightly darker period 1 than the other algorithms. Uniform8 and Algorithm 1 both present the description that "Water is mostly flowing to the Wetlands, Jungle and Plains". Uniform5 describes the longer period as water mostly flowing to the "Jungle, Wetlands and Plains" and so these are all similar. The dummy choices are reasonable in all three cases with one example being: "Water is mostly flowing to the Plains, Jungle and Wetlands". At time $t = 44s$, the end boundary for Algorithm 1, period 1, the students position the logs to direct a substantial amount of water to the Jungle and to the Plains. Algorithm 1 correctly detected this change and defined a new period 2. In the other algorithms, as period 1 was extended into this dominant Jungle/Plains dynamic, the descriptions became more difficult to choose from (hence the lowest probability for a successful choice is seen in Uniform5). Figure B.2 shows four snapshots around the $44s$ change point where after $44s$, the water mainly flows to the Jungle and the Plains. It qualitatively seems natural to have a change point at this time.

Session 5, Algorithm 1, has light periods 7 and 9 interspersed by darker periods. For both of these periods, the Algorithm 1 description is misleading. The ordering of the parameters conveys a hierarchical structure to the message that is being conveyed. This is not always correct. In both of these cases, the parameters were actually similar in magnitude

and thus the strict ordering of the description may have been misleading. In Section 4.1.2, we discuss how future work will entail correctly compiling the available information to design an assistive aid for teachers. Making decisions regarding how the data are presented will certainly form an important aspect of this work.

A final point that can be made from this session is that the rain events of the system are poorly identified by the algorithm and can thus lead to misleading descriptions that are generated. It remains a challenge to find a robust way for dealing with rain events. The simplest recommendation is to add these events to the log files that are stored as part of the output from CW.