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

Author:
Nicholas S. Hoernle

Change Point Modeling and Regime Learning in Markov Switching Systems
Modeling the Effects of Students' Interactions with Immersive Simulations

Abstract

TODO - abstract

Contents

List of Figures

Introduction

TODO

Chapter 1

Connected Worlds

1.1 Connected Worlds

1.1.1 Introduction

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 the 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/

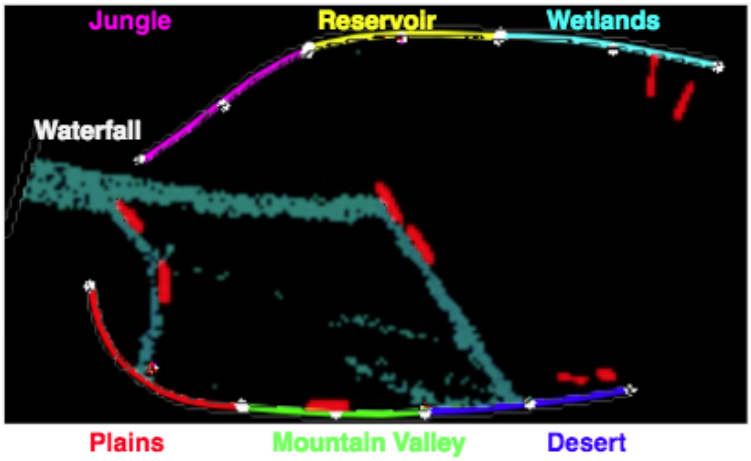


Figure 1.1: *Bird's eye view of the CW simulation. Biomes are labeled on the perimeter and logs appear as thick red lines. Water enters via the waterfall and in this image it is mainly flowing from left to right toward the Desert and the Plains.*

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 is a usable resource as it can then be directed to the biomes. Certain sources of water are under the students' direct control and others are out of their 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 specific 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. Figure 1.1 shows a bird's eye snapshot view of the state of logs and water in the simulation in CW. Not seen in this snapshot are the plant and animal counts or levels in the biomes. The amount of water that is stored in each of the biomes is also not shown.

1.1.2 Water Cycle

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 can be directed to the biomes where it is used to sustain plants and animals. Evaporation, dependent on the numbers and

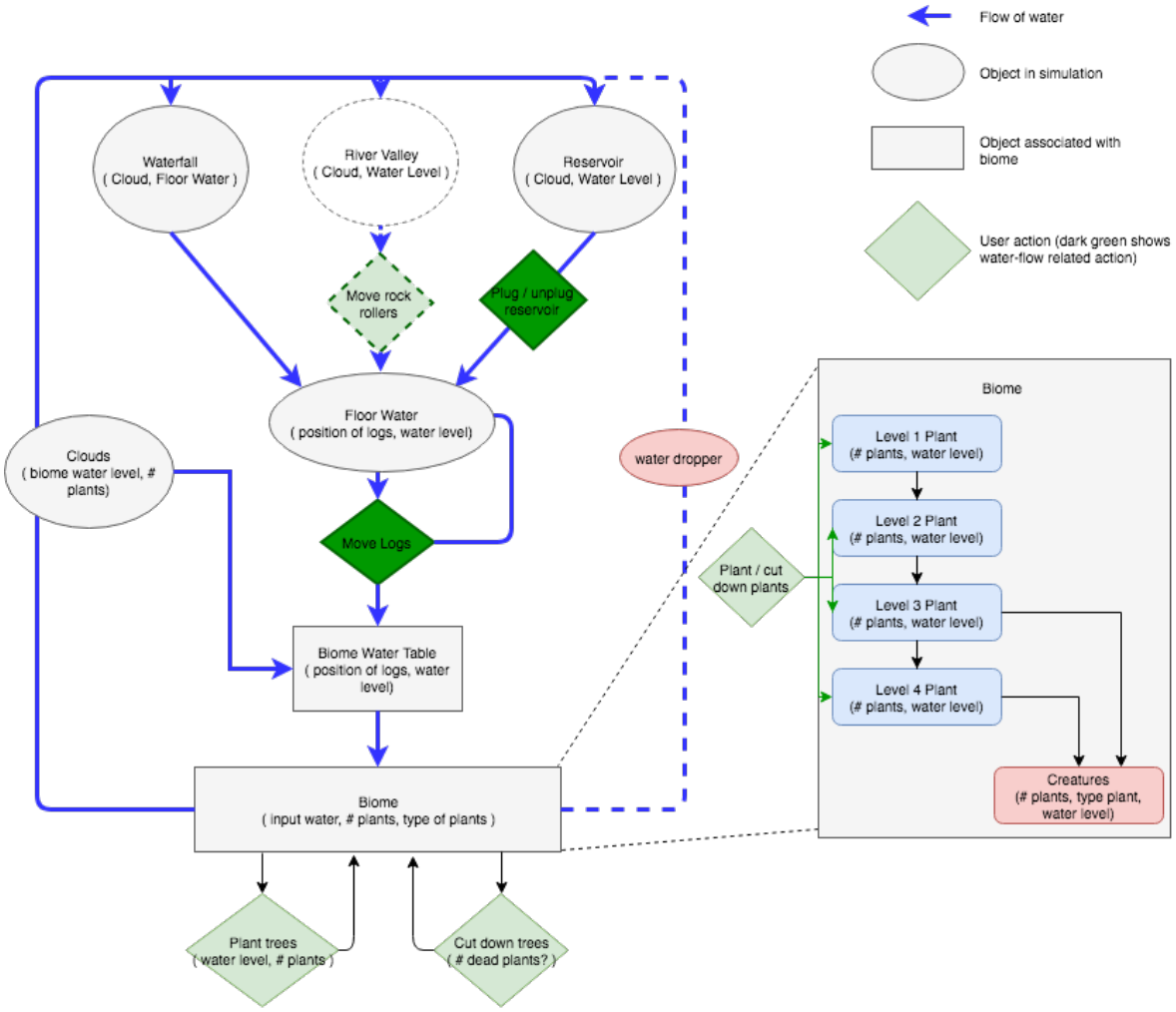
levels of plants, and rainfall, dependent on the amount of evaporation, brings the water back to the water sources where it once again becomes a usable resource.

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 Reservior. The Waterfall flows whenever water is available and thus the amount of input water from the Waterfall is out of the studnets' control. The Mountain Valley also flows when it rains in this area[2]. Students' choose when to release 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 the 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 are able to plant trees. Higher level trees require the presence of lower level trees. The number and level of trees affects the amount of water that the biome requires. Lastly, when water is present in a biome, studnets 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 depends on the flora, and the flora is 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 given simulation might be to attract these animals. Students are thus required to manage

[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



Water Cycle

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 are often not present or are negligable. Green, diamond boxes represent the actions that the students can take that will change the water cycles that are present. Grey circles are areas of the simulation and grey boxes are specifically one of the four biomes.*

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 Figure 1.3.

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 simulated environment. No one person - including the teacher or interpreter - can possibly follow what happens, even in a relatively short simulation. Each participant will have a different view of what transpired, depending on the actions he or she took and the state changes that resulted. We propose that it is important to develop tools that can support teachers' understanding of the effects of students' interactions in complex exploratory learning environments such as CW. We introduce a technique for decomposing a complex session into shorter periods where the dynamics of the simulation are stable. Our hypothesis is that these smaller periods will individually be more interpretable than the session as a whole and they will eventually form a subset of useful information that a teacher can draw on when leading a review of the session with groups of students.

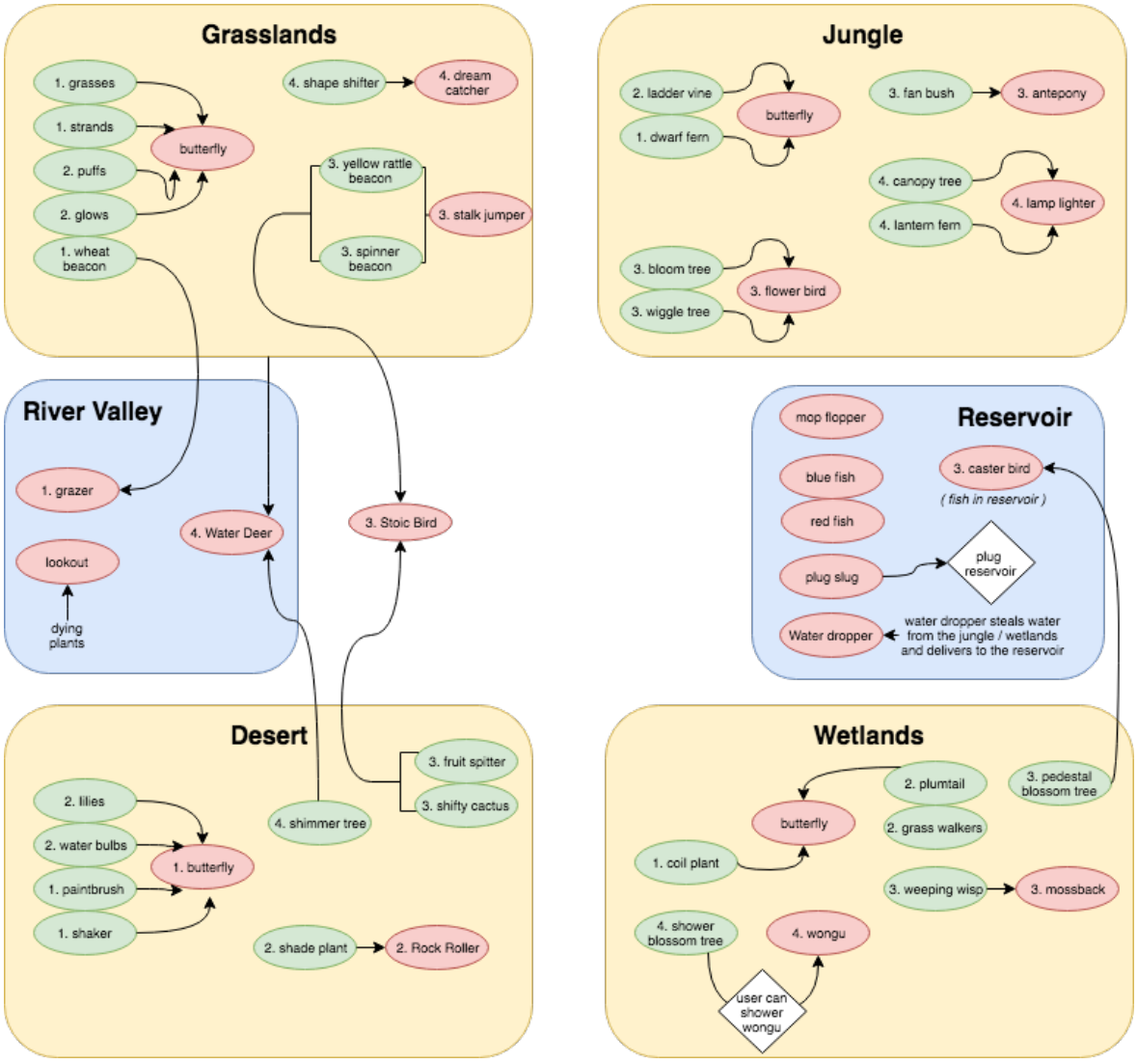


Figure 1.3: *Map of the relationships between plants, animals and biomes in the CW environment. The biomes are shown in yellow. The blue boxes correspond to areas of the simulation that do support animals but do not have plants that grow there. The fauna-flora specific relations can clearly be seen by the edges that link the plants (green ovals) and animals (red ovals). The plants and animals are endemic to their biome.*

Chapter 2

The Switching State Space Model

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 allowing 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 yet 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) but 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 purposes 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). In contrast to the switching systems that we discuss, 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. These previous 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.

Ghahramani and Hinton (2000) introduce and give a detailed presentation of the switching state space model. 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 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 and we propose an algorithm to perform posterior inference over the latent variables in this model.

Before describing the SSSM in Section 2.1.2, 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 towards constructing the SSSM.

2.1.1 Background

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 multinomial variable that can take one of M possible discrete values. 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 the number of possible observation states. Given the hidden state at time t in an HMM, the associated observation is independent from 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 $k \in 1, 2, \ldots, t-2$) given the value of state $\mathbf{X_{t-1}}$. The joint probability for the sequence of states and observations can be factored as[3]:

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

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 Guassian distributions (Ghahramani and Hinton, 2000). The linear Gaussian SSM can be defined by the tuple (A, C, Q, R) in:

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

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

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 $v_t \sim \mathcal{N}(0, R)$ 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 $1 \dots t$. 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.

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 $y_t \mid X_t \perp\!\!\!\perp y_{1:t-1}, X_{1:t-1}$ and $X_t \mid X_{t-1} \perp\!\!\!\perp y_{1:t-1}, X_{1:t-2}$. 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.

Backward Algorithm

It is not always the case that we need to analyze time series data in a streaming fashion (i.e. where data arrive in real time and we can only use past and present data to make state estimation inference). When the data is 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 forwards filter and a new term that can be factored similarly to equation 2.3.

$$\begin{aligned} P(X_t, y_{1:T}) &= P(y_{1:t}, y_{t+1:T}, X_t) = P(y_{1:t}, y_{t+1:T} \mid X_t)P(X_t) \\ &= P(y_{t+1:T} \mid X_t)[P(y_{1:t} \mid X_t)P(X_t)] \end{aligned} \tag{2.4}$$

$P(y_{t+1:T} \mid X_t)$ 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:

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

We can calculate the term $P(y_{t:T} \mid X_t)$ in terms of $P(y_{t+1:T} \mid X_{t+1})$ thereby defining the set of backward recursions that are needed to fully calculate the joint probability in

Equation 2.4. This recursion is given by:

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

One would therefore start the backward recursion from time T to successively calculate the joint probabilities of the states $X_{t:T}$. From Equation 2.4, it can be seen that the forward and backward recursions can both be 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.

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 $X_t = i$. 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.

$$\pi_{t,i} = \max_{\forall X_i} \left\{ \pi_{t-1,j} P(y_t \mid X_i) P(X_i \mid X_{t-1}) \right\} \tag{2.7}$$

The algorithm uses the maximum probability from the previous state along with the observation probabilities, for decoding the most likely present state. These can be stored in a table and the most likely state sequence is the maximum at each time step.

2.1.2 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:

$$\begin{aligned} \mathbf{X_t^{(m)}} &= \Phi^{(m)}\mathbf{X_{t-1}^{(m)}} + w_t^{(m)} \\ \mathbf{Y_t} &= S_t A^{(m)}\mathbf{X_t^{(m)}} + v_t \end{aligned} \tag{2.8}$$

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 the states and observations are assumed to follow Gaussian distributions, each regime can be viewed as a SSM as discussed in section 2.1.1.

Figure 2.1 presents a graphical representation of a SSSM. Edges between nodes represent conditional dependencies, gray nodes are observed variables and the 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.

2.1.3 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

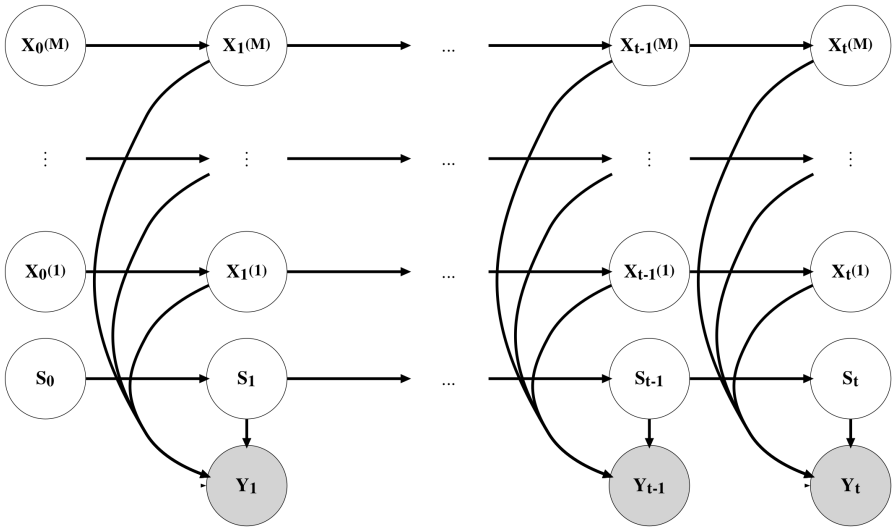


Figure 2.1: *Graphical model for the switching-state space model. A latent discrete switching variable (S_t) selects an active, real-valued state space model ($X_t^{(m)}$). The observation vector (Y_t) depends on the active regime at time t.*

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 level Y_t.

A single regime is insufficient for modeling the effects of students' interactions with CW. This is because students' actions have a complex impact 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 to 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. This duration is called a period. In our example, in one period, water is mainly flowing to the Plains and to the Desert. In the next period, students move the logs to re-route water flow to the Jungle (potentially because plant life is dying). The students may also release water from the Reservoir to direct that to the Desert. These dynamics will be captured by the second period's regime parameters. The output of the model is a number of regimes (that

are active for the corresponding periods) that together capture the system dynamics, but individually can easily be interpreted.

Note that 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 VAR process with observations $\mathbf{y_t}$ is defined by:

$$\mathbf{y_t} = \sum_{i=1}^{r} A^{(m)} y_{t-i} + w_t^{(m)} \qquad w_t^{(m)} \sim \mathcal{N}(0, Q^{(m)}) \tag{2.9}$$

Equation 2.9 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:

$$\mathbf{X_t} = \begin{bmatrix} A_1 & A_2 & \dots & A_r \\ I & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & I & 0 \end{bmatrix} \mathbf{X_{t-1}} + \begin{bmatrix} I \\ 0 \\ \vdots \\ 0 \end{bmatrix} w_t^{(m)} \tag{2.10}$$

$$\mathbf{y_t} = \begin{bmatrix} I & 0 & \dots & 0 \end{bmatrix} \mathbf{X_t} \tag{2.11}$$

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 2.2.3 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.

2.2 Inference for 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 2.8. Computing posterior distributions for SSSMs is computationally intractable (Murphy and Russell, 2002; Kim, 1994)[4]. To illustrate, in Figure 2.1 note that the graph 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 graph). 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).

Due to the intractability, previous approaches use approximation methods such as variational inference (Ghahramani and Hinton, 2000) or a 'merging of Gaussians' (Kim *et al.*, 1999; Murphy and Russell, 2002) to address the inference problem. The variational inference approximation transforms the intractable Bayesian expectation problem into an optimization problem by minimizing the KL 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 uses a single Gaussian to approximate the mixture of M Gaussians at each time step. While these methods have seen success in previous examples, they cannot be applied to our domain. This is because they allow the system to move back and forth between regimes, resulting in frequent regime changes that can hinder the interpretability of the model. This work takes a different approach by imposing structure on the model to address both inference and interpretability challenges. Another avenue for inference is the sticky hierarchical Dirichlet process hidden

[4]Discussed further in 2.2.1

Markov model (stick HDP-HMM) (Fox *et al.*, 2009, 2007), an extension of the hierarchical Dirichlet process (HDP) (Teh *et al.*, 2005), that relaxes the within cluster exchangeability assumption of the HDP. We discuss this model in further detail in Section 4.1.1 and present it as a promising avenue for future work.

To perform inference over the regime parameters and the switch assignments, 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., do we stay in the current regime or do we 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. We further hypothesize that the forced reduction in complexity of the fitted model would significantly simplify the interpretability of the model for a human.

2.2.1 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 2.1.1). 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 makes a 'Gaussian merging' assumption (Ghahramani and Hinton, 2000) to make the inference tractible. 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:

$$X_{t|t} = E[X_t \mid y_{1:t}] \tag{2.12}$$

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 $y_{1:t}$:

$$P_{t|t} = E[(X_t - X_{t|t})(X_t - X_{t|t})' \mid y_{1:t}] \tag{2.13}$$

With the introduction of the switching variable S_t, we require the expectation $X_{t|t}^{(i,j)}$ 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 state and the covariances associated with these estimators are:

$$X_{t|t}^{(i,j)} = E[X_t \mid y_{1:t}, S_{t-1} = i, S_t = j] \tag{2.14}$$

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

In computing the Viterbi algorithm, we would require the expectations and covariances in Equations 2.14 and 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 2.2.3. 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 posterior 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. $X_{t|t}^{(j)}$ represents the $E[X_t \mid S_t = j, y_{1:t}]$ with the $S_t = i$ 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[5].

Finally, we can use numerical optimization to perform parameter estimation by using an approximate data log likelihood that is defined by the filter with the Gaussian merging approximation.

2.2.2 Background on Inference for Probabilistic Models

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 considered part of the prior knowledge and as such inference about the latent variables have a strong interpretability within the defined model's structure. Bayesian statistics is derived from Bayes rule:

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

[5] see the derivation in Kim (1994) for further details.

The denominator in Equation 2.17 represents the normalizing constant that is not dependent on the parameter vector θ. This integral is often intractable and thus evaluation of the posterior $P(\theta \mid X)$ becomes a challenging objective. Conjugate structure in certain problems allows for the direct computation of an analytical solution for the posterior distribution. This structure is often not available, or conjugacy imposes undesirable restrictions given the problem setting. Thus other means for *approximating* the posterior distribution are required when performing inference in Bayesian models. Variational inference (Attias, 2000; Saul and Jordan, 1996; Saul *et al.*, 1996; Blei *et al.*, 2003) is an approach that is 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). Monte Carlo (MC) approximations present a different avenue for the approximation of an intractable posterior distribution. MC techniques are appealing due to their strong convergence properties albeit at a cost of high computation (MacKay, 1998).

MC sampling is a popular technique for performing posterior inference in Bayesian statistics as it presents an 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 but these techniques are shown to scale poorly to high dimensional and tightly correlated distributions. 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 family of MC techniques that use current samples to propose new, dependent samples are termed Markov Chain Monte Carlo (MCMC). When explicit structure of the domain allows the derivation of complete conditionals[6] (MacKay, 1998), Gibbs sampling presents an attractive MCMC technique. More generally, the Metropolis method is used when complete conditionals are not available.

Markov Chain Monte Carlo

Metropolis is an algorithm for sampling from an unknown distribution with density $\frac{1}{Z}f(x)$. Here, $f(x)$ is an unnormalized 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:

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

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 $\frac{1}{Z}f(x)$. 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. Variants of the

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

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 which modifies HMC by tuning parameters automatically.

Hamiltonian Monte Carlo and the No U-Turn Sampler

Hamiltonian Monte Carlo (Neal *et al.*, 2011) is inspired by statistical mechanics where particles in space are known to interact and follow trajectories that are defined by Hamiltonian mechanics. 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 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 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.

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. The leapfrog method presents a small adjustment to Euler's numerical scheme by alternating between the momentum and position updates with a half-step spacing these integration 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 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. The NUTS sampler is implemented in a number of probabilistic programming languages to enable automatic inference in complex user defined models. We use Stan-MC[7] (Carpenter *et al.*, 2016) for conducting inference in the model defined in Section 2.2.3. All inference implemented in this study is done using the NUTS sampler and in implemented using Stan-MC.

2.2.3 Algorithm for Posterior Inference of Switching State Space Models

Nonidentifiability and Label Switching

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 $\mathbf{Y}$. A well known problem with MCMC Bayesian inference in complex graphical models with hidden variables is that of identifiability (Jasra *et al.*, 2005). Models are nonidentifiable 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

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

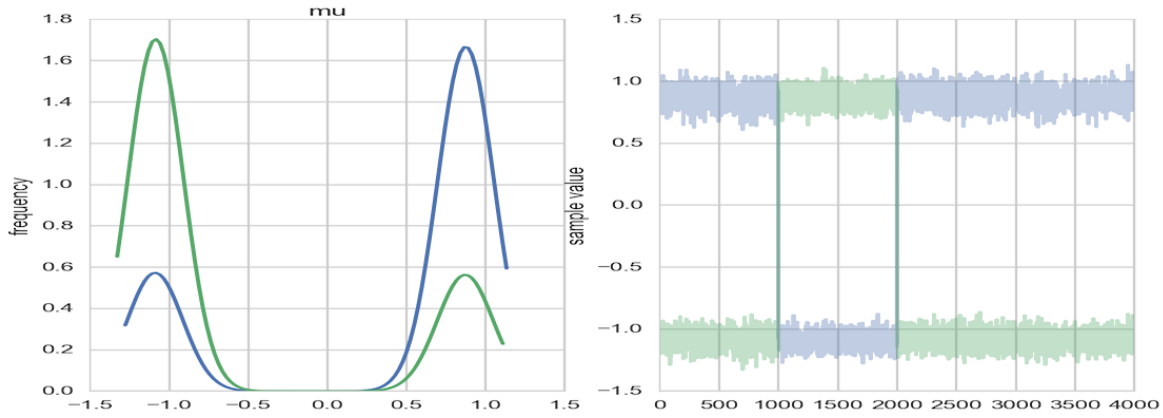


Figure 2.2: *Posterior samples and associated density plot from the posterior of a Gaussian mixture model with true parameters $\mu_1 = -1, \mu_2 = 1, \sigma_1 = \sigma_2 = 0.75$. Left shows the density plot for the posterior of the mean parameters. Right shows the posterior sample trace. The label switches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

sampling and the resulting posterior distribution is not class specific but rather multimodal 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 presents 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 clearly be seen at samples indexed 1000 and 2000 respectively. The smeared posterior distributions for the mean parameters is also clearly evident. If the chain was not truncated (and if other MCMC chains were run), the posterior 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.

Algorithm Sketch

Another solution for solving 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 $t < l \leq t+K$. Therefore, the value of S_l determines the values for all of the unknown states as S_t is assigned to regime m for $t < l$ and it is assigned to regime $m+1$ for $t \geq l$.

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 a posteriori 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 posterior distributions).

Algorithm 1 can be computed on a 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 (and related) as required by domain knowledge.

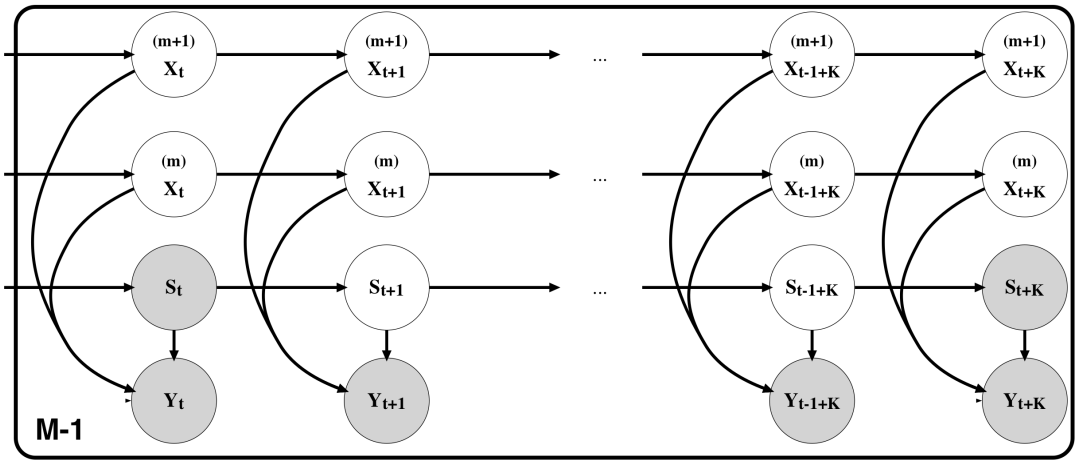


Figure 2.3: *Updated graphical model showing the semi-supervised switching labels, along with the choice of only two chains between two semi-supervised points. This representation is repeated $M-1$ times to describe the $M-1$ switches between the M regimes. Note that the labeled switch variables at the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

Chapter 3

User Study

3.1 User Study Empirical Validation

We evaluate two aspects of Algorithm 1. First, we show that it finds a good representation for the true change points in a synthetic dataset. Second, we 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. However, between the two experiments, we evaluate the interpretability of the joint inference over the periods and the associated regime parameters. The evaluations suggest that the proposed techniques produce results that do have interpretabilty in the CW domain.

3.2 Evaluation on Synthetic Data

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

Equation 3.1 (an adapted model from Ghahramani and Hinton (2000) where the state space is disjoint at regime switches) describes a SSSM with two regimes and a continuous state space. 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 switching variable that chooses between the two regimes at every time step. The prior probability of each of the regimes is 0.5[8] 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) that is discussed in Section 2.2.1. Each period is represented by a sequence of black and gray colored circles. Note that qualitatively it is a challenging task to distinguish between the regime labels by merely observing the time series values. 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 (high frequency 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] $P(S_1 = 1) = p_1 = P(S_1 = 2) = p_2 = 0.5$

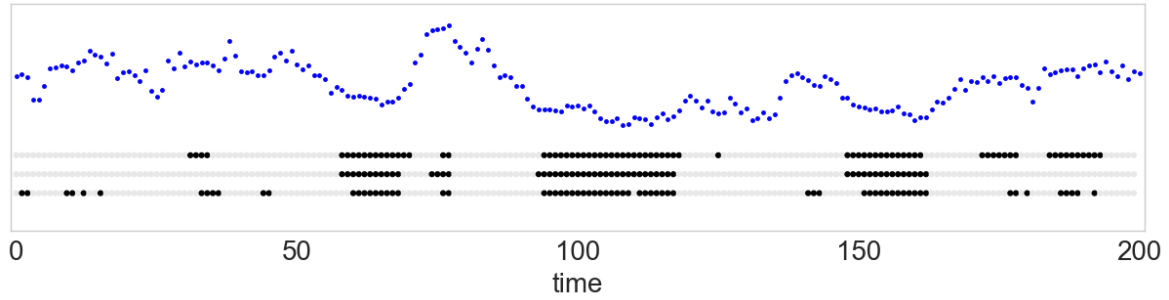


Figure 3.1: *An example of a generated time series from the SSSM model of Equation 3.1. The x axis represents time, and the y axis shows the observations. Regime labels are shown as black and gray dots representing the two label options. True labels (top) are compared to the inferred labels from Algorithm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

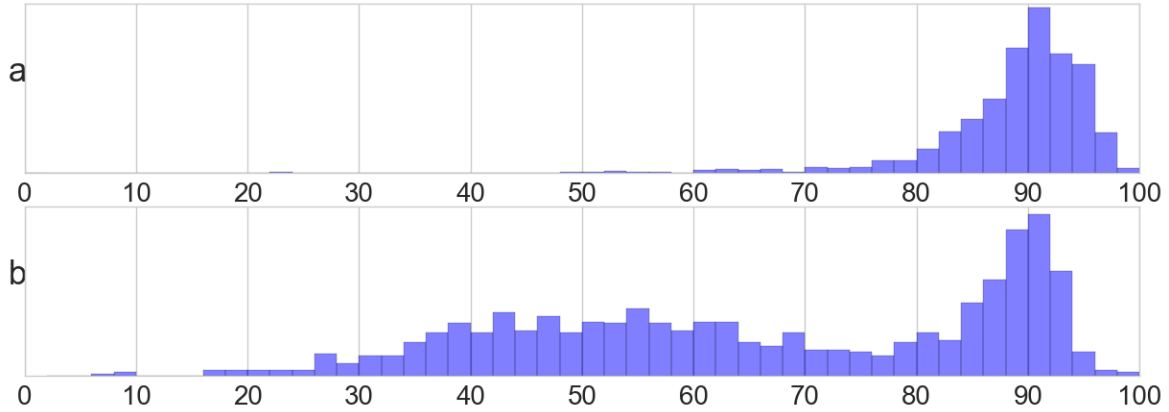


Figure 3.2: *Histogram of the percent of correctly inferred labels for the observed output. The structured sampling Algorithm 1 (a) learns the regime labels more accurately than the uninitialized Gaussian merging algorithm (b).*

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 the 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. We therefore use a hierarchical structure to link the interleaving regimes which force the model to switch between the two regimes at successive switch points. The number of regimes for the Gaussian merging approach is set to two.

3.3 Validation of Model Interpretability

Algorithm 1 is aimed 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 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 that are found by Algorithm 1 (presented in Section 3.3.1) and to verify the intelligibility of the automatically generated descriptions for the inferred periods (presented in Section 3.3.3).

3.3.1 Experiment 1

We designed an experiment[9] that asked evaluators to select one of three possible switch points between every pair of consecutive periods. Evaluators saw a composite of 1) the

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

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 changes in the system that indicate different dynamic regimes and those 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. The results indicate that the evaluators chose the switch point identified by Algorithm 1 significantly more often than the random baseline ($p < 1 \times 10^{-4}$), suggesting that the inferred switch points were, indeed, interpretable to a large extent as meaningful changes in the state of the system. The differences in interpretability seen in Figure 3.3 (e.g. Session 4 was more difficult to interpret than Session 3) can provide further guidance to us in how to support teachers and students in making sense of their experiences in CW.

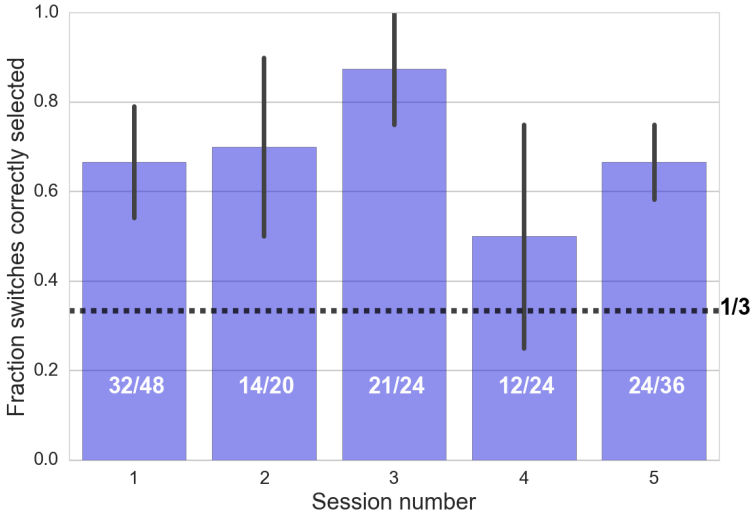


Figure 3.3: *Expert validation of five different test files from sessions with CW. The bar plot shows the fraction of correctly identified switches between automatically identified periods.*

3.3.3 Experiment 2

Experiment 2 aimed 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 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 screenshot of the visualization[10] 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.

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

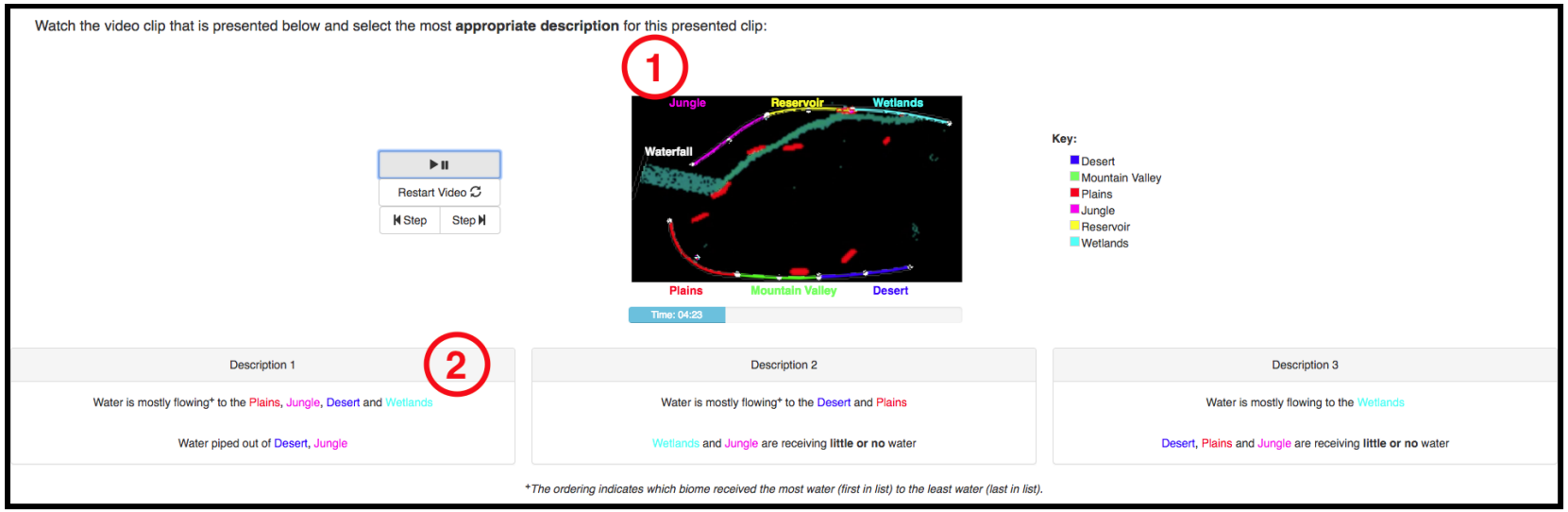


Figure 3.4: *Screenshot of the user interface designed to evaluate the interpretability of Algorithm 1. The video representation at (1) is played for the duration of the period. Three plausible descriptions (of which (2) is one of these) are presented and the validator is asked to select the description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

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 screenshot in figure 3.4) was presented 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 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 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 has a marked increase in the interpretability of the inferred description and video clip pairs. It is also interesting to note the same general structure of the bar plot as that seen in Figure 3.3, with file 3 having the highest success rate and file 4 having the lowest. The file appears to affect the interpretabilty of the periods in a similar maner for both experiments. This is intuitively correct as sessions where the system

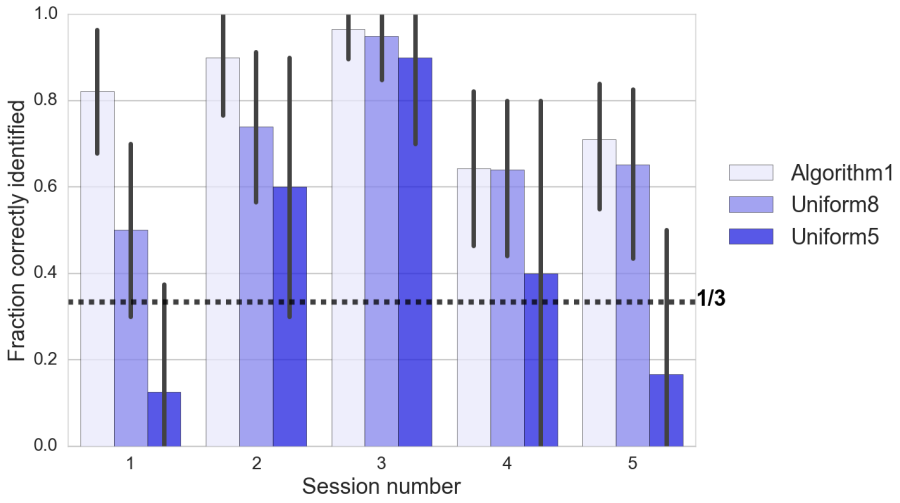


Figure 3.5: *Bar plot of the results from Experiment 2. There are 5 files that were tested. The fraction of correctly selected period descriptions from Algorithm 1 is compared to that from the control conditions. The control conditions include uniformly spaced periods for* 8 *and* 5 *periods respectively.*

dynamics are more stable (possibly due to students executing fewer actions or having a simpler goal) might make for simpler and more easily understand descriptions. However, in cases when there are possibly many conflicting student agendas, the resulting dynamics might be complex and hard to interpret.

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 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 and the effect of the algorithm used. Each period specific parameter p_i represents the log-odds of a successful

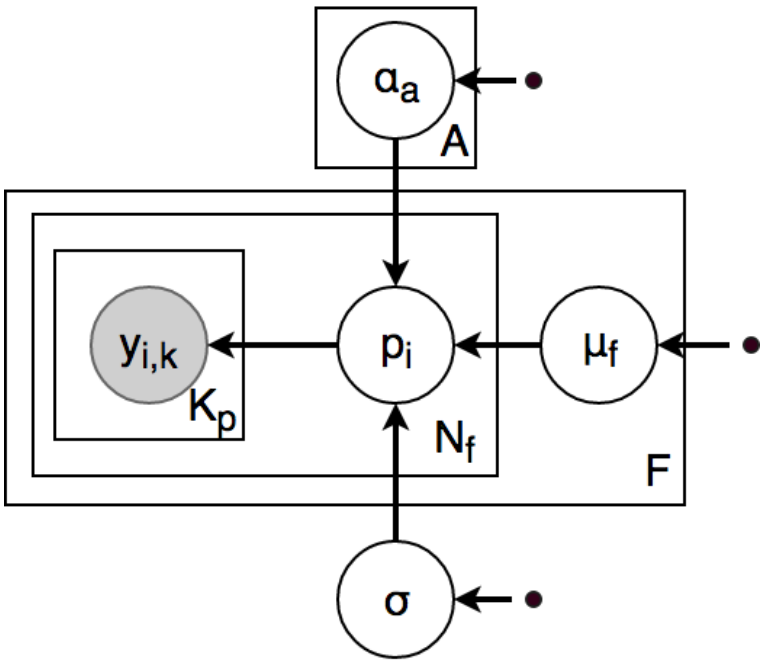


Figure 3.6: *Graphical model for the hierarchical logistic regression that is conducted to determine the intrpretability effect of Algorithm 1 and Uniform8 over the base Uniform5 condition. The observed data y corresponds to a Bernoulli trial where a description is selected to match a video clip or not. 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.*

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 $\mathcal{N}(0,1)$ prior on this parameter. We conduct inference over the algorithm intrpretability effect parameters (α_a) and the period specific log-odds parameters (presented in Appendix (TODO)).

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

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 with 5 splits. 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 discernable 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 otherwords, 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 Algotihm 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 pos-

terior probability for p_i for each of the presented periods. This discussion is given in Appendix(TODO).

Chapter 4

Conclusion

4.1 Future Work

4.1.1 Bayesian Non-parametric Learning for the SSSM

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

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

In Equation 4.1, δ refers to the derac delta function and represents the atoms of G that are distributed according to θ_k G_0 and have mass β_k associated with them. Note that in the Dirichlet process formalism, the number of atoms is assumed to be ∞, and thus the term

Bayesian non-parametrics stems from this countably infinite number of parameters. The Chinese Restaurant Process (CRP) (Neal, 2000; Gershman and Blei, 2012), is an abstraction that marginalizes over the probability associated with an atom and instead presents the Dirichlet process in terms of the clusters that are formed. The CRP imagines a restaurant with an infinite number of tables (components in the mixture model). Customers (data) arrive at the restaurant and choose a table proportional to the number of clients already at that table n_k, or choose a new table with probability proportional to α. As more customers enter the resturant, more tables are chosen with $E[N_t] = \alpha \log(N_c)$ (N_t and N_c refer to the number of tables and customers respectively). The DP can be used as a mixture model by assuming θ_k denotes some cluster specific paramters and where the number of components is random and grows as new data are observed. It is important to understand that while concepturally there are an infinite number of components, in practice a finite dataset exxhibits a finite number of clusters (only a finite number of tables can have customers seated at them).

The hierarchical Dirichlet process (HDP) extends Equation 4.1 by placing a Dirichlet process prior on the Dirichlet process (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. The HDP draws G_0 from a Dirichlet process $DP(\gamma, H)$, and then it draws group (restaurant) specific distributions $G_j \sim DP(\alpha, G_0)$. The base measure G_0 now acts as the expected value for encoding the frequency of each, global shared parameter ($E[G_j \mid G_0] = G_0$).

We may wish to use the HDP as the clustering prior to infer the HMM clustering parameters. 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 $X_t \mid X_{t-1}$ which is clearly not the case for the HMM. The transition to state X_t from X_{t-1} must depend on state dependent

probabilities π_X and not some global partition prior π. We therefore encode this regime dependent transition by using the HDP which still encourages structure in the individual transitions. Now each regime j might have its specific transition probabilities π_j but the different regimes might share the affinity to transition to certain 'dominant' regimes. Fox *et al.* (2009, 2007) demonstrate the problems with this approach. While we have allowed a reasonable clustering property to find the regimes in the HMM, and each regime is allowed to have its regime specific transition probabilities, it is impossible to encourage self transitions with simply the base hierarchical parameter H. The self transition states that the HMM should have an affinity for remaining in any given regime, when it is in that regime. This directly ties to Assumption 2 in Section 2.2.

Fox *et al.* (2009, 2007) present an adjustment to the HDP, by adding a self-transition affinity parameter K. 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 is allowed to grow with the length and/or complexity of the data. 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 cluters. Teh (2006) has applied this model to the HDP setting. It seems natural to extend these models to Fox *et al.* (2009) sticky-HDP HMM model.

4.1.2 The SSSM as an Assistive Classroom Tool

TODO: Discuss implementation in the classroom here

4.2 Conclusion

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

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.

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.

— *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.

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. (2006). A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, pp. 985–992.

—, 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*.