8-2009

Pattern Recognition for Command and Control Data Systems

Jason Schwier
Clemson University, jschwier@gmail.com

Pattern Recognition for Command and Control Data Systems

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Computer Engineering

---

by
Jason Montgomery Schwier
August 2009

Accepted by:
Dr. Richard R. Brooks, Committee Chair
Dr. Christopher Griffin
Dr. Adam Hoover
Dr. Christopher Post

Abstract

To analyze real-world events, researchers collect observation data from an underlying process and construct models to represent the observed situation. In this work, we consider issues that affect the construction and usage of a specific type of model. Markov models are commonly used because their combination of discrete states and stochastic transitions is suited to applications with both deterministic and stochastic components. Hidden Markov Models (HMMs) are a class of Markov model commonly used in pattern recognition. We first demonstrate how to construct HMMs using only the observation data, and no a priori information, by extending a previously developed approach from J.P. Crutchfield and C.R. Shalizi. We also show how to determine with a level of statistical confidence whether or not the model fully encapsulates the underlying process.

Once models are constructed from observation data, the models are used to identify other types of observations. Traditional approaches consider the maximum likelihood that the model matches the observation, solving a classification problem. We present a new method using confidence intervals and receiver operating characteristic curves. Our method solves a detection problem by determining if observation data matches zero, one, or more than one model.

To detect the occurrence of a behavior in observation data, one must consider the amount of data required. We consider behaviors to be "serial Markovian," when the behavior can change from one model to another at any time. When analyzing observation data, considering too much data induces high delay and could lead to confusion in the system if multiple behaviors are observed in the data stream. If too little data is used, the system has a high false positive rate and is unable to correctly detect behaviors.

We demonstrate the effectiveness of all methods using illustrative examples and consumer behavior data.

Dedication

To those who were, to those who are, and to those who will be.

Acknowledgments

First and foremost, this dissertation could not have become a real physical object without my advisor, Dr. Richard R. Brooks. I never understood all of your recommendations, but looking back now, I can see that everything was done in the best interests of myself and my fellow graduate students. Thank you for enabling me to achieve all that was done for this dissertation.

Second, I would like to thank Dr. Christopher Griffin for your contributions to this final product. Your mathematical insight and thoughts on how to approach the problems discussed in this dissertation made it possible for much of this work to be completed.

Third, thank you to my parents and Lisa for their dedication and time spent reading the drafts of this dissertation. Mom and Dad, you are always there to help me when my eyes and thoughts are not enough. I cannot thank you both enough for everything that you have done for me. Lisa, thank you for putting up with the many quiet evenings where I was not available to talk or spend time with you because of the work on this document.

Fourth, there are many individuals at Clemson that deserve my thanks for their contributions to my well-being. It is impossible to list everyone who helped me through my time here at Clemson University. I have met many individuals who enriched my life and gave me new perspectives on how the world works. It is unlikely you are reading this dissertation, but if you are, you will know if I am referring to you.

Fifth, this work would not have been possible without the monetary resources of the United States taxpayer. Thank you to the Congress for appropriating money to the Office of Naval Research. This material is based upon work supported by, or in part by, the Office of Naval Research Code 311 contract/grant number N00014-06-C-0022.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This dissertation is the culmination of several years of work on Markov models, the extension of hidden Markov models (HMMs), and a generalization developed by C.R. Shalizi and J.P. Crutchfield, the ϵ-machine. Markov models and HMMs are used extensively in pattern recognition applications or other systems with deterministic and stochastic components. Applications for Markov models include voice recognition [1, 2, 3, 4], texture recognition [5], biometrics [6], handwriting recognition [7, 8, 9, 10, 11], gait recognition [12], tracking [13, 14], and human behavior recognition [15, 16]. We refer the reader to [17] for a more detailed review of applications, training, and classification approaches.

1.1 Motivation

In this work, we consider the recognition of behaviors. We use the word "behavior" generically, not necessarily referring to human behaviors such as emotions (e.g. happy or sad) or actions (e.g. washing dishes or driving a car). Instead, we consider the behavior of objects, such as vehicles or ships. Most movements of objects of this type are predictable and are consistent for every occurrence of the behavior. For example, consider a humvee leaving a military base in a city. For simplicity, let us assume that three behaviors are possible for the humvee:

- Patrol base perimeter
- Patrol city

- Direct path to a location in the city, then return to base

By tracking the location of the humvee while it exhibits each of these behaviors, we can construct three different models. With a single vehicle, the usefulness of this task is unclear. If we consider a computer tracking the location of dozens of humvees simultaneously, it immediately becomes clear that having a set of known behaviors would be helpful to human operators watching the current situation. Computer aided analysis of situations such as those mentioned here may help to save lives.

In several of our examples, we show how approaches can be used indirectly in human behavior recognition. Consumer behavior recognition is the application of computer algorithms to predict the shopping patterns and purchasing power of different consumers. We specifically look at movie rental behaviors and assume that an individual's movie tastes remain constant for a given time period. For example, a couple's interest in movies may lie in romance and horror movies while dating, action and drama genres before children, and cartoon and family movies after children. We would construct three different behavior models to represent the three phases of the family's life. Computer aided analysis of consumer behavior may reduce the number of irritating advertisements thus benefiting consumers and decreasing marketing costs for corporations.

To accomplish the goal of computer aided analysis, we consider four questions in this dissertation:

1. Can models be constructed using no a priori information?
2. Can we be confident that the observations and model adequately represent the underlying process?
3. How can we determine if observations match zero, one, or more than one constructed model?
4. Do bounds exist on the amount of data we need to match observation data and models?

1.2 Organization

The outline of this dissertation is as follows. The remainder of Chapter 1 describes why questions one through four are important to the area of pattern recognition.

Chapter 2 explains the mathematics that form the foundation for this work. The models we use have deterministic and stochastic components. As such, these models with our extensions are

governed by probability theory. We introduce probability theory in a clear progression from basic concepts to the many probability distributions that will be referenced throughout the dissertation. We conclude the chapter with a discussion of Markov models and the most famous extension, the hidden Markov model.

Chapter 3 contains an overview of model construction. To construct models without a priori knowledge of the structure, we use the ϵ-machine construction process developed by J.P. Crutchfield and C.R. Shalizi that requires only a sequence of observations and a maximum data string length [18]. Values of the maximum data string length that are too small result in incorrect models being constructed. Values that are too large reduce the number of data samples that can be considered and exponentially increase the algorithm's computational complexity. We present a method for automatically inferring this parameter directly from training data as part of the model construction process [19]. This chapter answers the first of the four questions listed previously. The novel extension explained in this chapter removes the only user defined value of Shalizi's and Crutchfield's approach and allows models to be constructed directly from observation sequences without user input.

Chapter 4 elucidates how to calculate model confidence when constructing ϵ-machines from observed data sequences. If an insufficient amount of observation data is used to generate the HMM, the model will not be representative of the actual underlying process. Current methods assume the observations completely represent the underlying process; the methods presented in this chapter determine if the model matches the observation data. We do not assume that the observations gathered fully represent the underlying process. Therefore we desire a level of confidence that the constructed model is representative of the underlying process, not the observations. We present two methods that determine if the observation data and constructed model fully encapsulate the underlying process with a given level of statistical significance [20], answering the second question listed above. We use illustrative examples to demonstrate the effectiveness of the approaches.

Chapter 5 describes a novel approach for detecting matches between constructed models and new observation sequences. Currently, HMMs recognize patterns using a maximum likelihood method. One major drawback with this approach is that data observations are mapped to HMMs without considering the number of data samples available [1]. Another problem is that this method is only useful for choosing between HMMs. It does not provide a criteria for determining whether or not a given HMM adequately matches the data stream. In this work, we recognize complex behaviors

using HMMs and confidence intervals [21]. The certainty of a data match increases with the number of data samples considered. We detail an approach that has the ability to have more than one model or no models match the observation sequence. This extension solves a classification problem by translating the result into a detection problem, which answers the third question. We demonstrate the approach using both illustrative models and models from a consumer behavior dataset.

Chapter 6 explicates methods that find the bounds on the size of data windows used to match observation sequences and models. We consider how to detect patterns in data streams that are "serial Markovian," where target behaviors are Markovian but targets may switch from one Markovian behavior to another. Traditional Markov model based pattern detection approaches, such as hidden Markov models, use maximum likelihood techniques over the entire data stream to detect behaviors. To detect changes between behaviors, we use statistical pattern matching calculations performed on a sliding window of data samples. If the window size is too small, the system will suffer from excessive false positive rates. If the window is too large, change point detection is delayed. This dissertation finds both necessary and sufficient bounds on the window size. In this chapter, we develop the two mathematical approaches that derive the bounds directly from the models, mitigating the issue of changing window sizes due to changing observation data [22]. This development completes our solution set to the four questions posed in this introduction.

Chapter 7 concludes with a summary of our accomplishments stated in this dissertation. We also provide our thoughts for future extensions and paths that future researchers could explore.

1.3 Model Construction and Confidence

Traditionally, the Baum-Welch Algorithm is used to infer the state transition matrix of a Markov model and symbol output probabilities associated with the states, given an initial Markov model and a sequence of symbolic output values (see [1]). The Baum-Welch Algorithm uses expectation maximization to solve a non-linear optimization problem. The fundamental approach of constructing Markov models from data streams has been heavily researched for specific applications. Methods in [23] and [24], for example, illustrate construction and training in speech recognition applications.

To construct a Markov model without a priori structural information, we use an approach developed by J.P. Crutchfield and C.R. Shalizi [18, 25, 26], which derives the HMM state structure

and transition matrix from available data samples. Alternate methods use pruning to decrease the size of the HMM until it is optimal for the observation data [27]. To maximize the classification ability of the model, [28] proposes incorporating predictive measures into model development. Other approaches may be used to construct models from data streams for specific areas such as speech recognition [29]. The minimum description length principle from information theory [30] has been used in maximum-likelihood model construction [31]. In this dissertation, we only consider the approach from Shalizi et al.

Shalizi's approach finds statistically significant groupings of the training data that correspond to HMM states. This is accomplished by analyzing the conditional next symbol probabilities for a data window that slides over the training data. This data window increases gradually from a size of two to an a priori known maximum window size L. Except for the training data, the only initial information required to construct the HMM model using Shalizi's approach is the parameter L. The parameter L expresses the maximum number of symbols that are statistically relevant to the next symbol in the sequence. The state structure of the Markov model is inferred from the symbol groupings of length $\leq L$ by adding those states to the model that lower system entropy [32, 33]. To date, no one has considered how to dynamically find the parameter L. We extend the work of Crutchfield and Shalizi so that we determine parameter L with no prior knowledge and therefore derive minimum entropy HMMs with no a priori information.

When constructing models dynamically, issues arise that can affect the quality of the models. If an insufficient amount of observation data is used in construction, the model parameters may not allow enough variation in observation data for general use. If a sufficient amount of data is gathered, a topic commonly addressed is whether the constructed model is a representation of the observations. To determine this, the assumption that the observation data fully encapsulates the underlying process is made. If the observation data does not completely represent the underlying process, any model constructed from the data will not be representative of the process. Therefore we desire a level of confidence that the constructed model is representative of the underlying process, not just the observations.

Confidence is a concept derived from statistics typically used for mean and variance analysis. The allowable error of the mean for a set of values can be determined by measuring the standard deviation of the individual values. A model is simply a set of probabilistic transitions linking a set of states. We use model confidence on the transition probabilities and the likelihood that the system

is in a given model state to determine the probability of unknown events occurring.

1.4 Detection and Data Size

Traditionally HMMs are used for data classification, which assigns the observed data stream to one of a known set of models. This is most commonly done using a maximum likelihood approach [1]. Although detection and classification are similar problems in many respects, we note that detection is subtly different from classification. By definition, classification always returns one (and exactly one) model that matches the data stream. Detection may find that no model matches the data stream. It may also return more than one model.

We use confidence intervals for HMM analysis. This has the advantage in that we can consider the number of data samples available when comparing an HMM model with a sensor data stream. Our use of Receiver Operating Characteristic (ROC) curves to find detection thresholds is a novel approach when confidence intervals are used.

Detecting instances when observation data sequences correspond to behavior models is a form of the matching problem. Matching can be performed after all data has been collected for an a posteriori interpretation, such as with consumer behaviors, or during data collection for real-time data interpretation, such as in military situations. Providing methods to allow the application to adjust functionality between these two types of matching is critical for versatility.

We consider behavior data that is "serial Markovian." At any given time, the target being observed performs a behavior that can be expressed as a Markov model. However, the behavior being executed by the observed target may change over time. Consider the problem of matching cruise ships to cruise itineraries using a series of GPS coordinates. At any point in time, a ship may switch from one established itinerary to a different established itinerary. Reliability in recognizing any given or change in itinerary will require using several GPS readings.

This application is problematic for maximum likelihood approaches that consider the entire data stream. Consequently, we recognize Markovian behaviors by considering data within a sliding window of the data stream. If the window is too small, the process does not have sufficient data to distinguish between models so it suffers from an excessive false positive rate. If the window is too large, data collected after the change point is overwhelmed by data collected before the change point. This forces the system to wait for an excessively long period of time before recognizing the

new behavior [34, 35]. In other works, finding the proper window size to match models against input data streams is often left for future analysis [36, 37, 38], or the effect of different window sizes on results is not demonstrated [39]. In this dissertation, we show how to find the window size that is best able to detect changes when the target's behavior switches from one Markovian process to another.

Chapter 2

Background

Probability theory is a very profound area, much more complex than can be addressed in this chapter. All of the proofs and explanatory reasoning is omitted from the background as all the concepts here have been vetted by mathematicians and statisticians over several hundred years. This background chapter will cover relevant mathematical concepts which are necessary to understand the topics in following chapters.

2.1 Probability

This dissertation focuses on the fundamentals of probability. The study of probability allows us to represent many of life's seemingly unmeasurable situations with mathematical models and simulations. We will cover the basic tenets of probability necessary to understand the upcoming chapters. This discussion is succinct at best and the reader is encouraged to review textbooks on probability theory, such as [40], [41], and [42]. These references describe the derivation of these concepts and present a more complete discussion of the areas of probability which are not mentioned nor covered in this review.

2.1.1 Set Notation

Consider three sets S, S_1, and S_2, where S_1 and S_2 are subsets of S (i.e. $S_1 \subset S, S_2 \subset S$). The combination of S_1 and S_2 is called *union* and is written $S_1 \cup S_2$. The equivalent portion of S_1 and S_2 is called *intersection* and is written $S_1 \cap S_2$. A null intersection is represented by the

empty set, $S_1 \cap S_2 = \emptyset$. Unions and intersections of a small number of sets are often represented with Venn diagrams. Figure 2.1 demonstrates the union and intersection of two sets. The notation of set theory is used in probability theory, which we now discuss.

2.1.2 Events

The fundamental concept in probability is an *event*. An event is an action or an observable occurrence. Assume that we perform n experiments or trials to measure an event E. The probability of an event E occurring is estimated with

$$\Pr(E) = \lim_{n \to \infty} \frac{\#\text{ times E occurs}}{n}$$

All events occur within the space of all possible events, Ω. In practicality, we artificially limit Ω to fit all events within the context of our analysis. The probability of an event occurring in Ω is certain, or $\Pr(\Omega) = 1$. Consider two events $A, B \in \Omega$. The probability of both A **and** B occurring is $\Pr(A \cap B) = \Pr(AB)$. A and B are *independent* if the occurrence of B does not positively or negatively affect the occurrence of A. If this is true, then the joint occurrence, $\Pr(AB)$, can be represented by multiplying the probabilities of their individual occurrences, i.e. $\Pr(AB) = \Pr(A) \cdot \Pr(B)$. The probability of either A **or** B occurring is $\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(AB)$.

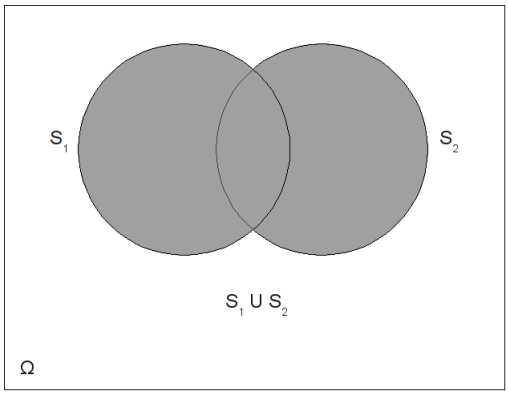
If the occurrence of B does affect the occurrence of A, Bayes Theorem is used to calculate the conditional probability. Bayes Theorem states the probability of A occurring given that B has occurred is the ratio of the joint occurrence and the probability of B occurring.

$$\Pr(A|B) = \frac{\Pr(AB)}{\Pr(B)}$$

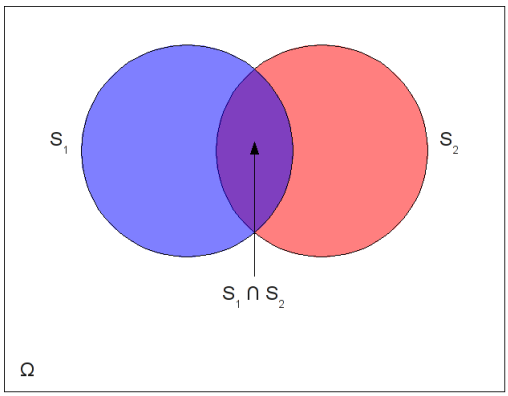
If A and B are independent, $\Pr(A|B) = \Pr(A)$. We note that $\Pr(A|B) = \Pr(A)$ is necessary but not sufficient for the independence of A and B. If we observe a series of events $A_1, A_2, \ldots$ where $A_i \cap A_j = \emptyset$ for $i \neq j$, then the series of events is independent and identically distributed (i.i.d.).

2.1.3 Random Variables

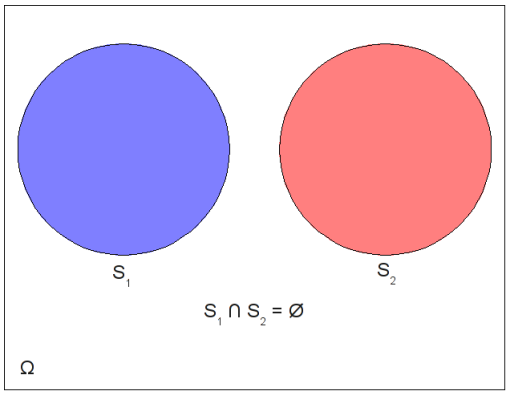
In complex systems, analyzing singular events often does not produce meaningful results. In many applications, we have a number of output values and thus a desire to know the probability



(a)



(b)



(c)

Figure 2.1: Venn diagrams illustrating (a) $S_1 \cup S_2$; (b) $S_1 \cap S_2$; (c) $S_1 \cap S_2 = \emptyset$

that a specific output occurs. To relate the space of possible events and the output space, we use a mapping function called a *random variable*. Formally, a random variable X is defined as

$$X : (\Omega, P) \rightarrow Y$$

where Ω is the event space, Y is the output space, and P contains the probability of a mapping from Ω to Y.

A random variable X has several properties to regulate the mapping. These properties are listed in Table 2.1 where x is a value that X can take.

Table 2.1: Properties of Random Variables

Discrete	Continuous
$p(x) = \Pr(X = x)$	$f(x) \geq 0$
$\Pr(X = x) \geq 0$	$\Pr(X = x) = 0$
$\sum_{i=0}^{x} \Pr(X = i) = 1$	$\int_{-\infty}^{\infty} f(x)\mathrm{d}x = 1$

The first equations for both the discrete and continuous cases are the functions used to represent the probability distribution functions. Notice that $p(x)$ represents the probability of a single value in the discrete case and $f(x)$ is only a real nonnegative function. In the discrete case, the second item simply states that probabilities can never be negative. The third states that for all possible values of x, the sum of the probabilities must equal one. In the continuous case, the second equation states that the probability of a continuous random variable taking on a single real value is zero. This is equivalent to the probability selecting a single point out of an infinite number of points. The third equation is the continuous equivalent to the discrete equation.

To explain how a random variable is assigned values, we use an example with two coins. There are two possible events for each coin, heads (H) and tails (T). The possible outcomes from flipping two coins simultaneously are: HH, HT, TH, and TT. If we let X equal the number of heads, X can take on three values: zero, one, or two. The function that determines if a random variable X equals x, $p(x)$, is called the *probability mass function*. In the continuous case, $f(x)$ is the *probability density function*. Both are different types of *probability distribution functions*. The function that determines if a random variable X is less than or equal to x, $\Pr(X \leq x)$, is called the *cumulative distribution function*.

When considering results from experiments with random variables, two commonly discussed features of probability distributions are the *expectation* and the *variance*. The expected value of a random variable, written as $E(X)$, is the value of x predicted to occur after a large number of trials given a probability distribution function. The expected value can be determined using Equation (2.1) for discrete distributions and Equation (2.2) for continuous distributions.

$$E(X) = \sum_{x} xp(x) \tag{2.1}$$

$$E(X) = \int_{-\infty}^{\infty} xf(x)\mathrm{d}x \tag{2.2}$$

The expected value of a probability distribution is written as μ in addition to $E(X)$.

Variance is the measure of the variability in the outcomes. It is the expectation of the difference between the random variable X and the calculated expected value μ. Variance is determined using one of the forms in Equation (2.3) for both discrete and continuous distributions.

$$V(X) = E(X-\mu)^2 = E(X^2) - \mu^2 \tag{2.3}$$

The variance of a probability distribution is written as σ^2 in addition to $V(X)$. The commonly known statistical measure of *standard deviation* is $\sigma = \sqrt{V(X)}$.

The phrase *random process* refers to a series of random variables X that may take a different value at different times. Instead of representing each time t with a different random variable, it is more convenient to write X_t and X_{t+1} to represent the random variable X at times t and $t+1$ respectively.

2.1.4 Bernoulli Distribution

The Bernoulli distribution is the simplest discrete probability distribution and is used to measure a success or a failure. Let a random variable X take on two values: 0 or 1. The probability distribution function for a Bernoulli distribution is

$$\Pr(X = 1) = p(x) = p$$

$$\Pr(X = 0) = 1 - p$$

where p is the probability of a one occurring. The expected value and variance of the Bernoulli distribution are $\mu = p$ and $\sigma^2 = p(1-p)$, respectively. As an example of this distribution, consider an unfair coin where heads occurs with probability 0.75. Let a random variable X represent a head occurring on a toss, then $\Pr(X = 1) = 0.75$ and $P(X = 0) = 0.25$. With the Bernoulli distribution, we can see the individual probability of an event, but we cannot determine trends.

2.1.5 Geometric Distribution

The geometric distribution is used to determine the probability that one success occurs given a series of failed repeating tests. Assume that we conduct n Bernoulli trials with probability of success p. Let a random variable X represent the first trial where the first success is recorded. The probability distribution function for a geometric distribution is

$$\Pr(X = n) = p(n) = p(1-p)^{n-1}$$

The expected value and variance of the geometric distribution are, respectively:

$$E(X) = \frac{1}{p}$$
$$V(X) = \frac{1-p}{p^2}$$

To demonstrate the geometric distribution, we continue the previous example with the unfair coin. Using a geometric distribution, we can determine the probability that the first tail occurs on the third toss

$$\Pr(X = 3) = (0.25) \cdot (0.75)^2 = 0.141$$

2.1.6 Binomial Distribution

The binomial distribution is used to determine the probability of a number of successes occurring in a number of trials. Assume we conduct n Bernoulli trials with probability of success p. Let a random variable X represent the number of times that k successes are registered in n trials. The probability distribution function for a binomial distribution is

$$\Pr(X = k) = p(k) = \binom{n}{k} p^k (1-p)^{n-k} = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$$

The expected value and variance of the binomial distribution are $\mu = np$ and $\sigma^2 = np(1-p)$, respectively. The cumulative distribution function for a binomial distribution is

$$\Pr(X \leq k) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}$$

We will write $B \sim (n, p)$ to refer to a binomial distribution B with a specific n and p.

To demonstrate the binomial distribution, we expand upon the previous example with the unfair coin. Using the binomial distribution, we can determine the probability of four heads occurring out of five tosses

$$p(4) = \binom{5}{4} (0.75)^4 \cdot (0.25)^1 = 0.396$$

2.1.7 Multinomial Distribution

The previously discussed Bernoulli, geometric, and binomial distributions are univariate (i.e. are represented with only one random variable). The multinomial distribution is similar to the binomial distribution but is multivariate. The multinomial distribution is used to determine the probability that a number of distinct outcomes occur in a number of trials. Assume we have an experiment where m distinct outcomes are possible, each with probability p_i for $i = 1, 2, \ldots, m$. If we conduct n trials of the experiment, let k_i for $i = 1, 2, \ldots, m$ represent the number of times that outcome i occurs such that $\sum_i k_i = n$, and let X_i for $i = 1, 2, \ldots, m$ be the random variable for outcome i. The probability distribution function for the multinomial distribution is

$$\Pr(X_1 = k_1, X_2 = k_2, \ldots, X_m = k_m) = \frac{n!}{k_1! k_2! \cdots k_m!} p_1^{k_1} p_2^{k_2} \cdots p_m^{k_m}$$

where

$$\sum_i p_i = 1 \quad \text{for } i = 1, 2, ..., m$$

Due to its multivariate nature, the multinomial distribution does not have a singular expectation or variance. Instead, each outcome is treated individually with expected value $\mu_i = k_i p_i$ and variance $\sigma_i^2 = k_i p_i (1 - p_i)$. With a multivariate distribution, we determine the relationship between the random variables with the *covariance*. The covariance between random variables X_1 and X_2

can be calculated with one of the forms in Equation (2.4).

$$\text{cov}(X_1, X_2) = E[(X_1 - \mu_1)(X_2 - \mu_2)] = E(X_1 X_2) - \mu_1 \mu_2 \tag{2.4}$$

If X_1 and X_2 are directly proportional (i.e. X_1 is small when X_2 is small, and vice versa), the covariance is positive. If X_1 and X_2 are inversely proportional, the covariance is negative. We will write $M \sim (n_i, k_1, \cdots, k_m, p_1, \cdots, p_m)$ to refer to a multinomial distribution with m random variables.

To demonstrate the multinomial distribution, we continue our example with the unfair coin. In addition to heads and tails, let us assume that the coin can also land on its side (S), all with probabilities $\Pr(H) = 0.6,\ \Pr(T) = 0.35,\ \Pr(S) = 0.05$. Using the Multinomial Distribution, we can find the probability that in five tosses, we see two heads, two tails, and one side with

$$\Pr(X_H = 2, X_T = 2, X_S = 1) = \frac{5!}{2!2!1!} 0.6^2 0.35^2 0.05^1 = 0.066$$

2.1.8 Student's t-distribution

All previously discussed distributions are discrete where the random variable(s) can only map to values in a discrete set. The t-distribution is a continuous distribution where the random variable maps to any real number. The t-distribution is heavily used in statistics to provide estimates on the confidence of the mean of a data set. The probability distribution function of the t-distribution is complex and is rarely used in calculations. Tables of values produced by the t-distribution are included in most statistics textbooks. Throughout this work, when we discuss reference variables for statistical comparison, we refer the reader to a statistics textbook, such as [43], for the desired reference value.

The degrees of freedom and a confidence value, α, are required to find a value for the distribution from a reference table. The degrees of freedom of a t-distribution are the number of data points n minus one and α is a user-defined value for the level of confidence desired. In this work, we typically use $\alpha = 0.05$ for a 95% confidence level in our calculations.

2.1.9 Normal Distribution

Like the t-distribution, the normal distribution is a continuous distribution. The normal distribution is one of the most widely used probability distributions and is the most recognizable due to its Gaussian curve or "bell" shape. The probability distribution function for the normal distribution is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

The cumulative distribution function for the normal distribution is

$$\Pr(X \leq b) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{b} e^{-(x-\mu)^2/2\sigma^2} \mathrm{d}x$$

The expected value μ and variance σ^2 are contained in the distribution function and are either known or calculated from the data. To estimate the mean, one should test the data for normality using a Quantile-Quantile plot [44] or Shapiro-Wilk test [45]. If the data passes either test, the sample mean can be calculated using Equation (2.5) and the sample variance can be found using either Equation (2.6) for an unbiased estimator or (2.7) for a maximum likelihood, where x_i are the data values for all i, n is the number of data values, and s^2 is the estimated variance found using Equation (2.8) [42].

$$\mu = \frac{\sum_i x_i}{n} \tag{2.5}$$

$$\sigma^2 = \frac{ns^2}{n-1} \tag{2.6}$$

$$\sigma^2 = s^2 \tag{2.7}$$

$$s^2 = \frac{1}{n} \sum_i (x_i - \mu)^2 \tag{2.8}$$

Due to the Central Limit Theorem [40], the normal distribution with mean $\mu = np$ and variance $\sigma^2 = np(1-p)$ may be used to approximate a binomial distribution as the number of trials $n \to \infty$ and $p \pm 2\sqrt{p(1-p)/n} \in [0, 1]$ [40]. In practice, the normal distribution can usually be used

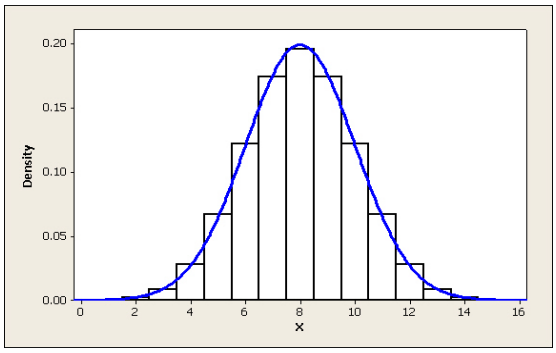


Figure 2.2: Example of a normal distribution $N \sim (8, 4)$ approximating a binomial distribution $B \sim (16, 0.5)$

if $n > 30$. Figure 2.2 shows an example of the normal approximation to the binomial distribution $B \sim (16, 0.5)$. Note that the normal distribution does not equal the binomial at every value of k. A continuity correction must be applied to correct and account for the discrepancies. Typically a correction of 0.5 is added to each k when calculating $\Pr(X < k)$ with the normal distribution [40].

If a t-distribution is symmetric about its mean and the degrees of freedom are sufficiently large, $n > 30$ is a standard rule, the normal distribution may also be used to approximate the t-distribution with $\mu = 0$ and $\sigma^2 = 1$. We will write $N \sim (\mu, \sigma^2)$ to refer to a normal distribution with mean μ and variance σ^2.

2.2 Models

Most situations and scenarios are too complex to model with a single probability distribution or may not fit the probability distribution function for any known distributions. For example, consider a line for the cash register at a vendor. For simplicity, we artificially limit the line to hold at most two people and represent the situation with the model in Figure 2.3.

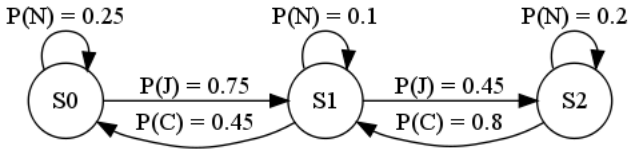


Figure 2.3: Example Markov model with three states and seven transitions

In our example, three observations are possible

1. J: a customer joins the line;

2. C: a customer is serviced at the register; and

3. N: no action is taken.

We let the vertices or states represent the number of individuals in line and the edges or transitions represent the probability of the length of the line changing. From the figure, we see if the line is empty, it is likely that a customer will enter the line. If there is one customer in line, we are equally likely to handle the customer and return to a line of zero or see the line grow to multiple customers. Modeling the situation in this manner is advantageous because we have the ability to represent the deterministic components of the situation (the number of customers the line supports) as states and the stochastic components (the probability of the line changing length) as transitions between the states. In these models, states with no outgoing transitions to other states are called *absorbing*. States with no incoming transitions from other states are called *emitting*.

A model consisting of states and probabilistic transitions is called a *Markov model*. Markov models and their various extensions are used heavily in many applications across engineering disciplines. We now introduce the concept of discrete sequences, which are used to describe the functionality of Markov models and, as we will see, to construct Markov models.

2.2.1 Sequences

Let *alphabet* $\mathcal{A}$ be a finite set of symbols representing observable events. For an individual *sequence* $\chi \in \mathcal{A}$, we use $\chi = \chi_1\chi_2 \ldots \chi_n$ to denote the elements that compose χ. By $\mathcal{A}^*$, we refer to the set of all sequences in $\mathcal{A}$. This notation is consistent with formal language theory [46]. Let λ be the empty sequence.

If χ is a sequence, then γ is a *subsequence* if

1. γ is a sequence in $\mathcal{A}$; and

2. There exist indices i and j such that $\gamma = \chi_i\chi_{i+1} \cdots \chi_{i+j}$.

2.3 Markov Models

A Markov model G is a tuple $G = (V, E, \mathcal{A}, \phi, \delta)$ where V is a set of vertices of a graph, E is a set of directed edges between the vertices, $\mathcal{A}$ is an alphabet, $\phi : \mathcal{A} \to E$ is a mapping function

of a symbol $a \in \mathcal{A}$ to an edge $e \in E$, and δ is a probability function such that

$$\sum_{a \in \mathcal{A}, v_j \in V} \delta(v_i, a, v_j) = 1 \quad \forall v_i \in V$$

The current definition of ϕ maps symbol outputs to the edges. Models with observable outputs on edges are similar to Mealy models from finite state machine literature [47]. An alternate definition of $\phi : \mathcal{A} \rightarrow V$ maps symbols to states, which is a Moore machine representation [48]. Both are valid Markov model interpretations. It is possible for any given state machine to construct an equivalent state machine that switches the role of states and transitions [49]. In the upcoming section on hidden Markov models, we will discuss both Mealy and Moore representations in more detail. In our experiments and all future chapters, we only consider Markov models following the Mealy machine mapping. We also note that Markov models may be unlabeled, where ϕ does not exist. In these models, there are no symbols corresponding to the edges or states in the model. The models that we use in this work are labeled Markov models.

A *path* through G following a sequence χ is an ordered set of vertices $(v_1, v_2, \ldots, v_{n+1})$ such that for each pair of vertices (v_i, v_j):

1. $(v_i, v_j) \in E$ and
2. $\phi(v_i, v_j) = \chi_i$ for $\chi_i \in \mathcal{A}$.

In Markov models, the vertices of G are referred to as states and the edges are referred to as transitions. V is the state space of size n and $\mathbf{P}$ is the $n \times n$ transition matrix for the model. Each element $p_{i,j} \in \mathbf{P}$ expresses the probability the model transitions from state i to state j. If $(v_i, v_j) \in E$, then

1. $p_{i,j} = \delta(v_i, a, v_j)$ for $a \in \mathcal{A}$;
2. $p_{i,j} \neq 0$; and
3. For any i, $\sum_j p_{i,j} = 1$.

Our definition of $p_{i,j}$ does not allow for a state v_i to have multiple transitions to state v_j, i.e. $\delta(v_i, a, v_j)$ and $\delta(v_i, a', v_j)$ both exist. Adding states to models with multiple edges between two states is typically sufficient to alter the model to be consistent with our definition of $p_{i,j}$.

We require that Markov models be deterministic in transition label; i.e., if there is a pair (v_i, v_j) with $\phi(v_i, v_j) = \chi_i$, then $\phi(v_i, v_k) \neq \chi_i$ for $\forall k \neq j$. Therefore, the probability that a particular symbol will occur next is the probability $p_{i,j}$ of moving to the next state using a transition associated with that symbol. If no transition exists between state i and state j, $p_{i,j} = 0$.

2.3.1 Properties of Markov models

A necessary but not sufficient condition for a model to be Markovian is the satisfaction of the Markov property. Let random variable X_t represent the state of the model at time t. Assuming that $(v_i, v_j) \in E$, the Markov property is as follows

$$\Pr(X_{t+1} = v_j | X_t = v_i, \ldots, X_0 = v_0) = \Pr(X_{t+1} = v_j | X_t = v_i) = p_{i,j}$$

The Markov property simply states that given all previous states the system has entered, the probability to transition to a new state is only dependent on the current state, not on the path taken to reach the current state. If this property is satisfied, the model is said to be *memoryless*.

A Markov model is said to be *irreducible* if any state v_j is reachable from every other state v_i within a finite number of time steps. A model where states cannot be reached is *reducible*. More formally, the model is irreducible if a sequence χ_{ij} with length k exists such that

$$\Pr(X_0 = v_i, X_k = v_j) = 1 \quad \forall v_i, v_j \in V \tag{2.9}$$

Figures 2.3 and 2.4 show examples of an irreducible model and reducible model, respectively. States that can and cannot be reached are *recurrent* and *transient*, respectively [50]. If a state v_i is recurrent and the expected value for the number of time steps needed before returning is finite, then v_i is said to be *positive recurrent*.

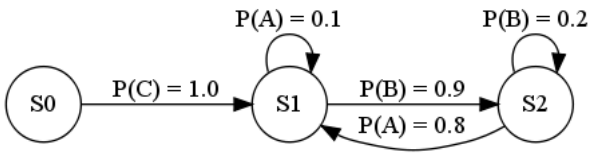


Figure 2.4: Example reducible Markov model

A Markov model is said to be *periodic* if there exists at least one state that is visited

every ck-time steps for $c \in \mathbb{N}_1$. If no states exist with this property, the model is said to be *aperiodic*. More formally, a state is periodic if indices i, j, and k exist such that given a sequence $\chi = \chi_1 \cdots \chi_i \cdots \chi_j \cdots \chi_k$ where at times i, j, and k, the current state is v_i and

$$\text{gcf}\{i, j, k\} > 1$$

where gcf$\{\cdot\}$ is the greatest common factor of the indices [41]. Figure 2.5 shows an example of a periodic Markov model.

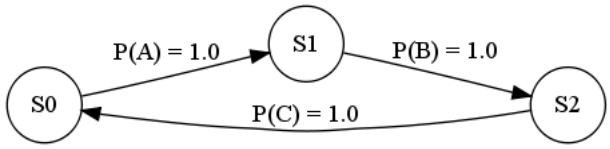


Figure 2.5: Example periodic Markov model

If all states in a Markov model are both aperiodic and positive recurrent, the model is said to be *ergodic*. The ergodic property is extremely important because it allows us to make predictions about Markov models representing systems where time steps k trend toward infinity.

2.3.2 Stationary Distribution

Recall that for a Markov model with state space size n, the probability matrix $\mathbf{P}$ is a square matrix where each constituent element $p_{i,j}$ $\forall i, j \in [0..n]$, represents the probability of the model transitioning from state v_i to state v_j. Let random variable X_k represent the current state of the system at time step k and π be a vector of size n with each element π_i representing the probability that $X_0 = v_i$ and $\sum_i \pi_i = 1$. The vector π is referred to as the initial probability vector. Unless otherwise specified, we assume that all states are equally likely to be the starting state at time 0, $\{\pi_i \in \pi : \pi_i = 1/n \quad \forall i \in [0 \ldots n]\}$.

We can calculate the probabilities for X_k using $\mathbf{P}$ and π using

$$\mathbf{x} = \pi \mathbf{P}^{(k)}$$

where $\mathbf{x}$ is a vector of size n whose elements x_i represent the probability of the model being in state

v_i at time step k. $\mathbf{P}^{(k)}$ is called the power matrix of $\mathbf{P}$. Interestingly, if the model is ergodic, then

$$\mathbf{S} = \lim_{k \to \infty} \mathbf{P}^{(k)} \tag{2.10}$$

where $\mathbf{S}$ eventually stabilizes such that each row is equivalent [41]. At this point, a row $\mathbf{s}$ of $\mathbf{S}$ is said to be the stationary distribution for the model. Each element $s_i \in \mathbf{s}$ is the asymptotic probability that the model is in state v_i, regardless of the initial probability vector π. Solving Equation (2.10), however, can be computationally intensive for large matrices.

Theorem 1 (modified from [41])**.** *The calculation of Equation* (2.10) *can be simplified by determining vector* $\mathbf{s}$ *such that*

$$\mathbf{s} = \mathbf{sP} \tag{2.11}$$

Proof. By definition,

$$\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)}\mathbf{P} \tag{2.12}$$

From Equation (2.10), $\mathbf{P}^{(k)} \to \mathbf{S}$ as $k \to \infty$. Consequently, since k is monotonically increasing, $\mathbf{P}^{(k+1)} \to \mathbf{S}$. Combining this with Equation (2.12) gives

$$\mathbf{S} = \mathbf{SP} \tag{2.13}$$

for a trend of $k \to \infty$. Since all rows in $\mathbf{S}$ are equivalent, selecting any one row $\mathbf{s} \in \mathbf{S}$ reduces Equation (2.13) to Equation (2.11). □

Remark 1. The solution to Equation (2.11) can be found by computing the solution to

$$\mathbf{s} = (\hat{\mathbf{P}} - \hat{\mathbf{I}})^{-1}\hat{\mathbf{0}}$$

$\hat{\mathbf{P}}$ is a modified form of $\mathbf{P}$ where the i^{th} column is a vector of ones. $\hat{\mathbf{I}}$ is a modified $n \times n$ identity matrix where the i^{th} column is a vector of zeros. $(\cdot)^{-1}$ is the inverse operation for matrices. $\hat{\mathbf{0}}$ is a $n \times 1$ vector where the first $i - 1$ elements are zero and the i^{th} element is one. i can be any value up to n but must be kept constant when creating the modified matrices and vectors.

2.3.3 Generating Sequences

Given a Markov model, the following procedure may be used to generate a sequence of length l:

i. Choose an initial state $v_i = v_0$

ii. Randomly select a transition $(v_i, v_j) \in E$ weighting the transitions by their probability $\delta(v_i, \chi_i, v_j)$ to move to state v_j from state v_i

iii. Record the label $\chi_i = \phi(v_i, v_j)$ associated with transition (v_i, v_j)

iv. Repeat steps *ii* and *iii* until l symbols have been recorded

v. Record the sequence $\chi = \chi_i \chi_{i+1} \chi_{i+2} \cdots \chi_l$

We note that if the model contains at least one absorbing state, depending on how symbols are recorded, it is possible for the model to be unable to generate a sequence of length l.

2.4 Hidden Markov Models

Markov models are used to model situations where the state and transition structure are directly observable or known a priori. In most real-world situations, the model structure that is generating output sequences is "hidden" and only events produced by the model are known. Hidden Markov models (HMMs) are a useful extension to Markov models that allow us to represent these situations by estimating the model and transition probabilities from observation data sequences.

In Markov models, we know the entire alphabet $\mathcal{A}$, whether or not all symbols in $\mathcal{A}$ are observed. In HMMs, we assume that the alphabet only consists of the symbols that we have collected in observations. Therefore, all symbols in paths through the HMM are associated with a known symbol alphabet [1, 51]. From this point forward, we will not acknowledge this difference between an observed alphabet and the actual alphabet; both will be referred to as the alphabet $\mathcal{A}$.

The generic HMM is represented as a random process X_t that produces observations in the form of another random process Y_t. This is depicted graphically in Figure 2.6. The generic model does not represent repeatable observations since it is a one-dimensional ordering of states with each state representing a single observation. Therefore, a fundamental question that all users of HMMs must answer is: what model structure should be used to represent the situation?

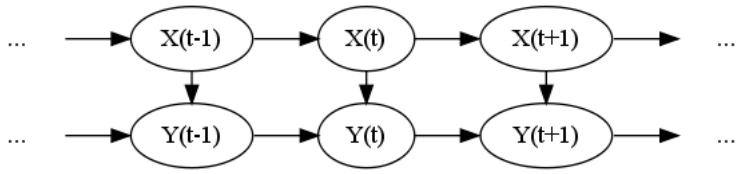


Figure 2.6: Generic representation of a HMM

Rabiner in [1] provides one of the best surveys of HMMs. The values that must be specified before a HMM can be used are given in Table 2.2. In classical HMM construction, the only value that is not user defined is $|\mathcal{A}|$, which is simply the size of the alphabet. The initial value for the number of states is extremely important because it sets the size of $\mathbf{P}$ and π. The initial values for $\mathbf{B}, \mathbf{P}$, and π can be set in any manner deemed appropriate, whether constructed or random[1]. We note that the Moore representation has two probability matrices for the model to represent the underlying process. Mealy machines are less complex, but at the sacrifice of model versatility.

Table 2.2: Values required for HMM construction

Moore Representation	Mealy Representation
• $\|\mathcal{A}\|$: Number of observable symbols	• $\|\mathcal{A}\|$: Number of observable symbols
• $\|V\|$: Number of states	• $\|V\|$: Number of states
• $\mathbf{B}$: Initial observation probability matrix	• ϕ: Symbol to edge mapping function
• $\mathbf{P}$: Initial state transition probability matrix	• $\mathbf{P}$: Initial state transition probability matrix
• π: Initial probability vector	• π: Initial probability vector

Once the initial structure is set, observation sequences used to refine or train the observation probabilities $\mathbf{B}$ and transition probabilities $\mathbf{P}$ in order for the model to represent the observations. The Baum-Welch Algorithm is the standard expectation maximization algorithm used to estimate the transition matrix $\mathbf{P}$ of a hidden Markov model [52, 1]. The algorithm is iterative and the stopping

[1]Provided that all rules of probability are followed, e.g. all outgoing transition probabilities must sum to one.

criteria may be given in terms of the convergence of the solution in any number of metrics.

Once the model is created and the different probability distributions are trained, the HMM can be used to answer three fundamental questions about the situation [1]:

1. Given an observation sequence χ, which HMM G best fits the sequence and what is the likelihood G generated χ;
2. What is the most likely path an observed sequence χ took through G; and
3. Which parameters should be altered to maximize $\Pr(\chi|G)$.

We now discuss the Forward-Backward Procedure, which is used to find the answers to these questions.

2.4.1 Forward-Backward Procedure

Given a model G, the maximum likelihood of a sequence χ can be calculated by iteratively exploring all possible paths and finding the probability that sequence χ occurs for each path [53, 15]. The Forward-Backward Procedure [54, 55, 56, 57] applies dynamic programming and induction to calculate the probability, $\Pr(\chi|G)$, reducing the algorithm complexity from exponential time to polynomial time.

The Forward-Backward Procedure is:

1. Initialization: $\alpha_1(i) = \pi_i$, $1 \leq i \leq n$
2. Induction: $\alpha_{t+1}(j) = \sum_{i=1}^{n} \alpha_t(i) p_{i,j}$, $1 \leq t \leq M-1$
3. Termination: $\Pr(\chi|G) = \sum_{i=1}^{n} \alpha_M(i)$

The α-values are called "forward" variables and the above three steps are considered the "forward" or maximum likelihood (ML) part of the algorithm. The "backward" or expectation maximization (EM) portion of the Forward-Backward Procedure calculates the "backward" variables and is used to find:

1. the path most likely taken to produce a sequence χ and
2. how the parameters of G can maximize $\Pr(\chi|G)$ [1].

In the case of EM, to find the path through G, the Viterbi Algorithm [58, 1] is the procedure designed to most efficiently find the state ordering. Maximum mutual information (MMI) [59, 60] and minimum classification error (MCE) [61, 62] are alternatives to EM to determine the optimal parameters. In our experiments, the most likely path is not considered nor do we train any models to more adequately match the observations. We refer the reader to [1] or the previous references on MMI and MCE for a discussion on these topics.

Chapter 3

Zero Knowledge ϵ-machines

3.1 Construction

ϵ-machines are an extension to HMMs and are used to generate a Markov structure from a series of observations. HMMs require an initial Markov structure and initial probability matrix. The Causal State Splitting and Reconstruction (CSSR) Algorithm [25] infers the state and transition structure given a sequence of symbolic output sequences and a string length L. This output model, called an ϵ-machine, is the minimum entropy estimation of the true underlying process dynamics. States are defined as conditional probability distributions over the next symbol that can be generated by the process. Defining the states in this manner allows the system to tolerate random noise in the observation sequence and still maintain the deterministic behavior of the system. The CSSR Algorithm has useful information-theoretic properties in that it maximizes the mutual information among state structure and the next output symbol and minimizes the remaining uncertainty (entropy).

The CSSR Algorithm requires two inputs: a series of observation data χ and a string length variable L. The parameter L defines the number of symbols in the past that are necessary to find the proper state structure of the model. In other words, a given symbol $\chi_i \in \chi$ is dependent on symbols $\chi_{i-1}, \chi_{i-2}, \ldots, \chi_{i-L}$.

Assume we are given a sequence $\chi \in \mathcal{A}^*$, and we select a value $L \in \mathbb{N}$. For each value of $\{i : 0 \leq i \leq L\}$, we determine the set W of all subsequences γ that have length i. W then contains all subsequences of χ from length 0 (the empty string λ) up to and including length L. For each

subsequence γ in W we prepend a symbol a to create a new sequence $a\gamma$. Using simple counting (see Equation (3.1)) we can compute the probability of seeing another symbol a' after seeing subsequence $a\gamma$ conditioned on the provided sequence χ. This yields a probability distribution function $f_{a\gamma|\chi}$ over $\mathcal{A}$. A similar distribution function can be computed for each state of the ϵ-machine (see Equation (3.2)). Using a two-parameter nonparametric statistical comparison test, such as the Kolmogorov-Smirnov test or χ^2 test (see [44]), we cluster the conditional distributions based on their similarity. The level of confidence chosen for the comparison affects the type I error rate [43] of matching the distributions. The state with the best p-value is assigned the new subsequence $a\gamma$. If no state has a sufficiently low p-value to be declared a match, a new state with subsequence $a\gamma$ is created and the process continues for the next string in W.

Once all subsequences are clustered and the states are defined, the splitting function divides clusters to ensure that each state has a deterministic probability function δ. Shalizi and Crutchfield state that there are more efficient algorithms for determinizing a finite state machine. However, these algorithms do not take into account a statistical structure associated to the state machine. To determinize, the states are further split apart based on antecedent; i.e., suppose subsequences γ and γ' are associated to state v. If γa is associated to state v' while $\gamma' a$ is associated to state v'', then any transition structure defined from the states would be non-deterministic. Reconstruction simply breaks the states apart so that the resulting antecedent states are the same for all subsequences associated to a state *given* a specific input symbol. Algorithm 3.1.1 shows Shalizi's and Crutchfield's procedure.

Note in a typical run of this algorithm, there may be transient states remaining after determinization. A final step could be added that removes these states if *only the recurrent system behavior is desired*. In this case, the resulting Markov model is irreducible. Crutchfield and Shalizi execute a step like this as described in [25].

The complexity of CSSR is $\mathrm{O}(k^{L+1})+\ \mathrm{O}(N)$, where k is the size of the alphabet, L is the maximum subsequence length considered, and N is the size of the input symbol sequence. Given a stream of symbols γ, of fixed length N, from alphabet $\mathcal{A}$, the algorithm is linear in the length of the input data set, but exponential in the size of the alphabet.

Note that [25] estimates conditional probabilities by analyzing grouped sets of outputs from a stochastic process. As long as Shalizi's assumption [25] that the volume of training data is sufficient, the law of large numbers dictates that this is almost surely true. This would not be the case if one

Algorithm 3.1.1 – CSSR Algorithm from [25]

Input: Observed sequence χ; Alphabet $\mathcal{A}$, Integer L;
Initialization:

1. Define state v_0 and add λ (the empty string) to state v_0. Set $\mathbf{V} = \{v_0\}$.
2. Set $N := 1$.

Splitting (For each $i : 0 \leq i \leq L$)

1. Let $W = \{\gamma | \exists v \in \mathbf{V}(\gamma \in v \wedge |\gamma| = i - 1)\}$, where W is the set of strings in states of the current model with length equal to $i - 1$.
2. Let N be the number of states.
3. For each $\gamma \in W$, for each $a \in \mathcal{A}$, if $a\gamma$ is a subsequence of χ, then
   (a) Estimate $f_{a\gamma|\chi} : \mathcal{A} \rightarrow [0, 1]$, the probability distribution over the next input symbol.
   (b) Let $f_{v_j|\chi} : \mathcal{A} \rightarrow [0, 1]$ be the joint state conditional probability distributions; that is, the probability given the system is in state v_i, that the next symbol observed will be a. For each j, compare $f_{v_j|\chi}$ with $f_{a\gamma|\chi}$ using an appropriate statistical test with confidence level α. Add $a\gamma$ to the state that has the most similar probability distribution as measured by the p-value of the test. If all tests reject the null hypothesis that $f_{v_j|\gamma}$ and $f_{a\gamma|\chi}$ are the same, then create a new state v_{N+1} and add $a\gamma$ to it. Set $N := N + 1$.

Reconstruction

1. Let $N_0 = 0$.
2. Let N be the number of states.
3. Repeat while $N_0 \neq N$:
   (a) For each $i \in 1, \ldots, N$: Set $k := 0$. Let M be the number of sequences in state v_i. Choose a sequence γ_0 from state v_i. Create state v_{ik} and add γ_0 to it. For all sequences γ_j $(j > 0)$ in state v_i:
      i. For each $a \in \mathcal{A}$ $\gamma_j a$ produces a sequence that is resident in another state v_k. Let $(\gamma_j, a, v_k) \in \delta$.
      ii. For $l = 0, \ldots, k$, choose γ from sequences within v_{ik}. If $\delta(\gamma_j, a) = \delta(\gamma, a)$ for all $a \in \mathcal{A}$, then add γ_j to v_{ik}. Otherwise, create a new state v_{ik+1} and add γ_j to it. Set $k := k + 1$.
   (b) Reset $V = \{v_{ik}\}$; recompute the state conditional probabilities $f_{v|\chi}$ for $v \in \mathbf{V}$ and assign transitions using the δ functions defined above.
   (c) Let $N_0 = N$.
   (d) Let N be the number of states.
4. The model of the system has state set $\mathbf{V}$ and transition probability function computed from the δ relations and state conditional probabilities.

were considering only one instance of an output string. As noted in [63], the dependencies in any single trace of a Markov process go back further than L states. In the same work, it is made clear that this is not true for the dependencies of the states themselves and therefore not true for the distributions that we estimate.

3.1.1 Computing $f_{v_i|\chi}$ and $f_{\gamma|\chi}$

The following formulas can be used to compute $f_{v_i|\chi}$ and $f_{a\gamma|\chi}$ in Algorithm 3.1.1. Let $\#(\gamma, \chi)$ be the number of times the sequence γ is observed as a subsequence of χ.

$$f_{\gamma|\chi}(a) = \Pr(a|\gamma, \chi) = \frac{\#(\gamma a, \chi)}{\#(\gamma, \chi)} \tag{3.1}$$

$$f_{v_i|\chi}(a) = \Pr(a|v_i, \chi) = \frac{\sum_{\gamma \in v_i} \#(\gamma a, \chi)}{\sum_{\gamma \in v_i} \#(\gamma, \chi)} \tag{3.2}$$

3.2 The Main Problem

One limitation to the CSSR Algorithm is the dependence on parameter L, which defines the maximum subsequence length considered when inferring the model. The model assumes that at time t, symbol χ_{t+1} is a random function of symbols $\chi_t, \chi_{t-1}, \ldots, \chi_{t-L}$. Currently, the choice of L is either ad hoc or arbitrary.

When inferring models with the CSSR Algorithm, if L is too small, the state structure of the inferred machine is incorrect because we do not capture all the statistical dependencies in the data. The number of states is incorrect and symbols are incorrectly assigned to states. Examples of this will be shown in Section 3.4. Since the parameter L defines the exponent of the algorithm complexity, it is imperative that L not be larger than absolutely necessary. This motivates our work in finding the correct value of L.

To identify incorrect state structures, we define a symbol-to-state mapping, which we observe as L increases. We show that this symbol-to-state mapping stabilizes once the correct string length is found. This occurs because the construction method has access to all the statistically relevant information. At which point, adding additional information by using a larger L simply reproduces the same correct model.

We note that this problem is quite different from the order problem discussed in [63]. Order estimation is a difficult problem that occurs when using the standard HMM. Since that model has two different sets of probability distributions (states and outputs), it is desirable to find the smallest number of states that can be used to express properly the set of output distributions. The smallest number of states is called the model order. Since the model we use has one set of probability distributions, where state transitions output symbols, the question of order does not arise. [18] proves that CSSR finds the most compact representation of the statistical dependencies in the data analyzed.

3.3 Solution to the Main Problem

To find the correct string length L, we check to see that the HMM inferred using CSSR with string length L is consistent with the model structure inferred using string length $L+1$. We verify the consistency of the models by seeing if their interpretation of the symbolic dataset χ output by the process being analyzed is consistent.

Algorithm 3.3.1 works iteratively. We start by inferring HMM G_2 by using the CSSR Algorithm with parameter $L = 2$. The training data is then input to G_2. For each symbol, we record the state in G_2 that is associated with the symbol. Since the HMM is deterministic, this mapping is unique. Since there is no specified start state, we perform this process for each state in G_2. If this process fails at some point for a specific start state, we do not store that information.

We then compute HMM G_3 using CSSR with $L = 3$ and calculate the mappings of states to symbols in exactly the same way it was done for G_2. For each pair of start states (i.e. each state in G_2 paired with each state in G_3), we determine how many times they agreed on the mapping of symbols to states. Since there is no clear start state, we keep the largest value m_3.

This value m_3 measures how similar state machines G_2 and G_3 are. (It is technically similar to the concept of *bisimulation* used in model checking literature [64].) Determining if two graphs are isomorphic is computationally challenging. Determining the equivalence of two Markov chains would be more difficult, since that would also require comparing probability distributions. We avoid that issue with this step; what is important in this process is the mapping of symbols to states. If two machines assign the symbols in the training data to the same states, then their interpretations are identical.

The process repeats with increasing values of L producing a new machine G_L and value m_L with each iteration. Informally, as L increases, the CSSR Algorithm has more information regarding the history of the system. The CSSR Algorithm monotonically improves the ability of the HMM G_L to explain the training data. As the HMMs asymptotically approach the true structure of the process that produced the data, the amount of agreement between G_L and G_{L-1} increases. When the correct value of L is found, there is no new information to be gained by using $L+1$. At this point, the mapping of symbols to states will remain stable (i.e. $m_L == m_{L+1} == m_{L+2} \dots$) and the process can terminate. We present these steps more formally in Algorithm 3.3.1.

Algorithm 3.3.1 – Zero Knowledge HMM Identification Algorithm

Input: Observed sequence χ; Alphabet $\mathcal{A}$;
Initialization:

1. Set $L = 1$.
2. The set $V_{L-1} = \{v_0\}$ and $G_{L-1} = \langle V_{L-1}, \mathcal{A}, \delta_{L-1}, p_{L-1} \rangle$, where $(v_0, \chi, v_0) \in \delta_{L-1}$ for all $\chi \in \mathcal{A}$ and $p_{L-1}(v_0, \chi, v_0)$ is the proportion of times symbol χ occurs in sequence χ. (This is the ϵ-machine that results when Algorithm 3.1.1 is run with $L = 0$.)
3. Let the length of $N = |\chi|$.

Main Loop:

1. Let $G_L = \langle V_L, \mathcal{A}, \delta_L, p_L \rangle$ be the ϵ-machine output of Algorithm 3.1.1 with χ, $\mathcal{A}$ and L;
2. For every state $v_0 \in V_L$, record the path $\mathbf{v}_L^{v_0} = \{v_1, v_2, \dots, v_N\}$ that occurs when $\hat{\delta}_L$ is recursively applied with input χ starting at state v. That is, $v_1 = \hat{\delta}_L(v_0, \chi_1)$, $v_2 = \hat{\delta}_L(v_1, \chi_2)$ etc. If there is some $i \leq N$ for which $v_i = \uparrow$, then we discard sequence $\mathbf{v}_L^{v_0}$ as undefined.
3. Each sequence $\mathbf{q}_L^{q_0}$ defines a partial function $f_L^{v_0} : [N] \times \mathcal{A} \to V_L$. If v_k is the k^{th} element of sequence $\mathbf{v}_L^{v_0}$, then $f_L^{v_0}(k, \chi_k) = v_k$. That is, position k with symbol χ_k is associated to state v_k. Let $\mathcal{F}_L$ be the set of functions $f_L^{v_0}$ defined in this way.
4. Compare the functions in $\mathcal{F}_L$ to the elements of $\mathcal{F}_{L-1}$: We will use these sets to define a matching problem whose optimal solution will be used to define a stopping criterion.
   (a) Let $\mathcal{I}$ be a set of indices corresponding to elements of V_{L-1} and $\mathcal{J}$ be a set of indices corresponding to elements of V_L.
   (b) Define binary variables x_{ij} ($i \in \mathcal{I}$, $j \in \mathcal{J}$). We will declare $x_{ij} = 1$ if and only if state v_i of V_{L-1} is matched with state v_j of V_L.
   (c) Define the following coefficients:
   $$r_{ij} = \sum_{v_{0_L} \in V_L, v_{0_{L-1}} \in V_{L-1}} \left| (f_L^{v_{0_L}})^{-1}(v_i) \cap (f_{L-1}^{v_{0_{L-1}}})^{-1}(v_j) \right|.$$
   (d) Solve the Matching Problem:
   $$\max_{x_{ij}} \; m_L = \sum_{ij} r_{ij} x_{ij}$$
   $$\text{s.t.} \sum_j x_{ij} = 1$$
   $$x_{ij} \in \{0, 1\}$$
   to obtain a matching between states in G_{L-1} and states in G_L.
5. If $|m_L - m_{L-1}| = 0$, then stop. The current value of L is the correct value.

Theorem 2. *For a model with correct string length L_{max}, the runtime operation of Algorithm 3.3.1 is*

$$O(k^{L_{max}+1}) + O((L_{max} - 1)N)$$

Proof. For each iteration of Algorithm 3.3.1 using string length L, the runtime operation [18] is

$$O(k^{L+1}) + O(N)$$

Each iteration of Algorithm 3.3.1 increases L by one unit. For L starting at length 2 and proceeding to L_{max}, the runtime operation is

$$\sum_{i=2}^{L_{max}} O(k^{i+1}) + O(N) = O(k^{2+1} + \cdots + k^{L_{max}+1}) + O((L_{max} - 1)N)$$

Taking the largest exponential term results in the provided runtime operation. □

3.3.1 Proof of Correctness

We now prove the correctness of the proposed algorithm. We use $H(X)$ to denote the entropy of random variable X. By $H(X|Y)$, we denote the conditional entropy of random variable X given Y [18]. First we list the results from [18] that are used in our proofs:

Lemma 1 (Theorem 1 [18])**.** *Let χ be an observed sequence of symbols and let G^* be the model of χ. Let G be any other Markov model for χ. Let $\chi_{n+1} \in \mathcal{A}$ be a random variable denoting the "next" symbol to be observed after χ. Then $H[\chi_{n+1}|G] \geq H[\chi_{n+1}|G^*]$. (That is, inferred HMMs are maximally prescient of the next symbol to be observed.)*

Lemma 2 (Theorem 3 [18])**.** *Let χ be an observed sequence of symbols* and assume χ is not drawn from a set of measure zero in the sample space*. Then there is a unique Markov model that is minimal in the number of states and obeys Lemma 1. (That is, there is one and only one model representation for each sequence χ with minimal entropy.)*

Theorem 3 (Stopping criteria)**.** *If Algorithm 3.3.1 is executed with a sufficiently long observation sequence χ, then the stopping criteria will be satisfied when the string length is correct.*

Proof. The proof is a consequence of Lemma 2 in [18]. Since Shalizi proves that the causal architecture is unique in Theorem 3, the set of mappings $\mathcal{F}_L$ of $[N] \times \mathcal{A}$ to states V_L are unique and

correctly summarize the statistical relationships in the data when L has the correct value.

The nature of the CSSR Algorithm ensures that the correctness of this assignment will not be changed by increasing the string length. Note that all subsequences of χ *up to* size L are evaluated in the Splitting step of Algorithm 3.1.1. Therefore, no additional information is obtained by considering strings of size $L+1$ and no additional states will be constructed *assuming that χ is of sufficient length and that it is not drawn from a set of measure zero in the sample space*; i.e., correlations among symbols that are not in the string length are not statistically significant). Hence, the inferred HMM found by CSSR using string length $L+1$ must correspond to the causal architecture found using string length L. □

Theorem 4 (Insufficiency)**.** *If Algorithm 3.3.1 is executed with a sufficiently long observation sequence χ, then the stopping criteria will not be satisfied for $L < L^*$ when L^* is the true value of the CSSR Algorithm parameter.*

Proof. This theorem is a direct consequence of combining the Pumping Lemma [46] and Lemmas 1 and 2. The Pumping Lemma states that for any word in a regular language, there is a constant value n such that any word in that language of length greater than n can be written in the form uv^iw such that all uv^iw belong to the language for any i. In other words, any word of length greater than n can be expressed as an instance of u followed by an arbitrary number of repetitions of instances of v followed by w. We recall that finite state machines are acceptors for the class of regular languages. In this case, u and w refer to transient parts of the finite state machine and v refers to the recurrent part of the finite state machine. The CSSR Algorithm removes u and w from the system, since they are transient states. In which case L^* is the same as $\max(|v|)$. Any path through the recurrent portion of the finite state machine can be expressed as system substrings of length L^* or less.

The Pumping Lemma explicitly states that the value L must exist. Lemma 1 proves that the correct state machine is optimal in the sense of the mutual information the states share with the next output symbol. If a value less than L^* is used then there are historical dependencies that will not have been used in constructing the HMM. An HMM constructed using a value less than L^* can not satisfy Lemma 1 and is not correct. HMMs constructed using values using any value L^* or greater contain all the history information in the regular language. Lemma 2 therefore proves that those HMMs must be equivalent. □

3.4 Experimental Demonstration of Algorithm 3.3.1

We now illustrate our results using models where we are certain of the value of L which is necessary for finding the true state structure of the input process.

Using the Markov model provided in Figure 3.1a, we generated a sequence of 2000 symbols. The model in Figure 3.1a is the "unknown" underlying process and the generated sequence is the observation data. Note that the underlying process should only produce two types of subsequences: "ACDY" and "BCDZ." We applied Algorithm 3.3.1 to the generated sequence for values of L varying from 1 to 36. We used an α value of 0.01 for the Kolmogorov-Smirnov test used to execute Step 2 of Algorithm 3.1.1.

Starting with $L = 2$, the CSSR Algorithm (Algorithm 3.1.1) was used to create the model in Figure 3.1b. The CSSR Algorithm constructed a model that will produce both subsequences from the original model and two additional subsequences: "ACDZ" and "BCDY." Rerunning the CSSR Algorithm using $L = 3$ and $L = 4$ produces models shown in Figures 3.1c and 3.1d, respectively. These models capture the structure of the input system perfectly.

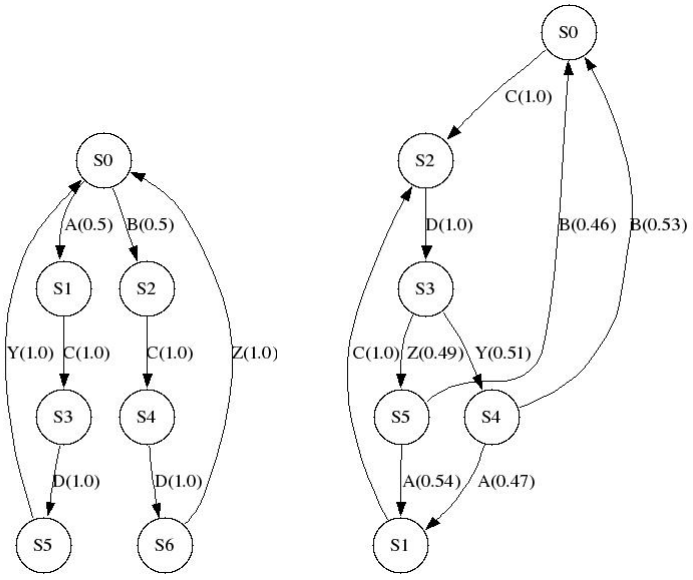


Figure 3.1: (a) A model that generates a sequence of random concatenations of two subsequences (ACDY and BCDZ). (b) The model generated with $L = 2$. ©Elsevier 2009.

A conclusive illustration presenting the validity of the assertion made in Theorem 4 is

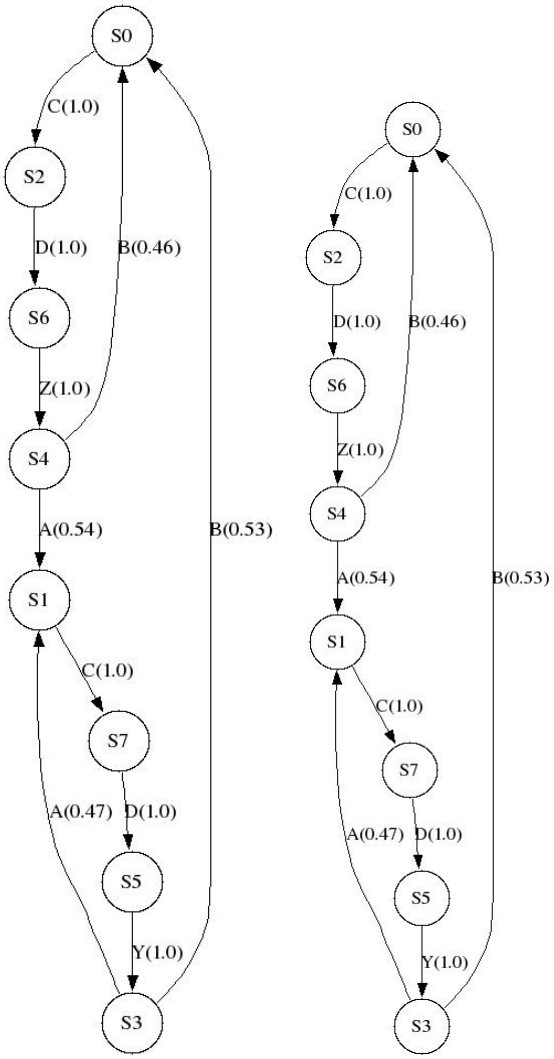


Figure 3.1: (c) The model generated with $L = 3$. (d) The model generated with $L = 4$. ©Elsevier 2009.

provided by the Markov model depicted in Figure 3.2a. Figures 3.2b, 3.2c, and 3.2d show models inferred using $L = 2$, 6, and 7 respectively. Following the same procedure as the previous example, the best metric values (number of symbols in the sequence string mapped to the same symbol) it finds as L increases are:

- $L = 3$ and $L - 1 = 2$ — CSSR (872) (Figure 3.2b)
- $L = 4$ and $L - 1 = 3$ — CSSR (1119)
- $L = 5$ and $L - 1 = 4$ — CSSR (1366)
- $L = 6$ and $L - 1 = 5$ — CSSR (1613) (Figure 3.2c)
- $L = 7$ and $L - 1 = 6$ — CSSR (1860) (Figure 3.2d)
- $L = 8$ and $L - 1 = 7$ — CSSR (1979).
- ...
- $L = 35$ and $L - 1 = 34$ — CSSR (1979).

For this specific instance, Figure 3.3 shows the general behavior of m_L^* in this problem. As is evident from the plot, m_L^* levels off once the correct L is detected and at the stopping point, $m_L^* - m_{L-1}^* = 0$.

3.5 Summary

In this chapter, we explained how to construct ϵ-machines from observation data and a string length value L using Shalizi's and Crutchfield's approach. With our extension, we showed how L can be determined dynamically during the construction process, allowing models to be constructed using zero knowledge. We continue our work on model construction in the next chapter with an analysis of confidence in the constructed models.

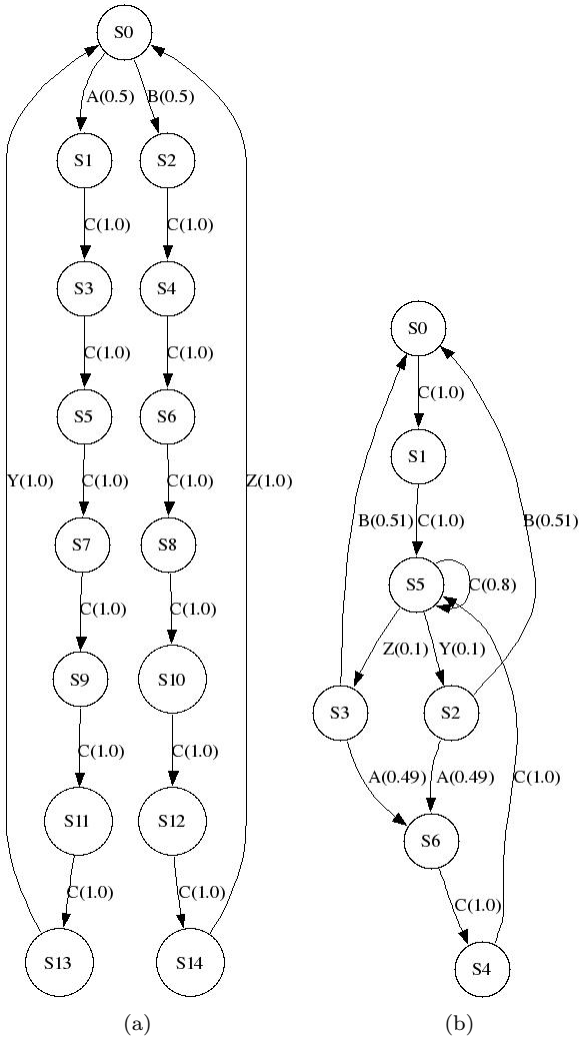


Figure 3.2: (a) A model used to validate result in Theorem 4. (b) The model generated with $L = 2$. ©Elsevier 2009.

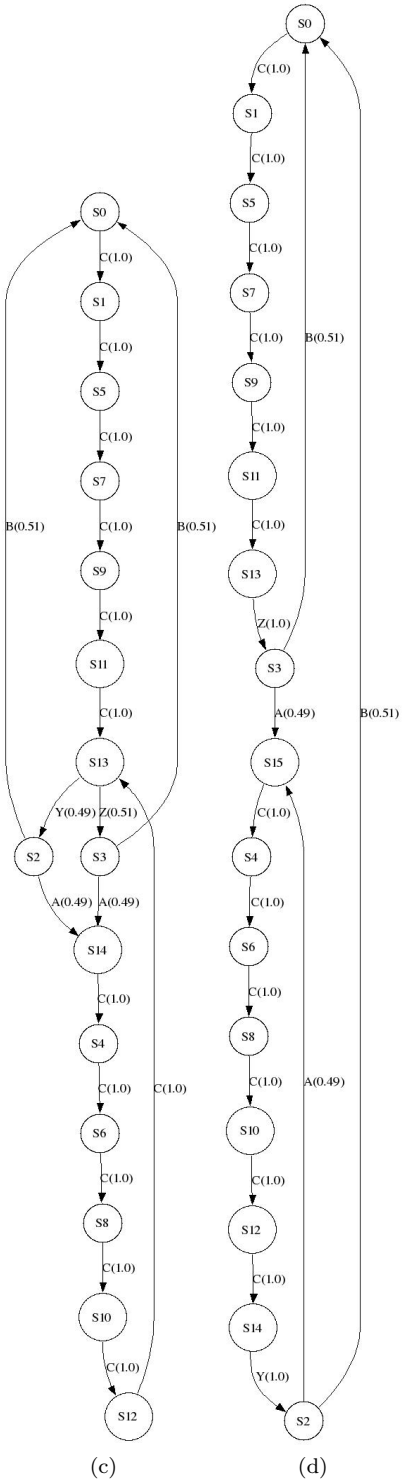


Figure 3.2: (c) The model generated with $L = 6$. (d) The model generated with $L = 7$. ©Elsevier 2009.

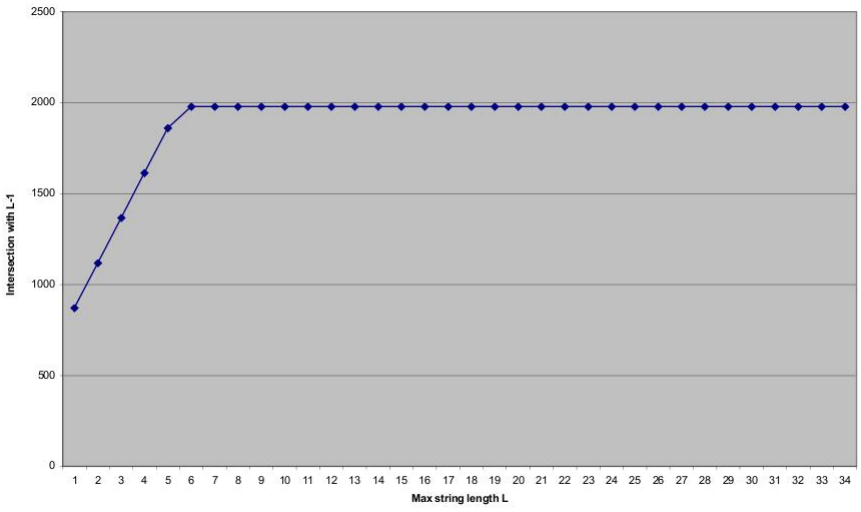


Figure 3.3: Plot showing the values of m_L^* for various values of L. Clearly at the optimal value of L, $m_{L+1}^* - m_L^* = 0$. ©Elsevier 2009.

Chapter 4

Model Confidence

4.1 Introduction

Models are used extensively in science and engineering to explain and predict natural processes. When using models, it is useful to ensure the models accurately represent the observations and the underlying process. The procedure to infer models is an open problem for many different applications, such as network modeling [65], traffic simulation [66], tumbling mill design [67], and crane simulation [68].

When models are dynamically constructed from observations, three questions are raised:

1. is a sufficient amount of observation data available for model creation/training;
2. does the model closely match the observations (model fidelity); and
3. are we confident that the model and observations represent the actual underlying process.

In this chapter, we present two approaches that address the third issue. To determine the model confidence, we consider the observation gathering and model construction procedure outlined in Figure 4.1. An underlying process is observed at some time interval, creating an ordered sequence of observations. The first question corresponds to the size of the observation sequence. The observations are used to construct a minimum entropy HMM. In model fidelity literature, the observations are assumed to completely represent the underlying process. Model fidelity is therefore a measure of how well the constructed model matches the observations. We propose model confidence

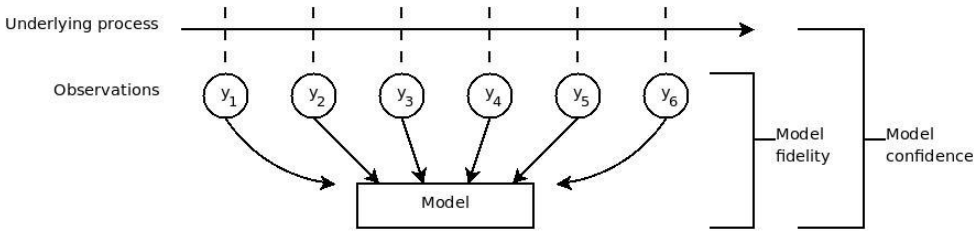


Figure 4.1: Hierarchy of the process, observations, and model showing the relationship between model fidelity and model confidence

to address when the observations may or may not completely represent the underlying process. In this case, model confidence is a measure of how well the model is believed to match the underlying process.

4.2 Current Research

A large body of research across many disciplines proposes solutions to finding a sufficient amount of training data and calculating model fidelity. Lack of training data is a common problem in pattern recognition. Recognition applications typically extract a finite set of features from the data in an attempt to detect known behaviors. If the training data is insufficient, the model parameters may only be representative of the small-scale data set, not the large-scale situation [69]. For speech recognition, [70] creates a definable weight and uses maximum likelihood to select features that are more likely to be representative of the situation instead of special training set cases. Zhu in [71] and Liwicki in [72] merge known, processed data sets with new, unprocessed data to increase the training set size. Yang in [73] generates synthetic data for gait recognition models to perform large-scale gait analysis. Fang in [69] proposes an elastic distortion model to generate more training samples from previously collected data, and a method to stabilize the covariance matrix of the training data. While these approaches enable the models to more effectively recognize a larger spectrum of input data streams, it is important to recognize that artifacts may be introduced into the model due to the artificiality of the data.

As stated in Chapter 1, an assumption for model fidelity is that the collected data is fully representative of the process under observation. Given a model that is believed to represent the situation, we desire a value to ensure that it matches sequential observations. We are not suggesting that high-fidelity models are the only models of value. Rather, our view coincides with [74], which

explains the need for using both low and high-fidelity systems depending on the requirements. Lee in [75] and Nechyba in [76] use a hidden Markov model (HMM) to generate a similarity value between the situation model and the observations. In their examples, continuous observation data is discretized and used to train a five-state HMM such that the HMM is the most likely model to represent the data. Given a model that exactly matches the data and the situation model, which is able to handle probabilistic variations, the similarity value is the ratio of the probabilities that the observation data was generated by both models. Nemirovsky in [77] proposes an approach that uses the frequencies of the deterministic components and the correlation between the stochastic components to produce a fidelity value.

In this chapter, we propose a solution to finding model confidence. The fidelity of a model is a measure of the similarity between the resulting model and the observed data, while accounting for probabilistic variations in the data stream and issues with over-training the model. In this work, we generate the model directly from the data, and consequently, produce the best match for the observations. We therefore must consider the confidence in the resulting model given the data that we have seen. While similar, this definition is slightly different from the definition of model fidelity. In system reliability, [78] phrases this as an optimization problem. Using the probabilities that different modules of a complex system will fail, [78] uses linear programming to find the number of input tests that should be performed on a module to ensure that the probability of failure is not above a defined threshold. To represent the Markov model in the linear program, Poisson distributions are used to simplify the representation by approximating the multiple binomial distributions of the system.

The approach that we present here is similar in purpose to that in [78] except we solve the binomial distribution directly. To use the Poisson approximation, [78] assumes the probabilities trend to zero as the number of samples increases to infinity. Over the long term with many large samples, this approximation is valid because the probabilities are typically significantly smaller than the number of samples. In pattern recognition, however, we assume that the probabilities are stationary and we may not have a sufficiently large number of samples for the approximation to hold. We also wish to find the amount of data needed to be confident that no unexpected events will occur with a defined probability. From the knowledge in the probability that no unexpected events will occur, we can derive our confidence in the model. Furthermore, as behavior models are not well-defined, we must also consider the model construction process in deriving any concept of

model confidence.

4.3 Determining Model Confidence

Assume we have collected a sequence of observations χ of length Z from a process that is representable by a Markov model. We also assume that we have a complete alphabet $\mathcal{A}$ representing all possible observations. We do not know if all symbols in $\mathcal{A}$ are found in χ.

Theorem 5. *Given an underlying process and a finite sequence of discrete observations χ, it is impossible to be absolutely certain that the process is completely encapsulated by χ.*

Proof. The proof is intuitive. Consider an event E where $P(E) = c$ where $0 < c \ll 1$ is a small, non-negligible constant. By the geometric distribution, a random variable R representing when E first occurs is $P(R = r) = c(1-c)^{r-1}$. As $r \to \infty$, $P(R = r) \to 0$ but $P(R = r) \neq 0$. Extending this to situations where we do not know if any events E exist or there are multiple events $E_1, E_2, \ldots$, to determine if χ contains all events of the underlying process requires knowledge of future events or an infinite knowledge source [79]. □

Let K_{v_i} be the set of outgoing transitions from state v_i. As an example using the model in Figure 4.2a, $K_{v_1} = \{(v_1, A), (v_1, C)\}$. Let U_{v_i} be the set of unobserved outgoing transitions from v_i. For the model in Figure 4.2b, $U_{v_1} = \{(v_1, B)\}$ and $U_{v_2} = \{(v_2, B), (v_2, C)\}$. Obviously, $K_{v_i} \cap U_{v_i} = \emptyset \quad \forall v_i \in V$. Our goal is to determine the total number of samples, Z, such that the probability that a transition in U_{v_i} exists is less than a user defined threshold.

If the model was not constructed with a sufficient amount of data, then as data is collected and the model is traversed, each state will be exited using either a transition from K_{v_i} or from U_{v_i}. For the current model to be ruled insufficient to represent the underlying process, a transition in U_{v_i} for any v_i need only be taken one time. Let the probability that a transition in U_{v_i} is taken be represented as β_{v_i}. We first consider a bound when determining values for β_{v_i}.

Lemma 3. *For $\beta^*_{v_i} < \beta_{v_i}$, the expected number of samples needed to predict within a given level of confidence that a transition in U_{v_i} exists with probability β_{v_i} may not be sufficient to predict a transition existing with probability $\beta^*_{v_i}$.*

Proof. We represent the situation with two geometric distributions. Let R_1 represent the number of samples needed to detect if a transition exists with probability β_{v_i} and R_2 represent the same for

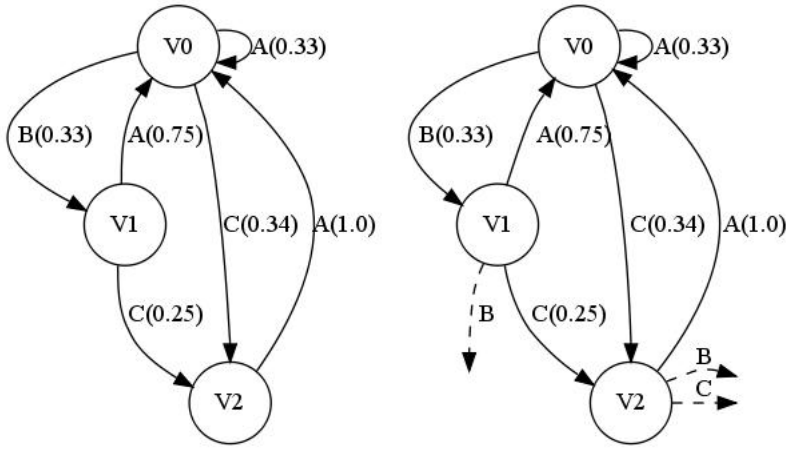


Figure 4.2: Example of known and unknown transitions for alphabet $\mathcal{A} = \{A, B, C\}$. Solid transitions are known and given probabilities calculated from input data. Dashed transitions are possible, but not yet seen. (a) Model generated using CSSR; (b) Model with unobserved transitions.

$\beta^*_{v_i}$. Since $\beta^*_{v_i} < \beta_{v_i}$, we use the expectation of the geometric random variables:

$$E[R_1] = 1/\beta_{v_i}$$

$$E[R_2] = 1/\beta^*_{v_i}$$

$$E[R_1] < E[R_2]$$

As $\beta^*_{v_i} \to \beta_{v_i}$, the expectations become equal, then intuitively the probability that the number of samples needed for β_{v_i} will work for $\beta^*_{v_i}$ increases. In the general case, however, this is unlikely to occur. □

By Lemma 3, we show that as a value is calculated for β_{v_i}, we cannot provide any certainties about values of $\beta^*_{v_i} < \beta_{v_i}$. Furthermore, as a probability, β_{v_i} is bounded between $[0, 1]$.

We define joint event J to be the joint occurrence the system is currently in state v_i and will exit the state using a transition in U_{v_i}.

Definition 1.

$$P(J) = s_i P(e \in U_{v_i}) = s_i \beta_{v_i}$$

where $s_i \in \mathbf{s}$ is the asymptotic state probability for v_i and $\beta_{v_i} = P(e \in U_{v_i})$ is the probability of an unobserved outgoing transition e being taken to leave v_i for another state v_j. $P(J)$ represents the

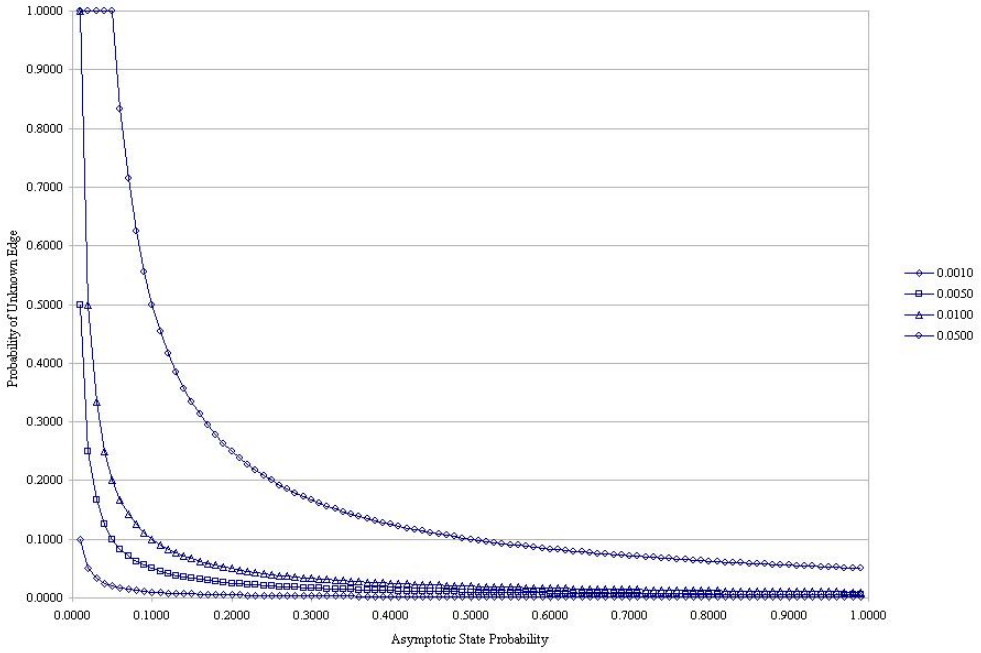


Figure 4.3: Relationship between the asymptotic state probability and the probability of an unknown transition for different levels of κ. κ equals (diamond) 0.001; (square) 0.005; (triangle) 0.01; (circle) 0.05.

probability that the system is in state v_i and takes a transition in U_{v_i} to leave the state.

The probability that the system will take a transition in U_{v_i} is unknown. We could set β_{v_i} as a user-defined value, but with different values for s_i, $P(J)$ would potentially hold a different value for each state. An alternative is to let the probability of the joint event be a user-defined threshold $\kappa = P(J)$. A larger value of κ decreases the amount of data needed for confidence that χ represents the underlying process but increases the risk of an event $e \in U_{v_i}$ occurring. A smaller joint probability decreases this risk but may require significant amounts of data. We can determine the value of β_{v_i} needed to meet the selected value of κ by rewriting Definition 1

$$\beta_{v_i} = \frac{\kappa}{s_i} \tag{4.1}$$

If the model was not constructed with a sufficient amount of data, then as data is collected and the model is traversed, each state will be exited using either a transition from K_{v_i} or from U_{v_i}. For the current model to be ruled insufficient to represent the underlying process, a transition in U_{v_i} for any v_i need only be taken one time. We propose two different algorithms for addressing this

problem.

Both approaches calculate the number of samples required for each state, n_i, to have a desired level of confidence that a transition U_{v_i} does not exist. We must scale up the maximum n_i to find the length of χ.

$$Z = \max_{v_i \in V} \left\lceil \frac{n_i}{s_i} \right\rceil \tag{4.2}$$

We make assumptions about the observation data and knowledge about the underlying process. First, the alphabet $\mathcal{A}$ is complete and contains all expected observations. By assuming this, our approach is restricted to finding "known unknowns" [80] within a given level of statistical confidence. If an observation is not in the alphabet, i.e. is an "unknown unknown," [80] the event does not factor into the confidence in or probability of an unknown event. In addition, if $K_{v_i} = \mathcal{A}$ and $U_{v_i} = \emptyset$, the state does not have any possible untaken outgoing transitions. No more transitions are available to exit the state and testing the state does not change the confidence in the model. We reference state 0 in both models of Figure 4.2 as an example.

4.3.1 Confidence that $e \in U_{v_i}$ does not occur

To determine if the amount of data collected is statistically sufficient, we use a one-sided binomial test on β_{v_i}. The one-sided binomial test allows us to find the smallest number of samples needed for the probability of a transition in U_{v_i} existing is nonzero. The null hypothesis of $H_0 : \beta_{v_i} = 0$ is tested against $H_1 : \beta_{v_i} \neq 0$. We reject H_0 for H_1 when z is greater than the reference statistic, z_α. At this point, if no transitions in U_{v_i} for all v_i are observed, the model is assumed to be with a user-defined level of confidence.

Definition 2. The z-statistic for each state $v_i \in V$ is determined by

$$z_{v_i} = \frac{\beta_{v_i}}{\sqrt{\frac{\beta_{v_i}(1-\beta_{v_i})}{n_i}}}$$

where n_i is the number of times state v_i is entered. The reference normal distribution statistic, z_α with confidence α can be found in a statistics textbook, such as [43].

Lemma 4. *For given z-statistics z_{v_1} and z_{v_2}, the statistic satisfying $\min\{z_{v_1}, z_{v_2}\}$ may be compared to the reference z-statistic in lieu of testing z_{v_1} and z_{v_2} individually.*

Proof. β_{v_i} is held constant for the experiment and is the same for z_{v_1} and z_{v_2}. Therefore from Definition 2, $z_{v_i} \propto 1/\sqrt{n_i}$. Without loss of generality, let us assume that $z_{v_1} = \min\{z_{v_1}, z_{v_2}\}$. Since $n_i \in \mathbb{I}^+$ and $\sqrt{n_i} > 1$, in order for $z_{v_1} < z_{v_2}$ to hold, $n_1 > n_2$. The objective of comparing to the reference statistic is to find the amount of data needed for a specified level of confidence. If $z_{v_1} > z_\alpha$ then consequently $z_{v_2} > z_\alpha$. Since more data is required for z_{v_1} to meet this condition, the minimum of z_{v_1} and z_{v_2} satisfies both the statistical and application requirements. □

Lemma 5.

$$z_{exp} = \min_{v_i} z_{v_i}$$

Proof. The proof is a simple extension of the proof for Lemma 4 which accounts for more than two z-statistics and is omitted. □

Definition 3. Using the calculation for familywise error [81], the model confidence can be determined by

$$\alpha_f = 1 - \prod_{v_i \in V} (1 - P(Z < z_{v_i}))$$

where $P(Z < z_{v_i})$ is the probability that a normal distribution has the value of at least z_{v_i}.

To use the binomial test in this manner, we propose a simple algorithm to perform on-line testing of the observation sequence. The algorithm determines if a constructed model statistically represents a data stream in the process of being collected. We first collect a sequence of observation data χ and construct a model from the collected data. If $|\chi|$ is not sufficiently long, we will be unable to construct a model from the data; additional data should be gathered. If a model is constructed, we determine the z-statistics and find if the experimental statistic provides $100 \cdot (1-\alpha)\%$ confidence that a transition with probability β_{v_i} does not occur. The algorithm is provided in Algorithm 4.3.1.

Once the value of n_i is determined using Algorithm 4.3.1, Equation (4.2) can be used to find the total amount of data required for the model.

4.3.2 Probability that $e \in U_{v_i}$ does not occur

This approach determines the amount of data needed to be certain the probability a transition in U_{v_i} occurs is less than α_p. While similar to the previous approach, we note that confidence

Algorithm 4.3.1 – Proposed Algorithm: z-test

Input: Recurring observation y_t for time t; Alphabet $\mathcal{A}$; User defined κ, α;
From time $t = 1$:

1. Construct model G_t from sequence $\chi = \chi_1 \cdots \chi_t$
2. Calculate the asymptotic state probabilities $\mathbf{s}$
3. Use Equation (4.1) to determine the values for β_{v_i}
4. Calculate the experimental statistics for each state using Definition 2
5. Find z_{exp} using Lemma 5
6. If $z_{exp} > z_\alpha$, conclude G_t and χ sufficiently represent the underlying process with the desired level of confidence

uses the cumulative distribution function in comparisons with a user defined error rate. In this approach, we use the probability mass function of the binomial distribution for comparison.

Definition 4. For a given state v_i, the probability β_{v_i} that the state is exited using a transition from U_{v_i} is

$$\Pr(X = 1) = \binom{n_i}{1} \beta_{v_i}^1 (1 - \beta_{v_i})^{n_i - 1} = n_i \beta_{v_i} (1 - \beta_{v_i})^{n_i - 1}$$

where n_i is the frequency state v_i is entered.

Due to the symmetry of the binomial distribution, this is equivalent to finding $\Pr(X = n_i - 1)$ using $\beta_{v_i}^* = 1 - \beta_{v_i}$. If the state is not exited by a transition in U_{v_i}, then as n_i increases, the probability of such an exit decreases similarly. β_{v_i} is the probability of an unknown transition calculated using the user-defined threshold κ.

Definition 5. The probability that a state v_i is exited using a transition in U_{v_i} can be bounded by a user-defined threshold α_p such that

$$\Pr(X = 1) < \alpha_p$$

Combining Definitions 4 and 5 and rewriting the result provides

$$n_i \ln(1 - \beta_{v_i}) e^{n_i \ln(1 - \beta_{v_i})} < \frac{\alpha_p (1 - \beta_{v_i}) \ln(1 - \beta_{v_i})}{\beta_{v_i}} \tag{4.3}$$

Theorem 6. *For state v_i, the number of samples needed for the probability that an unobserved*

transition in U_{v_i} occurs is at most α_p is

$$n_i > \left\lceil \frac{1}{\ln(1-\beta_{v_i})} \cdot W\left(\frac{\alpha_p(1-\beta_{v_i})\ln(1-\beta_{v_i})}{\beta_{v_i}}\right)\right\rceil \tag{4.4}$$

Proof. Using the Lambert W function [82], we isolate n_i from Equation (4.3). Due to the fact that $1-\beta_{v_i} < 1$ and $\ln(1-\beta_{v_i}) < 0$, we reverse the inequality. Furthermore, $n_i \in \mathbb{N}^+$; we incorporate the ceiling function for this restriction. □

Remark 2. We note that the probability mass function for the binomial distribution is symmetric about the mean, and as such, two values of n_i exist that fulfill the condition in Definition 5. Of the two real-valued branches, the 0 branch of the Lambert W family corresponded to the lower of the n_i values, while the -1 branch produced the maximum value of the two possible values that solve Equation (4.4).

Like the previous approach, once n_i is determined by Equation (4.4), we use the asymptotic state probabilities and Equation (4.2) to find the total amount of data required for the model.

Equation (4.2) gives the amount of data needed to be confident in one state. If Definition 4 is used for a given β_{v_i} for each state, the confidence in the model can be determined for an increasing amount of data Z. We use the combination of the experimentwise error to produce the familywise error [81].

Definition 6.

$$\alpha_f(Z) = 1 - \prod_{v_i \in V} (1 - Z s_i \beta_{v_i}(1-\beta_{v_i})^{Z s_i - 1}) \tag{4.5}$$

where for state v_i, s_i is the asymptotic state probability and β_{v_i} is determined from Equation (4.1).

Remark 3. If a specific model confidence is desired, Definition 6 can be rewritten as the Sidak equation [81] with the assumption that the same level of confidence is desired across all states:

$$\alpha_p = 1 - (1-\alpha_f)^{\frac{1}{|V|}}$$

where $|V|$ is the size of the state space. Using the value of α_p calculated in this manner, we use Equations (4.4) and (4.2) to find the amount of data required for a determined level of model confidence α_f.

To use the proposed equations, we present a simple algorithm to perform on-line testing of the observation sequence. The algorithm determines if a constructed model sufficiently represents a data stream in the process of being collected. We begin by creating a sequence χ from the observations. We construct a model from the gathered data. If $|\chi|$ is not sufficiently long, we will be unable to construct a model from the data set, therefore additional data should be gathered. If a model is constructed, we determine if the predicted amount of data is greater than the amount of observations. This procedure is described in Algorithm 4.3.2.

Algorithm 4.3.2 – Proposed Algorithm

Input: Recurring observation χ_t for time t; Alphabet $\mathcal{A}$; User defined κ, α_p;
From time $t = 1$:

1. Construct model G_t from sequence $\chi = \chi_1 \cdots \chi_t$
2. If G_t the state or transition space of G is of measure zero, collect next observation χ_{t+1}
3. Else, given G_t, use Equations (4.4) and (4.2) to find Z
4. If $Z > |\chi|$, continue to the next χ_{t+1}
5. Else, the probability of a transition in U_{v_i} occurring is less than α_p for all states and model G_t represents χ with confidence specified in Definition 6

4.4 Algorithm Demonstrations

We demonstrate the utility of confidence in the model with an illustrative example. Consider the model shown in Figure 4.4a with asymptotic state probabilities given in Table 4.1. Our alphabet for this experiment was $\mathcal{A} = \{A, B, C, D\}$. We varied the value of $p = \Pr(D) = \{0.1, 0.01, 0.001\}$ to produce three different models. From each of the three models, we selected a random start state (occurring at time 0) and, using the probabilities of the transitions, stepped through the model to generate a series of observations. Each step represented an integer increase in time, i.e. the first symbol occurred at time 1, etc. With the series of observations, we used Algorithms 4.3.1 and 4.3.2 to find the value of Z predicted to be needed for selected values of κ. We set $\alpha_c = \alpha_p = 0.05$ for all tests for a model confidence of 81.45%. The sequence χ generated was kept constant for all values of κ to allow us to compare the results across κ. The amount of data predicted to be necessary for a 95% confidence per state is given in the two final columns of Table 4.2 for various selections of κ. As expected, the predicted amount of data increases as κ decreases. We also note that statistical confidence from Algorithm 4.3.1 does not require as many samples as the direct probability calculation from Algorithm 4.3.2.

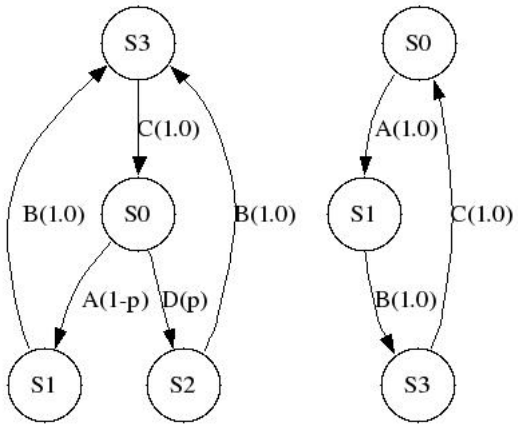


Figure 4.4: Models observed in illustrative tests; (a) Initial model; (b) Model without "D" transition.

Table 4.1: Asymptotic state probabilities for models with structure in Figure 4.4a

State	$p = 0.1$	$p = 0.01$	$p = 0.001$
0	$0.3\overline{3}$	$0.33\overline{3}$	$0.333\overline{3}$
1	0.3	0.33	0.333
2	$0.0\overline{3}$	$0.00\overline{3}$	$0.000\overline{3}$
3	$0.3\overline{3}$	$0.33\overline{3}$	$0.333\overline{3}$

Table 4.2: Predicted Z and calculated β_{v_i} for various κ

κ	β_{v_i}	Z (Alg 4.3.1)	Z (Alg 4.3.2)
$0.0\overline{5}$	0.15	49	87
$0.03\overline{6}$	0.11	67	118
$0.0\overline{3}$	0.10	76	131
0.03	0.09	85	146
0.01	0.03	265	447
0.005	0.015	535	897
$0.003\overline{6}$	0.011	733	1223
$0.00\overline{3}$	0.01	805	1346
0.003	0.009	895	1496
0.001	0.003	2701	4496
0.0005	0.0015	5404	8996
$0.0003\overline{6}$	0.0011	7372	12268
$0.000\overline{3}$	0.001	8110	13495
0.0003	0.0009	9013	14995

Our selections for κ were primarily based on their relationship with β_{v_i}. Using a variation of Equation (4.1), we can determine the κ necessary for a specific β_{v_i} and an asymptotic state probability of $0.3\overline{3}$ (the maximum from Table 4.1). Column two of Table 4.2 provides a list of β_{v_i} values and their corresponding κ values. Note that we chose values of κ such that β_{v_i} was equivalent,

Table 4.3: Number of matches of models in Figure 4.4 using Algorithm 4.3.1 (out of 50 trials)

	$p = 0.1$		$p = 0.01$		$p = 0.001$	
κ	Fig. 4.4a	Fig. 4.4b	Fig. 4.4a	Fig. 4.4b	Fig. 4.4a	Fig. 4.4b
0.05	39	11	0	50	0	50
$0.03\overline{6}$	42	8	0	50	0	50
$0.0\overline{3}$	48	2	0	50	0	50
0.03	48	2	0	50	0	50
0.01	50	0	28	22	6	44
0.005	50	0	42	8	9	41
$0.003\overline{6}$	50	0	44	6	14	36
$0.00\overline{3}$	50	0	46	4	16	34
0.003	50	0	47	3	16	34
0.001	50	0	50	0	30	20
0.0005	50	0	50	0	37	13
$0.0003\overline{6}$	50	0	50	0	45	5
$0.000\overline{3}$	50	0	50	0	46	4
0.0003	50	0	50	0	48	2

just greater than, and just less than the values of p for our three models. We chose additional values of κ arbitrarily to provide a more uniform list.

As the defined value for κ decreased, the smallest β_{v_i} values decreased correspondingly. The smallest κ (0.0003) predicted an extremely large amount of data was needed to be 95% confident that each state independently did not have a transition greater than 0.0009. Based on probability, we expected to be unable to detect the transitions $p = \{0.1, 0.01, 0.001\}$ using values of κ greater than $0.0\overline{3}$. When the "D" transition is not detected, the model takes on the structure in Figure 4.4b. We recorded the number of times the constructed models matched those in Figure 4.4 from fifty independent tests for each experiment. Tables 4.3 and 4.4 present the summary values when using Algorithms 4.3.1 and 4.3.2, respectively.

A cursory analysis of Table 4.3 reveals that when a value of κ is selected such that β_{v_i} is equivalent to a value of p, Algorithm 4.3.1 fully reconstructs the underlying process in approximately 95% of the cases. This supports our choice of α_c. For the smaller selections of p, the algorithm is also able to reconstruct the underlying process for many of the sequences when β_{v_i} is slightly larger than p. When β_{v_i} is considerably larger than p (i.e. the chosen κ is large, the sequences do not contain a "D" symbol. The "D"-loop is hidden from the constructed model because it had not yet occurred. We provide a statistical analysis of the generated sequences for the three models in Section 4.5.

Analyzing Table 4.4, we observe similar results to those discussed previously. When a

Table 4.4: Number of matches of models in Figure 4.4 using Algorithm 4.3.2 (out of 50 trials)

	$p = 0.1$		$p = 0.01$		$p = 0.001$	
κ	Fig. 4.4a	Fig. 4.4b	Fig. 4.4a	Fig. 4.4b	Fig. 4.4a	Fig. 4.4b
0.05	0	50	0	50	0	50
$0.03\overline{6}$	23	37	0	50	0	50
$0.0\overline{3}$	50	0	9	41	2	48
0.03	50	0	12	38	3	47
0.01	50	0	40	20	8	42
0.005	50	0	47	3	16	34
$0.003\overline{6}$	50	0	49	1	18	32
$0.00\overline{3}$	50	0	50	0	20	30
0.003	50	0	50	0	21	29
0.001	50	0	50	0	33	17
0.0005	50	0	50	0	48	2
$0.0003\overline{6}$	50	0	50	0	50	0
$0.000\overline{3}$	50	0	50	0	50	0
0.0003	50	0	50	0	50	0

Table 4.5: Asymptotic state probabilities for models with structure in Figure 4.5

State	$p = 0.1$	$p = 0.01$	$p = 0.001$
0	0.25	0.25	0.25
1	0.225	0.2475	0.24975
2	0.025	0.0025	0.00025
3	0.225	0.2475	0.24975
4	0.025	0.0025	0.00025
5	0.225	0.2475	0.24975
6	0.025	0.0025	0.00025

value of κ is selected such that β_{v_i} is equivalent to a value of p, Algorithm 4.3.2 fully reconstructs the underlying process in all fifty independent tests, an improvement over Algorithm 4.3.1. The algorithm is also able to reconstruct the underlying process for many of the sequences when β_{v_i} is slightly larger than p. When β_{v_i} is considerably larger than p (i.e. the chosen κ is large, the sequences do not contain a "D" symbol. We note, however, that Algorithm 4.3.2 states more data is required than Algorithm 4.3.1 and thus should produce better results.

We now consider a second example demonstrating the functionality Algorithms 4.3.1 and 4.3.2. Using the model in Figure 4.5, with asymptotic state probabilities given in Table 4.5, to represent the underlying process, we let $p = \{0.1, 0.01, 0.001\}$ for three independent experiments. Table 4.6 provides a summary of values for κ, their corresponding β_{v_i} values, and the number of samples predicted by the algorithms.

We followed the same procedure as in the previous demonstration and generated fifty se-

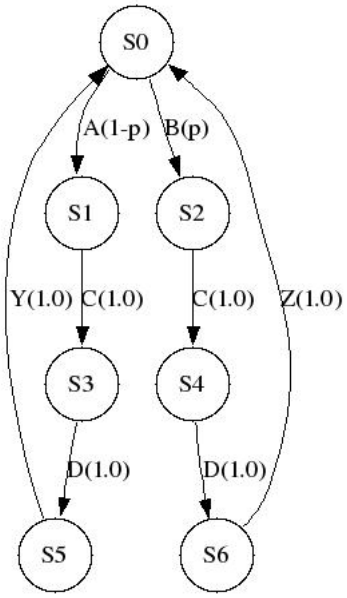


Figure 4.5: Model used in second illustrative test to represent the underlying process

Table 4.6: Predicted Z and calculated β_{v_i} for various κ

κ	β_{v_i}	Z (Alg 4.3.1)	Z (Alg 4.3.2)
0.05	0.2	44	84
0.0275	0.11	88	158
0.025	0.10	100	174
0.0225	0.09	112	194
0.01	0.04	260	444
0.005	0.02	532	894
0.00275	0.011	976	1630
0.0025	0.01	1072	1794
0.00225	0.009	1192	1994
0.001	0.004	2696	4494
0.0005	0.002	5404	8994
0.000275	0.0011	9828	16357
0.00025	0.001	10812	17993
0.000225	0.0009	12016	19993

quences of 50,000 symbols. Information and summary statistics about the sequences can be found in Section 4.6. Due to the larger state space of the model, a greater number of variations were generated by the CSSR Algorithm when an insufficient number of samples were used to construct the model. We recorded the number of times when the CSSR Algorithm reconstructed the model depicted in Figure 4.5 and when it did not. Furthermore, as p decreased, a smaller number of sequences contained occurrences of "B,C,D,Z" to an extent that allowed reconstruction. When $p = 0.1$, the

Table 4.7: Number of matches of models in Figure 4.5 using Algorithm 4.3.1 (out of 50 trials)

	$p = 0.1$		$p = 0.01$		$p = 0.001$	
κ	Fig. 4.5	Not	Fig. 4.5	Not	Fig. 4.5	Not
0.05	0	50	0	39	0	10
0.0275	0	50	0	39	0	10
0.025	3	47	0	39	0	10
0.0225	5	45	0	39	0	10
0.01	23	27	2	37	0	10
0.005	39	11	3	36	0	10
0.00275	46	4	3	36	0	10
0.0025	48	2	3	36	0	10
0.00225	49	1	3	36	0	10
0.001	50	0	4	35	1	9
0.0005	50	0	8	31	1	9
0.000275	50	0	13	26	3	7
0.00025	50	0	14	15	3	7
0.000225	50	0	14	15	4	6

CSSR Algorithm was able to reconstruct the model representing the underlying process in all fifty sequences. The total sequences decreased to 39 and 10 for $p = 0.01$ and $p = 0.001$, respectively. Tables 4.7 and 4.8 list our results when using Algorithms 4.3.1 and 4.3.2, respectively.

Analyzing the results, we see that both algorithms correctly predicted the amount of data for large p. As with the previous example, Algorithm 4.3.2 produced slightly better results because it predicted additional data would be necessary to meet the defined value for α_p. Neither algorithm was able to reconstruct the model representing the underlying process for the latter two values of p. This was most likely due to the decreased occurrence of the subsequence "B,C,D,Z" in the observation data. This subsequence only had a chance of occurring every fourth symbol. In observation data of 50,000 symbols, this equates to 12,500 chances for the "B" transition to be taken. As the probability of the transition decreased, the number of samples needed to determine if the "B" branch of the model was statistically significant had not been gathered. This does not mean that the algorithms are incorrect. Rather it means that given the observation data, the minimum-entropy Markov model to represent the data did not include a separate "B" branch from the "A" branch.

4.5 String Information for model in Figure 4.4a

For each of the fifty sequences generated using the model in Figure 4.4a, we determined the amount of data to fully reconstruct the model by visually checking the constructed models. Table

Table 4.8: Number of matches of models in Figure 4.5 using Algorithm 4.3.2 (out of 50 trials)

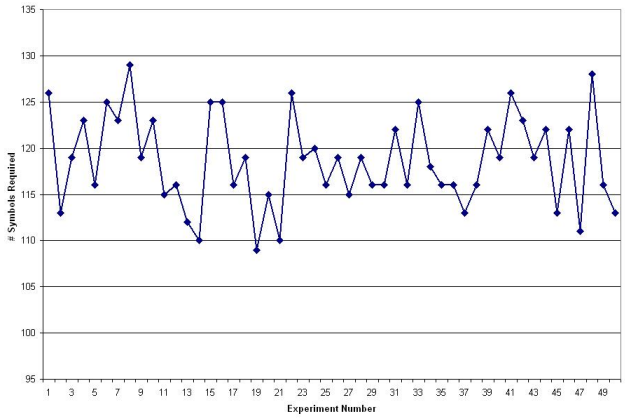
	$p = 0.1$		$p = 0.01$		$p = 0.001$	
κ	Fig. 4.5	Not	Fig. 4.5	Not	Fig. 4.5	Not
0.05	0	50	0	39	0	10
0.0275	12	38	1	38	0	10
0.025	14	36	1	38	0	10
0.0225	17	33	1	38	0	10
0.01	36	14	3	36	0	10
0.005	45	5	3	36	0	10
0.00275	49	4	4	35	0	10
0.0025	50	0	4	35	0	10
0.00225	50	0	4	35	0	10
0.001	50	0	8	31	1	9
0.0005	50	0	11	31	3	7
0.000275	50	0	19	20	5	5
0.00025	50	0	24	15	5	5
0.000225	50	0	25	14	5	5

Table 4.9: Summary of results from visual inspection

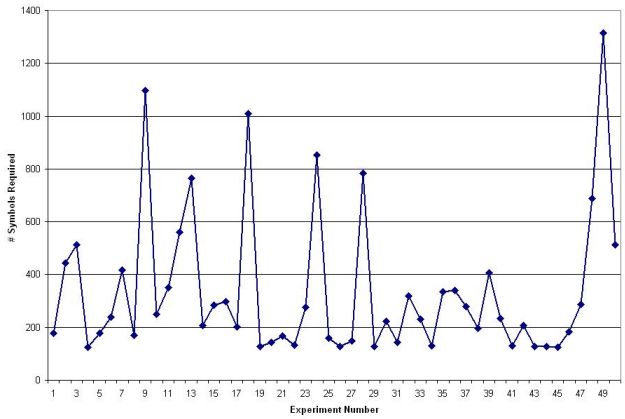
	$p = 0.1$	$p = 0.01$	$p = 0.001$
Minimum	109	125	128
Median	119	232	1782
Maximum	129	1314	9738
Mean	118.6	337.44	3105.24
St. Dev.	5.01	276.27	2897.07

4.9 provides a summary of statistical results for the visual inspection. Figure 4.6 shows the amount of data required for each sequence as determined visually. Note that the minimums for all three models are roughly equivalent. The minimum cases occurred when the symbol "D" appeared near the beginning of the observations. The high variation in the amount of data needed is the reason why methods that are too dependent on the data stream are inaccurate.

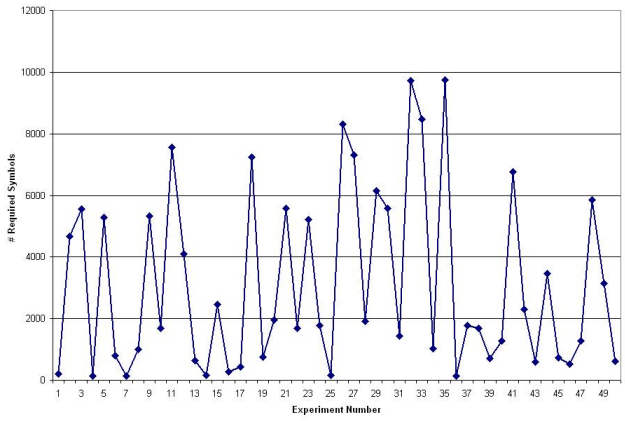
We next considered the statistics for the first occurrence and number of occurrences of "D" in the sequences. Summary statistics are provided in Table 4.10. The first occurrence column provides the index number of the first time "D" is observed in the sequence. The number of occurrences column shows the number of times that "D" occurred in the subsequence $\gamma = \gamma_0 \cdots \gamma_k$ where $k = \{129, 1314, 9738\}$ for $p = \{0.1, 0.01, 0.001\}$, respectively. The values of k are the maximum values from the visual inspection in Table 4.9. This data shows that "D" occurred very early in some sequences, but the first occurrence tended to have high variability. Using the values of k, the number of times we expect to see the symbol "D" for the three models is 4.3, 4.4, and 3.2,



(a)



(b)



(c)

Figure 4.6: Minimum number of symbols needed to recreate model from Figure 4.4a using visual inspection for all 50 generated sequences. (a) $p = 0.1$; (b) $p = 0.01$; (c) $p = 0.001$

Table 4.10: Summary information about generated sequences

	$p = 0.1$		$p = 0.01$		$p = 0.001$	
	First occur.	# occur	First occur.	# occur	First occur.	# occur
Minimum	0	1	0	1	20	1
Median	15	5	224	4	1779	3
Maximum	110	8	1311	10	9735	8
Mean	26.66	4.52	319.28	4.42	3099.3	3.08
St. Dev.	27.11	1.71	289.11	2.01	2900.2	1.72

respectively, for the different values of p. The mean values for the number of occurrences in Table 4.10 are similar. This shows that the sequences are behaving as we would expect in the number of occurrences that "D" is observed.

4.6 String Information for model in Figure 4.5

For each of the fifty sequences generated using the model in Figure 4.5, we determined the amount of data to fully reconstruct the model by visually checking the constructed models. Table 4.11 provides a list of some statistics about the visual inspection results. The count data provides the number of sequences that we could visually determine matched the underlying process. For the latter two models ($p = 0.01$ and $p = 0.001$), sequences that did not return a model matching the underlying process within 50,000 symbols were not included in the statistics. For observation data of 50,000 symbols, the amount of variability increased considerably as the probability of observing "B,C,D,Z" decreased. We believe that the variability is due to the extreme commonality between the two observed sequences. The last symbol of a subsequence of four depended only on the preceding three symbols (in many cases only on the first symbol, e.g. Z had a 1:1 correlation with B). Figure 4.7 shows the variability in the visual inspection results. Points shown as zero were sequences where we could not determine the value through visual inspection.

Table 4.11: Summary of results from visual inspection

	$p = 0.1$	$p = 0.01$	$p = 0.001$
Count	50	39	10
Minimum	94	136	2024
Median	312	16510	17744.5
Maximum	1739	48590	45417
Mean	392.08	18462.36	21725.5
St. Dev.	324.08	13678.7	16150.14

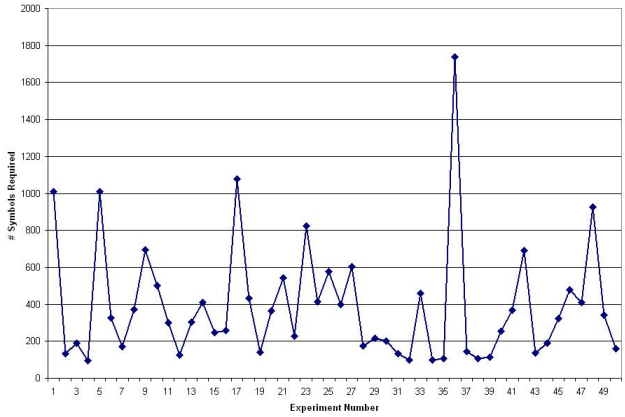
We next considered the statistics for the first occurrence and number of occurrences of "B,C,D,Z" in the sequences. Summary statistics are provided in Table 4.12. As in the previous section, the first occurrence column provides the index number when the first subsequence is observed in the sequence. The # occurrences column shows the frequency that "B,C,D,Z" occurred in the subsequence $\gamma = \gamma_0 \cdots \gamma_k$ where $k = \{1739, 48590, 45417\}$ for $p = \{0.1, 0.01, 0.001\}$, respectively. The values of k are the maximum values from the visual inspection in Table 4.11. This data shows that "B,C,D,Z" occurred very early in some sequences, but the first occurrence tended to have high variability. Using the values of k, the number of times we expect to see the symbol "B,C,D,Z" for the three models is 43.5, 121.5, and 11.4, respectively, for the different values of p. The mean values for the number of occurrences in Table 4.12 are similar with the exception of $p = 0.001$. In this case, the subsequence "B,C,D,Z" occurred more frequently than expected in slightly more than 45,000 symbols.

Table 4.12: Summary information about generated sequences

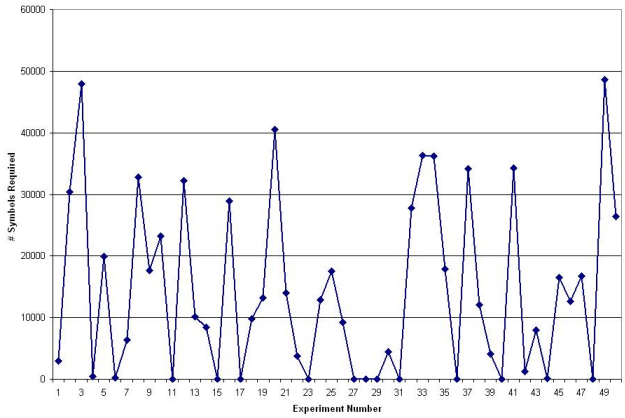
	$p = 0.1$		$p = 0.01$		$p = 0.001$	
	First occur.	# occur	First occur.	# occur	First occur.	# occur
Minimum	1	32	3	99	3	39
Median	26	41	223.5	123	635.5	55
Maximum	134	59	1705	154	4379	77
Mean	35.50	42.38	403.96	122.52	966.08	55.44
St. Dev.	33.55	5.97	433.45	12.14	936.82	8.44

4.7 Summary

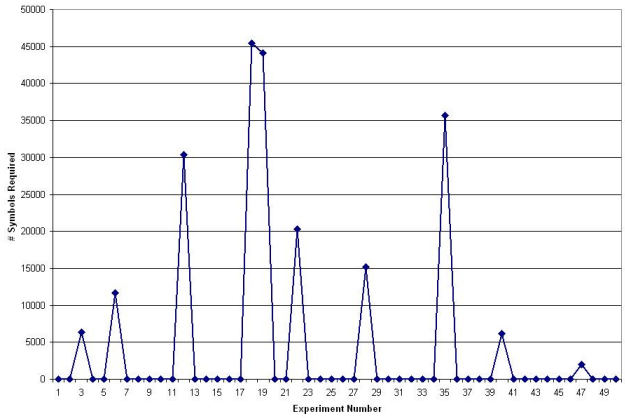
In this chapter, we explained two algorithms that we used to develop a level of confidence in the constructed models. The algorithms run in parallel with data collection. When the number of samples gathered causes different calculations to exceed user-defined thresholds, the expected level of confidence is achieved. This concludes our work on model construction. In the next chapter, we discuss a new method to solve the problem of matching constructed models and new observation sequences.



(a)



(b)



(c)

Figure 4.7: Minimum number of symbols needed to recreate model from Figure 4.5 using visual inspection for all 50 generated sequences. A value of zero indicates that more than 50,000 symbols were needed. (a) $p = 0.1$; (b) $p = 0.01$; (c) $p = 0.001$

Chapter 5

Pattern Detection

5.1 Introduction

Given a set of Markov models $\{G_k\} = G^*$ for $k = 1, 2, \ldots,$ $\Pr(\chi|G_k)$ is a conditional maximum likelihood estimate for all $G_k \in G^*$, and the Markov model G_k associated with the largest probability is the HMM most likely to have generated the observation [83].

If multiple HMMs have the same $\Pr(\chi|G)$ value, the estimator may return either the set of HMMs or an individual HMM. All of the experiments we demonstrate returned an individual HMM. It is somewhat unlikely, in realistic applications, that more than one HMM in a set returns the same value because of floating point precision.

The Forward-Backward Procedure discussed in Chapter 2.4.1 computes a weighted average of products of probabilities. As the observation sequence length increases, the probability that any given HMM generated the observation decreases monotonically. This has the following drawbacks:

- When the observed output sequence is short, the Forward-Backward approach ignores the uncertainties associated with making decisions based on a small number of data samples.
- When the number of data samples is large, the Forward-Backward Procedure makes decisions by comparing the values of progressively shrinking floating point values (see Figure 5.1). It is counterintuitive that the metric for matching observations to HMMs decreases as the number of samples used increases. In addition, the number of valid significant digits in the answer decreases with each multiplication. This makes comparison of the infinitesimal values produced

suspect, when considering long output sequences. This can be mitigated through the use of scaling factors by normalizing (Equation (5.1)) and calculating a scaling factor c_i (Equation (5.2)) for each α-value found in the Forward-Backward Procedure. The Forward-Backward Procedure can then be modified to account for the modified α-values [84].

$$\hat{\alpha}_i = \frac{\alpha_i}{\Pr(\chi_1 \dots \chi_i)} \tag{5.1}$$

$$c_i = \Pr(\chi_i | \chi_1 \dots \chi_{i-i}) \tag{5.2}$$

$$\text{New Induction: } c_{t+1}\hat{\alpha}_{t+1}(j) = \sum_{i=1}^{n} \hat{\alpha}_t(i) p_{i,j},\ 1 \leq t \leq M-1$$

$$\text{New Termination: } \Pr(\chi|G) = \Pi_{j=1}^{n} c_j$$

The scaling modification does not address the issue of uncertainty with small sample sizes and cannot indicate whether a sequence is adequately represented by any models in a dictionary. Additionally, if used in actual behavior recognition software, errors will be magnified due to the increased number of multiplications and the inherent inaccuracies with computer floating-point calculations leading to potential instability of the final result [85].

5.2 The Main Problem

We look at the problem of identifying the presence of a behavior in the data stream that is modelable as a Markov model. We accomplish this by matching the data stream with a set of existing Markov models. We calculate a confidence interval for each of the transitions in the model and determine when the model sufficiently matches the input data. Our approach is not concerned with maximizing the performance of the model with additional data streams. Therefore, no training or tuning of the model parameters is performed.

Our problem is subtly different from the problems discussed above and previously in Chapter 2.4.1. The first two tasks are typically performed using the "forward" portion of the Forward-Backward Procedure. The Forward-Backward Procedure uses a maximum likelihood algorithm that determines the model that best matches the input data stream. The "backward" portion of the Procedure automatically adjusts the model parameters to maximize the performance of the model

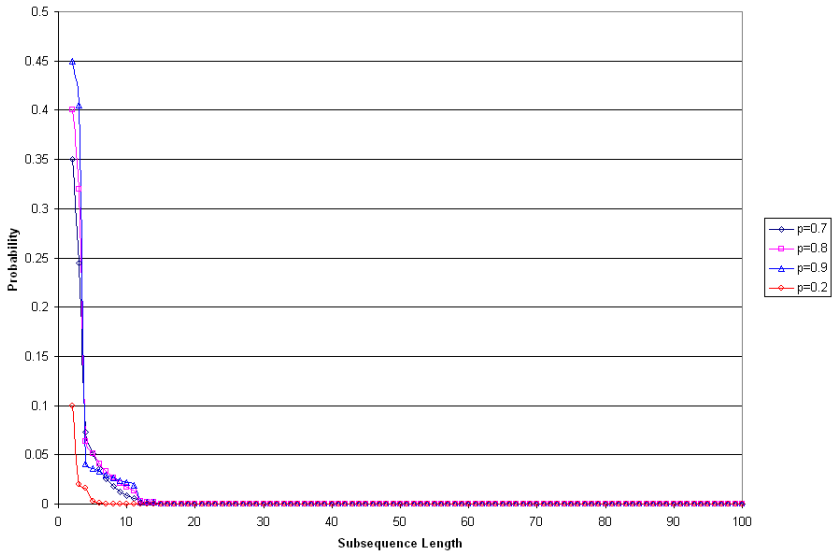


Figure 5.1: Maximum Likelihood Probabilities (y axis) for multiple Markov models versus the length of the string considered (x axis). The data sets used are generated using the models described in Section 5.4, with parameters: (Square) Markov model $p = 0.8$. (Diamond) Markov model $p = 0.7$. (Triangle) Markov model $p = 0.9$. (Circle) Markov model $p = 0.2$. Image reproduced with permission from [21]. ©IEEE 2009.

with additional input data streams.

The confidence interval based approach does not have the drawbacks associated with the Forward-Backward Procedure. The decisions made consider the number of samples used and become more certain as the number of samples considered increases. For short observation sequences, uncertainty in the decision is reflected by a larger variance and, correspondingly, a larger confidence interval. Furthermore, as the observation sequence increases in length, the number of multiplications necessary to calculate the confidence interval remains constant, alleviating the floating point value issue.

The problem we address is slightly different from the one solved by the Forward-Backward Procedure. Strictly speaking, ML solves a classification problem where an observation is mapped to one of a number of known classes. EM, MMI, and MCE then alter the parameters of the model to give the highest number of true positives and the lowest number of false positives. Our approach solves a detection problem where the behavior described by a HMM is either found or not found in a given data stream. In practice, as we will show, the two algorithms can be used for many of the same applications.

5.3 Confidence Interval Analysis

This section presents our approach that uses confidence intervals to identify HMM described behaviors in sensor streams. First, we explain how to calculate confidence intervals for HMM transitions. We then explain how to use them for identifying behaviors in sensor streams.

5.3.1 Calculating Confidence Interval Bounds

Given a Markov model and a sequence of observations $\chi = \chi_1\chi_2\cdots\chi_n$, we construct approximations of the transition probability matrix $\mathbf{P}$ for every possible start state v_0.

Starting at state v_0, follow the transitions (v_i, v_j) associated with each symbol $\chi_i \in \chi$ in turn, giving the path taken by the output sequence through the HMM. Since our HMMs are all deterministic, this path is unique. First initialize all counters to zero, and then, starting with $i = 0$, for each χ_i:

1. If the probability associated with transition (v_i, v_j) in $\mathbf{P}$ is zero, then the sequence could not have been generated by the HMM starting at v_0. The mapping is rejected and the process stops. When using HMMs, we must consider all possible starting states using an initial probability vector π. By eliminating the states that cannot produce the observed sequence, we limit the number of results.

2. Else, add one to the counter c_i for state v_i and add one to the counter $c_{i,j}$ for transition (v_i, v_j).

3. The current estimate for $p_{i,j}$ is: $\hat{p}_{i,j} = \frac{c_{i,j}}{c_i}$.

4. Since c_i is the number of observations of the system in state v_i, the confidence interval around $p_{i,j}$ is (where $Z_{\alpha/2}$ is from the standard normal distribution or student's t-distribution with degrees-of-freedom $c_i - 1$):

$$\left[p_{i,j} - Z_{\alpha/2}\sqrt{p_{i,j}(1-p_{i,j})/c_i}, p_{i,j} + Z_{\alpha/2}\sqrt{p_{i,j}(1-p_{i,j})/c_i}\right] \tag{5.3}$$

   We use the value $p_{i,j}$ from matrix $\mathbf{P}$ instead of the estimated value $\hat{p}_{i,j}$. Although either value could be used, the variance $p_{i,j}(1-p_{i,j})$ is typically less subject to sampling errors. Note that these values are constrained to remain within the set $[0, 1]$ and the interval is the asymptotic limit of the binomial, decreasing in size as the data length increases. We note Equation (5.3)

is the Wald confidence interval. Other confidence intervals may be substituted for the Wald interval [86, 87]. For greater accuracy in the representation, multinomial confidence intervals may be used [88]. This allows the system to account for the covariance and will most likely decrease the size of the confidence intervals calculated.

5. If $\hat{p}_{i,j}$ is within the confidence interval, we accept the null hypothesis that $\hat{p}_{i,j} = p_{i,j}$ with the probability of a type I error for that transition equal to α [43]. The observed sequence is consistent with the HMM model for that transition.

5.3.2 Using Confidence Interval Bounds

We use the following procedure to determine if the observed sequence matches Markov models:

1. Select a HMM to match with the observed sequence.

2. Use ROC curves [89, 90] to find the optimal threshold for accepting or rejecting the HMM mapping[1].

3. Use frequency counting to estimate the transition probabilities and calculate the confidence interval bounds as described in Section 5.3.1.

4. Determine the percent of transition probabilities from the originally selected Markov model that fall within their respective confidence interval.

5. If the percentage of transitions taken by the observed sequence that fall within the confidence interval is greater than the threshold value from step 2, reject the hypothesis that the observed sequence is not an occurrence of the HMM, and register a detection event.

This approach can either analyze the entire sequence or consider windows of the observed sequence. Windowing selects w symbols starting at a given symbol in the observation sequence. For example, the first window covers the subsequence $[\chi_0 \ldots \chi_w]$ and the second window covers subsequence $[\chi_1 \ldots \chi_{w+1}]$. In our application, objects may change their behavior during observation. As we increase the window size w, the confidence interval around each transition tightens causing a decrease in the false positive rate. A larger w also increases the amount of time needed to recognize a

[1]This step is done off-line. Example ROC curves are in Section 5.4

transition between behavior modes. A more complete description of windowing with Markov models is given in Chapter 6.

5.3.3 Discussion

Each transition taken by an HMM can be viewed as a set of Bernoulli trials. The asymptotic statistics describing confidence intervals of random variables are one of the best established aspects of probability theory. The HMM provides a state space structure uniting a set of independent Bernoulli distributions.

Given the *a priori* transition probability $p_{i,j}$, 95% of the randomly chosen samples of size c_i will be within the confidence interval asymptotically when the value 1.96 is used for constant $Z_{\alpha/2}$. This provides us with a metric for accepting or rejecting mappings that has a firm theoretical grounding.

We can analytically determine a threshold for any desirable false negative rate from this fact alone. This analytical threshold compensates for issues due to using simultaneous confidence intervals. When the 95% confidence interval is used, data generated by the correct HMM process will fall within any of the confidence intervals 95% of the time. It will therefore be within the confidence intervals of n transitions:

$$(0.95)^n \cdot 100$$

percent of the time.

Unfortunately, decisions cannot be based solely on the false negative rate. We want to make the decision that best separates true positives from false positives. The threshold is therefore sensitive to the set of samples being treated. If the data streams to be rejected are from processes quite different from the HMM model, a very high threshold can be used. If they are generated by processes quite similar to the HMM model, a lower threshold value will have to be used and a higher false negative rate tolerated. This is why ROC analysis is needed to empirically find the proper threshold value for a given detection problem. We note that the use of ROC curves implicitly compensates for issues related to using simultaneous confidence intervals.

5.4 Illustrative Example

To illustrate our approach, we present a simple example using nine artificially generated Markov models with the structure depicted in Figure 5.2. Our approach does not require a bound on the size of the state or transition space. Increasing the number of transitions simply increases the number of confidence intervals that must be calculated. For each model, the probability of remaining in the same state and repeating the last symbol, p, has one of the values: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9. From each of these nine models, we generated forty sequences of 100 symbols. We generated the sequences from the model for illustrative purposes only. The confidence interval approach works equivalently with a set of training data to determine the thresholds.

We used both our proposed algorithm and the Forward-Backward Procedure to determine which model generated the observation sequence. If the correct mapping is accepted, we have a true positive. If an incorrect mapping is accepted, we count that as a false positive. We compare the performance of our approach with the ML portion of the Forward-Backward Procedure (although the ML approach solves a slightly different problem in that it chooses one model from a set of possible mappings). We did not employ the scaling approach discussed in Chapter 2.4.1.

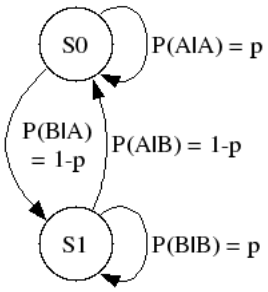


Figure 5.2: Our test Markov model. Note how the self-looping and state change probabilities for both state 0 and state 1 are equivalent, respectively.

For both approaches, we calculated results from windows of the observed sequence. If the window size is too small, the false positive rate will be too high. Conversely, window sizes that are too large react too slowly when the target under observation switches behavior modes. Since these data sets do not include data streams that modify their behavior, larger window sizes will have an advantage in this example. We arbitrarily chose a maximum window size of 30 symbols to maintain a sufficiently large number of samples.

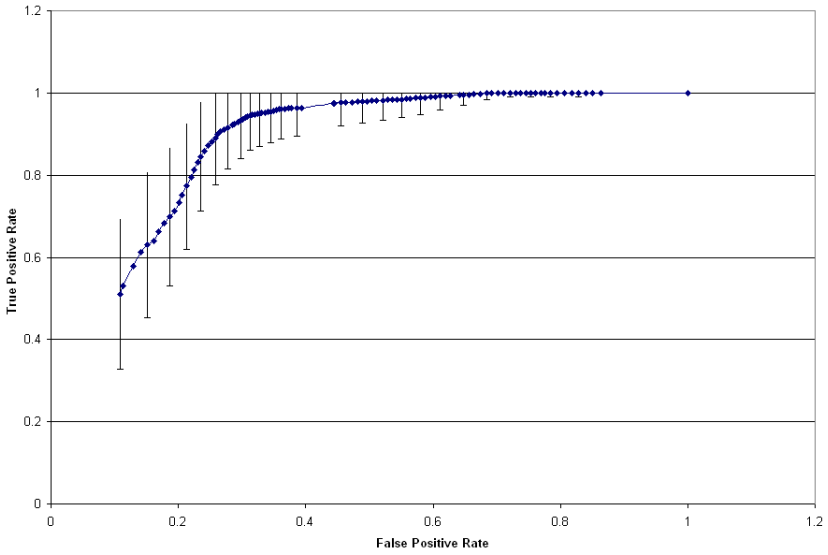
5.4.1 Confidence Interval Bounds

We performed our analysis by using different window sizes on the observed sequence to determine the optimal threshold. We considered each window size independently and calculated true and false positive rates for each window size. For each of the 40 valid sequences, we found the percent of valid transitions for each subsequence. We kept a running average at each window by summing the percent of valid transitions and dividing by the number of windows analyzed. At any given window, we used the running average to determine if the portion of the sequence analyzed thus far could be considered to be a match by being greater than or equal to a desired threshold. To calculate the true positive rate, we counted the number of valid sequences and divided by the total number of possible valid sequences. Note that the number of possible valid sequences varies with the size of the window. We calculated the false positive rate in a similar fashion except we considered invalid sequences and their respective subsequences.

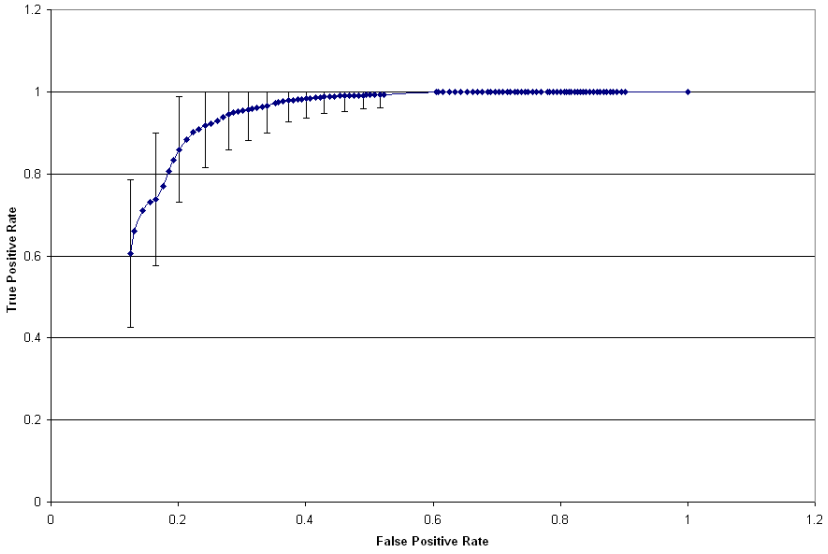
To find the optimal threshold we constructed ROC curves. We varied the window size from 10 to 30 symbols and the threshold from 0% to 100%. A subset of the ROC curves from these tests is provided in Figure 5.3. To find the optimal point on the ROC curves, we calculated the Euclidean distance between each point on the curve and the optimal value of (0, 1) where there are neither false positives nor false negatives. We used the threshold with the minimum Euclidean distance across all window sizes and thresholds for a particular Markov model as the optimal threshold. The window size and threshold value for the minimum Euclidean distance for each Markov model is in Table 5.1.

Table 5.1: Optimal Thresholds of ROC Curves

MM $p, (1-p)$	Min Distance	TP Rate	FP Rate	Window Size	Threshold @ window size
0.1, 0.9	0.0578	0.974	0.051	30	70
0.2, 0.8	0.2061	0.948	0.199	20	86
0.3, 0.7	0.2783	0.872	0.247	30	82
0.4, 0.6	0.2156	0.889	0.185	29	92
0.5, 0.5	0.2425	0.883	0.212	27	91
0.6, 0.4	0.2774	0.841	0.227	29	89
0.7, 0.3	0.2222	0.867	0.178	26	89
0.8, 0.2	0.1861	0.891	0.151	30	84
0.9, 0.1	0.0837	0.951	0.068	30	71

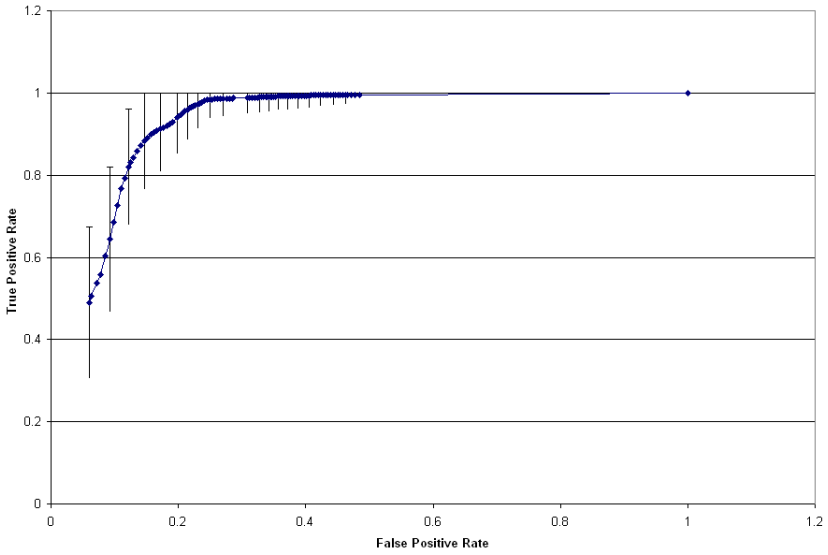


(a)

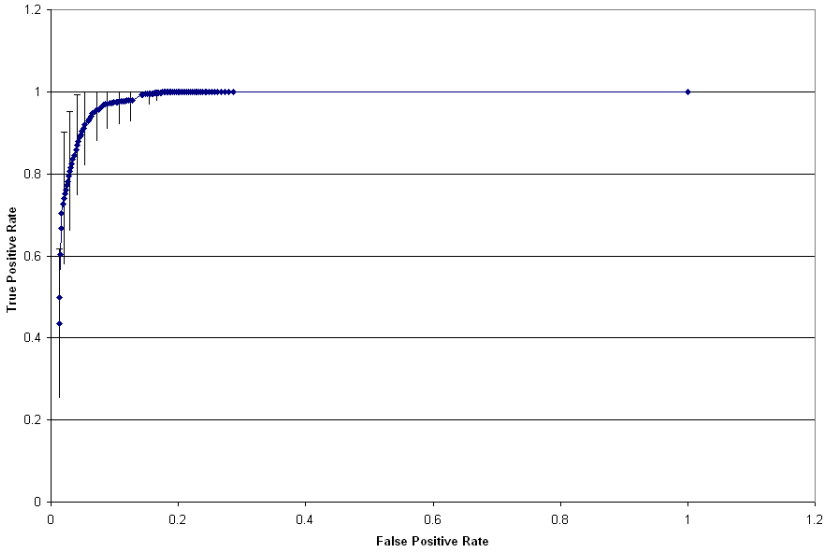


(b)

Figure 5.3: ROC curves used to find the optimal threshold values for the Markov models given in Table 5.1. (a) curve for MM 0.3-0.7; (b) curve for MM 0.5-0.5. All ROC curves are shown with 95% confidence intervals around selected threshold values. Images reproduced with permission from [21]. ©IEEE 2009.



(c)



(d)

Figure 5.3: ROC curves used to find the optimal threshold values for the Markov models given in Table 5.1. (c) curve for MM 0.8-0.2; (d) curve for MM 0.9-0.1. All ROC curves are shown with 95% confidence intervals around selected threshold values. Images reproduced with permission from [21]. ©IEEE 2009.

5.4.2 Maximum Likelihood Results

To provide a baseline result for our confidence interval approach, we analyzed the 360 sequences and determined which Markov model had the highest probability of matching each sequence. We used different window sizes on the output sequence and considered all window sizes from 10 to 30. This maintained consistency in our results. We calculated true and false positive rates for each window size.

To calculate the true positive rate for window size w, we considered only the valid sequences for each Markov model. We counted the number of times the model generating the subsequence had the largest probability of matching when using the Forward-Backward Procedure. Summing across all nine models, we had the total number of true positive results from all valid subsequences of window size w. Dividing by the total possible number of true positives gave us the true positive rate.

The false positive rate was calculated in a similar fashion. We counted the number of times a model, that did not generate the subsequence, had either the largest probability or belonged to the set of Markov models that had the largest probability. (In our tests, the process always chose a single HMM). We again summed across all nine models and divided by the total possible number of false positives for window size w to find the false positive rate at each subsequence length.

Table 5.2 provides the true and false positive rates at the optimal window sizes from Table 5.1 when using the ML approach. The ML approach has a lower false positive rate than the confidence interval approach at all window sizes, which is desirable for applications. Unfortunately, the true positive rate rises above 50% in only three cases. Since a set of Markov models can be returned by the maximum likelihood estimator, it is possible for a true positive and multiple false positives to correspond to the same subsequence (we note that this never occurred in our tests).

5.4.3 Comparisons

Table 5.1 provides the true and false positive rates at the optimal thresholds for each HMM with the confidence interval approach. All HMMs have high true positive detection rates.

The HMMs in Table 5.1 have relatively high false positive rates compared to the ML results. Further analysis showed that the false positive rates increase when the HMMs considered have similar distributions. Note that the false positive rates are higher for HMMs with $0.2 \leq p \leq 0.8$ and higher

Table 5.2: Maximum Likelihood Rates

MM $p, (1-p)$	Window Size	TP Rate	FP Rate
0.1, 0.9	30	0.826	0.043
0.2, 0.8	20	0.415	0.073
0.3, 0.7	30	0.394	0.072
0.4, 0.6	29	0.438	0.071
0.5, 0.5	27	0.437	0.089
0.6, 0.4	29	0.440	0.078
0.7, 0.3	26	0.336	0.054
0.8, 0.2	30	0.508	0.057
0.9, 0.1	30	0.897	0.039

than when $p = 0.1$ or $p = 0.9$. This is because sequences from the first set of HMMs can be confused with two other HMMs with similar parameters. It is more difficult to differentiate between two data sets when they are generated by subtly different processes. If we ignore sequences generated by HMMs when the p parameters differ by less than 0.2, the false positive rate for all nine Markov models drops to at most 5%.

Table 5.2 provides the true and false positive rates of the ML approach at the optimal window sizes of the confidence interval approach. All models maintain low false positive rates but the true positive rate is only greater than 80% for the models when $p = 0.1$ and $p = 0.9$. The ML approach performs better for those two data sets for the same reasons that we found for the confidence interval approach.

This comparison illustrates the difference between the two approaches. In our tests, the confidence interval approach performs better for the identification task. Maximum likelihood may be better for some classification tasks, since it is more likely to provide a single response than the confidence interval approach. We recognize that the confidence interval method will be the better solution for many applications, since it provides clearer criteria for rejecting a detection event.

5.5 Use on Consumer Activity Data

The previous example illustrated our approach. We now apply our approach to data models that we extracted from the Netflix challenge data set [91]. Each state $v_i \in V$ represents a portion of the consumer's rental history classified by movie genres, and each transition $p_{i,j} \in \mathbf{P}$ represents the probability of a rental of a specific genre. Each observation χ_i is a type of rental from the

set of rental categories χ. The goal is to identify classes of behaviors for clients. We generated the models using the CSSR Algorithm outlined in Chapter 3 and a data stream from the Netflix dataset. Alternatively, one could use the Baum-Welch algorithm on the data stream to find the conditional probabilities and create a HMM. Both approaches are equally valid with the confidence interval approach.

The two models for this test are shown in Figure 5.4. We followed a similar procedure as the previous example to calculate the thresholds and the true and false positive rates for both the confidence interval approach and the maximum likelihood approach. From each behavior, we generated 200 sequences of 1000 symbols to determine the threshold necessary for each model. Our results are given in Tables 5.3 and 5.4. The ROC curve for each model is shown in Figure 5.5.

The confidence interval approach had a high true positive rate and a high false positive rate. This is consistent with the results in Section 5.4. The ML approach had a high true positive rate and a higher false positive rate than the confidence interval approach. The Euclidean distance between the ROC curve and point (0, 1) can be used to measure the quality of a detection method. In this example, the confidence interval approach outperforms maximum likelihood (0.289 vs. 0.394 and 0.271 vs. 0.401).

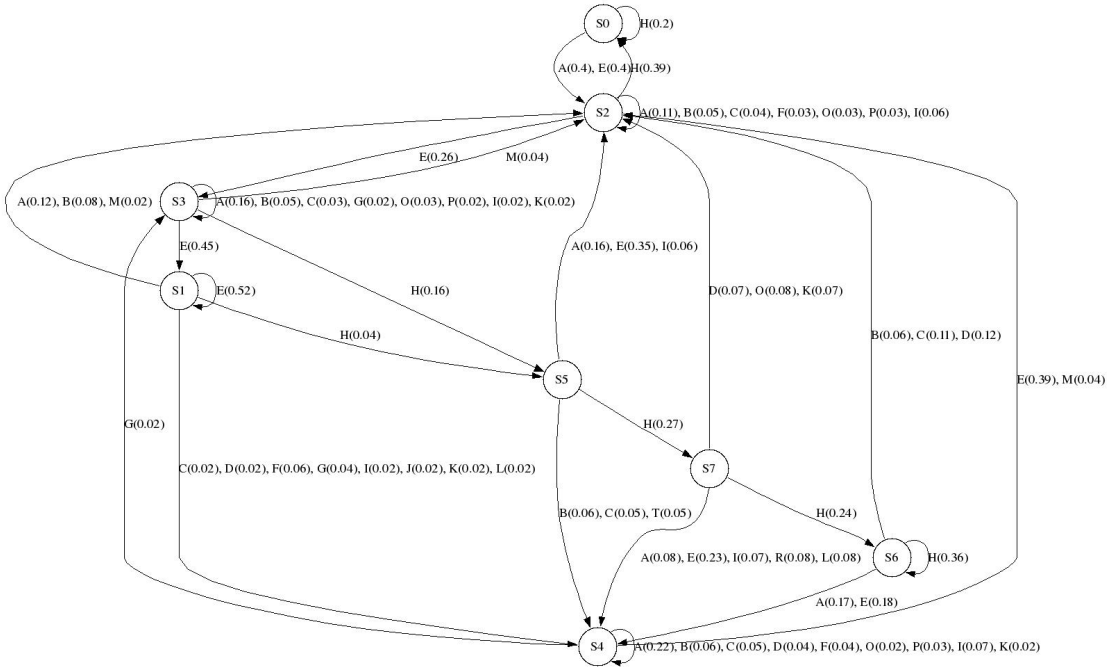
We also note that it is possible to tune the threshold for the confidence interval approach. Decreasing the threshold to 93 for model 1 allows us to achieve a 100%-true positive rate. Increasing the threshold to 95 for model 1 eliminates all false positives. The true positive and false positive rates for these thresholds for model 1 are given in Table 5.5. Tuning the detection threshold makes the confidence interval approach attractive for a wide range of applications.

Table 5.3: Consumer Activity Results

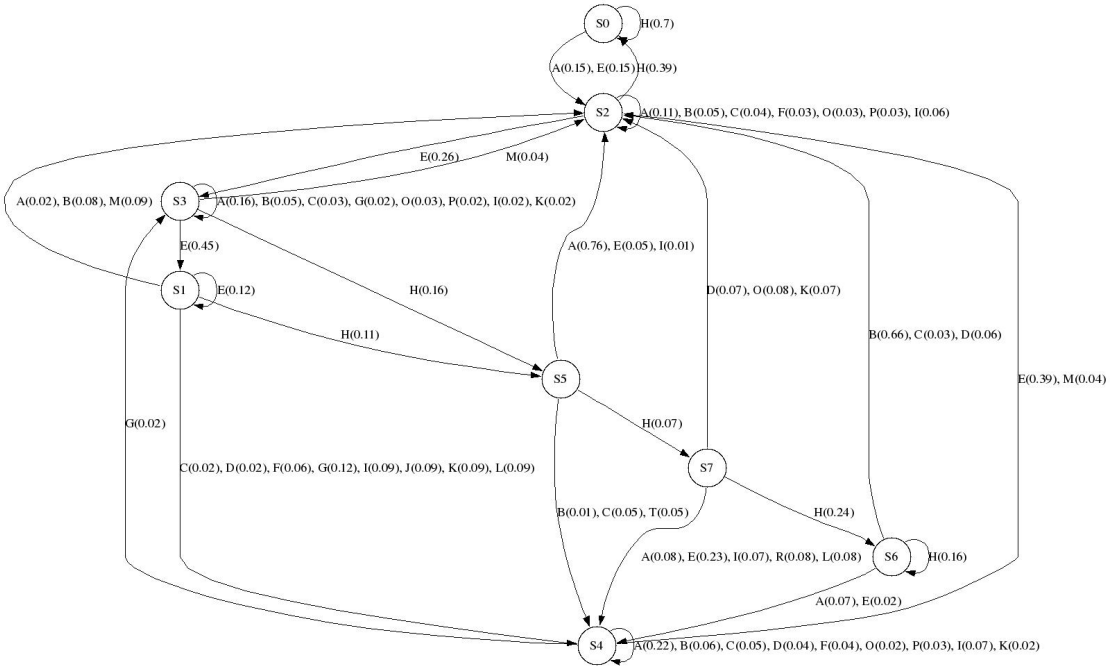
	Threshold	Window Size	TP Rate	FP Rate	Distance
Model 1	94	25	0.930	0.280	0.289
Model 2	96	15	0.855	0.229	0.271

Table 5.4: Consumer Activity Maximum Likelihood Results

	ML TP Rate	ML FP Rate	ML Distance
Model 1	0.973	0.394	0.394
Model 2	0.958	0.399	0.401

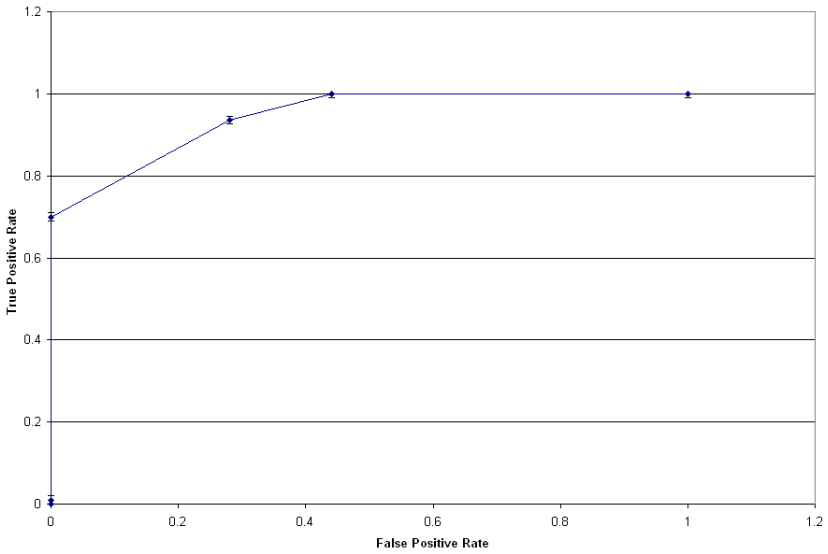


(a)

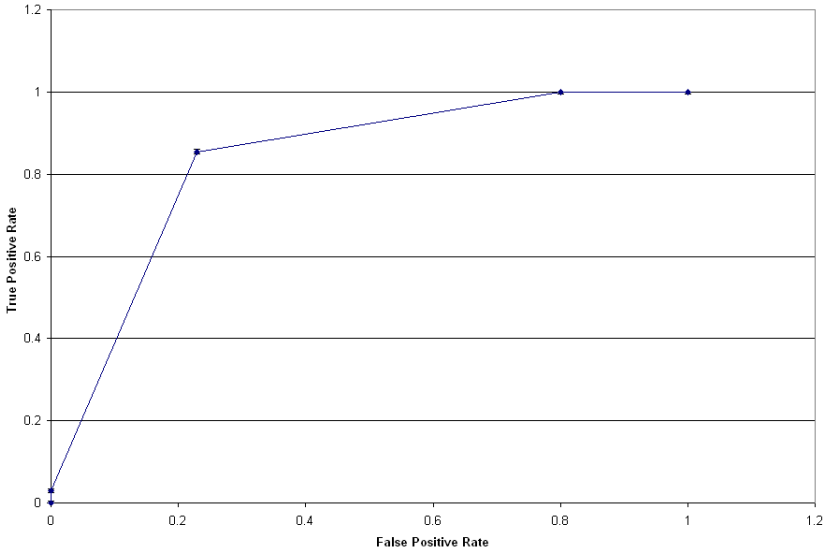


(b)

Figure 5.4: Behavior models extracted from the Netflix data set. (a) Model 1; (b) Model 2. Images reproduced with permission from [21]. ©IEEE 2009.



(a)



(b)

Figure 5.5: ROC curves depicting the thresholds for each model. (a) ROC curve showing the true positive and false positive rate for a window size of 25 for model 1; (b) ROC curve showing the true positive and false positive rate for a window size of 15 for model 2. 95%-confidence intervals are shown in both images. Images reproduced with permission from [21]. ©IEEE 2009.

Table 5.5: Alternate Thresholds

	Threshold	Window Size	TP Rate	FP Rate
Model 1	93	25	1.0	0.44
	95	25	0.70	0.0

5.6 Summary

In this chapter, we showed how to apply confidence intervals to Markov models and solve the pattern detection problem. We noted how pattern detection is slightly different from classification. Our experimental results demonstrate how window size impacts the results by changing the amount of data analyzed, and consequently, changes the size of the confidence intervals. We discuss the importance of and how to calculate the window size in the next chapter.

Chapter 6

Determining the Window Size

6.1 Current Approaches

Windowing is often used in signal processing. In audio signal analysis, speech is classified into three frequency categories: formants, plosives, and fricatives. Each class has a finite length and the proper window size is critical to accurate identification of speech [92]. In audio signal processing, windows that are too small do not capture enough information to positively identify the speech category. Likewise, windows that are too large, capture too much information and are also unable to identify the category. Researchers have created methods to determine window sizes based upon discrete segments of the data stream [93] and also upon analysis of the analog signal stream [94]. Windowing issues faced by behavior recognition systems are slightly different than signal processing. Larger window sizes do not affect the accuracy of the system, but instead affect the processing time.

The conventional method of selecting a window size is to arbitrarily choose a value and empirically determine if the selected window size provides the desired true and false positive rates. If the window size does not provide an acceptable level of recognition, a new value is chosen to improve the results and the process is repeated. A number of works illustrate the initial selection of the window size when performing behavior recognition [95, 96, 97, 98, 99]. Zhou in [34] uses simulations to demonstrate the effectiveness of selecting the proper window size for the application. While this method is acceptable to find a working window size, it is inefficient to search all possible window sizes in order to determine the optimal window size necessary for the application to function

with minimum errors and delay. Allowing the target to change from one behavior to another is not discussed.

Baillie in [35] uses a modified Bayesian Information Criterion algorithm called BICseg to determine the amount of data that finds statistically likely change points in the data. Baillie specifically finds the change points in audio data where the change is from a distinct type of noise to another type of noise. Our work is slightly different in that we do not require a distinct characteristic to differentiate between the two data processes. The BICseg algorithm sets the minimum window size to be one frame (1/30th of a second) and the maximum to be the entire data stream. It iteratively determines the best window size to use on the data stream using a maximum likelihood approach. This approach can be used to determine the window size best able to detect changes in a specific input data stream and determine when noise levels have changed. The algorithm must be rerun for additional data streams.

Other methods to find the window size involve dynamically determining the window size from the input data stream. Sarma in [100] finds the largest window size needed to improve the performance of a detector using the statistical properties of the input data. If the window size passes a two-sided parametric rank based statistical significance test, then the maximum needed window size to properly analyze the data is found. If the test fails, the window size is reduced and the significance test is performed again. The minimum window size is defined to be the smallest number of samples needed for statistical significance of a pre-specified α.

Yu in [101] proposes an algorithm to set window sizes proportional to measurable distances in radar images to reduce noise. Smaller distances in the contours of the image result in smaller window sizes being used to filter the data. Their algorithm is a combination of selecting an arbitrary maximum value to limit the amount of information processed but allowing the window size to be dynamically chosen below the maximum value. The topic of the minimum window size needed was not discussed explicitly. The window size found is dependent on the data stream being analyzed.

The approaches used by [100] and [101] set the minimum and/or maximum bounds on the window size from the data stream with limits chosen a priori. The window sizes determined by these algorithms are data dependent, requiring new calculations for a new window size for new data streams. In behavior recognition, the dependency of the window size on the data stream is not practical due to the variability and inconsistency of the data. Selecting a window size solely on a positive match of the data to a model limits the usefulness of the recognition system. In this chapter,

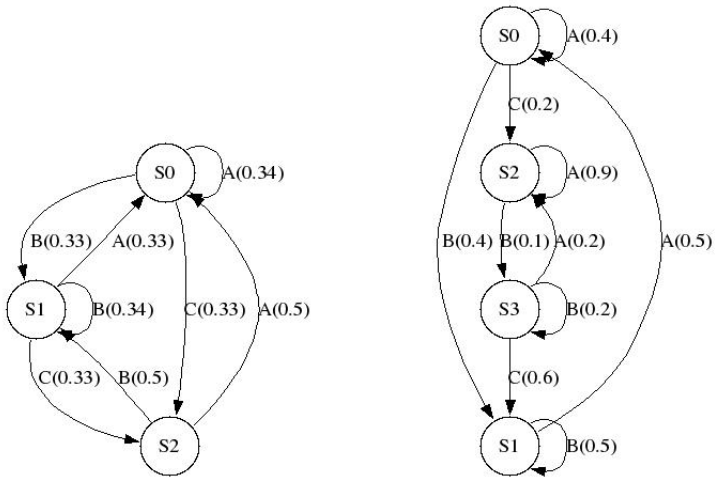


Figure 6.1: Illustrative models. (a) Model 1: 3-states, 8 transitions; (b) Model 2: 4 states, 10 transitions

our techniques calculate window sizes that are independent from the input data. We calculate the window size using only the models.

6.2 The Main Problem and Solutions

Our goal is to determine the window size needed to differentiate between two Markov models G_1 and G_2, with structure $(V_1, E_1, \mathbf{P}_1)$ and $(V_2, E_2, \mathbf{P}_2)$, respectively. We use two approaches to calculate the window size. The first approach calculates the statistical power of each binomial distribution representing a transition in the model. The algorithm iterates through possible window sizes until the desired statistical power is reached. An alternate approach uses the confidence interval method defined in Chapter 5 to find a closed form solution that determines the window size needed to prevent the transition probabilities from approximating one another. Both approaches are designed to consider only two models at a time. We note that window sizes calculated to differentiate between G_1 and G_2 can only be determined if $\mathbf{P}_1 \neq \mathbf{P}_2$.

A special case exists where G_1 and G_2 have the same structure ($V_1 = V_2$ and $E_1 = E_2$). To differentiate between two models with the same structure, we directly compare the matching transitions. In later sections, we provide an example of our approach working on models with the same structure. We do note that finding matching transitions is a graph matching problem, and in

worst case is in **NP** [102]. We omit discussion about graph matching as it is beyond the scope of this dissertation.

In the general case, G_1 and G_2 have a different state and transition structure ($V_1 \neq V_2$ and $E_1 \neq E_2$). If $V_1 \neq V_2$ and $E_1 = E_2$, we simply ignore the transitions that do not have matches and only use transitions that have corresponding matches in E_1 and E_2.

To differentiate between two models, we compare the transitions that have the same conditional probability distributions. Each transition can be described by its associated label and the label associated with the previously taken transition. For example, using the model in Figure 6.1a, we consider the transition $\Pr(A|A)$, meaning that the transition taken previously had label A and the transition to take has a label A. Given a second model of any structure, we find all transitions with the conditional distribution $\Pr(A|A)$ (in Figure 6.1b, there are two), to compare against the transition from the first model. We calculate window sizes using the approaches defined below and use the maximum window size as the window size necessary to differentiate conditional distributions with the selected labels. By selecting the maximum window size to represent the set of equivalent conditional probability distributions, we guarantee that the window size is able to differentiate between all transitions in the set. We note that in considering only the conditional probabilities, we ignore the state structure of the Markov model and assume that higher order effects in the data stream do not affect the calculated window sizes. We can use longer histories of previous transitions (such as $\Pr(A|A, B, A, A, A)$) to incorporate the higher order effects. This has the potential to make the problem intractable, however.

6.2.1 Binomial Distribution Approach

Upon entering a state $v_i \in V$, a state is exited through one of j-outgoing transitions. We generalize for all transitions, including those that exit and enter the same state. Each state can be modeled with a multinomial distribution $M_i \sim (n_i, c_{i,1}, \cdots, c_{i,j}, p_{i,1}, \cdots, p_{i,j})$, where n_i is the number of times state v_i is entered, $c_{i,j}$ is the number of times the edge (v_i, v_j) is taken, and $p_{i,j}$ is the probability of the transition between (v_i, v_j). Each transition can be individually modeled as a binomial distribution $B_{i,j} \sim (n_i, p_{i,j})$. To determine n_i for each state, we first calculate the asymptotic state probability vector $\mathbf{s}$ for G, where $s_i \in \mathbf{s}$ is the asymptotic state probability for state

v_i. For a given window size N, we can calculate the expected number of times we are in state v_i by

$$n_i = \lfloor s_i \cdot N \rfloor \tag{6.1}$$

Note that this infers that the asymptotic state probability vector $\mathbf{s}$ must have stabilized for n_i to be valid.

For binomial distributions $B_{1:i,j}$ and $B_{2:i,j}$ with respective means μ_1 and μ_2 and variances σ_1^2 and σ_2^2, we make the following claims.

Lemma 6.

$$\lim_{N \to 0} |\mu_1 - \mu_2| = 0$$

Proof. This proof is a consequence of Equation (6.1). Since $\mu_1 = n_{1:i}p_{1:i,j} = Ns_{1:i}p_{1:i,j}$ and $\mu_2 = n_i p_{2:i,j} = Ns_{2:i}p_{2:i,j}$, then $|Ns_{1:i}p_{1:i,j} - Ns_{2:i}p_{2:i,j}| = N|s_{1:i}p_{1:i,j} - s_{2:i}p_{2:i,j}|$, $N \in \mathbb{N}_0$. A substitution of $N = 0$ completes the proof. □

Lemma 7.

$$\lim_{N \to 0} |\sigma_1^2 - \sigma_2^2| = 0$$

Proof. This proof is a consequence of Equation (6.1). Since $\sigma_1^2 = n_{1:i}p_{1:i,j}(1-p_{1:i,j}) = Ns_{1:i}p_{1:i,j}(1-p_{1:i,j})$ and $\sigma_2^2 = n_i p_{2:i,j}(1-p_{2:i,j}) = Ns_{2:i}p_{2:i,j}(1-p_{2:i,j})$, then $|Ns_{1:i}p_{1:i,j}(1-p_{1:i,j}) - Ns_{2:i}p_{2:i,j}(1-p_{2:i,j})| = N|s_{1:i}p_{1:i,j}(1-p_{1:i,j}) - s_{2:i}p_{2:i,j}(1-p_{2:i,j})|$, $N \in \mathbb{N}_0$. A substitution of $N = 0$ completes the proof. □

Corollary 1. *The function $f(N) = |\mu_1 - \mu_2| = |Ns_{1:i}p_{1:i,j} - Ns_{2:i}p_{2:i,j}|$ is an increasing function with growth $|s_{1:i}p_{1:i,j} - s_{2:i}p_{2:i,j}|$ for each unit increase in N.*

Proof. From Lemma 6, this can be proven using mathematical induction to show that $f(N) < f(N+1)$.

Trivial Case: $f(0) < f(1)$

$$|0 \cdot s_{1:i}p_{1:i,j} - 0 \cdot s_{2:i}p_{2:i,j}| < |1 \cdot s_{1:i}p_{1:i,j} - 1 \cdot s_{2:i}p_{2:i,j}|$$
$$0 < |s_{1:i}p_{1:i,j} - s_{2:i}p_{2:i,j}|$$

Case: $f(N) < f(N+1)$

$$\begin{aligned} f(N) &= |N \cdot s_{1:i}p_{1:i,j} - N \cdot s_{2:i}p_{2:i,j}| = N \cdot |s_{1:i}p_{1:i,j} - s_{2:i}p_{2:i,j}| \\ f(N+1) &= |(N+1) \cdot s_{1:i}p_{1:i,j} - (N+1) \cdot s_{2:i}p_{2:i,j}| \\ &= |N \cdot s_{1:i}p_{1:i,j} + s_{1:i}p_{1:i,j} - (N \cdot s_{2:i}p_{2:i,j} + s_{2:i}p_{2:i,j})| \\ &= N \cdot |s_{1:i}p_{1:i,j} - s_{2:i}p_{2:i,j}| + |s_{1:i}p_{1:i,j} - s_{2:i}p_{2:i,j}| \end{aligned}$$

$$N \cdot |s_{1:i}p_{1:i,j} - s_{2:i}p_{2:i,j}| < N \cdot |s_{1:i}p_{1:i,j} - s_{2:i}p_{2:i,j}| + |s_{1:i}p_{1:i,j} - s_{2:i}p_{2:i,j}|$$
$$0 < |s_{1:i}p_{1:i,j} - s_{2:i}p_{2:i,j}|$$

By definition, $p_{1:i,j}$ and $p_{2:i,j}$ must be greater than zero in order for $B_{1:i,j}$ and $B_{2:i,j}$ to exist. Likewise, members of $\mathbf{s}$ cannot be zero in order for n_i to exist. Therefore, the proof is as given.

□

To compare the binomial distributions of matching transitions, we find the value of k where the probability distributions overlap. Figure 6.2 shows an example of distribution overlap using normal distributions to approximate the binomial distributions. The amount of overlap $\beta_{i,j}$ for the transition with probability $p_{1:i,j}$ indicates the probability of an error because it is a measure of the amount of similarity in the probability distribution functions of the binomial distributions. We postulate the proper window size is one where the overlap is less than a chosen significance value α. The selection of α affects the type II error rate. We can make the following determinations about the overlap values $\beta_{i,j}$.

Lemma 8.

$$\lim_{N \to 0} \beta_{i,j} = 1$$

Proof. $\beta_{i,j}$ is the probability of making an incorrect decision and, as overlap, is a measurement of the similarity between $B_{1:i,j}$ and $B_{2:i,j}$. By Lemmas 6 and 7, the difference between the distributions approaches zero with the window size. The probability of an error being made while differentiating between the two distributions (uncertainty) will approach the maximum as the distributions become increasingly similar. □

Theorem 7.

$$\lim_{N\to\infty} \beta_{i,j} \to 0$$

Proof. The proof for this theorem is provided in Section 6.5. □

Theorem 8.

$$\lim_{N\to\infty} \frac{\partial \beta_{i,j}}{\partial N} \to 0$$

Proof. The proof for this theorem is provided in Section 6.5. □

With the computed overlap values $\beta_{i,j}$, we may establish the bounds on an algorithm using the overlap values to calculate the window sizes needed to differentiate between two Markov models and find the level of certainty with our result.

Theorem 9. *For a state space of size m and a maximum window size of N, the algorithm has a complexity of*

$$O(m^2 N^2 \log N)$$

Proof. For one transition, the algorithm must compute $s_i N$ overlap values using the binomial distribution, which contains a factorial. The worst case complexity of the factorial is $O(n^2 \log n)$ [103]. If this occurs for all m^2 possible transitions, complexity is as given. □

Theorem 10. *Given all $\beta_{i,j}$ values for a model calculated for a given N, the level of certainty for the result is*

$$\beta_{cert} = 1 - \prod_{i=1}^{m} (1 - (s_i \beta_{i,j})) \quad \forall j$$

Proof. By Lemma 8, the $\beta_{i,j}$ values are the probability of making an incorrect decision for a given transition. The equation is produced by weighting each $\beta_{i,j}$ value by the probability of being in the state v_i to take the transition from state v_i to state v_j and using the inclusion-exclusion principle [40] to find the total probability of making an incorrect decision. This equation is a computationally simpler form of the inclusion-exclusion principle. □

As the tested window size increases and/or the size of the model increases, the amount of computations needed to determine the binomial overlap increases correspondingly. Due to this complexity, heuristics are necessary to enable the algorithm to find an appropriate window size and not perform an exhaustive search. We select key window sizes according to the following heuristics: the first window size when

- At least one transition in the first model has less than $(100 \cdot \alpha)\%$ overlap with its corresponding transition in the second model
- A majority of transitions in the first model have less than $(100 \cdot \alpha)\%$ overlap with their corresponding transitions in the second model
- All of the transitions in the first model have less than $(100 \cdot \alpha)\%$ overlap with their corresponding transitions in the second model

The third condition is the worst case bound discussed above. We demonstrate how this bound becomes impractical for larger Markov models in Section 6.4. We note that in all three of these heuristics, the algorithm can be adjusted to include both models, instead of one model (e.g. all of the transitions from both models are less than $(100 \cdot \alpha)\%$ overlap). Without this enhancement, only the dark grey area in Figure 6.2 is required to be less than α. If the enhancement is included, both the light and dark grey areas must each be less than α for the window size to meet the criteria. This approach is equivalent to two one-sided statistical tests. In this case, the selection of α also affects the type I error rate. We did not use this enhancement in our work.

Recall in Chapter 2.1.9 that the normal distribution may be used to approximate the binomial distribution, with a proper continuity correction. We used the normal approximation in place of the binomial distributions when n_i and $p_{i,j}$ were such that the rule specified in Chapter 2.1.9 held and used the binomial distribution when not.

Algorithm 6.2.1 – Binomial Algorithm

Input: Observed sequence χ; Markov model G_1 with state space V_1 and transition matrix $\mathbf{P}_1$; Markov model G_2 with state space V_2 and transition matrix $\mathbf{P}_2$; Significance level α

Initialization:

1. Set $N_{one} = -1$ // At least one transition has less than α overlap
2. Set $N_{maj} = -1$ // Majority of transitions have less than α overlap
3. Set $N_{all} = -1$ // All transitions have less than α overlap

Execution:

1. Set $N = 2$
2. Set $\beta_{err} = 1$
3. For model G_1:
   (a) Calculate the steady state probabilities $\mathbf{s_1}$
   (b) Calculate $n_{1:i}$ for each state $v_{1:i} \in V_1$ using Equation (6.1)
   (c) Create a binomial distribution $B_{1:i,j}$ with mean $\mu_1 = n_{1:i} \cdot p_{1:i,j}$ and variance $n_{1:i} \cdot p_{1:i,j}(1 - p_{1:i,j})$ for each transition $p_{1:i,j} \in \mathbf{P}_1$
4. For model G_2:
   (a) Calculate the steady state probabilities $\mathbf{s_2}$
   (b) Calculate $n_{1:i}$ for each state $v_{2:i} \in V_2$ using Equation (6.1)
   (c) Create a binomial distribution $B_{2:i,j}$ with mean $\mu_2 = n_{2:i} \cdot p_{2:i,j}$ and variance $n_{2:i} \cdot p_{2:i,j}(1 - p_{2:i,j})$ for each transition $p_{2:i,j} \in \mathbf{P}_2$
5. For each pair of matching transitions $p_{1:i,j}$ and $p_{2:i,j}$:
   (a) Find the k-value where $B_{1:i,j}$ and $B_{2:i,j}$ intersect
   (b) If $\mu_1 > \mu_2$ then
      i. Calculate: $\beta_{i,j} = \Pr(B_{1:i,j} < k)$
      ii. Mark transition $p_{1:i,j}$ if $\beta_{i,j} < \alpha$ holds
   (c) Else if $\mu_1 < \mu_2$ then
      i. Calculate: $\beta_{i,j} = \Pr(B_{1:i,j} > k)$
      ii. Mark transition $p_{1:i,j}$ if $\beta_{i,j} < \alpha$ holds
   (d) Else no intersection between the two distributions
6. If at least one transition is marked
   (a) Set $N_{one} = N$ if $N_{one} == -1$
7. If a majority of transitions are marked
   (a) Set $N_{maj} = N$ if $N_{maj} == -1$
8. If all transitions are marked
   (a) Set $N_{all} = N$ if $N_{all} == -1$
9. If $N_{all} \neq -1$ terminate the program
10. Else increment N

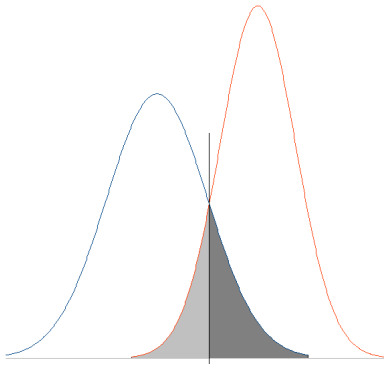


Figure 6.2: Example of overlap using normal distributions. The dark grey area indicates the $\beta_{i,j}$ (overlap) value from the first distribution B_1. The light grey area indicates the $\beta_{i,j}$ (overlap) value from the second distribution B_2. With binomial distributions, the intersection point would be the value of k where $\Pr(B_1 = k) \approx \Pr(B_2 = k)$.

6.2.2 Confidence Interval Approach

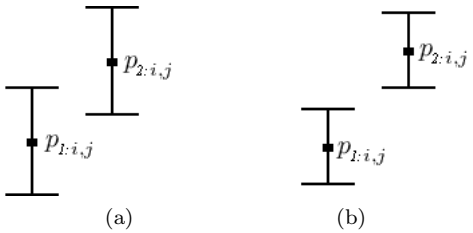


Figure 6.3: Illustration of confidence interval intersection. (a) Confidence intervals intersect and confusion between the intervals is possible; (b) Confidence intervals do not intersect and there can be no confusion between the intervals.

With the previous approach, we illustrated how to decrease the overlap between the transition binomial probability distributions to be less than a pre-determined value of α. We now consider the confidence interval that can be calculated for the transition probability $p_{i,j}$ in the binomial distribution $B_{i,j} \sim (n_i, p_{i,j})$ [21]. Our goal in this approach is to determine window sizes that eliminate the intersection between the confidence intervals. We compare the transition probabilities and their confidence intervals from matching transitions. In Figure 6.3a, we show what we mean by confidence interval intersection. An intersection in coverage of the real number line exists between the upper half of the confidence interval for transition with probability $p_{1:i,j}$ and the lower half of the confidence interval for transition with probability $p_{2:i,j}$. Appropriately selecting the window size decreases the size of the confidence intervals and removes any intersection, shown in Figure 6.3b.

Building upon the work in Chapter 5 and [21], confidence intervals are used to match input data streams to Markov models representing behaviors. The confidence interval is created using the model and the observations, where the range of the confidence interval is given by

$$\left[p_{i,j} - Z_{\alpha/2}\sqrt{\frac{p_{i,j}(1-p_{i,j})}{n_i}}, p_{i,j} + Z_{\alpha/2}\sqrt{\frac{p_{i,j}(1-p_{i,j})}{n_i}}\right] \tag{6.2}$$

where $Z_{\alpha/2}$ is from either the normal or t-distribution, $p_{i,j}$ is the probability of the transition, and α is the level of confidence. Note that these values are constrained to remain within the range [0, 1] and the interval is the asymptotic limit of the binomial, decreasing in size as the n_i increases. In this work, we use $\alpha = 0.05$ for 95% confidence intervals.

Theorem 11. *If the confidence intervals for transitions $p_{1:i,j} \in \mathbf{P}_1$ and $p_{2:i,j} \in \mathbf{P}_2$ have statistical significance α_1 and α_2, respectively, then the level of certainty for N calculated with the confidence interval approach is:*

$$\alpha_{cert} = \frac{\alpha_1}{2} + \frac{\alpha_2}{2} - \frac{\alpha_1\alpha_2}{4}$$

Proof. From Equation (6.2), the confidence interval is calculated with a two-sided test. The intervals may only overlap on one side (see Figure 6.3a for a visual depiction). The level of certainty is the probability that a tested value falls outside the confidence interval. We use the inclusion-exclusion principle to calculate this probability and the fact that the confidence intervals are calculated independently to find the value of α_{cert}. □

From Equation (6.2), we know that the size of the confidence interval for a transition with probability $p_{1:i,j} \in \mathbf{P}_1$ is:

$$CI = Z_{\alpha/2}\sqrt{\frac{p_{i,j}(1-p_{i,j})}{n_i}} \tag{6.3}$$

The range of the confidence interval is then $p_{1i,j} \pm CI$. The confidence interval represents the range about a particular probability within which another probability $p_{2:i,j} \in \mathbf{P}_2$ must fall to be considered an acceptable approximation for the given significance α. Therefore, the absolute difference between the two probabilities must be:

$$|p_{1:i,j} - p_{2:i,j}| \leq CI \tag{6.4}$$

If we want to guarantee enough input data is present, then we rewrite Equation (6.4)

$$|p_{1:i,j} - p_{2:i,j}| > CI \tag{6.5}$$

Combining Equations (6.3) and (6.5) provides us with

$$|p_{1:i,j} - p_{2:i,j}| > Z_{\alpha/2}\sqrt{p_{1:i,j}(1 - p_{1:i,j})/n_i} \tag{6.6}$$

Rewriting Equation (6.6) to calculate the amount of data samples present in calculating the confidence interval results in

$$n_i > (Z_{\alpha/2})^2 \cdot \frac{p_{1:i,j}(1 - p_{1:i,j})}{|p_{1:i,j} - p_{2:i,j}|^2} \tag{6.7}$$

The minimum amount of data needed for a particular confidence interval to be constructed with enough information to be able to differentiate between a match and a failure to match is inversely proportional to the square of the absolute difference of the transition probabilities. Note that $p_{1:i,j} \neq p_{2:i,j}$ is required or it is not possible to differentiate between $p_{1:i,j}$ and $p_{2:i,j}$ using confidence interval intersection.

Calculating all values of n_i, we then use the asymptotic state probability vector $\mathbf{s}$ to determine the window size N_i needed in order for Equation (6.7) to hold for the transition.

$$N_i = \left\lceil \frac{n_i}{s_i} \right\rceil \quad \forall i \tag{6.8}$$

The minimum (maximum) window size needed in order for one transition (all transitions) to not fall within the confidence interval is either the minimum (maximum) N_i, respectively. An extension for this approach is to also consider when a majority of the transitions do not intersect with matching transitions from a second model. We did not use this enhancement in this work.

6.2.3 Establishing Bounds on the Window Size

There are two cases where the approaches described previously break down, and it is not possible to find the window size that can adequately differentiate between the transition probabilities.

Theorem 12. *If $\mu_1 = \mu_2$, neither approach will be able to differentiate between matching transitions $(v_{1:i}, v_{1:j}) \in E_1$ and $(v_{2:i}, v_{2:j}) \in E_2$.*

Proof. By definition, $\mu_1 = Ns_{1:i}p_{1:i,j}$; the equation for μ_2 is similar. If using the binomial approach, Lemma 6 results in zero and there is no statistical difference between the means of the binomial distributions. The overlap $\beta_{i,j}$ cannot be calculated. If using confidence intervals, Equation (6.7) is undefined and the minimum number of samples to differentiate between transition probabilities cannot be calculated. □

Remark 4. For the means to be equivalent, either $s_{1:i} = s_{2:i}$ and $p_{1:i,j} = p_{2:i,j}$ or $s_{1:i}p_{1:i,j} = s_{2:i}p_{2:i,j}$. If the asymptotic state probabilities and transition probabilities are equivalent, state v_i is equivalent in both G_1 and G_2 and it is not possible to differentiate at this state. For complex multiple state models, it is unlikely for the combination of the asymptotic state probabilities and transition probabilities to be equivalent. Our approaches cannot differentiate between the transitions if this occurs.

Theorem 13. *If $\sigma_1^2 = \sigma_2^2$, neither approach will be able to guarantee matching transitions $(v_{1:i}, v_{1:j})$ and $(v_{2:i}, v_{2:j})$ can be differentiated.*

Proof. By definition, $\sigma_1^2 = Ns_{1:i}p_{1:i,j}(1-p_{1:i,j})$; the equation for σ_2^2 is similar. If using the binomial approach, Lemma 7 results in zero and there is no statistical difference between the variances of the binomial distributions. For the variances to be equivalent, $s_{1:i} = s_{2:i}$ and either $p_{1:i,j} = p_{2:i,j}$ or $p_{1:i,j} = (1-p_{2:i,j})$. The case of $p_{1:i,j} = p_{2:i,j}$ is covered by the proof for Theorem 12. For the latter, $p_{1:i,j}$ is the opposite probability of $p_{2:i,j}$. This will result in the overlap $\beta_{i,j}$ decreasing to some constant value, most likely near zero, and the rate of change decreasing to zero. This is confirmed in the proofs of Theorems 7 and 8. As N increases, a point exists where the overlap will not decrease. If the transitions cannot be discriminated before this point, it is not possible to differentiate between the transitions by further increasing N. If $p_{1:i,j}$ is the opposite probability of $p_{2:i,j}$, the confidence interval approach is not affected. □

Remark 5. We note that these limitations affect the ability for the system to differentiate between specific transitions in the models. While unlikely, it is possible for all transitions to fall within the above two limitations. The system would be completely unable to discriminate between G_1 and G_2. In the general case, some transitions will be ignored because they suffer from one of the above limitations. Transitions where $\mu_1 \neq \mu_2$ and $\sigma_1^2 \neq \sigma_2^2$ would then be used to calculate the window size needed for differentiation.

Assuming that transitions of G_1 and G_2 exist that can be used to differentiate the models, we desire bounds on the window size. We first show how the window sizes found using Algorithm 6.2.1 and Equation (6.8) can be used to differentiate at the transition level and then generalize the result to differentiate between G_1 and G_2.

Corollary 2. *(Necessary Condition) If $N_{i,j}$ is the smallest window size able to differentiate between $B_{1:i,j}$ and $B_{2:i,j}$ with a given statistical significance α, then any window size $N^*_{i,j} < N_{i,j}$ will not be able to differentiate between $B_{1:i,j}$ and $B_{2:i,j}$ with statistical significance α.*

Proof. We consider a window size $N_{i,j}$ able to differentiate between transitions when $\beta_{i,j} < \alpha$. For window sizes ranging from $\{0, ..., \infty\}$, $N_{i,j}$ is the first window size for this condition to hold. A consequence of Theorems 7 and 8 is that $\beta^*_{i,j}$ will be greater than α for $N^*_{i,j} < N_{i,j}$. □

Corollary 3. *(Sufficient Condition) If $N_{i,j}$ is the smallest window size able to differentiate between $B_{1:i,j}$ and $B_{2:i,j}$ with a given statistical significance α, then any window size $N^*_{i,j} > N_{i,j}$ will be able to differentiate between $B_{1:i,j}$ and $B_{2:i,j}$ with statistical significance α.*

Proof. The proof is similar to the proof of Corollary 2 and is omitted. □

Theorem 14. *Given the set $\{N_{i,j}\}$ that expresses calculated window sizes for all matching transitions in $\mathbf{P}_1$ and $\mathbf{P}_2$, then any window size $N^* < \min\{N_{i,j}\}$ will not be able to differentiate between G_1 and G_2 and any window size $N^* > \max\{N_{i,j}\}$ will be able to differentiate between G_1 and G_2 with significance level α.*

Proof. By Corollary 2, the minimum window size of the set can differentiate between one transition pair and window sizes below this cannot differentiate between any transition pairs because there is no statistical difference between any of the transitions in G_1 and G_2. By Corollary 3, the maximum window size of the set can differentiate between every transition pair and window sizes above this amount maintain this property. □

The mathematical derivations do not consider the true and false positive rates or the delays associated with the choice of the window size. Furthermore, there is no mathematical verification whether window sizes occurring between these bounds are able to differentiate between G_1 and G_2 with a desired level of accuracy. Heuristics and iterative algorithms that account for the true and false positive rates and delays are the method we use to determine if the window sizes are appropriate

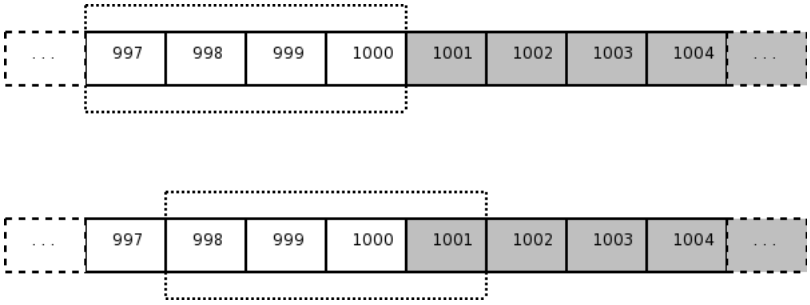


Figure 6.4: Example of windowing a data stream. White blocks represent data from the first model; grey blocks represent data from the second model. (Top) The last "pure" window of data from the first model for a window size of four. (Bottom) The first "mixed" window (the "change point") for a window size of four.

for the application. We illustrate this with experimental results of an illustrative model and models from a consumer behavior dataset.

6.3 Illustrative Example

Using Equation (6.8) and Algorithm 6.2.1, we determine the window sizes necessary to differentiate between two models with the structures shown in Figure 6.1. The calculated window sizes are shown in Table 6.1. To test the window sizes, we generate a sequence of 1000 symbols from each model and concatenate the sequences together (i.e. 1000 symbols from Model 1 immediately followed by 1000 symbols from Model 2. The goal is to use a sliding window to maximize the true positive rate while minimizing the false positive rate and the delay. We calculate the true positive rate by counting the number of windows correctly matching the model that generated the sequence divided by the total number of possible correct matches. We use the confidence interval approach defined in Chapter 5 and [21], to match behaviors with the data stream. We do not include windows that contain symbols from both the first and second model; for example, windows that contain symbols 999 and 1000 from the first sequence and symbols 1 and 2 from the second sequence of symbols. These "mixed" windows are the unstable portion of the system where the results cannot be predicted. We consider the first "mixed" window to be the "change point." Figure 6.4 shows an example of "pure" and "mixed" windows. We calculate the false positive rate in a similar fashion except we count the number of matching windows containing only symbols from the set of 1,000 symbols the model did not generate. In this work, we only look at the true and false positive rates

for Model 1. We generated the sequences from the model for illustrative purposes only. The window size approach works equivalently with a set of training data to determine the proper window sizes.

Each model has an associated threshold value. We calculate the delay by calculating the average acceptance value for each model at each window. When the acceptance value falls below the threshold for the model, the matching between the sequence and the model is rejected. If the acceptance value is greater than the threshold, the matching is accepted. We measure rejection delay by measuring the number of symbols required after the first "mixed" window until the acceptance value from the first model is lower than its threshold. Likewise, the acceptance delay is measured by finding the number of symbols required after the first "mixed" window until the acceptance value from the second model is higher than its threshold. We calculated the thresholds for Models 1 and 2 using the procedure specified in [21]. This process was repeated fifty-one times and the results averaged.

For example, with a window size of four, we use "pure" windows starting with the first through the 997th symbol to calculate the true positive rate for Model 1. When the 998th symbol from the sequence is the first symbol in the window, the window now contains one symbol from the second sequence. When the 1001st symbol is the first symbol in the window, the window contains only symbols from the second sequence and this and all consequent windows are used to calculate the false positive rate for Model 1. For a visual reference of symbols and their positions in this example, see Figure 6.4. We measure the rejection delay of the first model by finding the number of symbols after the 997th symbol until its acceptance value falls below the threshold of 91. The acceptance delay of the second model is the number of symbols after the 997th symbol until its acceptance value is greater than 92 for Model 2. The average true and false positive rates are shown in Figure 6.5a, and delays are shown in Figure 6.5b. The results from the different window sizes are given in Table 6.2.

The true positive rate in both tests does not drop below 0.9 and, as expected, tends toward 1.0 as the window size increases. The false positive rate in both tests decreases at an exponential rate as the window size increases. The initial decrease of the true positive rate is likely due to the asymptotic state probability vector being incorrect for small window sizes. The asymptotic state probability vector stabilizes as N increases. For Models 1 and 2, the vector stabilized at $N = 10$ and $N = 45$, respectively. This decrease is not observed in the more complex models described in Section 6.4. Our hypothesis that larger window sizes incur increasing delay is confirmed when considering

the results from Figure 6.5b. The high delay for small window sizes is due to the inability of the system to discriminate between the models when using a small number of data samples.

Similar to how the threshold of Receiver Operating Characteristic (ROC) curves [89, 90] is calculated, we find the window size that simultaneously has the highest true positive rate, lowest false positive rate, the minimum acceptance delay, and the smallest rejection delay. This corresponds to the minimum Euclidean distance to the point (100,0,0,0). We scale the true and false positive rates by 100 to place them on a similar scale to the delay values. An acceptance delay of zero means that the second model was incorrectly matched with the data from process one. This indicates the window size is too small. A rejection delay of zero indicates that the first model was matched with all of the data from the second process. We chose to stop testing the calculated window sizes after 170 samples because the true and false positive rates were virtually ideal (100% and 0% respectively) and increasing the window size only increased the delay values. The distances are shown in Table 6.2. Window sizes marked with a "*" indicate non-calculated experimental values to contrast performance of calculated window sizes.

After reviewing the data, window sizes with an acceptance delay of zero accepted the second model well before the "change point" and thus do not meet the criteria specified in Chapter 1.4. We disregard all window sizes that have an acceptance delay of zero because of the inability of the system to distinguish between the first and second model when only data from the first model is present in the window. For completeness, window sizes with a rejection delay of zero are unable to eliminate the first model well after the "change point" and are disregarded as well. Tested window sizes under 20 suffer from a high false positive rate, but this is overshadowed in the distance calculation by the very low delay in both acceptance and rejection. The distance metric is not a valuable evaluation tool with the illustrative models shown here because the low delays overshadow the high false positive rates for small window sizes. All window sizes had extremely high true and false positive rates, again confirming the hypothesis that small window sizes are unsuitable for differentiating between models.

6.4 Consumer Data Example

We now test the approaches on data collected from the Netflix consumer data set [91]. A portion of the consumer's movie rental history classified by movie genres is represented by each state

Table 6.1: Illustrative model window sizes: different model structure

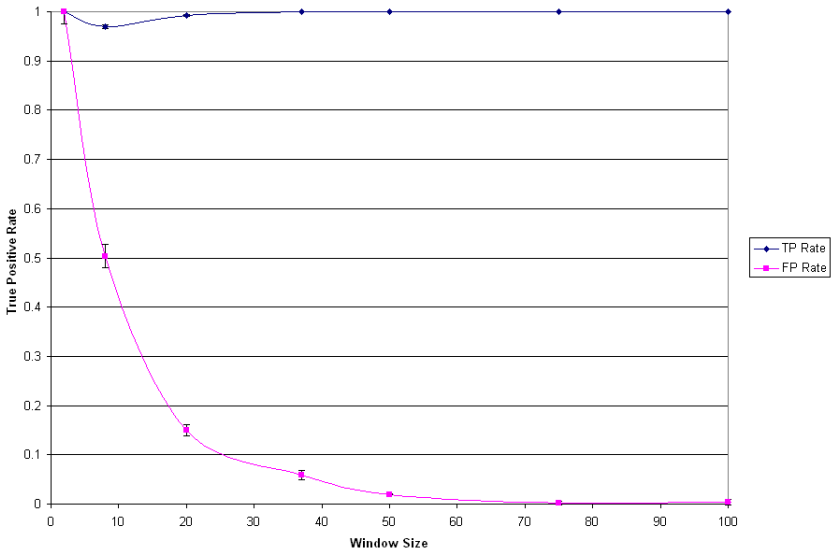
Model 1 vs. Model 2		Window size
$\alpha = 0.05$	At least one transition	2
	Majority transitions	37
	All transitions	78,924
$\alpha = 0.01$	At least one transition	2
	Majority transitions	170
	All transitions	157,512
CI	Minimum	8
	Maximum	979

Table 6.2: Summary of results for illustrative tests: different model structure

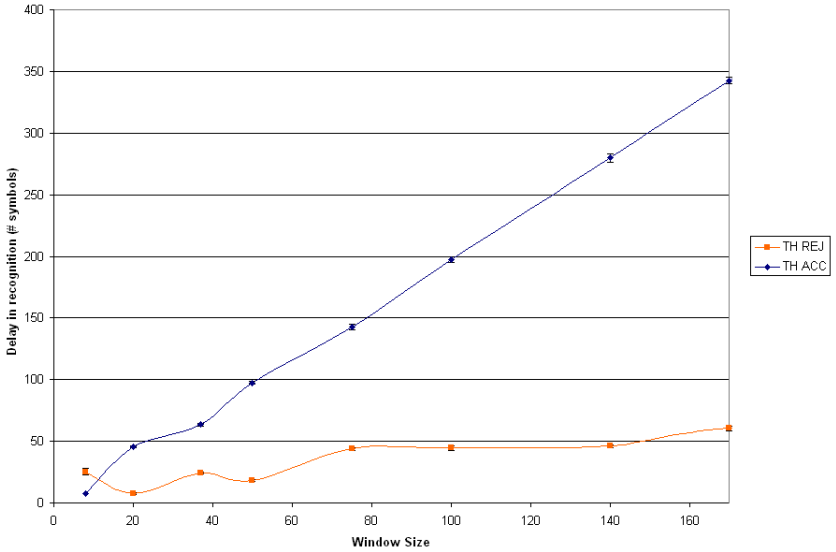
Model 1 vs Model 2					
Window Size	TP Rate	FP Rate	Acc. Delay	Rej. Delay	Distance
2	100	100	0.0	0.0	100
8	97.0	50.4	7.82	25.27	56.99
20*	99.2	14.9	45.64	7.73	48.64
37	99.9	5.8	63.64	24.36	68.39
50*	100	1.9	97.45	18.18	99.15
75*	100	0.14	142.64	43.91	149.25
100*	100	0.40	197.18	44.45	202.13
140*	100	0.58	279.73	46.0	283.49
170	100	0.09	342.64	60.45	347.93

$v_i \in V$. The probability of a rental of a specific genre is represented by each transition $p_{i,j} \in \mathbf{P}$. Each observation χ_i is a type of rental from the set of rental categories χ. The goal is to identify the number of previous rentals to consider when matching rental data to consumers. We generated the models using the Causal State Splitting and Reconstruction (CSSR) Algorithm [26, 32] presented in Chapter 3 and data streams from the Netflix dataset. Other methods to derive Markov models, such as the Baum-Welch algorithm [52] to find Hidden Markov models [1, 51] may be used instead with no change to our approaches in this paper. The models tested in this section have the same structure with 8 states and 70 transitions.

The two models for this experiment are shown in Figure 6.6. We followed a similar procedure to the previous experiment by first calculating the window sizes according to the two approaches. The window sizes are shown in Table 6.3. We chose to terminate Algorithm 6.2.1 after 500,000 iterations. It is unlikely that a consumer would have a history of rentals greater than 500,000 and the window size would not make sense. This also shows the impractical nature of using the sufficient window size for general applications.



(a)



(b)

Figure 6.5: 95% confidence intervals are shown in both images. (a) True and false positive results at selected window sizes for Model 1 vs Model 2; (b) Delay at selected window sizes for accepting and rejecting Models 1 and 2 as matches to the data stream.

To test the window sizes, we followed a similar procedure to the one used in Section 6.3. Thresholds of 94 and 96 were used for Models A and B, respectively. The true and false positive rates and the delays for the small window sizes are given in Table 6.4. We chose to discontinue testing above a window size of 192 because a window size of 192 has a 100% true positive rate

Table 6.3: Consumer data model window sizes

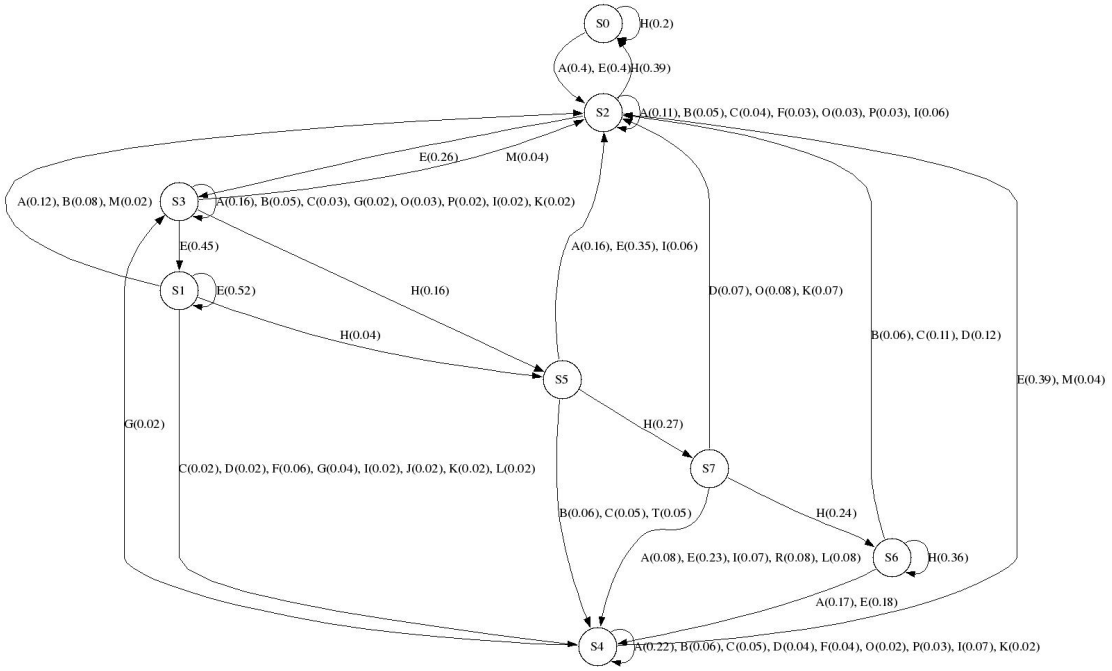
Model A vs. Model B		Window size
$\alpha = 0.05$	At least one transition	7
	Majority transitions	119
	All transitions	$> 500{,}000$
$\alpha = 0.01$	At least one transition	13
	Majority transitions	192
	All transitions	$> 500{,}000$
CI	Minimum	15
	Maximum	17,369

Table 6.4: Summary of results for consumer data tests

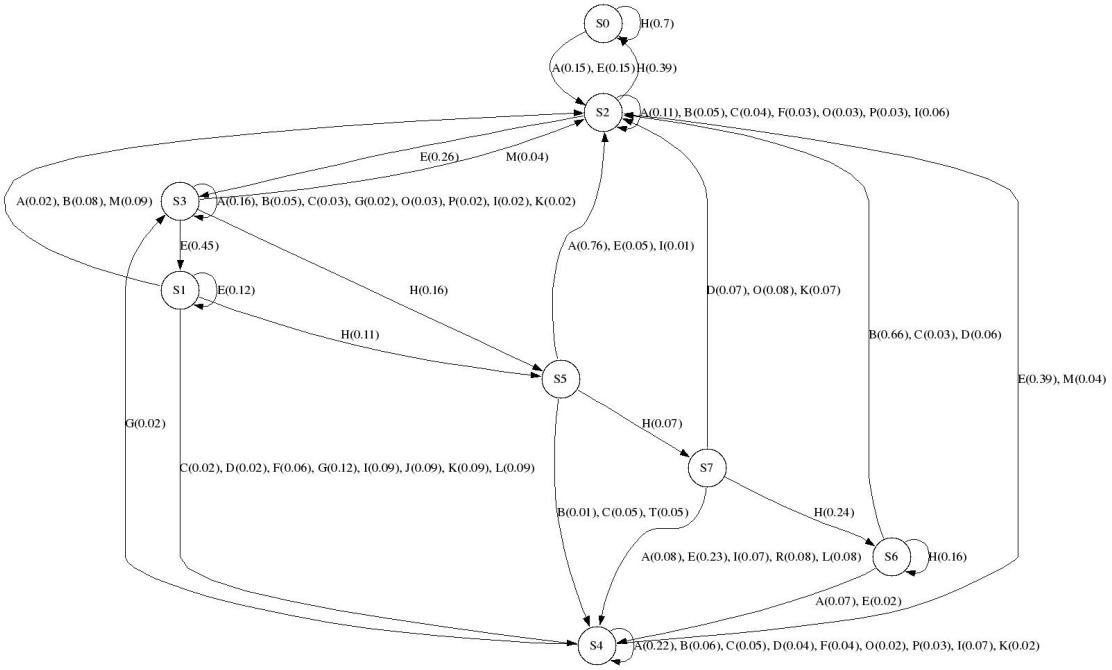
Model A vs Model B					
Window Size	TP Rate	FP Rate	Acc. Delay	Rej. Delay	Distance
7	92.6	61.8	0.45	403.0	407.77
13	93.0	37.4	12.64	92.91	101.19
15	94.1	28.3	20.0	54.73	65.06
50*	98.2	3.3	131.91	26.36	134.57
75*	99.8	1.5	164.18	42.0	169.48
96*	100	0.4	221.64	46.55	226.47
119	100	0.2	237.45	70.45	247.69
192	100	0.0	309.91	111.45	329.34

and 0% false positive rate. Larger window sizes only increase the delay of recognition because the true and false positive rates are at their ideal bounds. A window size of seven causes the average acceptance delay to be near zero indicating that more than seven rentals must be considered when matching the consumer behavior against the models to avoid a crippling false positive rate. The true and false positive rates are shown in Figure 6.7a, and the delays are shown in Figure 6.7b.

The distance metric shows us that for our calculated window sizes, we should consider the consumer's fifteen previous choices when matching the data against these particular behavior models. Window sizes of seven and thirteen had high rejection delays, while window sizes of fifty or greater had significantly high acceptance delays. We note that in both this example and the illustrative example, it is possible to tune the window size. A window size of fifteen provides the maximum true positive rate while simultaneously minimizing the false positive rate and delays. Increasing the window size to fifty significantly decreases the false positive rate and the rejection delay at a cost of increasing the acceptance delay. Tuning the window size makes our approaches attractive for a variety of applications.

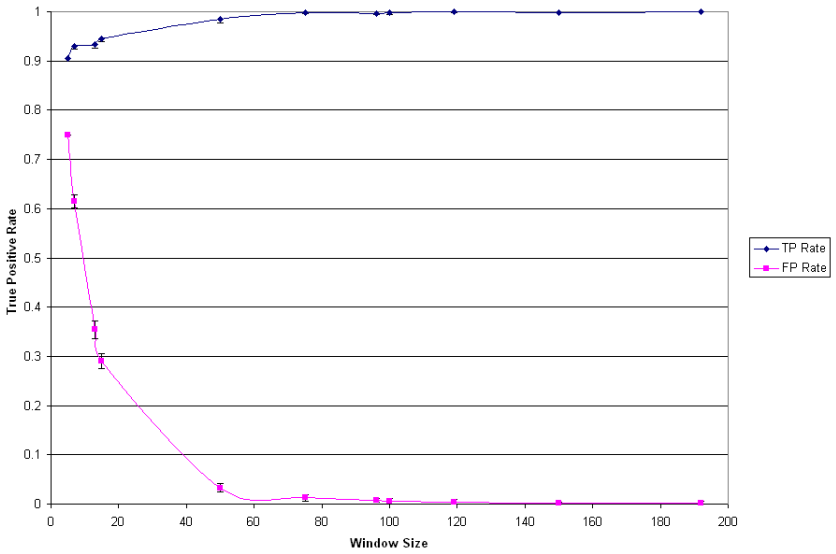


(a)

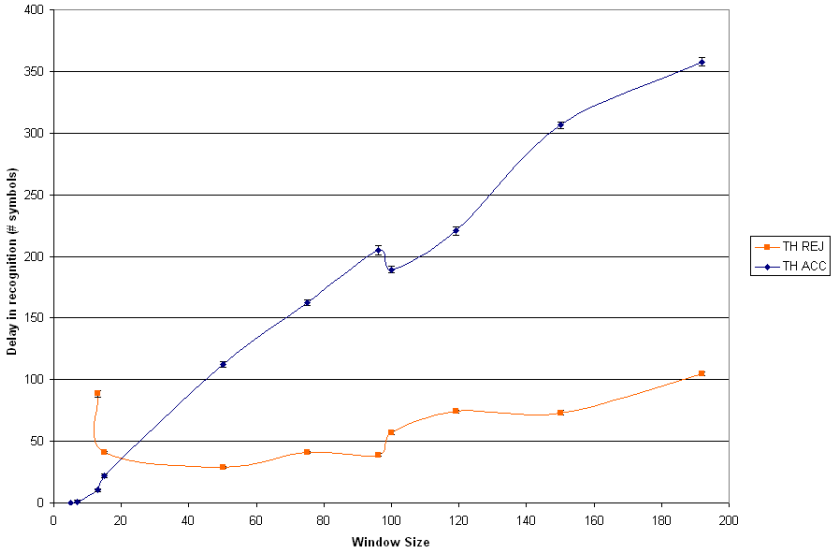


(b)

Figure 6.6: Behavior models extracted from Netflix data. (a) Model A; (b) Model B. Images reproduced with permission from [21]. ©IEEE 2009.



(a)



(b)

Figure 6.7: 95% confidence intervals are shown in both images. (a) True and false positive results at selected window sizes for Model A vs Model B; (b) Delay at selected window sizes for accepting and rejecting models A and B as matches to the data stream.

6.5 Proofs

Define β as a measure of overlap between two binomial distributions corresponding to the occurrence or non-occurrence of a specific transition in a Markov chain. Refer to the probability

of the first transition as p_1 and the probability of the second transition as p_2. Define $n_1 = \lfloor Ns_1 \rfloor$ and $n_2 = \lfloor Ns_2 \rfloor$, where N is an integer window size and $0 \leq s_1, s_2 \leq 1$ are the stationary state probabilities for some state of the prescribed Markov chains.

From this we may define:

$$\pi_1(j, n_1) = \binom{n_1}{j} p_1^j (1 - p_1)^{n_1 - j} \tag{6.9}$$

$$\pi_2(j, n_2) = \binom{n_2}{j} p_2^j (1 - p_2)^{n_2 - j} \tag{6.10}$$

$$\tag{6.11}$$

From these two terms, we may define β as:

$$\beta(n_1, n_2) = \sum_{j=0}^{\max\{n_1, n_2\}} \min\{\pi_1(j, n_1), \pi_2(j, n_2)\} \tag{6.12}$$

This value is the over-lapping area under the discrete curves corresponding to the two binomial distributions.

Our objective is to prove the following theorems:

Theorem 7: As N approaches infinity, $\beta(n_1, n_2)$ approaches 0.

Theorem 8: As N approaches infinity, the discrete derivative of β as a function of N approaches 0.

6.5.1 Proof of Theorem 7

We begin by showing that the theorem holds in the case when $s_1 = s_2 = 1$. Hence, we may replace $\beta(n_1, n_2)$ by the function $\beta(N)$. Then we have:

$$\beta(N) = \sum_{j=0}^{N} \min\{\pi_1(j, N), \pi_2(j, N)\} \tag{6.13}$$

Assuming that $p_1 \neq p_2$ and $p_2 > p_1$, then there is an integer $r(N)$ such that:

$$\beta(N) = \sum_{j=0}^{r(N)} \pi_2(j, N) + \sum_{j=r(N)+1}^{N} \pi_1(j, N) \tag{6.14}$$

To prove Theorem 7 in this special case, we require two key facts: first, simple algebra tells

us that the point at which the two functions $\pi_1(r, N)$ and $\pi_2(r, N)$ occurs at:

$$r(N) = \frac{N\left(-\ln\left(1-p_1\right)+\ln\left(1-p_2\right)\right)}{\ln\left(p_1\right)-\ln\left(1-p_1\right)-\ln\left(p_2\right)+\ln\left(1-p_2\right)} \tag{6.15}$$

Second, we use a relation between the tails of the F-distribution and the tails of the Binomial distribution [104]:

$$F_B(s; p, N) = F_F((s+1)(1-p)/p(N-s); 2(N-s), 2(s+1)) \tag{6.16}$$

where F_B and F_F are the cumulative distribution functions of the Binomial and F-distribution respectively and the degrees of freedom of the F-distribution are $2(n-s)$ and $2(s+1)$.

Without loss of generality, assume that $p_2 > p_1$. Then when $r(N)$ is an integer (or we replace it by its floor), we may evaluate:

$$\sum_{j=0}^{r(N)} \pi_2(j, N) = F_F((r+1)(1-p_2)/p_2(N-s); 2(N-r), 2(r+1)) \tag{6.17}$$

It is clear from the structure of the expressions that as N approaches infinity, $2(N-r)$ and $2(r+1)$ both approach infinity. It is also known that as the degrees of freedom of an F-distribution approach infinity, the cumulative distribution function (CDF) approaches $H(t-1)$ where H is the Heaveside step function. That is, the CDF is zero on the left side of 1 and 1 on the right side of 1.

Now, consider $(r+1)(1-p_2)/p_2(N-s)$ as N approaches infinity. This is the input to the F-distribution in Equation (6.17) and this value approaches:

$$\frac{\ln\left(\frac{1-q}{1-p}\right)(1-q)}{\ln\left(\frac{p}{q}\right)q} \tag{6.18}$$

When $p_1 < p_2$, this quantity is strictly less than 1 but approaches 1 as $p_1 \to p_2$, as is illustrated by its graph in the region $p_1 \in [0, 1]$ and $p_2 > p_1$:

Applying this fact, together with the knowledge that the F-distribution approaches the Heaveside step function, demonstrates that the contribution to $\beta(N)$ by π_2 approaches 0 as N approaches infinity. Since this argument holds in a completely symmetric fashion for π_1, it follows at once that when $s_1 = s_2 = 1$, then $\lim_{N\to\infty} \beta(N) = 0$. We can now prove Theorem 7.

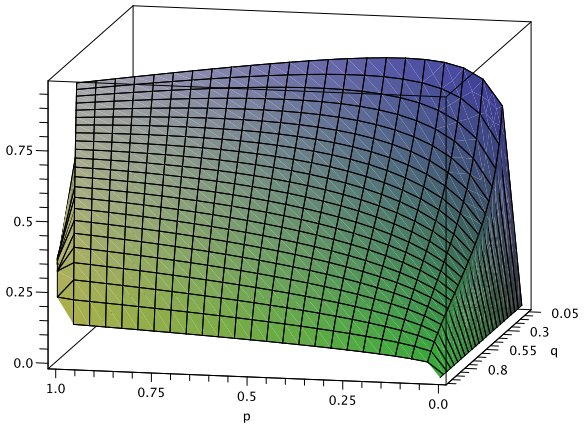


Figure 6.8: A plot of the coordinate input to the F-distribution as N approaches infinity.

Proof of Theorem 7. Suppose now that $s_1, s_2 \neq 1$. It is still the case that n_1 and n_2 both approach infinity as N approaches infinity. Using this fact, we may simply apply the results demonstrated above to argue that

$$\sum_{j=0}^{r(n_1,n_2)} \pi_2(j, n_2) \to 0$$

as N approaches infinity where $r(N)$ has been replaced by an increasing function of n_1 and n_2. By symmetry it follows that

$$\sum_{j=r(n_1,n_2)+1}^{\max n_1,n_2} \pi_1(j, n_1) \to 0$$

whenever $p_1 \neq p_2$ and hence $\beta(n_1, n_2)$ approaches 0 as N approaches infinity as required. □

6.5.2 Proof of Theorem 8

We again proceed by showing that the theorem holds in the case when $s_1 = s_2 = 1$. From Equation (6.14), we may compute $\beta(N+1) - \beta(N)$ as:

$$\sum_{j=0}^{\min\{r(N),r(N+1)\}} \{[\pi_1(j, N+1) - \pi_1(j, N)] + [\pi_2(j, N) - \pi_2(j, N+1)]\} + t(N) \tag{6.19}$$

where $t(N)$ is a term whose value approaches zero as N approaches infinity. That is, $t(N)$ is the finite number of terms missing from the sum because we sum only from $j = 0$ to $\min\{r(N), r(N+1)\}$;

hence $t(N)$ has binomial probability form. Clearly, $t(N)$ approaches zero as N approaches infinity.

Thus, we are interested in evaluating: $\pi_1(j, N+1) - \pi_1(j, N)$. Algebraic manipulation allows us to see that:

$$\pi_1(j, N+1) - \pi_1(j, N) = \binom{N}{j} p_1^j (1-p_1)^{(N-j)} \frac{(N+1)(-p_1)+j}{N+1-j} \tag{6.20}$$

$$\pi_2(j, N) - \pi_2(j, N+1) = \binom{N}{j} p_2^j (1-p_2)^{(N-j)} \frac{(N+1)(p_2)-j}{N+1-j} \tag{6.21}$$

Substituting these results into Equation (6.19) yields:

$$\beta(N+1) - \beta(N) = t(N) + \sum_{j=0}^{r} \left\{ \binom{N}{j} p_1^j (1-p_1)^{(N-j)} \frac{(N+1)(-p_1)+j}{N+1-j} + \binom{N}{j} p_2^j (1-p_2)^{(N-j)} \left\{ \frac{(N+1)(p_2)-j}{N+1-j} \right\} \right. \tag{6.22}$$

Evaluating the sum yields:

$$\begin{aligned} \beta(N+1) - \beta(N) = t(N) &- \frac{(r+1)(-1+p_1)\binom{N}{r+1} {p_1}^{r+1} (1-p_1)^{N-r-1} (-p_1(N+1)+r+1)}{(p_1 N + p_1 - r - 1)(N-r)} \\ &- \frac{(r+1)(-1+p_2)\binom{N}{r+1} {p_2}^{r+1} (1-p_2)^{N-r-1} (p_2(N+1)-r-1)}{(p_2 N + p_2 - r - 1)(N-r)} \end{aligned} \tag{6.23}$$

Taking the limit as N approaches infinity in the previous equation allows us to see that $\beta(N+1) - \beta(N)$ approaches zero as N approaches infinity. Hence, when $s_1 = s_2 = 1$, we have proved Theorem 8. We can now prove Theorem 8 in the general case.

Proof of Theorem 8. As in the proof of Theorem 7, both n_1 and n_2 approach infinity as N approaches infinity. We may apply the same set of arguments in this case to show that $\beta(n_1+k, n_2+k) - \beta(n_1, n_2)$ approaches 0 as N approaches infinity and for some large positive k. □

6.5.3 Interpretation of the Proofs

Theorem 7 shows us that if we choose an arbitrarily large value of N, then the value of β measuring the overlap between our two binomial distributions will collapse to zero. Likewise, Theorem 8 shows that the rate at which β approaches zero is decreasing. Hence we have a measure of Type I error (β) that approaches zero at a rate approaching zero. This suggests a natural point

of diminishing returns for allowing N to become larger. That is, there will be a point at which the amount of time we have to wait for each additional unit of window size N is not justified by the amount of decrease in the value of β we see. This fact justifies our hypothesis that ROC curves should be used to determine appropriate parameters for identifying model change-points. The ROC curves are used to empirically identify the point of diminishing returns. Once this point is identified, we have optimized the trade-off between decreasing the value of β and increasing the amount of time that must pass before a change-point is identified.

6.6 Summary

In this chapter, we showed how to determine bounds on the window size using two different methods. The proper window size affects the true and false positive rate of detection events and influences the delay of the system. Our experimental results demonstrate how the window size can be tuned for specific applications.

Chapter 7

Conclusion

7.1 Concluding Summary

This dissertation presented a complete set of work from model construction through optimal model detection given observation sequences of an underlying process. Our work focused on hidden Markov models (HMMs) and ϵ-machines, a type of Markov model created by J.P. Crutchfield and C.R. Shalizi. We examined four questions and demonstrated how solutions to these questions enabled applications to increase efficiency in analysis or to develop new models.

Our work covered two issues that affect model construction. Markov models are important tools for pattern recognition. In our opinion, the fact that expectation maximization methods require the initial state structure to identify a HMM has been a traditional weakness of this approach. The CSSR Algorithm has alleviated much of this problem. Unfortunately, this approach still depends on the a priori knowledge of the parameter L. We extended the CSSR Algorithm to identify the parameter L and consequently, a Markov model of the dynamics of a process that generated an observation sequence. With this extension, we can now generate a Markov model of a stochastic process from a sequence of symbolic observations with *zero knowledge*. Thus, we have provided a method of going from symbol stream to an optimal Markov model of that stream that requires zero knowledge by the user.

The second issue with model construction dealt with the quality of the models. Traditional model fidelity work assumes that any sequence of observations fully encapsulates the variability of the underlying process. Many researchers have found ways to mitigate the issues caused by insufficient

observation data, and have developed methods to determine if the situation model appropriately matches the data. We have added to this body of work by developing two methods that find level of confidence such that the model and data are representative of the actual underlying process under observation. We have shown how to determine within a given level of statistical confidence if a "known unknown" transition does not occur given two user-defined thresholds. Bounding the probability of an unknown transition to a specific threshold produces better results than statistical confidence but also requires more data. Used in conjunction with the zero knowledge extension, models can be constructed to the user's desired levels of confidence without user involvement.

Once the model is constructed from an observation sequence, we explained how the models can be matched against observation sequences. The maximum likelihood (ML) approach is frequently used to find which HMM most likely generated a data sequence. Calculating the probability that a data stream was generated by a Markov model has the unfortunate property that long data streams are always found to be less likely than short data streams unless scaling factors are included in the algorithm. When scaling factors are used, the algorithm is subject to instability due to accumulating floating point errors. Furthermore, ML gives no intuition in the case when available HMMs do not provide a good fit for the observed data stream.

We presented a new method for finding HMM patterns in data streams by using confidence intervals. We compared our method to the maximum likelihood approach and demonstrated how receiver operating characteristic (ROC) curves can be used to find an optimal detection threshold. The proposed approach is useful in that it considers the number of data samples used in determining whether or not a data stream could match a given HMM. It is also capable of determining when a given data set matches none of the available HMMs. As a result of this research, we were able to provide examples in which the confidence interval approach we derived gave a better true positive rate than the existing maximum likelihood method for assigning observed output streams to HMMs. We also provided an example using our approach on consumer activity datasets. This example shows that our approach can out perform maximum likelihood methods in recognizing behaviors.

From the illustrative examples, the fact that our confidence interval based approach yields a high true positive rate and combined with the fact that other more traditional HMM assignment methods have low false positive rates, suggests that it may be attractive to combine the two methods and benefit from the advantages of both. This requires reconciling conflicts when the two approaches disagree. If we accept detections only when both approaches agree (logical-AND), this would keep

the low true positive rate from the ML approach. Accepting detections from either approach (logical-OR) would keep the high false positive rate of the confidence interval approach. We therefore expect combined methods to perform worse than either of the two individual methods.

It is not surprising that the confidence interval approach performs better with large window sizes, since more data is available for making decisions. We concluded the experimental work of this dissertation with an explanation of how to determine window size values analytically. Representing the transitions in a Markov model as binomial distributions allows us to use statistical analyses to find the probability of making an error for each transition and allows us to calculate an error value for the model. We described how to determine both the necessary and sufficient window sizes to be able to differentiate between two Markov models with a level of statistical significance. Window sizes greater than the minimum necessary and less than the sufficient window size cannot be shown mathematically to differentiate between the models. We demonstrated our heuristics and how the choice of the window size affects the true and false positive rates and the acceptance and rejection delays. We explained how this approach is able to calculate window sizes for Markov models with any structure. This was accomplished by operating on sets of transitions instead of matching to a single transition. We matched transitions with the set having the same conditional probability distribution and found the window size that represented all the transitions in the set. For the special case when the models have the same structure, we match transitions with the same conditional probability connecting matching states. We noted that we only used the single previous transition to perform the match.

7.2 Future Research

Many topics for further research exist. We did not address an issue, mentioned in [18], with model construction. When determining the correct value for L, how large data sets need to be for reliable estimation is unknown. This is a different but similar problem to the one we addressed with model confidence. We considered how the size of the observation sequence can provide a confidence level that the model represented the underlying process. This asks how to calculate a statistical confidence value on whether or not L is correct. This confidence value would be determined in a similar fashion as our approach with model confidence: by considering if the size of the observation sequence is sufficient.

In our demonstrations of the model confidence algorithms, we specifically looked to see if the constructed model matched the model acting as the underlying process. It is possible that other models with different state and transition structures would produce equivalent sequences to the underlying process. Future work could look at using inexact graph matching to determine if the constructed models are in fact equivalent to the initial model. Additional future directions for this work could extend the approach for use on other processes that handle finite state automata. It would be interesting to see if this approach could be adapted to provide a level of confidence to probabilistic context free grammars generated from observations. A possible extension to refine this approach is to create a second user-defined threshold on the asymptotic state probabilities. If the asymptotic state probability is less than a given threshold, then we would have the algorithm ignore the state.

In the confidence interval approach, we stated that we used the Wald confidence interval because of our representation of the transitions as binomial distributions. We noted when discussing the bounds on the window size that the state could be represented as a multinomial distribution because the transitions are not completely independent. The Wald confidence interval and the binomial assumption are valid and produce fairly good results, but we note that incorporating a multinomial distribution and multinomial confidence interval will produce better results. A multinomial confidence interval should be smaller in range and will represent the relationships between the outgoing transitions with greater accuracy.

In our approaches to find bounds on the window size, we stated that if the model structure is different, we use the conditional probabilities and ignore higher order effects. Future work should be performed to determine if our assumption about higher order effects is accurate. While our approach is designed to operate dynamically, to allow for the time necessary to determine the proper window sizes, several improvements could be made to increase the efficiency. We believe that adding additional limits into the algorithm to set N would decrease the time needed to calculate the different window sizes. Additionally, when the model state structure is different, we use the maximum window size from each set of equivalent conditional probability distributions. It is possible that this limit is too strict and an adjustable limit, such as a majority of the set is covered with the window size, would be more efficient. We also show how the window size calculation depends on the asymptotic state probability vector stabilizing. Other options, such as work decreasing the number of samples needed for the vector to stabilize, could allow for smaller window sizes to be

considered. We also demonstrated how our simple Euclidean distance calculation did not produce meaningful results because we equivalently weighted the factors of true positive rate, false positive rate, acceptance delay, and rejection delay. As we stated in the proofs, the trends support the use of a distance metric. We still believe that the distance metric could be modified to provide a meaningful, consistent value to find the proper window size. One possible way to alter the distance metric is to adopt variable weights for the different factors. Future research could explore this and determine if there is an optimal weighting for specific applications.

Our work focused on discrete Markov models, which are a form of a probabilistic finite state machine. Another interesting extension would be to see if similar approaches could be found for more complex processes, such as probabilistic pushdown automata that generate probabilistic context free languages. These models have a memory by incorporating a stack. We believe that our approaches could be modified to work with these types of state machines.

Appendices

Appendix A Related Publications

The dissertation is an amalgamation of multiple individual works that cover distinct but related areas in pattern recognition and ϵ-machines. In developing this work, I attempted to combine the information for continuity and completeness. There are two clear segments to this work, as I outlined in the introduction: model construction and detection. The first two papers discuss model construction and extensions made in this endeavor. The third paper discusses a new method for detecting the occurrence of behaviors and leaves to future work the discussion that we present in the fourth paper. The fifth paper details a method to dynamically determine the alphabet from collected spatial data. This development, while fascinating, is beyond the scope of this dissertation. The final three papers are segments of this work or related work that were presented at various conferences.

- J.M. Schwier, R.R. Brooks, C. Griffin, and S. Bukkapatnam, "Zero knowledge hidden Markov model inference," *Pattern Recognition Letters*, Accepted for publication, 2009.
- J.M. Schwier, R.R. Brooks, and C. Griffin, "On the confidence of constructed hidden Markov models," *Under development for submission*, 2009.
- R. Brooks, J. Schwier, and C. Griffin, "Behavior detection using confidence intervals of hidden Markov models," *IEEE Transactions on Systems, Man, and Cybernetics Part B*, Accepted for publication, 2009.
- J.M. Schwier, R.R. Brooks, and C. Griffin, "Determining the window size to differentiate Markov models," Under review, 2009.
- C. Griffin, R. Brooks, and J. Schwier, "A hybrid statistical technique for modeling non-linear recurrent tracks in a compact set," Under review, 2009.
- C. Griffin, R. Brooks, and J. Schwier, "Determining a purely symbolic transfer function from symbol streams: Theory and algorithms," *American Control Conference*, pp. 4065-4067, 11-13 June 2008.
- R. R. Brooks, J. Schwier, and C. Griffin, "Sensor Stream Analysis for Behavior Recognition," *3rd Annual Innovations and Commercial Applications of Distributed Sensor Networks Symposium*, Shreveport, LA, November 2007.

- R. Brooks, J. Schwier, and C. Griffin, "On-line behavior recognition for situation and threat assessment," *Conf. Proc. AIAA Infotech@Aerospace*, 7-10 May 2007.

Appendix B Curriculum Vitae

EDUCATION

Clemson University Clemson, SC August 2006 – August 2009

Doctor of Philosophy, Computer Engineering

- Advisor: Richard R. Brooks, PhD
- Dissertation topic: Pattern Recognition for Command and Control Data Systems
- Degree conferred: August 2009
- Cumulative GPA: 4.00 / 4.00
- Major courses: Probability, Linear programming, Statistical methods, Computer network security, Game theory, Markov decision processes, Markov chain Monte Carlo

Rensselaer Polytechnic Institute Troy, NY August 2004 – December 2005

Master of Science, Computer and Systems Engineering

- Advisor: Kenneth Vastola, PhD
- Thesis topic: Improvement of the OSPFv2 intra-AS routing module for the SSFNet network simulator
- Degree conferred: December 2005
- Cumulative GPA: 3.76 / 4.00
- Major courses: Network security, Applied cryptography, Internet protocols, Broadband and optical networking, Database management

Penn State University University Park, PA May 2000 – May 2004

Bachelor of Science, Computer Engineering

- Degree focus: Networking and microcontroller devices
- Degree conferred: May 2004
- Cumulative GPA: 3.66 / 4.00

WORK EXPERIENCE

Clemson University Clemson, SC August 2006 – August 2009

Graduate Research Assistant

- Performed research for Office of Naval Research grant on behavior recognition.
- Conducted research on generation of Markov models to represent behaviors taken from input data streams.
- Applied novel extensions to Markov models to solve behavior recognition problems.
- Determined bounds on the parameters necessary to complete behavior recognition.
- Managed computational and data storage servers helping to ensure the computers were operational for research purposes.

Applied Research Lab, Penn State University Park, PA January 2006 – July 2006

Assistant Research Engineer, CCAT Program

- Designed and integrated computer backup strategies for disaster mitigation and recovery by integrating Norton Ghost software onto a private network.
- Responsible for upgrading and ensuring the error-free run-time of simulation programs by debugging and modifying LabVIEW code of integrated systems.
- Updated simulation programs to function in deterministic environments for increased efficiency.
- Converted a pre-existing simulator to function on a real-time computer system.

Rensselaer Polytechnic Institute Troy, NY May 2005 – August 2005

Course Administrator

- Updated Engineering course curriculum for embedded control laboratory, converted exams and lab worksheets from paper to interactive online forms by integrating question randomization to prevent undergraduate students from cheating.
- Created online course components for embedded engineering laboratories, which required reworking and updating the course web site to fit the new course curriculum.

- Developed testing materials and exams for College of Engineering students.
- Supported faculty by developing new computer programs to demonstrate and explain complex topics.

Rensselaer Polytechnic Institute Troy, NY August 2004 – May 2005

Teaching Assistant (TA)

- Performed TA duties for two computer organization and architecture courses for 400 students by grading assignments and exams and assisting sophomore undergraduate students with the course material.
- Coordinated and managed two hands-on computer design laboratory courses for 70 students by grading laboratory reports and instructing sophomore undergraduate students with laboratory procedure.
- Instructor for computer aided design (CAD) course of 34 students, where I instructed and guided freshman undergraduate students on proper engineering practices when using CAD software.

Applied Research Lab, Penn State University Park, PA May 2003 – May 2004

Undergraduate Research Assistant

- Designed and executed ns-2 simulations on DDoS attack patterns.
- Developed and performed main research simulations to identify the multiple characteristics inherent to DoS-class network attacks.
- Maintained and improved DDoS attack detection algorithm by streamlining code to maximize detection and decrease processing time thus resulting in an increase of stability for overall program performance.
- Evaluated alternate simulation tools for application performance and functionality.
- Provided an objective analysis of ns-2 against other simulation tools - information which was included in a presentation to management to identify focus for future simulation research.

PUBLICATIONS

- R. Brooks, **J. Schwier**, and C. Griffin, "Behavior detection using confidence intervals for hidden Markov models," *IEEE Trans. on Systems, Man, and Cybernetics Part B*, Accepted for publication, 2009.
- **J.M. Schwier**, R.R. Brooks, C. Griffin, and S. Bukkapatnam, "Zero knowledge hidden Markov model inference," *Pattern Recognition Letters*, Accepted for publication, 2009.
- R. Brooks, **J. Schwier**, and C. Griffin, "Markovian search games in heterogeneous spaces," *IEEE Trans. on Systems, Man, and Cybernetics Part B*, vol. 39, no. 3, pp. 626-635, 2009.
- C. Griffin, R. Brooks, and **J. Schwier**, "Determining a purely symbolic transfer function from symbol streams: Theory and algorithms," *American Control Conference*, pp. 4065-4067, 11-13 June 2008.
- R. R. Brooks, **J. Schwier**, and C. Griffin, "Sensor Stream Analysis for Behavior Recognition," *3rd Annual Innovations and Commercial Applications of Distributed Sensor Networks Symposium*, Shreveport, LA, November 2007.
- R. Brooks, **J. Schwier**, and C. Griffin, "On-line behavior recognition for situation and threat assessment," *Conf. Proc. AIAA Infotech@Aerospace*, 7-10 May 2007.

PRESENTATIONS

- **On-line behavior recognition for situation and threat assessment**, Doubletree Hotel Sonoma Wine Country, Rohnert Park, CA, May 7-10, 2007. Co-authored and presented for the AIAA Infotech@Aerospace Conference.
- **Empirical DDoS Evaluation**, Applied Research Laboratory, Washington DC, November 24, 2003. Authored and presented for the Mobile Ubiquitous Security Environment MURI, Annual Program Review.
- **Attack Detection**, Applied Research Laboratory, Washington DC, November 24, 2003. Presented for the Mobile Ubiquitous Security Environment MURI, Annual Program Review.

CONFERENCES

- **Best Practices in Building City/University Relations**, Murray State University / City of Murray; June 1-4, 2009; Murray, KY to represent graduate students and discuss how students can participate in improving town-gown relations.
- **Network Encryption User Conference & Training**, General Dynamics; June 2-3, 2004; Las Vegas, NV to enhance knowledge of network security encryption devices.

PROFICIENCIES

- **Operating Systems:** Microsoft Windows Family, Debian Linux, Red Hat Linux
- **Tools:** C, C++, Java, 68HC12 Assembly, TCL, Perl, VHDL, ns-2, SSFNet, LabVEW, LaTeX
- **Hardware Devices:** General Dynamics TACLANE (KG-175), Sectéra (KG-235), FASTLANE (KG-75)

AWARDS

- Clemson University Graduate Dean's Fellowship 2006
- RPI ECSE Dept. "TA of the Year" 2004–2005

AFFILIATIONS

- Alpha Epsilon Lambda, Graduate honors society Member
- Eta Kappa Nu, Electrical and computer engineering honors society Member
- Omicron Delta Kappa, University honors society Member
- Tau Beta Pi, Engineering honors society Member

COMMITTEES AND BOARDS

- iTiger Student Board of Directors November 2008 – August 2009
- Clemson University Foundation Board of Directors July 2008 – July 2009

- Academic Council — July 2008 – June 2009
- Int'l Town and Gown Assoc. Board of Directors — Sept 2008 – June 2009
- Joint City University Board — August 2007 – March 2009
- Graduate Council — August 2007 – March 2009
- Clemson President's Cabinet — April 2008 – March 2009
- EMPower Student Advisory Board — April 2008 – March 2009
- Student Body Administrative Council — April 2008 – March 2009
- Alumni Association Board of Directors — April 2008 – March 2009
- Continuing & Executive Education Task Force — November 2008 – Feb 2009
- Consumer Health Advisory Board, Co-Founder — August 2007 – May 2008
- Graduate Advisory Committee — August 2007 – May 2008
- Martin Luther King, Jr. Award Committee — November 2008

ACTIVITIES

- ECE Department Tour Guide — August 2007 – August 2009
- Clemson Graduate Student Body President — April 2008 – March 2009
- Vice-President, Clemson Graduate Student Body — August 2007 – March 2008
- Senate President, Clemson Graduate Student Body — August 2007 – March 2008
- Technical Judge FIRST Lego League SC State Competition — February 2008
- Teamwork Judge FIRST Lego League Upstate Competition — January 2008
- Senator, Clemson Graduate Student Body — October 2006 – July 2007
- Honorary Judge Philadelphia FIRST Robotics Competition — January 2005
- Honorary Judge Philadelphia BEST Robotics Competition — November 2004

- Mars Desert Research Station Crew 25 Member — March 2004
- Vice-President / Treasurer, Penn State Mars Society — September 2003 – May 2004

Appendix C Permissions for Republication

ELSEVIER LICENSE
TERMS AND CONDITIONS

Jul 19, 2009

This is a License Agreement between Jason M Schwier ("You") and Elsevier ("Elsevier") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Elsevier, and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

Supplier	Elsevier Limited The Boulevard,Langford Lane Kidlington,Oxford,OX5 1GB,UK
Registered Company Number	1982084
Customer name	Jason M Schwier
Customer address	ECE Department Clemson, SC 29634
License Number	2230041334247
License date	Jul 15, 2009
Licensed content publisher	Elsevier
Licensed content publication	Pattern Recognition Letters
Licensed content title	Zero Knowledge Hidden Markov Model Inference
Licensed content author	J.M. Schwier, R.R. Brooks, C. Griffin and S. Bukkapatnam
Licensed content date	27 June 2009
Volume number	n/a
Issue number	n/a
Pages	1
Type of Use	Thesis / Dissertation
Portion	Full article
Format	Both print and electronic
You are an author of the Elsevier article	Yes
Are you translating?	No
Order Reference Number	
Expected publication date	
Elsevier VAT number	GB 494 6272 12

Almost all of Chapter 3 was reprinted from *Pattern Recognition Letters*, J.M. Schwier, R.R. Brooks, C. Griffin, and S. Bukkapatnam, "Zero knowledge hidden Markov model inference," Copyright 2009, with permission from Elsevier.

Comments/Response to Case ID:

ReplyTo:

From: Jacqueline Hansson Date: 06/19/2009

Subject: Re: Request Permission for Republish Material

Send To: "Jason M. Schwier" <jschwie@clemson.edu>

cc:

Dear Jason M. Schwier:

This is in response to your letter below, in which you have requested permission to reprint, in your upcoming thesis/dissertation, the described IEEE copyrighted figures. We are happy to grant this permission.

Our only requirements are that you credit the original source (author, paper, and publication), and that the IEEE copyright line ([Year] IEEE) appears prominently with each reprinted figure.

Sincerely yours,

Jacqueline Hansson

IEEE Intellectual Property Rights Office
445 Hoes Lane
Piscataway, NJ 08855-1331 USA

+1 732 562 3966 (phone)

+1 732 562 1746 (fax)

IEEE-- Fostering technological innovation
and excellence for the benefit of humanity.

Hello,

I would like to request permission to republish figures from an IEEE
paper which has been accepted and is forthcoming to print.

Authors: R.R. Brooks, J.M. Schwier (me), C. Griffin
Title: Behavior Detection Using Confidence Intervals of Hidden Markov Models
Journal: IEEE Transactions on Systems. Man, and Cybernetics, Part B: Cybernetics

Usage: I would like to use Figure 1 (page 3), Figure 3 (page 5), Figure
4 (page 6), and Figure 5 (page 7) and their respective captions in my
PhD dissertation.

If you could please get back to me by June 30, I would greatly
appreciate it.

Thank you,
Jason Schwier

Bibliography

[1] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[2] R. Damper and J. Higgins, "Improving speaker identification in noise by subband processing and decision fusion," *Pattern Recognition Letters*, vol. 24, no. 13, pp. 2167–2173, 2003.

[3] K. Pulasinghe, K. Watanabe, K. Izumi, and K. Kiguchi, "Modular fuzzy-neuro controller driven by spoken language commands," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 34, no. 1, pp. 293–302, 2004.

[4] I. Sanches, "Noise-compensated hidden markov models," *IEEE Trans. on Speech and Audio Processing*, vol. 8, pp. 533–540, September 2000.

[5] J. Chen and A. Kundu, "Rotation and gray scale transform invariant texture identification using wavelet decomposition and hidden markov model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 208–214, February 1994.

[6] F. Perronnin, J. Dugelay, and K. Rose, "A probabilistic model of face mapping with local transformations and its application to person recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1157–1171, July 2005.

[7] H. Xue and V. Govindaraju, "Hidden markov models combining discrete symbols and continuous attributes in handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 458–462, March 2006.

[8] S. Mozaffari, K. Faez, V. Märgner, and H. El-Abed, "Lexicon reduction using dots for off-line farsi/arabic handwritten word recognition," *Pattern Recognition Letters*, vol. 29, no. 6, pp. 724–734, 2008.

[9] B. Sin, J. Ha, S. Oh, and J. Kim, "Network-based approach to online cursive script recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 29, no. 2, pp. 321–328, 1999.

[10] B. Van, S. Garcia-Salicetti, and B. Dorizzi, "On using the viterbi path along with hmm likelihood information for online signature verification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 37, no. 5, pp. 1237–1247, 2007.

[11] H. Jianying, M. Brown, and W. Turin, "Hmm based online handwriting recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 1039–1045, October 1996.

[12] Z. Liu and S. Sarkar, "Improved gait recognition by gait dynamics normalization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 863–876, June 2006.

[13] Y. Chen, Y. Rui, and T. Huang, "Muticue hmm-ukf for real-time contour tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1525–1529, September 2006.

[14] S. Lefevre, E. Bouton, T. Brouard, and N. Vincent, "A new way to use hidden markov models for object tracking in video sequences," *Proc. on Image Processing*, 2003.

[15] J. Yang, Y. Xu, and C. Chen, "Human action learning via hidden markov model," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 27, no. 1, pp. 34–44, 1997.

[16] T. Hu, L. De Silva, and K. Sengupta, "A hybrid approach of nn and hmm for facial emotion classification," *Pattern Recognition Letters*, vol. 23, no. 11, pp. 1303–1310, 2002.

[17] X. He, L. Deng, and W. Chou, "Discriminative learning in sequential pattern recognition – a unifying review for optimization-oriented speech recognition," *IEEE Signal Processing Magazine*, vol. 25, pp. 14–36, September 2008.

[18] C. Shalizi, *Causal architecture, complexity, and self-organization in time series and cellular automata*. PhD thesis, Unviersity of Wisconsin-Madison, 2001.

[19] J. Schwier, R. Brooks, C. Griffin, and S. Bukkapatnam, "Zero knowledge hidden markov model inference," *Pattern Recognition Letters*, 2009. Accepted for publication.

[20] J. Schwier, R. Brooks, and C. Griffin, "On the confidence of constructed hidden markov models," *To be submitted to IEEE Trans. on SMC Part B*, 2009. Under development.

[21] R. Brooks, J. Schwier, and C. Griffin, "Behavior detection using confidence intervals of hidden markov models," *IEEE Trans. on Systems, Man, and Cybernetics Part B*, 2009. Accepted for publication.

[22] J. Schwier, R. Brooks, and C. Griffin, "Determining the window size to differentiate markov models," *IEEE Transactions on Systems, Man, and Cybernetics: Part B*, 2009. Submitted for review.

[23] L. Deng and K. Erler, "Structural design of a hidden markov model based speech recognizer using multi-valued phonetic features: Comparison with segmental speech units," *Journal of the Acoustical Society of America*, vol. 92, pp. 3058–3067, December 1992.

[24] L. Deng and D. Sun, "A statistical approach to automatic speech recognition using the atomic speech units constructed from overlapping articulatory features," *Journal of the Acoustical Society of America*, vol. 95, pp. 2702–2719, May 1994.

[25] C. Shalizi and J. Crutchfield, "Computational mechanics: Patterns and prediction, structure and simplicity," *Santa Fe Institute Working Paper 99-07-044*, 2001.

[26] C. Shalizi, K. Shalizi, and J. Crutcheld, "Pattern discovery in time series, part i: Theory, algorithm, analysis, and convergence," *Santa Fe Institute, Tech. Rep.*, 2002.

[27] B. Mak and K. Chan, "Pruning hidden markov models with optimal brain surgeon," *IEEE Transactions on Speech and Audio Processing*, vol. 13, pp. 993–1003, September 2005.

[28] J. Chien and S. Fururi, "Predictive hidden markov model selection for speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 13, pp. 377–387, May 2005.

[29] M. Ostendorf and H. Singer, "Hmm topology design using maximum likelihood successive state splitting," *Computer Speech and Language*, vol. 11, no. 1, pp. 17–42, 1997.

[30] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and matching," *IEEE Transactions on Information Theory*, vol. 44, pp. 2743–2760, October 1998.

[31] V. Kumar, J. Heikkonen, J. Rissanen, and K. Kaski, "Minimum description length denoising with histogram models," *IEEE Transactions on Signal Processing*, vol. 54, pp. 2922–2928, August 2006.

[32] C. Shalizi, K. Shalizi, and J. Crutcheld, "An algorithm for pattern discovery in time series," *arXiv:cs.LG/0210025 v3*, November 2002.

[33] C. Shalizi and K. Shalizi, "Blind construction of optimal nonlinear recursive predictors for discrete sequences," *arXiv:cs.LG/0406011 v1*, June 2004.

[34] S. Zhou, J. Zhang, and S. Wang, "Fault diagnosis in industrial processes using principal component analysis and hidden markov model," in *Proc. of American Control Conference*, vol. 6, pp. 5680–5685, 2004.

[35] M. Baillie and J. Jose, "An audio-based sports video segmentation and event detection algorithm," in *Conf. on Computer Vision and Pattern Recognition Workshop*, p. 110, June 2004.

[36] A. Bobick and Y. Ivanov, "Action recognition using probabilistic parsing," in *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, pp. 196–202, 23-25 June 1998.

[37] S. Jia, Y. Qian, and G. Dai, "An advanced segmental semi-markov model based online series pattern detection," in *Proc. of 17th Intl. Conf. on Pattern Recognition*, vol. 3, pp. 634–637, 23-26 August 2004.

[38] F. Lukaszewski and K. Nagorko, "Pattern classification with incremental class learning and hidden markov models," in *Proc. of 5th Intl. Conf. on Intelligent Systems Design and Applications*, pp. 216–221, 8-10 September 2005.

[39] R. Huang, C. Kuo, L. Tsai, and O. Chen, "Eeg pattern recognition-arousal states detection and classification," in *IEEE Intl. Conf. on Neural Networks*, vol. 2, pp. 641–646, 3-6 June 1996.

[40] R. Scheaffer, *Introduction to Probability and Its Applications, Second Edition.* Duxbury Press, 1995.

[41] C. Grinstead and J. Snell, *Introduction to Probability.* American Mathematical Society, 1997.

[42] M. Evans, N. Hastings, and B. Peacock, *Statistical Distributions Third Edition.* John Wiley and Sons, Inc., 2000.

[43] B. Bowerman and R. O'Connell, *Linear Statistical Models: An Applied Approach, 2nd Edition.* PWS-Kent Publishing Company, 1990.

[44] J. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference Fourth Edition.* CRC Press, 2003.

[45] S. Shapiro and M. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 54, pp. 591–611, December 1965.

[46] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[47] G. Mealy, "A method for synthesizing sequential circuits," *Bell Systems Technical Journal*, vol. 34, pp. 1045–1079, 1955.

[48] E. Moore, "Gedanken-experiments on sequential machines," *Automata Studies, Annals of Mathematical Studies*, vol. 34, pp. 129–153, 1956.

[49] A. Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[50] A. Papoulis and S. Pillai, *Probability, Random Variables, and Stochastic Processes Fourth Edition*. McGraw-Hill, 2002.

[51] D. Upper, *Theory and algorithms for hidden Markov models and generalized hidden Markov models*. PhD thesis, Unviersity of California-Berkeley, 1989.

[52] L. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[53] M. Zaki, S. Jin, and C. Bystroff, "Mining residue contacts in proteins using local structure predictions," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 33, no. 5, pp. 789–801, 2003.

[54] L. Baum and J. Egon, "An inequality with applications to statistical estimation for probabilistic functions of a markov process and to a model for ecology," *Bull. Amer. Meteorol. Soc.*, vol. 73, pp. 360–363, 1967.

[55] L. Baum and G. Sell, "Growth functions for transformations on manifolds," *Pacific Journal of Mathematics*, vol. 27, no. 2, pp. 211–227, 1967.

[56] L. Min and Y. Shun-Zheng, "A network-wide traffice anomaly detection method based on hsmm," in *Proceedings of the 2006 International Conference on Communications, Circuits, and Systems*, vol. 3, pp. 1636–1640, June 2006.

[57] Y. Shun-Zheng and H. Kobayashi, "An efficient forward-backward algorithm for an explicit-duration hidden markov model," *IEEE Signal Processing Letters*, vol. 10, pp. 11–14, 2003.

[58] G. Forney, Jr., "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, pp. 268–278, March 1973.

[59] Q. Dan, W. Bingxi, Y. Honggang, and D. Guannan, "Discriminative training of gmm based on maximum mutual information for language identification," *Proc. 6th World Congress on Intelligent Control and Automation*, Jun. 21-23 2006.

[60] D. Kim and D. Yook, "Spectral transformation for robust speech recognition using maximum mutual information," *IEEE Signal Processing Letters*, vol. 14, pp. 496–499, July 2007.

[61] A. Biem, "Minimum classification error training for online handwriting recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1041–1051, July 2006.

[62] B. Juang, "Discriminative learning for minimum error classification," *IEEE Trans. on Signal Processing*, vol. 40, pp. 3043–3054, December 1992.

[63] O. Cappe, E. Moulines, and T. Ryden, *Inference in Hidden Markov Models*. Springer Verlag, 2005.

[64] R. Milner, *Communications and Concurrency*. Prentice Hall, 1989.

[65] S. Wang and H. Kung, "A new methodology for easily constructing extensible and high-fidelity tcp/ip network simulators," *Computer Networks*, vol. 40, pp. 257–278, 2002.

[66] G. Bham and R. Benekohal, "A high fidelity traffic simulation model based on cellular automata and car-following concepts," *Transportation Research Part C*, vol. 12, pp. 1–32, 2004.

[67] J. Herbst, "A microscale look at tumbling mill scale-up using high fidelity simulation," *Int'l Journal of Mineral Processing*, vol. 74S, pp. S299–S306, 2004.

[68] J. Huang and C. Gau, "Modelling and designing a low-cost high-fidelity mobile crane simulator," *Int'l Journal of Human-Computer Studies*, vol. 58, pp. 151–176, 2003.

[69] B. Fang and Y. Tang, "Improved class statistics estimation for sparse data problems in offline signature verification," *IEEE Trans. on Systems, Man, and Cybernetics, Part C*, vol. 35, pp. 276–286, August 2005.

[70] S. Wang, A. Liew, W. Lau, and S. Leung, "An automatic lipreading system for spoken digits with limited training data," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 18, pp. 1760–1765, December 2008.

[71] J. Zhu, S. Hoi, and M. Lyu, "Face annotation using transductive kernel fisher discriminant," *IEEE Trans. on Multimedia*, vol. 10, pp. 86–96, January 2008.

[72] M. Liwicki and H. Bunke, "Handwriting recognition of whiteboard notes - studying the influence of training set size and type," *International Journal of Pattern Recognition and Articial Intelligence*, vol. 21, no. 1, pp. 83–98, 2007.

[73] X. Yang, T. Zhang, Y. Zhou, and J. Yang, "Gabor phase embedding of gait energy image for identity recognition," *IEEE Int'l Conf. on Computer and Information Technology*, pp. 361–366, July 2008.

[74] G. Guruswamy, "A review of numerical fluids/structures interface methods for computations using high-fidelity equations," *Computers and Structures*, vol. 80, pp. 31–41, 2002.

[75] K. Lee and Y. Xu, "Human sensation modeling in virtual environments," *Proc. Int'l Conf. on Intelligent Robots and Systems*, vol. 1, pp. 151–156, 2000.

[76] M. Nechyba and Y. Xu, "On the fidelity of human skill models," *Proc. IEEE Int'l Conf. on Robotics and Automation*, vol. 3, pp. 2688–2693, 1996.

[77] S. Nemirovsky and M. Porat, "On texture and image interpolation using markov models," *Signal Processing: Image Communication*, vol. 24, pp. 139–157, 2009.

[78] J. Rajgopal and M. Mazumdar, "Modular operational test plans for inferences on software reliability based on a markov model," *IEEE Trans. on Software Engineering*, vol. 28, pp. 358–363, April 2002.

[79] E. Steinhart, "A mathematical model of divine infinity," *Theology and Science*, vol. 7, pp. 261–274, 2009.

[80] D. Rumsfeld, "Department of defense news briefing." News Transcript, February 2002.

[81] N. Salkind, ed., *Encyclopedia of Measurement and Statistics*, pp. 103–107. Thousand Oaks, CA: Sage, 2007.

[82] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, and D. Knuth, "On the lambert w function," *Advances in computational mathematics*, vol. 5, pp. 329–359, 1996.

[83] M. Inoue and N. Ueda, "Exploitation of unlabled sequences in hidden markov models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1570–1581, 2003.

[84] C. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[85] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Computing Survey*, vol. 23, no. 1, 1991.

[86] L. Brown, T. Cai, and A. DasGupta, "Interval estimation for a binomial proportion," *Statistical Science*, vol. 16, no. 2, pp. 101–117, 2001.

[87] A. Agresti and B. Coull, "Approximate is better than 'exact' for interval estimation of binomial proportions," *The American Statistician*, vol. 52, no. 2, pp. 119–126, 1998.

[88] L. Goodman, "On simultaneous confidence intervals for multinomial proportions," *Technometrics*, vol. 7, pp. 247–254, May 1965.

[89] S. Han and S. Cho, "Evolutionary neural networks for anomaly detection based on the behavior of a program," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 36, no. 3, pp. 559–570, 2006.

[90] K. Huang, H. Yang, I. King, and M. Lyu, "Imbalanced learning with a biased minimax probability machine," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 36, no. 4, pp. 913–923, 2006.

[91] J. Bennett and S. Lanning, "The netflix prize," *Proceedings of the KDD Cup and Workshop*, 2007.

[92] X. Huang, G. Duncan, and M. Jack, "Formant estimation system based on weighted least-squares lattice filters," *IEE Proc. F on Radar and Signal Processing*, vol. 135, pp. 539–546, December 1988.

[93] R. Far and S. Gazor, "Am-fm decomposition of speech signal using mwl criterion," in *Canadian Conf. on Electrical and Computer Engineering*, vol. 3, pp. 1769–1772, 2-5 May 2004.

[94] P. O'Shea, "The use of sliding spectral windows for parameter estimation of decaying sinusoidal signals," in *Proc. of IEEE Region 10 Annual Conf. on Speech and Image Technologies for Computing and Telecommunications*, vol. 2, pp. 827–830, 2-4 December 1997.

[95] R. Seiler, M. Schenkel, and F. Eggimann, "Off-line cursive handwriting recognition compared with on-line recognition," in *Proc. of 13th Intl. Conf. on Pattern Recognition*, vol. 4, pp. 505–509, 25-29 August 1996.

[96] J. Song and D. Kim, "Simultaneous gesture segmentation and recognition based on forward spotting accumulative hmms," in *Proc. of 18th Intl. Conf. on Pattern Recognition*, vol. 1, pp. 1231–1235, 2006.

[97] M. Dehghan, K. Faez, M. Ahmadi, and M. Shridhar, "Off-line unconstrained farsi handwritten word recognition using fuzzy vector quantization and hidden markov word models," in *Proc. of 15th Intl. Conf. on Pattern Recognition*, vol. 2, pp. 351–354, 2000.

[98] E. McDermott, T. Hazen, J. Le Roux, A. Nakamura, and S. Katagiri, "Discriminative training for large-vocabulary speech recognition using minimum classification error," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 15, no. 1, pp. 203–223, 2007.

[99] P. Peursum, H. Bui, S. Venkatesh, and G. West, "Human action segmentation via controlled use of missing data in hmms," in *Proc. of 17th Intl. Conf. on Pattern Recognition*, vol. 4, pp. 440–445, 23-26 August 2004.

[100] A. Sarma and D. Tufts, "Improving cfar detection through adaptive determination of reference window extents," in *Proc. of MTS/IEEE on OCEANS*, vol. 2, pp. 1501–1507, 2005.

[101] Q. Yu, X. Yang, S. Fu, X. Liu, and X. Sun, "An adaptive contoured window filter for interferometric synthetic aperture radar," *IEEE Geoscience and Remote Sensing Letters*, vol. 4, pp. 23–26, January 2007.

[102] J. Gross and J. Yellen, *Handbook of Graph Theory*. CRC Press, 2004.

[103] P. Borwein, "On the complexity of calculating factorials," *Journal of Algorithms*, vol. 6, pp. 376–380, September 1985.

[104] R. Ling, "Just say no to the binomial (and other discrete distribution) tables," *The American Statistician*, vol. 46, pp. 53–54, February 1992.