TIME INTERVALS AS A BEHAVIORAL BIOMETRIC

A DISSERTATION
SUBMITTED TO THE SEIDENBERG SCHOOL OF COMPUTER
SCIENCE AND INFORMATION SYSTEMS
OF PACE UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

John Vincent Monaco
November 2015

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Dr. Charles C. Tappert) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Dr. Lixin Tao)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Dr. Meikang Qiu)

Abstract

Timestamped events from human behavior are truly ubiquitous. In the Information Age, most human-computer interactions generate timestamped events in some way. From the keys pressed on a keyboard and the transmission of an email message, to the submission of a research article through an online submission system, a timestamp is generated and stored for each event. In many scenarios, timestamps can be observed without user cooperation or knowledge, further increasing the ubiquity of human temporal behavior. It is thought that temporal behavior is unique to an individual and the time intervals between events can be utilized as a behavioral biometric.

This work introduces the concept of and methodology surrounding time interval biometrics. Temporal behavioral patterns are analyzed and captured by a proposed generative model, the *partially observable hidden Markov model*, which generalizes the well-known hidden Markov model to account for partially observable hidden states through event types. Experimental results on publicly available data, including keystrokes, Bitcoin transactions, Linux kernel commits, White House visits, and terrorist activity, indicate a strong potential for time intervals as a behavioral biometric and present an improvement over previously published results. Along with these results comes greater incentive to remain anonymous, and some strategies to conceal temporal behavior are proposed.

Preface

There is a vast array of literature that attempts to model the way humans behave in time. Temporal behavior occurs on a wide range of scales, from high-frequency actions, such as individual neurons firing, to low-frequency group dynamics. Many works try to uncover the underlying mechanisms that are responsible for human actions and understanding of time. Fewer works analyze the individuality of human temporal behavior, and a literature review finds no work that coherently analyzes temporal behavior over all scales for biometric applications. This work attempts to fill that void by analyzing human temporal behavior across a range of time scales for the purpose of biometric identification and verification.

There has been much recent interest in keystroke dynamics, and as of now there are several commercial systems available. There is generally a growing demand for secure authentication and intrusion detection systems, with a recent focus on behavioral applications. Typing behavior is just one avenue for such a system, in which the timings between keystrokes are thought to be individually unique. Since many applications require a keyboard to interact with, typing behavior is a natural choice of biometric modality to verify the legitimacy of the end user. The aim of this work is to strengthen the current state-of-the-art in keystroke dynamics and to open the door to other modalities for a safer and more secure cyber space.

This dissertation was written primarily for biometrics professionals and assumes some prior knowledge in statistics. It should be accessible to anyone familiar with biometrics and may be of interest to those studying cyber security, group dynamics, and human behavior.

Acknowledgement

This work would not have been possible without the support of three people. The first is my dissertation adviser, Dr. Charles Tappert, who introduced me to biometrics and keystroke dynamics. I have had the pleasure of working alongside Dr. Tappert since I was an undergraduate student, and much of what I know about research and academia is the result of his efforts. I am also indebted to Dr. Allen Stix, who discovered my interest in computer science and spent countless summer afternoons feeding my curiosity as an undergraduate. Dr. Stix sparked my interest in research and continuing education, and I can only hope to one day have the impact he has had. Lastly, I owe a great deal to my loving wife who has come to realize that research can be performed any time of day or night. Her support has allowed me to focus on completing this work and begin the next chapter of our lives.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

"You are ~~what~~ *when* you eat"

Imagine an application that records the time when a person eats. Over several weeks, circadian and weekly patterns would emerge, and if this data is available for several people, individual differences in eating habits might also become apparent. Maybe some people eat frequently throughout the day or eat early dinners on Sundays, while others adhere to religious or cultural traditions that dictate eating schedules. Over time, the individual eating behavior could be used to uniquely identify a person from month to month. This type of application is not so far-fetched, as it is common to upload pictures of food to social networks where the timestamps can be extracted from metadata or the image itself.

Minkowski was one of the first to suggest time as the fourth dimension of spacetime, a theory later refined by Einstein. Now, time plays a critical role in human life. There are a number of factors that determine how humans behave in time. Timed human actions are subject to physiological processes, social norms, and workplace deadlines. There are also various mechanisms that allow humans to reason about and reconstruct time. A major scientific goal is to quantify the temporal behavior of humans (Barabási, 2005).

1.1 Background

1.1.1 Biometrics

Many applications call for the identification or verification[1] of an individual, such as bank account access, border control, and police investigations linking a suspect to a crime. The identification problem requires determining the identity of one person out of U possible choices. Verification is a binary problem in which the legitimacy of a claimed identity must be either verified or rejected. There are generally three ways through which a person can be identified or verified.

1. **Demonstration of knowledge:** having secret knowledge, such as a password.
2. **Demonstration of possession:** having a unique item, such as a key or an access card.
3. **Demonstration of being:** having unique physiological or behavior characteristics.

Biometric recognition is concerned with individually-unique physiological and behavioral characteristics. While knowledge and possession are transferable, physiological and behavioral characteristics are generally more difficult to reproduce. Biometric recognition can often be seamlessly integrated into an application and requires little effort on behalf of the user, whereas recognition through knowledge or possession places a burden on the user to remember some information or locate an item. Despite this, biometric applications deal with personal information and come at the expense of increased responsibility of the application authority. If a password is compromised, it can easily be reset, and if token is lost, it can be replaced. A biometric database may contain potentially sensitive information and offers greater risk if compromised. One solution to this problem is homomorphic encryption, which would allow the application authority to securely manipulate biometric data (Gentry et al., 2009), however these techniques are not yet practical due to computational requirements.

[1] The term *verification* is used instead of *authentication* to emphasize both forensic and real-time applications. Authentication typically implies an explicit claim of identity, while verification is more generally the process through which an identity is established (Wayman, 2009).

Biometrics is multidisciplinary, drawing concepts from biology, statistics, machine learning, computer science, and psychology. Biometric recognition requires that a physiological or behavioral trait can be quantified in some way. There are numerous biometric modalities utilized in practice, such as face, iris, and DNA, and many others that have been proposed, such as ear shape, gait, and smell. In general, a biometric modality should possess the following characteristics (Jain et al., 2004).

1. **Ubiquitous:** the modality is present for every person.
2. **Permanent:** the modality does not change over time.
3. **Unique:** the modality is measurably unique for each person.

A biometric modality is ubiquitous if it can be measured for every person and permanent if that characteristic does not change over time. Finally, the biometric modality is unique if any two individuals are measurably different. There are few biometric modalities that meet all three criteria. DNA comes close since every person possesses it and a person's DNA will not change over time. With the exception of monozygotic (or identical) twins, DNA is also distinct.

In practice, a biometric system must meet additional criteria to be of high utility. The system must be computationally efficient, widely accepted, and difficult to circumvent. Although DNA is difficult to reproduce, DNA analyses take at least several hours to complete with most current methods on the scale of 24-72 hours (Hopwood et al., 2010). This limits DNA to primarily forensic applications. On the other hand, fingerprints can be captured and compared to a large database in seconds. Methods for liveliness detection are usually employed to reduce the risk of spoofing, in which a fake fingerprint can be used to gain unauthorized access (Schuckers, 2002).

Humans perform biometric recognition on a daily basis. When people meet face-to-face, they are typically recognized by facial and voice characteristics. Over the phone, recognition is performed by voice alone. In practice, both humans and computers are subject to differences in accuracy. For example in a face recognition task, varying levels of accuracy are observed across forensic experts, industry experts, and laymen. Acting cooperatively, greater accuracy can be achieved. Using a simple fusion strategy, the accuracy of

one expert can be obtained by two industry experts or four laymen (Phillips and O'toole, 2014; White et al., 2015).

1.1.2 Behavioral biometrics

Most current biometric modalities are physiological characteristics, such as face, fingerprint, and iris. Behavioral biometrics are concerned with measuring human behavior for the purpose of identification or verification. While physiological biometrics capture only spatial information, behavioral biometrics capture both spatial and temporal information. For example, gait recognition aims to identify a person based on unique walking patterns. The trajectory of the feet and various points on the legs through time are utilized to isolate the individually-unique walking pattern.

Since behavioral biometrics require measurements over time, they cannot be captured instantaneously. Despite this, behavioral biometrics offer a number of advantages. They can be seamlessly integrated into an application by utilizing the interactions that naturally occur and remain unobtrusive to the user. For example, typing behavior during an online exam can be used to verify the identify of the test-taker. Utilizing behavior also allows for continuous authentication, in which the interactions are continuously monitored for suspicious activity. Consider an intrusion detection system on a mobile device. Figure 1.1 shows the touchscreen gestures of two subjects performing the same task. User A switches between hands to navigate a web page, seen by the gestures densely clustered on the left and right sides of the screen, while User B performs most gestures rather quickly with the right hand.

A behavioral biometric can be either continuous or discrete. *Continuous behavioral biometrics* utilize a sensor that detects some spatial information at fixed intervals, such as gait measured by a video camera or voice measured by a microphone. *Discrete behavioral biometrics* utilize a sensor that detects spatial information only when the user performs some action, such as pressing a button. In this case, the time and location of the pressed button are recorded. Discrete behavioral biometrics are event driven since they are only measured when the user interacts with the system. This work is concerned only with the discrete behavior.

(a) User A (b) User B

Figure 1.1: Touchscreen gestures from two users, A (left) and B (right), performing the same task. The users exhibit different touchscreen behaviors while navigating a web page. (Alotaibi et al., 2014).

1.1.3 Time intervals

Time interval biometrics utilize the timestamps from a sequence of timed events for the purpose of identification and verification. The time intervals, or the time between events, is of interest. Let t_n be the time of the n^{th} event. The series of time intervals is given by

$$\tau_n = t_n - t_{n-1} \, . \tag{1.1}$$

The original event times can be reconstructed from the time intervals and the time of the first event, t_0. In this work, the time intervals will be used to create a model of temporal behavior. In the case an event has duration, *i.e.* it is not instantaneous, then t_n marks the time of onset. Events may also contain additional information, such as intensity and location, which can be incorporated into the model to capture spatiotemporal behavior.

1.1.4 Psychology of time

Humans perform actions across a wide range of time scales, from the firing of individual neurons approximately 200 times per second, to email correspondence several times a day,

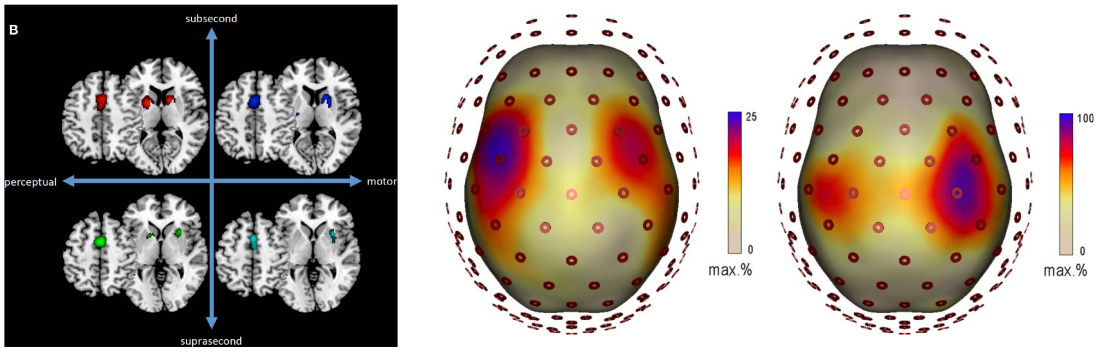
Scale	Time units	System	World (theory)	Example
10^7	months			
10^6	weeks		Social band	Email correspondence
10^5	days			
10^4	hours	Task		
10^3	10 min	Task	Rational band	Respond to one email
10^2	minutes	Task		
10^1	10 sec	Unit task		
10^0	1 sec	Operations	Cognitive band	Type one word
10^{-1}	100 ms	Deliberate act		
10^{-2}	10 ms	Neural circuit		
10^{-3}	1 ms	Neuron	Biological band	Muscle fiber response
10^{-4}	100 μs	Organelle		

Table 1.1: Newell's time scale of human action (Newell, 1994).

and the submission of a research paper which may occur only several times per year. Human actions occur in a hierarchy of time scales separated by orders of magnitude, with low-frequency events emerging from many high-frequency ones. This is demonstrated by Newell's time scale of human action, shown in Table 1.1.

Actions can be involuntary, driven by physiological processes, or performed deliberately by the conscious mind. Consider email correspondence, which requires actions on a wide range of time scales. At the lowest level, neurotransmitters active the muscle fibers in the hand to manipulate the mouse and keyboard. These signals are received at the fingertips, which coordinate to strike each key in the correct sequence. The time an email is finally sent depends on the daily schedule, reading speed, typing speed, and cognitive processing capabilities of the individual. It is typical for several emails to be answered in a single session, and at the highest level temporal behavior appears to be bursty and nonrandom (Barabási, 2005).

Humans also process time at different scales. At higher scales, actions are dominated by circadian rhythm and time is subjectively reconstructed (Buonomano, 2007). At lower scales, there are thought to be several different mechanisms responsible for time interval



(a) Wiener et al. (2011) (b) Praamstra et al. (2006)

Figure 1.2: Multiple mechanisms responsible for timing. (a) shows active areas of the brain detected through neuroimaging, with the regions responsible for perceptual vs motor (left, right) and subsecond vs suprasecond (up, down) tasks. (b) shows active areas of the brain detected through EEG recordings during a time judgment task (left) and movement-related task (right).

production and assessment. Sound localization requires temporal processing on the order of microseconds, as it takes little more than 500 μs for sound to travel from the left to the right ear (Carr, 1993). A distinction can be made between implicit and explicit timing (Zelaznik et al., 2002). As an example of explicit timing, consider a sprinter anticipating the start of a race. The estimation of when the starter's gun will fire requires an explicit representation of time. On the other hand, consider a distance runner aiming for a specific lap time. The lap time is an implicit representation of time since it emerges as a result of the runner's speed and endurance. Several researchers have attempted to map the various timing mechanisms in the brain, shown in Figure 1.2[2].

It is believed that movement initiation is explicitly timed, while the duration of movement is implicitly timed (Zelaznik et al., 2002). Many actions contain a mix of implicit and explicit time representations. In email correspondence, the longer time intervals from day to day are likely the result of explicit timing, as a user may choose when to begin an email correspondence session. The shorter time intervals within each email correspondence session are implicitly timed, depending on a number of factors such as reading and typing speed, message importance, and priority of other tasks.

[2]Figure 1.2a © 2011 Frontiers in Integrative Neuroscience. Figure 1.2b © 2006 Society for Neuroscience.

1.2 Motivation

Time intervals have a number of desirable properties that make them attractive as a behavioral biometric.

Timestamps are ubiquitous. Everyone is capable of producing a timestamped event in some form. At the very least, a living heartbeat can be recorded and the resulting time intervals used for identification (Israel et al., 2005). Most human computer interactions generate timestamped events, such as keystrokes, web surfing, SMS, and phone calls. Given the number of electronic devices people interact with on a daily basis, timestamps can be extracted from numerous sources.

Timestamps are persistent. It's standard practice to record the timestamp of an event and store that value in a database. A look at email, blog, and SMS database schemas will reveal timecreated, timemodified, and timedeleted columns. This information typically persists even after deleting or deactivating an account. Since timestamps are generally not considered sensitive information, they are commonly publicly available or easy to obtain.

Timestamps are minimally intrusive. This is true for actions that occur in the upper bands of Newell's time scale, as many low-frequency actions can be measured remotely and do not require the user to be present. Such examples are email and Bitcoin transactions, which occur on the order of hours and days. On the contrary, actions that occur on the lower end of Newell's time scale, such as heartbeat or eye movement, typically require special sensors or laboratory conditions to measure.

Timestamps can be collected without cooperation. Many spatial biometrics, such as fingerprint and iris, require at least some degree of cooperation from the user. Time interval biometrics utilizing actions in the upper bands of Newell's time scale can be measured remotely and without such cooperation. Many actions are publicly available by default, such as Twitter postings and Bitcoin transactions. Others, such as keystrokes, can be detected remotely through network traffic.

Timestamps are resilient to encryption and masking. While spatial information (*e.g.* IP address or GPS coordinates) can be masked, temporal information is more difficult to hide. For example, a message can be securely transmitted by encrypting its contents, however the time the message is sent is still visible to an eavesdropping third party. Masking temporal behavior requires introducing a lag between the user and the application, thus is not practical for real-time applications. This issue is discussed in Chapter 7.

Timestamps can be incorporated into domain-specific models. Models that capture temporal behavior through event timestamps can be incorporated into domain-specific models, such as those that capture motor behavior or other physiological processes. The resulting spatiotemporal model may offer greater statistical power than spatial or temporal models alone.

1.3 Problem statement

There are primarily three problems associated with time interval biometrics.

Identification Given a sequence of timed events, decide who they belong to (1 out of U).

Verification Given a sequence of timed events with claimed responsibility, decide whether the claim is legitimate (binary classification).

Prediction Given a sequence of timed events, predict the time of a future event.

The first two problems are typical in biometrics literature, with applications such as access control, customer tracking, and forensics, while the third problem is more generally related to behavioral analysis. Event prediction plays an important role in resource allocation and decision making. For example, predicting the time a cell phone user will place a call can help network providers manage their resources effectively, and predicting the time of the next attack from a terrorist group can help mitigate risk and aid in decision making.

1.4 Thesis

The thesis of this work is summarized as follows.

> Timestamped human events, generally characterized by periods of activity and inactivity, can be used for the purpose of biometric identification, verification, and prediction.

Exploratory and confirmatory analyses in Chapter 3 will show that human temporal behavior is generally bursty and nonrandom. This motivates the development of a model that captures temporal behavior with a dependency on event types. Empirical results in Chapter 6 further support this thesis and demonstrate the proposed model is an improvement over existing models. Given these findings, the problem of masking temporal behavior is analyzed in Chapter 7.

1.5 Roadmap

In this work, publicly available and mostly real-world datasets are utilized. Behavioral data collection is a tedious task, as experiments utilizing behavioral data are subject to a number of confounds that may invalidate experimental results. Confounds are often manifested through data collection procedures. In dealing with human subjects, the criteria for selecting participants, instructions given to participants, and reward structure may influence the outcome of an experiment (Read, 2005). Utilizing real-world data avoids some of these issues since the data originates from a natural environment. It also allows for comparison to previously published results.

Some data is only obtainable under real-world conditions. This is especially true for low-frequency events, such as email correspondence and Bitcoin transactions, which would require extensive longitudinal studies to analyze under laboratory conditions. Real-world data is desirable for temporal behavior analysis since laboratory conditions can impose an artificial upper bound on the time intervals. Such upper bounds are due to the timing constraints of a data collection session or the length of the entire study.

This work starts by reviewing the related efforts in modeling temporal behavior. Several datasets are then introduced and analyzed in an exploratory and confirmatory analysis. A novel statistical model is introduced, as well as model evaluation criteria. Experimental results are obtained and some privacy issues are introduced, accompanied by some strategies to mask temporal behavior. The major contributions of this work include the following.

1. Unified empirical analysis of human time intervals that span Newell's time scale.
2. Novel statistical model, the *partially observable hidden Markov model*, that utilizes partially observable states revealed through event types.
3. Novel strategies to mask temporal behavior supported by theory and empirical results.

The work is organized as follows.

Chapter 2 reviews the works related to time interval biometrics, including modeling techniques and privacy issues.

Chapter 3 analyzes behavioral data that span Newell's time scale and motivates the development of a new model.

Chapter 4 proposes the partially observable hidden Markov model for time intervals.

Chapter 5 defines the criteria under which the proposed model will be evaluated.

Chapter 6 presents experimental results on several datasets.

Chapter 7 discusses some privacy issues and possible mechanisms to circumvent time interval biometrics.

Chapter 8 draws conclusions and discusses future work.

Appendix A describes the time delay embedding procedure and the determined embedding parameters used in Chapter 3.

Appendix B describes the nonlinear prediction error used in some of the hypothesis tests in Chapter 3.

Appendix C contains a list of abbreviations and symbols.

Chapter 2

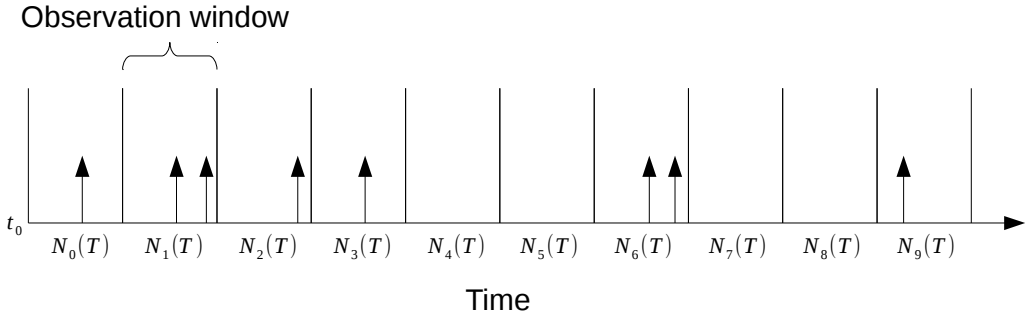
Related work

The use of time intervals for biometric identification dates back to World War II, in which a technique described as the "fist of the sender" was used to identify Morse code operators based on the rhythm of the message (Dunstone and Yager, 2008; Vacca, 2007). Keystroke dynamics later evolved, in which typing behavior is used for identification or verification. Gaines et al. (1980) was one of the first works to utilize keystroke timings for authentication, following a technical disclosure from IBM (Spillane, 1975). During the 1980's and 1990's, there was also much interest in transcription typing behavior (Salthouse, 1986; Cooper, 1983), which coincides with the growing popularity of the personal computer. Since 2000, numerous sources of temporal behavior have been explored and considered as biometric modalities. This chapter reviews the efforts in modeling temporal behavior using time intervals.

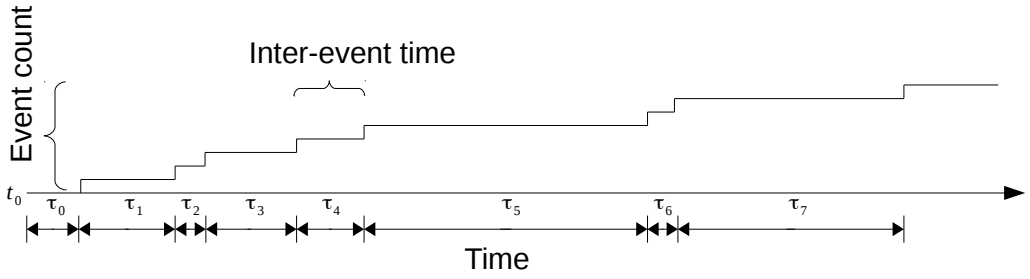
2.1 Modeling techniques

Point processes

A point process is a stochastic process whose realizations consist of point events in time or space (Cox and Isham, 1980). Point processes are well studied, dating back to at least the 1950's (Cox, 1955) with a theory of modeling established by David Cox (Cox and Isham, 1980). More recently, point process theory has been refined and extended in Daley and



(a) Windowed observations and event intensity



(b) Time interval observations and event count

Figure 2.1: Point process modeling techniques (original figure).

Vere-Jones (2003, 2007) and Lowen and Teich (2005). A sequence of event timestamps is a point process, and this univariate representation of temporal behavior allows one to take advantage of techniques in modeling point processes.

There are primarily two ways of modeling a point process, shown in Figure 2.1. In the first, the number of events that occur within a window of length T is observed, given by $N(T)$. In this type of model, the event intensity is modeled as a function of time. An alternative representation is through the time intervals between events, in which the time interval τ is a function of the event count. While the windowed observations require the discretization of time, the time intervals do not.

In practice, timed events are measured by a digital computer with a discrete clock and technically fall under the first type of model, *i.e.* discretization occurs at a very low level. Both the clock resolution and precision play a role in timing accuracy. Clock resolution is the degree to which a measurement can be made and clock precision is the degree to which a measurement can be repeated. Typically, the resolution of the system clock is fine enough

to have little effect on measurements. For example, millisecond precision is not needed to capture the correspondence behavior of an email sender, and second or minute resolution will usually suffice. The clock precision is typically courser grained than the resolution. Standard desktop computer clocks have a precision of approximately 16 ms, which has a noticeable effect in modeling typing behavior (Killourhy and Maxion, 2008).

The model proposed in this work is of the form shown in Figure 2.1b, where time intervals are modeled explicitly. This representation avoids the inevitable discretization of time and information loss when two events occur in the same interval. It also has a more compact representation, since long periods of inactivity can be summarized by a single scalar instead of a sequence of observation windows with empty event counts.

2.1.1 Micro behavior

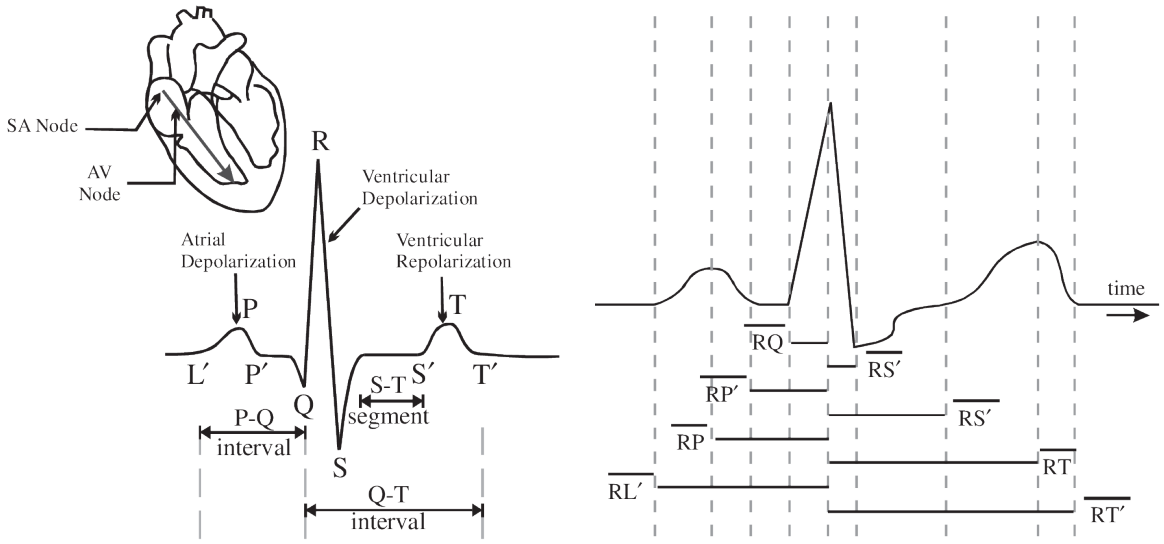
Micro behavior consists of the actions that occur in the biological band of Newell's time scale (Figure 1.1). These actions typically occur on a microsecond or millisecond scale and are only indirectly controlled by an individual. Examples of micro behavior are heartbeat, measured through electrocardiogram (ECG), and network traffic.

Electrocardiogram

ECG offers robust and secure biometric identification in medical applications. A living heartbeat is difficult to reproduce or disguise, and ECG is frequently used for monitoring a patient's condition. ECG measures the change in electrical potential over time as the heart muscle contracts due to the firing of an electrical signal beginning at the sinoatrial (SA) node and traveling to the atrioventricular (AV) node, shown in Figure 2.2a[1]. There are 5 fiducial markers commonly used in medical science to describe a single heartbeat, including the P wave, QRS complex, and T wave, also shown in Figure 2.2a.

Kyoso and Uchiyama (2001) showed that the durations of and time intervals between wave components can be used for biometric identification. The P wave duration, PQ interval, QRS interval, and QT interval are thought to be unique to an individual and invariant

[1]Reprinted from *Pattern Recognition*, 38/1, Steven A. Israel, John M. Irvine, Andrew Cheng, Mark D. Wiederhold, Brenda K. Wiederhold, ECG to identify individuals, 133-142, © 2004, with permission from Elsevier.



(a) ECG fiducial points (b) ECG time intervals

Figure 2.2: ECG fiducial points and time interval features (Israel et al., 2005).

to changes in heartbeat frequency, given by the R-R intervals. A series of experiments with ECG data collected from 9 normal users in 2 separate sessions showed that the QRS and QT intervals from a single heartbeat could be used to correctly identify users at over 94% accuracy with a generalized Mahalanobis distance criterion in discriminant analysis.

Additional timing features between fiducial points were defined in Israel et al. (2005). ECG was captured from 29 users at a high temporal resolution of 1000 Hz over 12 separate sessions, each consisting of 7 two-minute tasks designed to evoke various levels of anxiety. After 2-40 Hz bandpass filtering to isolate the heartbeat signal, the individual heartbeats were normalized to unit length by the L'T' distance, which corresponds to the total heartbeat duration. The time intervals from each fiducial point to the R position were calculated, shown in Figure 2.2b. After stepwise feature selection using Wilkes' lambda, 12 features were selected and classified using linear discriminant analysis. Users were identified with 81% accuracy using single heartbeats and 100% accuracy using 20 s blocks, invariant to the ECG sensor placement locations. Under high anxiety conditions, 78% of individual heartbeats and 97% of users were correctly classified. When training on a low anxiety task and testing on a high anxiety task, heartbeat classification accuracy reduced to 66% and

user classification accuracy remained at 98%. This demonstrated the time interval features to be robust under various conditions.

Network traffic

The time intervals between packet arrivals over a network can be used for device identification and verification. Network traffic is considered a micro behavior since the user typically can only indirectly control the packet arrival time intervals, which occur at the millisecond scale. The arriving packet time intervals are subject to a number of factors, such as device system load, installed applications, and wireless environment conditions. The time intervals can be used for device identification or device type identification, in which the device make and model are of interest as opposed to the physical device itself.

Methods that use the time intervals between arriving packets for device identification have several advantages. Most important is that these methods do not require payload analysis and can be applied to encrypted network traffic (*i.e.* the packet contents are not analyzed). Utilizing only the time intervals between packets is also computationally inexpensive compared to methods that require payload analysis and avoids relying on features that are often uninformative, such as port number and host address information (Auld et al., 2007).

In Uluagac et al. (2013) and Radhakrishnan et al. (2014) a method is developed that relies only on packet arrival time intervals from wireless devices to identify a specific device or device type. A feedforward neural network is trained for host identification using a feature vector of 300 time intervals as input. The technique can be applied passively, in which network traffic is observed as a third party, or actively, by pinging the host and recording the response times. With 37 devices on an isolated network, device identification accuracy was 98% on average using a passive analysis. On an open network, identification accuracy was 94% using a passive analysis and 97% using the active (ping response) technique.

The time intervals between beacon frames of a wireless access point can additionally be used to detect unauthorized wireless access points (AP) through estimation of the clock skew (Jana and Kasera, 2010). The clock skew is thought to be a unique characteristic of an AP and can be determined with as few as 50-100 packets. This technique can be applied to prevent MAC, BSSID, and SSID spoofing from rouge wireless access points.

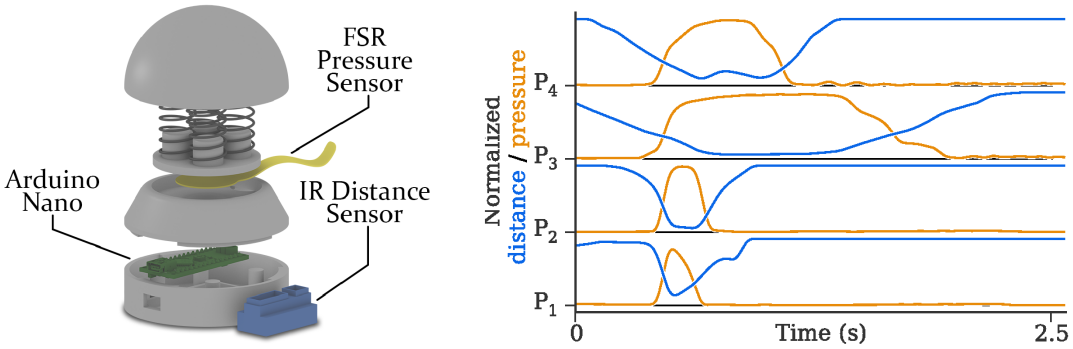


Figure 2.3: One-button design and press behavior of 4 users (Pohl et al., 2015).

2.1.2 Motor behavior

Motor behavior consists of the actions that occur in the cognitive band of Newell's time scale (Figure 1.1). These actions are typically driven by task goals and demonstrate a learned skill or behavior. Most current behavioral biometrics fall into this category, such as keystroke, mobile device behavior, and gait.

Button press

The behavior captured during a single button press can be sufficiently discriminative to unique identify an individual. Pohl et al. (2015) performed a study to evaluate biometric identification accuracy in non-engaging tasks during low-security scenarios, such as pressing a button to activate a coffee machine. A low-cost desktop button was designed, which included an infrared distance sensor with a 4-30 cm operating range and a pressure sensor with a 0.2-20 N range, both sampled at 400 Hz. The button design and normalized pressure and distance measures for 4 users are shown in Figure 2.3[2].

[2]Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org. UbiComp '15, September 7–11, 2015, Osaka, Japan. Copyright is held by the owner/author(s). Publication rights licensed to ACM. ACM 978-1-4503-3574-4/15/09. . . $15.00. http://dx.doi.org/10.1145/2750858.2804270.

Several novel timing features were extracted from the button presses. This included the total interaction time and press and release phase durations. Using a random forest classifier, the system was iteratively trained over 4 weeks in a production environment with 5 users. It was shown that after 3 weeks, the 5 users were correctly identified with 95% accuracy with only one button press. Allowing two button presses increased accuracy to 99%.

While a single button press offers convenience and minimal security, repeated button presses capture inter-press dynamics and offer more robust identification and verification. Laskaris et al. (2009) is one of the first works to explicitly propose time intervals as a behavioral biometric. A dynamical systems approach was used to verify the identity of 40 users who generated random time intervals by repeatedly pressing a single key on a keyboard. Users were asked to press the key as irregularly as possible after being given a definition of randomness and produced 10 different samples collected over 5 different days. The samples each contained 128 key presses, or alternatively 127 time intervals. It has been shown that humans are inherently bad at generating random numbers (Schulz et al., 2012), and perhaps the inability to be random contributes to the effectiveness of rhythm generation for biometric authentication.

Laskaris et al. (2009) employed Takens' Theorem by reconstructing a multi-dimensional system through time delay embedding of the time intervals. Parameters for time delay embedding were determined by the net class separability in phase space. The multivariate Wald-Wolfowitz (MWW) test (Friedman and Rafsky, 1979) was used to measure distances between the samples in reconstructed phase space. The distances were used to place the samples back in vector space by multidimensional scaling. With the samples in feature vector space, the author used standard techniques for validation; first a simple thresholding scheme followed by a minimum class variance support vector machine (MCVSVM), a SVM variant aimed at minimizing in-class variance. Verification performance was reported in terms of the equal error rate (EER), with error rates as low as 5.40%.

The MWW test statistic is more powerful in high dimensions than other multivariate test statistics (Friedman and Rafsky, 1979). The MWW is a natural extension of the Wald-Wolfowitz runs test, calculated as follows. Consider two samples of dimension d_e. The

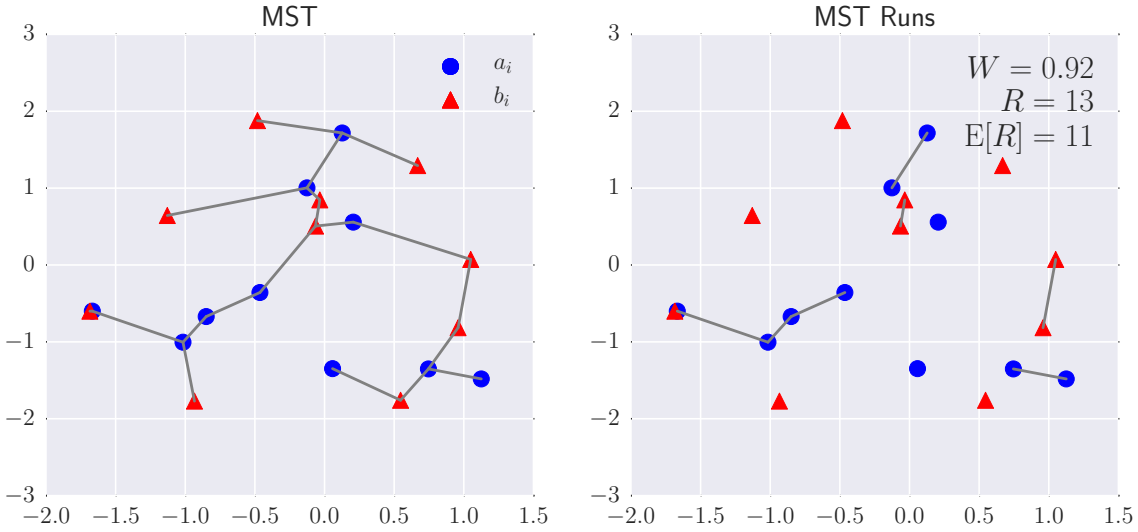


Figure 2.4: Multivariate Wald-Wolfowitz test. The runs are obtained by removing edges connecting vertices from different samples in the MST (original figure).

observations in both samples are pooled together and the minimum spanning tree (MST) is constructed over all observations in $\mathbb{R}^{d_e}$. A *run* is a segment of the tree that traverses vectors from only one sample. Runs are separated by edges that connect vertices (*i.e.* embedded vectors) from different samples. In the case that both samples originated from the same distribution, the branches of the tree will likely encounter vectors from each sample, resulting in a large number of runs. If the observations originated from different distributions, then the branches will traverse the vectors of one sample and then the other sample, resulting in relatively few runs.

An example of the MST and runs for a pair of bivariate samples is shown in Figure 2.4. Both samples were generated from the same distribution, resulting in a large number of runs. The number of runs, R, is used in the MWW test statistic as follows,

$$W = \frac{R - \frac{2mn}{N} - 1}{\left(\frac{2mn(2mn-N)}{N^2(N-1)}\right)^{\frac{1}{2}}} \tag{2.1}$$

where m and n are the number of vectors in each sample and $N = m + n$. The expected number of runs when the the samples follow the same distribution is given by

$$\mathsf{E}[R] = \frac{2mn}{N} + 1 \,. \tag{2.2}$$

It has been shown that $\mathsf{E}[R]$ has a standard normal distribution (Friedman and Rafsky, 1979). Since the MWW test relies on the construction of the MST, the total cost of the test is $O(N^3)$. In Monaco (2014), the MWW test statistic is approximated with a KD-tree, reducing the computational cost to $O(Nk\log(Nk))$ without loss of statistical power.

Keystroke

Keystroke dynamics refers to the way a person types on a keyboard. Typing behavior is thought to be unique enough to allow for user identification and verification. Keystroke dynamics can be used for both continuous and static verification. In continuous verification applications, it is desirable to verify a user as soon as possible, such as in an intrusion detection system. Since longer samples generally give a better estimate of a user's typing characteristics, there is a tradeoff between verification time and accuracy. Static verification applications include test-taker verification and password hardening, in which keystroke dynamics add an extra layer of security to username/password authentication.

Models of keystroke dynamics in the biometrics literature have predominantly been discriminative, with little attention paid to the generative mechanisms underlying typing behavior. Discriminative models are generally favored due to better asymptotic performance and efficiency (Jordan, 2002). Despite this, generative models have a wide range of applications and offer insight where discriminative models do not. Salthouse (1986) identified four major components that encapsulate transcription typing behavior, including input, parsing, translation, and execution phases. A survey of keystroke biometric systems can be found in Banerjee and Woodard (2012).

There are primarily three categories of keystroke biometric applications: short fixed-text, long fixed-text, and free-text. Short-fixed text applications include short fixed text entry that must be entered exactly, such as password or PIN entry. Incorrect sequences are

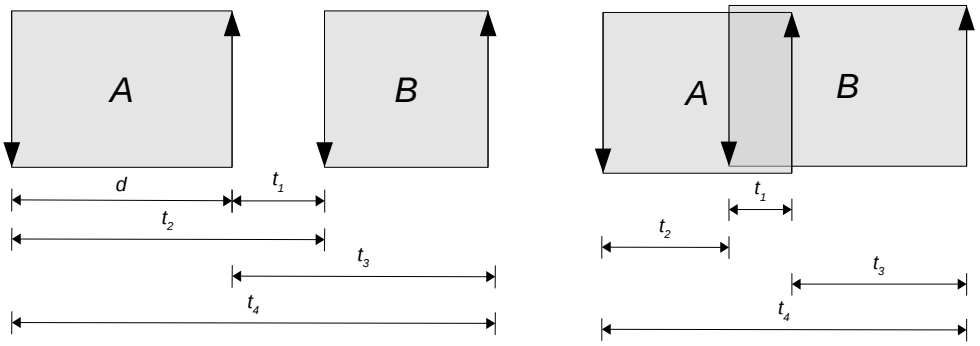


Figure 2.5: Non-overlapping and overlapping keystroke time intervals (original figure).

typically discarded and typing behavior is used as an additional verification mechanism. Long fixed-text requires a user to copy a passage, usually several sentences. Spelling and grammatical errors are tolerated and sometimes used as additional features to measure the user's behavior. Long fixed-text applications are used by some online course providers to verify the identify of online students (Maas et al., 2014). Free-text applications have no restrictions on the input and typically deal with longer text, such as the response to an essay-style question during an exam.

There are a number of time interval features that can be extracted from a sequence of keystrokes, including duration (press time to release time of a key) and transition (time interval between successive key presses or releases) features. Figure 2.5 shows the typical features used in modern keystroke biometric systems for non-overlapping and overlapping keystrokes. Duration and press-press transition times are always positive, while release-press transition is negative for overlapping keystrokes. Some works, such as Brizan et al. (2015), utilize trigram and higher n-gram features.

Tappert et al. (2009) performed a longitudinal study to evaluate several factors in keystroke biometric identification and verification systems. Free-text and long fixed-text samples were collected from 118 participants. Factors such as keyboard type, fixed vs free-text, and template degradation over time were evaluated in simulated experiments. A dichotomy model k-nearest neighbor (kNN) classifier was used, which transforms a multi-class problem into a two-class problem by taking feature vector differences. There were 239 features consisting mostly of mean and standard deviations of timing features for various key groups and a fallback hierarchy to account for missing observations. On a standard

desktop keyboard, identification accuracies were as high as 99% for the long fixed-text input 93% for free-text input. Training and testing on different types of input reduced the identification accuracy to 70-80%. Template degradation had a noticeable effect on identification accuracy in a subset of the collected data, where accuracy dropped from 90% to 83% after 4 weeks and from 84% to 67% after 2 years.

Killourhy and Maxion (2009) evaluated the performance of several classifiers on short fixed-text input. Using a database of 400 password samples from 51 participants, they found the best EER to be 0.096% with a scaled Manhattan detector as described in Araujo et al. (2005). In Killourhy (2012), the authors determined a 1% EER for 200 PIN entries from each of 28 participants. In this case, 2 out of 3 repetitions of the PIN are considered, requiring that 2 match the user's template. Using the same database as Killourhy and Maxion (2009), they analyzed the effect of the system clock on keystroke biometric authentication systems in Killourhy and Maxion (2008) and found a 4.2% increase when using a standard low-resolution clock of 15 ms compared to a high resolution clock with an accuracy of 200 μs. They hypothesized that other factors, such as bus contention, system load, and networking delays, may have an effect on a user's keystroke dynamics.

2.1.3 Social behavior

Social behavior consists of the actions that occur in the social band of Newell's time scale (Figure 1.1). These actions are driven by longer term goals and often depend less on physiology than motor behavior. Some examples of social behavior that are of interest in biometric applications are email correspondence, file sharing, and financial transactions.

Email correspondence

Malmgren et al. (2008) examined the heavy tails observed in email correspondence time intervals and propose a nonhomogeneous Poisson process that accounts for circadian and weekly periodicities. They analyzed a database of 394 email accounts that sent at least 41 emails over 83 days. In a Monte Carlo simulation, a best-fit truncated power law distribution was rejected for 344 users mostly due to the underestimation of intervals in the range

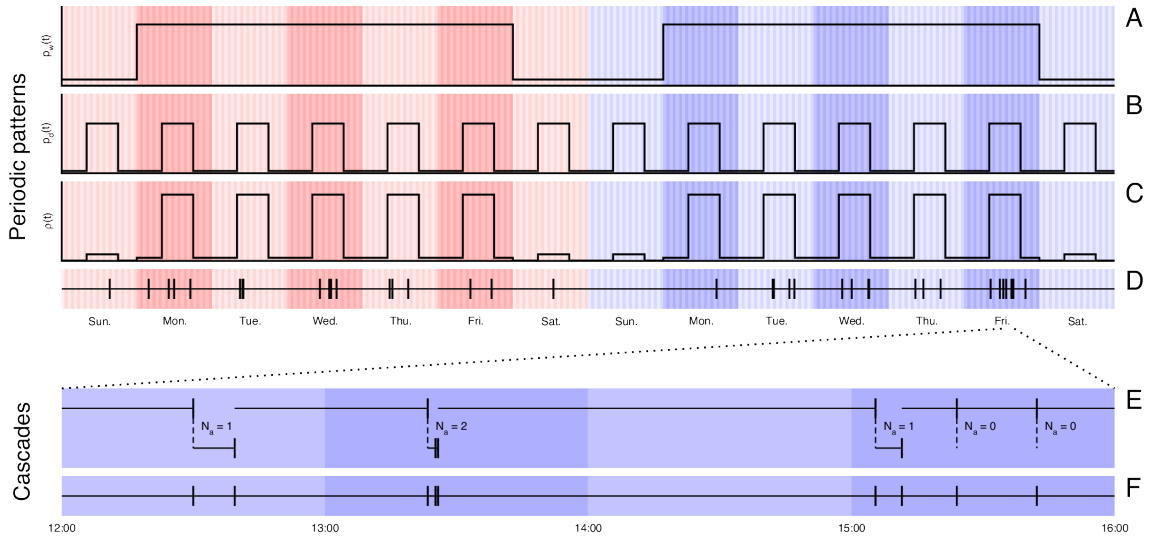


Figure 2.6: Cascading stochastic process model. The time series of events (D) is modeled by the activity rate (C), which is a combination of weekly (A) and daily (B) activity patterns. In the cascading model (E), the activity rate increases when an email is sent after a period of inactivity. The resulting events follow a nonhomogeneous Poisson process (F). (Malmgren et al., 2008).

of 24 hours. The proposed model accounts for the nonhomogeneities due to circadian and weekly rhythm.

The proposed model, show in Figure 2.6[3], was a cascading nonhomogeneous Poisson process where the cascades represent periods of activity. The rationale of the model is as follows. A user will generally respond to several emails in one sitting, and during that time the individual is considered to be an active state. During the active state, email correspondence follows a homogeneous process with random time intervals between sent emails. The initiation of the active state is governed by a nonhomogeneous process with its intensity described by a square-pulse distribution. Circadian and weekly patterns are captured by the heightened activity of the square pulse on weekdays and during daylight hours. In testing the goodness of fit, this model was rejected for only one user at the 0.05 significance level through Monte Carlo hypothesis testing. This result established the model as a possible explanation of email correspondence behavior.

[3] © 2008 The National Academy of Sciences of the USA.

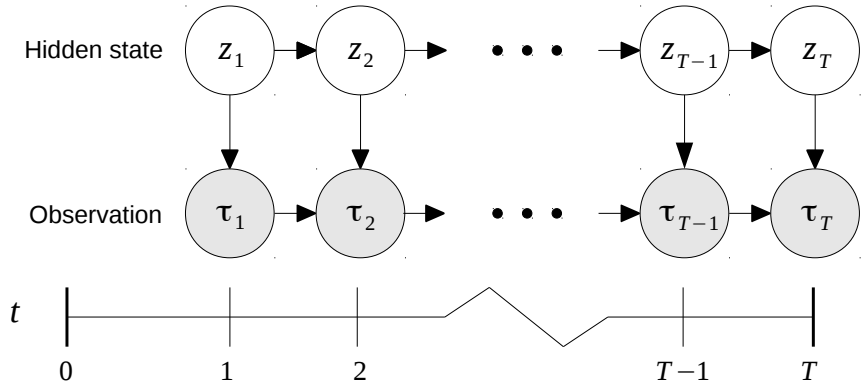
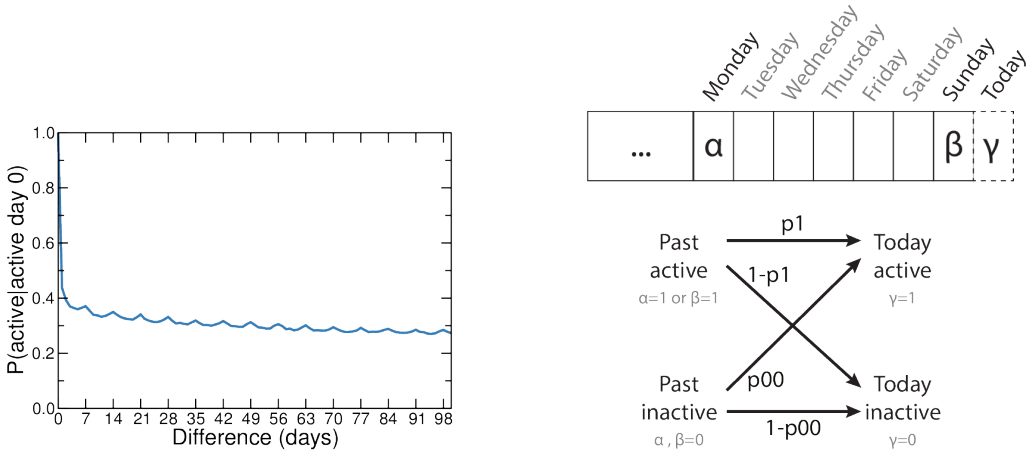


Figure 2.7: Double-chain hidden Markov model structure. The hidden state is given by z_n, and a first-order dependency exists between states and observations resulting in a double-chain structure (original figure).

A drawback of the cascading model is the computational cost in estimating the model parameters. This was performed through a simulated annealing procedure, which yielded strong candidates in each fitting through a computationally expensive procedure. In Malmgren et al. (2009), the same authors proposed a double chain hidden Markov model (DCHMM), which is similar in nature to Malmgren et al. (2008) and takes advantage of efficient parameter estimation through the expectation maximization (EM) algorithm. Unlike a typical first-order HMM, the DCHMM introduces a first-order dependency between observations, as shown in Figure 2.7. The probably of sending an email at any given time depends on the state of the individual and the time since the last email sent. Despite the double chain, the maximum likelihood parameter estimates can still be efficiently obtained (Berchtold, 1999).

File sharing

Gavaldà-Miralles et al. (2014) performed a study to determine whether copyright laws are effective in limiting the sharing of copyrighted material. A large database of peer-to-peer file sharing activity was analyzed. A two-state second-order hidden Markov model was proposed, with daily download activity dependent on the download activity of one day prior and seven days prior.



(a) Autocorrelation of download activity.

(b) Second-order hidden Markov model.

Figure 2.8: File sharing behavior follows circadian and weekly patterns, evident in the autocorrelation of download activity. A second-order hidden Markov model captures this behavior (Gavaldà-Miralles et al., 2014).

The mode structure was motivated by the autocorrelation of download activity, which showed high correlation for a lag of one day and peak correlations at 7 day intervals, shown in Figure 2.8[4]. A goodness of fit test was performed visually, in which synthetic data appeared very similar to real data. Several international regions were analyzed in an effort to understand the effect of copyright laws on file sharers. It was determined that the implementation of stricter copyright laws has only a short term effect on the behavior of file sharers, suggesting other means of managing illegal file sharing are necessary.

[4]Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org. CoNEXT'14, December 2–5, 2014, Sydney, Australia. Copyright 2014 ACM 978-1-4503-3279-8/14/12 ...$15.00. http://dx.doi.org/10.1145/2674005.2675009.

	ACC	EER
Time interval	0.30	0.23
Hour of day	0.25	0.25
Time of hour	0.04	0.49
Time of day	0.21	0.28
Coin flow	0.50	0.13
Inputs and outputs	0.22	0.28

Table 2.1: Bitcoin user identification and verification results (Monaco, 2015).

Financial transactions

Modeling financial transaction behavior is of interest in several applications, such as fraud detection and customer identification. An intrusion detection technique that uses only timing information has been proposed. In Scott (2001), a nonhomogeneous Poisson hidden Markov model (NPHMM) is used to model spending behavior and detect anomalous activity in a user's account. A similar NPHMM is used in Lu and Zeng (2012) to predict insurance claim counts based on weather patterns.

More recently, methods for identifying users in the Bitcoin network based on transaction behavior were proposed in Monaco (2015). Similar to Monaco (2014) and Laskaris et al. (2009), samples were time delay embedded and compared using an approximate MWW in the reconstructed phase space. Most of the samples were shown to be nonrandom and some to be nonlinear through Monte Carlo hypothesis testing. Using a population 61 users with 6 months of activity each, a linear-weighted kNN classifier with the MWW as a distance measure was able to correctly identify users with up to 75% accuracy. The identification accuracy (ACC) and EER for each feature are shown in Table 2.1. The time intervals alone allowed 30% of users to be correctly identified.

2.1.4 Group behavior

Group behavior consists of the actions performed by groups in the social band of Newell's time scale (Figure 1.1). While social behavior is driven by individual goals, group behavior emerges from multiple agents acting cooperatively.

Terrorist activity

Generative models of terrorist activity time intervals were proposed in several works. In Clauset et al. (2007), the frequency of severe terrorist events was shown to be scale free with roughly 13-year oscillations. The power law behavior was shown to be robust and remained intact when controlling for such variables as country and weapon type. Later, the same author showed the probability of a 9/11-sized event to be much greater than what is commonly perceived (Clauset et al., 2013). Using historical data from 1968, the probability of observing at least one 9/11-size event in the near future was estimated to be 11-35%. In this range, the presence a 9/11 event is not an outlier. Global terrorist events also appear to be clustered in time, demonstrated in Telesca and Lovallo (2006). While severe terrorist events were shown to be memoryless, global terrorist events indicated bursts of activity and time-clusterization. The Allan Factor was used to reveal the non-randomness of terrorist events, while severe events appeared to follow a Poisson process.

A self-exciting model of terrorist activity was proposed in Porter et al. (2012). The authors reasoned that the probability of a terrorist attack quickly increases and then decays after a recent attack. The daily number of attacks that occurred in Indonesia over 13 years were used to make successive predictions of the probability of a future attack. Using the log-probability gain as a performance metric, the proposed model outperformed a baseline model in making predictions. Figure 2.9[5] shows the best-fit self-exciting and baseline models with the empirical data.

In Raghavan et al. (2013), a two-state hidden Markov model for terrorist group activity is proposed. Model states correspond to the active and inactive periods of activity within the group. They hypothesized that group dynamics can be better understood if the underlying state is known. Using data from several terrorist groups, the model was iteratively trained and then used to predict the time to the next event. The asymptotic prediction errors of the two-state hidden Markov model were lower than a self-exciting hurdle model and baseline predictions using the sample mean, justifying the authors' hypothesis.

[5] © 2012 Institute of Mathematical Statistics.

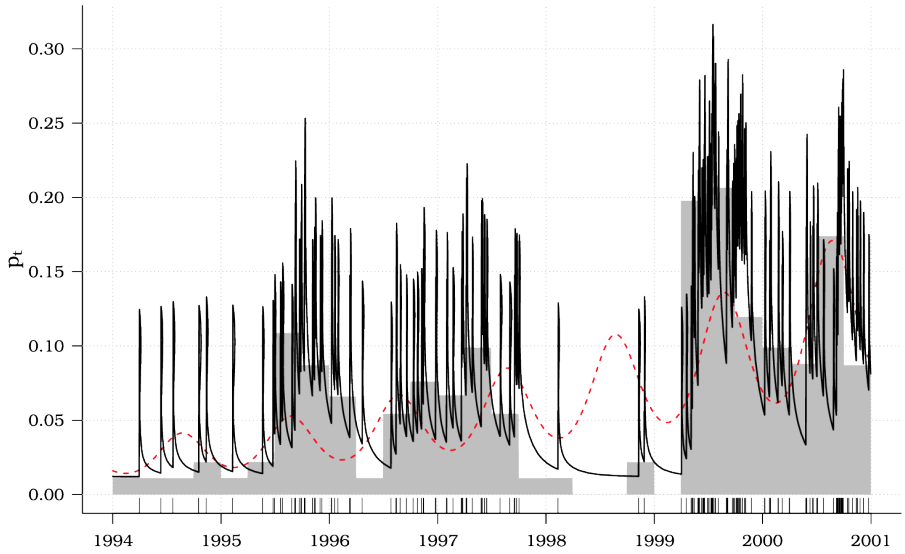


Figure 2.9: Self-exciting model of terrorist activity. The best-fit self-exciting (solid black) and baseline (dotted red) are shown with the observed piecewise constant probabilities (gray) (Porter et al., 2012).

2.2 Privacy

There is generally less privacy as one moves up in Newell's time scale. Social and group behavior are relatively easy to observe, while motor and micro behavior are personal and harder to measure without a user's consent. Since keystrokes are ubiquitous with modern computers and often contain sensitive information, they are the subject of some privacy issues.

Key-press times can be obtained remotely without installing a keylogger on the victim's computer by observing the network traffic generated from an interactive application, such as SSH in interactive mode (Song et al., 2001) and Google Suggestions service (Tey et al., 2013). A timing attack on SSH network traffic timestamps collected during password entry can provide about 1 bit of information in cracking a password. Song et al. (2001) verified that the key-press latencies can be reliably determined from the packet inter-arrival time of interactive SSH traffic, as the time between the actual press of a key and the creation of the packet by the kernel is negligible. In Tey et al. (2013), the key-press timestamps are

masked by a client-side buffering mechanism, however they still allow for the estimation of key-press latency statistics.

With access to the victim's computer, a spoofing attack such as in Rahman et al. (2013) can be utilized. Observing a victim's keystrokes directly allows for typing behavior to be easily replicated. Without any knowledge of the victim's typing pattern, an attack such as Serwadda and Phoha (2013) can be used. The latter work used a template enumeration technique, in which an independent keystroke database was used to reduce the search space of typing behavior. This works well for shorter strings and assumes that the verification system allows multiple attempts.

Monaco et al. (2015) provides empirical evidence for a two-state generative model of typing behavior in which the user can be in either a passive or active state. Given key-press latencies with missing key names, the model is then used to spoof the key-press latencies of a user by exploiting the scaling behavior between inter-key distance and key-press latency. The proposed generative model uses this partial information to perform a key-press-only sample-level attack on a victim's keystroke dynamics template. Results show that some users are more susceptible to this type of attack than others. For about 10% of users, the spoofed samples obtain classifier output scores of at least 50% of those obtained by authentic samples. With at least 50 observed keystrokes, the chance of success over a zero-effort attack doubles on average.

2.3 Limitations

Most of the proposed models have not been directly applied to all three problems mentioned in Section 1.3. Additionally, model selection has primarily been driven by statistical methods. Selection criteria for the problems described in Section 1.3 should be based on the relevant performance measures, such as empirical identification and verification accuracies. Lastly, many works employ models of the first type in Figure 2.1, in which windowed observations are made. This results in the discretization of time and seems less appropriate for time interval modeling.

Chapter 3

Empirical patterns

This chapter describes the experimental datasets and some preliminary modeling results. Behavioral time interval data is analyzed through exploratory and confirmatory techniques. Model design is guided by the findings, and data attributes, such as clustering and time-dependence, are determined.

3.1 Experimental data

Time intervals are ubiquitous and there are a number of publicly available time interval datasets. These range from high-frequency events, such as keystrokes and eye movement, to low-frequency events, such as cyber crime and visitation records. Table 3.1 summarizes the datasets used in this work. These datasets were chosen because of their relevance to biometric security applications and coverage of Newell's time scale.

3.1.1 Keystrokes

Keystrokes are detected by a *keylogger*, a program that records the sequence of typed keys along with the key-press and key-release timestamps. A keylogger can be installed on a client machine and record events in a system-wide context, or it can deployed over the web and record events in a sandbox environment. Event times are usually measured with millisecond resolution and approximately 16 ms precision Killourhy and Maxion (2008).

Dataset	Source	Events	Users	Time interval
Keystroke fixed-text	Monaco et al. (2013)	24k	60	263 ms
Keystroke free-text	Villani et al. (2006)	251k	56	310 ms
Mobile	Jain et al. (2014)	11k	52	366 ms
Keypad	Bakelman et al. (2013)	6.6k	30	376 ms
Bitcoin transactions	Reid et al. (2013)	239k	61	134 mins
Linux kernel commits	Passos et al. (2014)	16k	52	5.2 days
White House visits	Hudson (2015)	2.7k	18	9.9 days
Terrorist events	LaFree et al. (2007)	1.8k	10	125 days

Table 3.1: Publicly available time interval datasets. The number of events, users, and mean time interval for each user are shown for the subsets of each dataset selected for this work.

Keystrokes are typically categorized by input type as follows.

Short fixed-text The text input is exactly known beforehand for short input and is generally within the range of 4-16 characters. This includes passwords, phone numbers, and personal identification numbers (PIN).

Long fixed-text The text input is approximately known beforehand and usually consists of several sentences. A common scenario is the keystroke verification mechanism used by massively open online course (MOOC) providers in which the student must copy several sentences (Maas et al., 2014). Spelling and grammar mistakes are often ignored, and the actual input only approximately matches the expected input.

Long free-text The text input is unknown beforehand and consists of at least several sentences. The keystrokes collected as part of an essay question in an online test would be considered long free-text.

Further distinction is made by the input device, such as laptop, desktop, or mobile keyboard, as classification performance varies across devices (Tappert et al., 2009). The soft keyboards on mobile devices are different from desktop and laptop keyboards in several ways. On desktop and laptop computers, a key must be released after it is pressed, whereas on a mobile device, a key can be pressed without being released. This happens when the user's finger slides out of the soft key boundary before it is released.

Dataset	Users	Samples/User	Keystrokes/Sample	Time interval
Keystroke-fixed	60	4	100	263 ms
Keystroke-free	56	6	748±98	310 ms
Mobile	52	20	11	366 ms
Keypad	30	20	11	376 ms

Table 3.2: Keystroke data summary.

Four keystroke datasets are analyzed in this work, summarized in Table 3.2. They consist of long free-text, long fixed-text and short fixed-text input collected on desktop and laptop keyboards, as well as short fixed-text input collected on a mobile device. The long free-text *(keystroke-free)* dataset contains input from users who answered essay-style questions as part of a class exercise (Villani et al., 2006). A subset of the original dataset was randomly selected to obtain 56 users who answered 10 questions each. Of this dataset, each sample contained 748 ± 98 keystrokes. The long fixed-text dataset *(keystroke-fixed)* contains 60 users who each copied 4 different nursery rhymes or fables (Villani et al., 2006). Each sample is truncated so that it contains exactly 100 keystrokes. The short fixed-text dataset (*keypad*) contains keystrokes recorded using only the keypad on a desktop keyboard. This dataset contains 30 users who entered the phone number 9141937761 followed by the Enter key. Each user provided 20 correct entries, and each sample contains 11 keystrokes (10 digits + Enter). Incorrect entries were discarded. The mobile short fixed-text dataset (*mobile*) contains keystrokes recorded on a mobile device from 52 users entering the same digit sequence, 9141937761, followed by the Enter key (Jain et al., 2014). Data was recorded on 5 identical LG-D820 Nexus 5 devices which have 4.95 inch touch screens and 1080x1920 pixel resolution.

Each keystroke is described by the key that was pressed, the press time, and the duration. The keystroke features are described in Table 3.3. The time of word, time of sentence, and time of paragraph features can be extracted for the long text datasets, where the key press times are normalized to the respective word, sentence, or paragraph interval.

Feature	Description
Time interval	Time between events (millisecond resolution)
Duration	Duration the key was held down
Time of word	Normalized time of word
Time of sentence	Normalized time of sentence
Time of paragraph	Normalized time of paragraph

Table 3.3: Keystroke features.

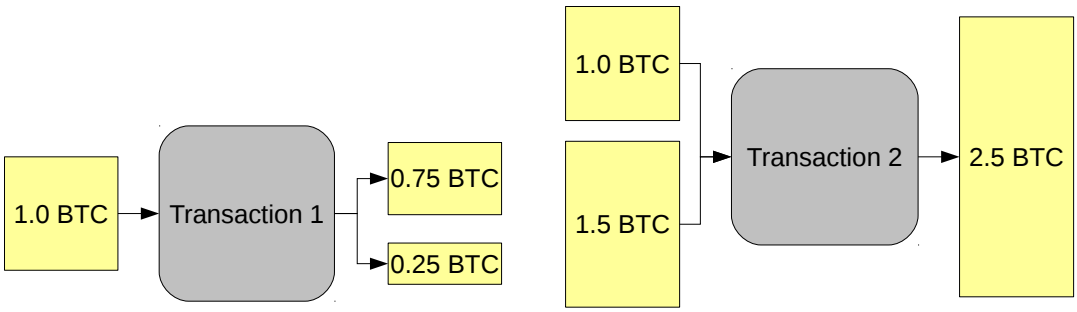
3.1.2 Bitcoin transactions

Bitcoin[1] was first proposed in 2008 and implemented in 2009 (Nakamoto, 2008). It was originally designed to solve the double-spending problem and eliminate the central authority responsible for verifying the legitimacy of online transactions, such as in traditional credit and banking systems. What makes the Bitcoin network especially unique is that its entire transaction history is publicly available giving one the opportunity to analyze economic and spending behavior (Kondor et al., 2014). The network operates in a peer-to-peer fashion, where clients collectively verify the legitimacy of each transaction.

A Bitcoin address is similar to a bank account number. Anyone can generate an unlimited number of Bitcoin addresses; thus it is common for users to control multiple addresses. A Bitcoin transaction is a many-to-many function, able to have multiple inputs and multiple outputs. Transactions are specified by the input and output addresses, the bitcoin value of each output address, and the time the transaction takes place. The cumulative bitcoin value from the inputs must be distributed to the outputs. Therefore, it is common for a sender to use an output address to receive change from a transaction. These addresses are normally referred to as *change addresses* (Meiklejohn et al., 2013). The sender must prove ownership of all input addresses for a transaction to be verified by the network. After the transaction is verified, the bitcoins are distributed to the output addresses as specified. Figure 3.1 shows two transactions with multiple outputs and multiple inputs, respectively.

A Bitcoin transaction is a behavioral event initiated on behalf of the sender. The sender could be a single person, organization, or a business. Additionally, a transaction could

[1]The proper noun *Bitcoin* refers to the peer-to-peer network, while lowercase *bitcoin* refers to the individual unit of currency.



(a) Multiple outputs (b) Multiple inputs

Figure 3.1: Bitcoin transactions with multiple outputs and multiple inputs. The total value of the inputs must be distributed to the outputs (original figure).

initiated by a computer program. The anonymity that can potentially be achieved using Bitcoin is only a side effect of the ability to create an unlimited number of addresses that act similarly to bank account numbers (Meiklejohn et al., 2013). It is recommended that users create a new address for every transaction, although many fail to do so. Because of this, multi-input transactions can be used to link addresses to the same owner since ownership of each address must be proven through public key cryptography. This was shown in a work where the *user network* was reconstructed from the publicly available transaction network (Reid and Harrigan, 2013).

The dataset used in this work is a subset of the user network compiled by Ivan Brugere[2], using tools modified from Martin Harrigan (Reid and Harrigan, 2013). Blockchain 230686 includes Bitcoin transactions made up to April 7, 2013, a network of 6.3M vertices and 37.4M edges. From the raw dataset, only users with between 100 and 1000 unique outgoing transactions per month, for at least separate 6 months (not necessarily consecutive), are kept. This yields 61 users, and the most recent 6 month-long samples for each user are kept, for a total of 366 samples. Restricting sample sizes to be within the same order of magnitude avoids discrimination by transaction frequency alone.

The Bitcoin transaction features are summarized in Table 3.4. These features capture both timing and network information of a each transaction. Transactions with 0 outputs and a positive number of inputs have a strictly positive bitcoin value; this represents a payment

[2] `http://compbio.cs.uic.edu/data/bitcoin/`

Feature	Description
Time interval	Time between events (second resolution)
Minute of hour	Minutes elapsed since the start of the hour
Hour of day	Hours elapsed since the start of the day
Day of week	The day of week, where Monday = 1, . . . , Sunday = 7
฿ value	Bitcoin value, positive for gain, negative for loss
Num. outputs	Number of output edges to other users
Num. inputs	Number of input edges from other users
Input/output balance	Number of inputs minus the number of outputs

Table 3.4: Bitcoin transaction features.

to the user. Similarly, transactions with 0 inputs and at least one output have a negative bitcoin value and represent a payment from the user. A transaction with 0 outputs and 0 inputs has a 0 bitcoin value, and represents a self-loop, *i.e.* a transaction between addresses belonging to the same user. This may occur if a user chooses to consolidate bitcoin to a single address or distribute bitcoin to multiple addresses.

3.1.3 Linux kernel commits

The commit history of the Linux kernel is analyzed (Passos and Czarnecki, 2014). This dataset contains over 300k commits from approximately 13.5k authors over a 7 period. The commits include patches, feature additions, and feature removals made to the Linux kernel. Timestamps are provided with day resolution, and it is possible for an author to make more than one edit per day. After an edit is made and submitted, it must be reviewed and approved by a committer before being integrated into the repository.

A subset of the dataset is selected for this work. Yearly samples are formed and users who made at least 160 commits for four years, with commits on at least 40 days per year, are selected. The resulting subset contains 52 users, with 4 years of activity each, and an average of 77 days of activity per year for each user. The event features are summarized in Table 3.5.

Name	Description
Time interval	Time between events (day resolution)
Day of week	The day of week, where Monday = 1, . . . , Sunday = 7
Week of month	The week of the month, from 1, . . . , 5
Time to commit	Mean time between authorship repository integration
Num. commits	Number of unique commits
Num. committers	Number of unique committers that reviewed the edits
Description chars	Number of words in the commit description
Description words	Number of characters in the commit description

Table 3.5: Linux kernel commit features.

3.1.4 White House visits

Starting in 2009, the White House has made its visitor logs publicly available (Hudson, 2015). The dataset contains 4.67m visits from over 200k people. Many of these visits are tour groups, in which case the tour guide is logged. Each event contains the time of arrival, time of departure, the scheduled time of arrival, and time the appointment was scheduled. It also contains the number of people accompanying the visitor, the check-in and check-out locations, and the scheduled meeting room. All timestamps have minute resolution.

A subset of the original dataset is analyzed in this work. Users who made at least 8 separate visits for 4 years, with at least 20 visits per year are selected. Four yearly samples are created for each user. The resulting dataset contains 18 users, with an average of 38 visits per year. The features of each visit are summarized in Table 3.6.

3.1.5 Terrorist activity

The Global Terrorism Database (GTD) is an open source database of worldwide terrorist events that have occurred since 1970 (LaFree and Dugan, 2007). Each event is described by over 100 variables, including the time of the event, suspected and confirmed participating groups, and claims of responsibility. According to GTD data collection methodology, a terrorist event is an intentional act of violence committed by a sub-national group or individual. The event must have at least two of the following attributes for inclusion in the GTD:

Feature	Description
Time interval	Time between events (minute resolution)
Hour of day	Hours elapsed since the start of the day
Day of week	The day of week, where Monday = 1, . . . , Sunday = 7
Week of month	The week of the month, from 1, . . . , 5
Duration	Duration of the visit
Tardiness	Time between the scheduled appointment and the actual time of arrival. Negative for early arrivals.
Forecast	Time between when the appointment was scheduled and the scheduled time of the appointment.
Num. people	Number of people accompanying the visitor

Table 3.6: White House visit features.

Feature	Description
Time interval	Time between events
Day of week	The day of week, where Monday = 1, . . . , Sunday = 7
Week of month	The week of the month, from 1, . . . , 5
Month of year	Month of the year, where January = 1, . . . , December = 12
Num. perpetrators	Number of perpetrators
Num. killed	Number of people killed
Num. captured	Number of perpetrators captured
Num. related events	Number of related events

Table 3.7: Terrorist activity features.

- Political, economic, religious, or social motivations
- Intent to coerce or intimidate a large audience
- Actions that violate international humanitarian law

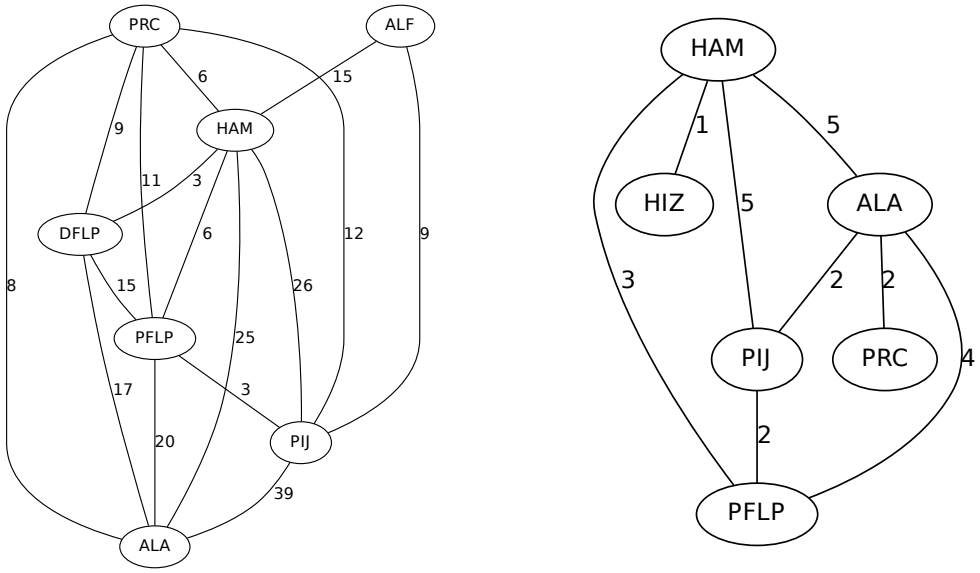
The database is overseen by a panel of 12 terrorism research experts and is generally considered of high quality (Sheehan, 2012). As of January, 2015, the GTD contains 129,017 events from January 1, 1970, to October 15, 2013. Table 3.7 summarizes the event features used in this work. Additional attributes include the criteria, the attack type, whether the attack was a success or not, the type of weapon used, and the target of the attack.

Group	Confirmed	Suspected	Claimed	Competing
Al-Aqsa Martyrs Brigade (ALA)	161	6	14	5
al-Fatah (ALF)	51	8	2	0
Democratic Front for the Liberation of Palestine (DFLP)	48	0	2	0
Hamas Islamic Resistance Movement (HAM)	288	24	14	5
Hizballah (HIZ)	277	48	2	1
Palestine Liberation Organization (PLO)	152	13	1	1
Palestinian Islamic Jihad (PIJ)	176	9	13	5
Palestinians (PAL)	1086	18	0	0
Popular Front for the Liberation of Palestine (PFLP)	105	5	8	3
Popular Resistance Committees (PRC)	52	0	2	1
Total unique events	2315	128	46	13
Grand total (region)	2485 (5509)			

Table 3.8: Middle Eastern active terrorist groups. Number of confirmed, suspected, claimed, and competing events from active terrorist groups in Israel, the West Bank and Gaza Strip, and Lebanon.

In this work, only the events that have occurred in Israel, the West Bank and Gaza Strip, and Lebanon are considered. This region was selected for several reasons. Besides being an area of ongoing conflict, it contains a relatively high number of active terrorist groups that operate solely within that region. The active groups and event counts are summarized in Table 3.8.

The active groups in this region were determined according to the following criteria. The group must have confirmed responsibility for at least 40 events in the region, where they must also have committed at least half of all their confirmed events. This eliminates active groups operating primarily outside the region. After applying these criteria, 10 active groups are identified. This is a relatively dense area of active terrorist groups, as the region consists of only 3 states. By contrast, an area such as Sub-Saharan Africa, which contains 46 states, has 11 groups that meet the criteria. The selected region also contains a relatively high number of events with competing claims of responsibility.



(a) Cooperation network. Edges represent the number of events with mutual confirmed responsibility.

(b) Competition network. Edges represent the number of mutually exclusive competing claims.

Figure 3.2: Terrorist group cooperation and competition networks (original figure).

An event in the GTD can have up to three different participating groups, and the participation of each group can be either confirmed or suspected[3]. A group may claim responsibility for an event even though they were not involved. This automatically makes the group a suspect, however it does not confirm their involvement. In some cases, events have competing claims of responsibility from suspected groups, and the claims of responsibility in these events are mutually exclusive (*i.e.* one group claims sole responsibility for an event).

The network of groups with confirmed mutual responsibility for an event can be constructed. Likewise the network of competing claims to an event can be constructed. The cooperation and competing-claims networks are shown in Figure 3.2. Many of the groups that have confirmed mutual responsibility for an event also have competing claims of responsibility.

[3]Suspected participation is based on a dubious claim of responsibility or speculation of involvement (LaFree and Dugan, 2014).

3.2 Motivation for a two-state model

3.2.1 Heavy tails

A heavy-tailed distribution is one whose tail is not exponentially bounded (Asmussen, 2008). The time intervals between human actions are generally modeled by heavy-tailed distributions (Vázquez et al., 2006). It has been shown that numerous human activities, such as email correspondence, phone calls, and print job submissions, follow power law distributions (Barabási, 2005). The power law distribution is given by probability density function

$$P(\tau) \sim \tau^{-\alpha} \qquad \tau > \tau_{min}, \alpha > 1 \tag{3.1}$$

where α is the scaling exponent. For $\alpha < 3$, the standard deviation is undefined, and for $\alpha < 2$, the distribution has infinite mean. Such a distribution is often characterized as being *scale-free* since all values are likely to be present. Evidence of scale-free behavior is thought to be a sign of complexity (Newman, 2005).

It is common for observed time intervals to have some maximum value, due to either a theoretical maximum or some constraint on the maximum time between events. A truncated power law is one whose tail follows an exponential cutoff, with density function

$$P(\tau) \sim \tau^{-\alpha} e^{-\beta\tau} \qquad \tau > \tau_{min}, \alpha > 1, \beta > 0 \tag{3.2}$$

where parameter β determines the exponential cutoff.

Accurately fitting a power law distribution is generally difficult (Clauset et al., 2009). To further complicate the matter, it is common for power law behavior to only occur in the tail of an empirical distribution, *i.e.* for values $\tau > \tau_{min}$. Using the methods developed in Clauset et al. (2009), the best-fit scaling parameter α, truncation parameter β, and minimum τ_{min} are determined for each user in each dataset. Figure 3.3 shows the distributions of scaling exponents for the users in each dataset.

A loglikelihood ratio test is performed for each fit, comparing the truncated power law to the alternative exponential distribution. Note that by definition, since a distribution is heavy-tailed if its tail is not bounded by the exponential, the exponential is the minimum

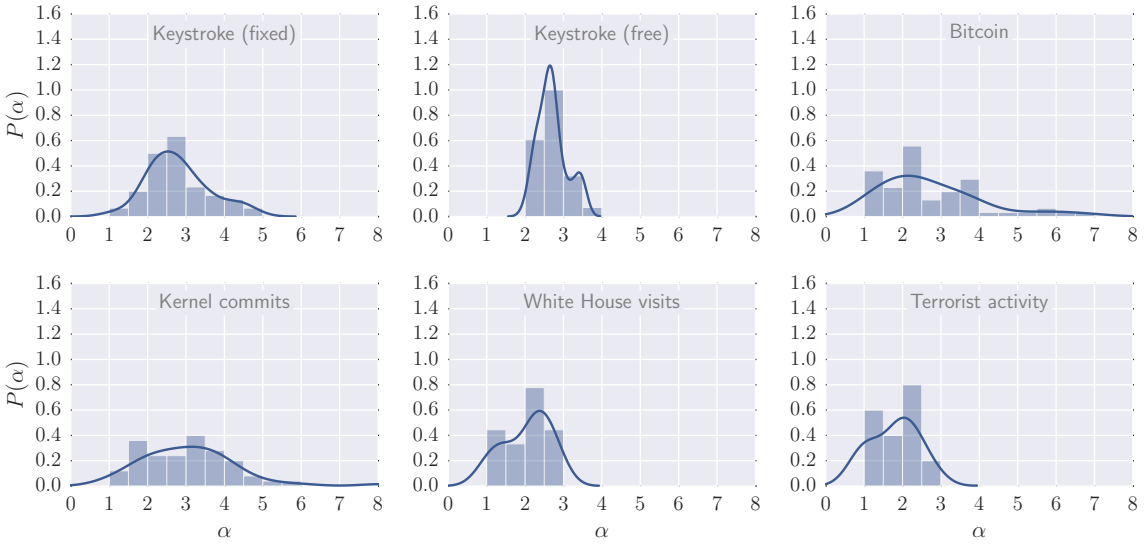


Figure 3.3: Distribution of scaling exponents for each dataset. The best-fit truncated power law is determined (original figure).

possible alternative to test against (Asmussen, 2008). Table 3.9 shows the proportion of users for which the exponential or power law distribution is preferred in the loglikelihood ratio test, and (in parentheses) the proportion of users for which that preference is significant at the 0.05 level. p-values are determined using standard techniques in comparing simple-vs-simple hypotheses (Vuong, 1989).

Despite showing evidence of heavy tails, the power law would be a poor choice of model for the purpose of identification and verification. The reason for this is that only the

Dataset	Power law	Exponential
Keystroke (free)	1.00 (0.63)	0.00 (0.00)
Keystroke (fixed)	1.00 (0.88)	0.00 (0.00)
Bitcoin	0.98 (0.44)	0.02 (0.00)
Kernel commits	1.00 (0.40)	0.00 (0.00)
White House visits	1.00 (0.11)	0.00 (0.00)
Terrorist activity	1.00 (0.50)	0.00 (0.00)

Table 3.9: Power law vs exponential loglikelihood ratio test results showing the proportion of users better modeled by each distribution and (in parenthesis) the proportion of users for which that preference is significant at the 0.05 level.

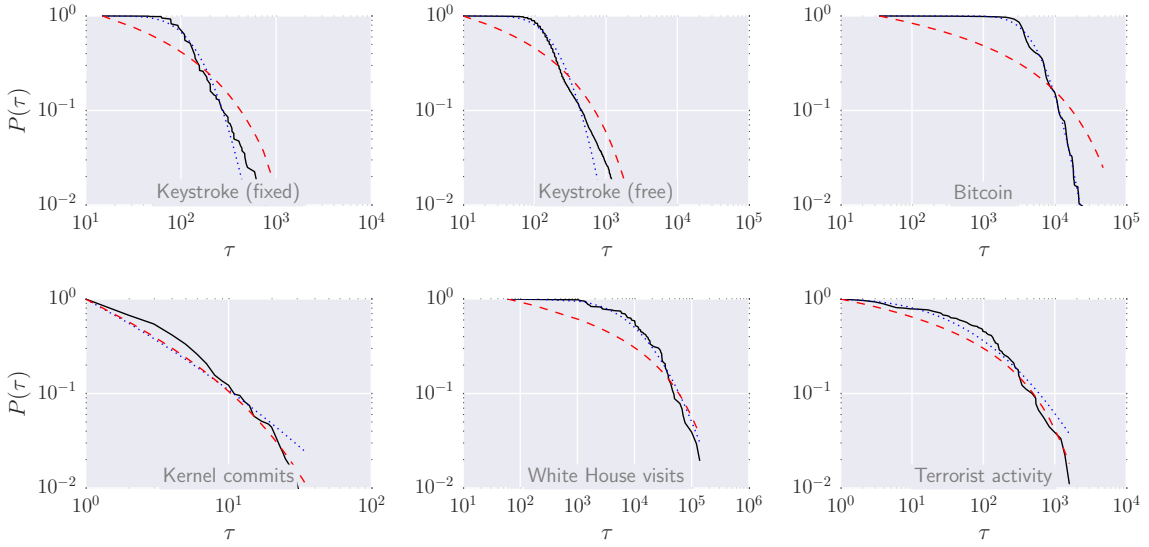


Figure 3.4: Best-fit power law (dashed red) vs log-normal (dotted blue) time interval distribution examples for a randomly selected user (solid blue) from each dataset (original figure).

tail of the distribution follows a power law. The power law does not specify how the values below τ_{min} are distributed, which represents a major portion of the empirical time interval distribution for many users. The log-normal is often an adequate alternative to the truncated power law. Both distributions exhibit heavy tails and can become indistinguishable from each other (Perline, 2005). Whether one model is more appropriate than the other is somewhat a matter of debate. In Stouffer et al. (2005) and Barabási et al. (2005) it is disputed that the log-normal distribution provides a better fit for time intervals of email correspondence. Log-normal distributions are usually the result of a multiplicative process (Mitzenmacher, 2004), and power law distributions may arise from nonhomogeneous Poisson processes (Hidalgo and César, 2006).

As a demonstration, Figure 3.4 shows the best-fit truncated power law and log-normal distributions for a randomly selected user from each dataset. The log-normal provides an adequate fit, while the truncated power law is unable to capture the complete empirical distribution. A loglikelihood ratio test confirms this finding, with results shown in Table 3.10. The log-normal is usually preferred over the truncated power law when τ_{min} is set to the minimum value in the distribution, *i.e.* all of the data is used in determining distribution

Dataset	Power law	Log-normal
Keystroke (free)	0.00 (0.00)	1.00 (1.00)
Keystroke (fixed)	0.00 (0.00)	1.00 (1.00)
Bitcoin	0.00 (0.00)	1.00 (1.00)
Kernel commits	0.75 (0.56)	0.25 (0.08)
White House visits	0.00 (0.00)	1.00 (1.00)
Terrorist activity	0.70 (0.20)	0.30 (0.00)

Table 3.10: Power law vs log-normal loglikelihood ratio test results showing the proportion of users better modeled by each distribution and (in parenthesis) the proportion of users for which that preference is significant at the 0.05 level.

parameters. The only exceptions to this are kernel commits and terrorist activity, where about half the users significantly prefer the truncated power law at the 0.05 level.

3.2.2 Temporal clustering

Temporal clustering of a point process can be evaluated in several ways. The coefficient of variation (CV) measures the relative variation of time intervals, given by

$$C_v = \frac{\sigma_\tau}{\langle \tau \rangle} \tag{3.3}$$

where σ_τ is the time interval standard deviation and $\langle \tau \rangle$ is the expected time interval. A Poisson process has $C_v = 1$, and temporal clustering occurs when there is a deviation from a random process such that $C_v > 1$. The CV is determined for each user in each dataset. The mean CV of each dataset is shown in Table 3.11. The free-text keystroke dataset shows higher CV than fixed-text due to the presence of longer time intervals that occur during free-text production.

While the CV indicates the presence of temporal clustering, it is not clear at what timescales the clustering occurs. The Allan Factor (AF) can be used to determine the time scale at which clustering occurs (Barnes and Allan, 1966). The AF is given by

$$\mathrm{AF}(T) = \frac{\mathsf{E}\left[N_{k+1}(T) - N_k(T)\right]^2}{2\mathsf{E}\left[N_k(T)\right]} \tag{3.4}$$

Dataset	C_v
Keystroke (fixed)	1.34
Keystroke (free)	3.98
Bitcoin	1.65
Kernel commits	3.26
White House visits	1.70
Terrorist activity	2.83

Table 3.11: Mean coefficients of variation.

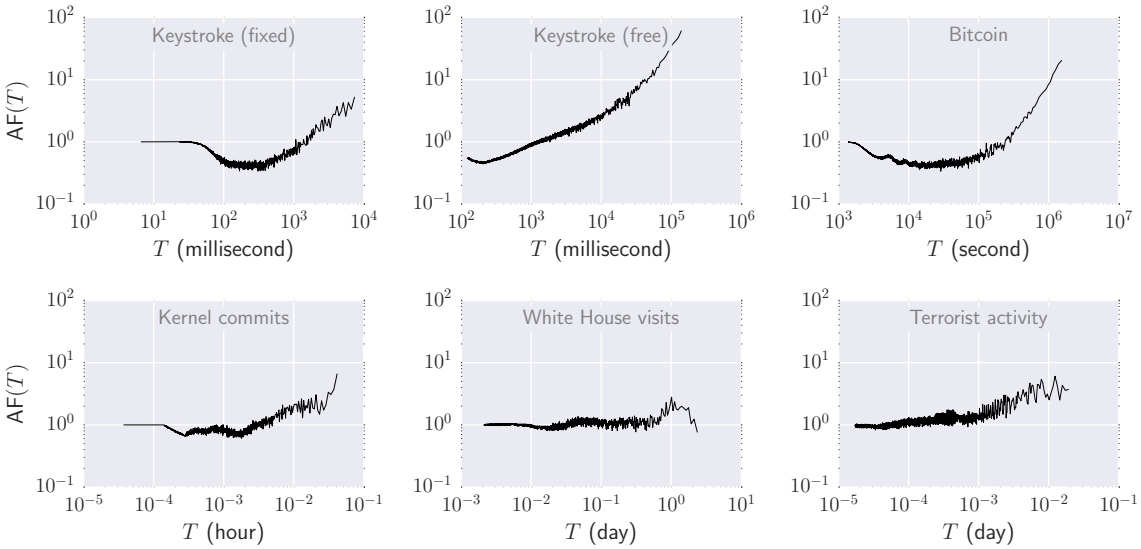


Figure 3.5: Allan Factor of a randomly selected user from each dataset (original figure).

where $N_k(T)$ is the number of events that occur in the k^{th} window of length T. The AF can be interpreted as the frequency-deviation of a process with sample period T, where higher values indicate frequency instability. Constant AF for any T indicates a memoryless, or random, process. If the events are clustered in time, then the AF will vary as a power law by

$$\mathrm{AF}(T) = 1 + \left(\frac{T}{T_1}\right)^{\alpha}$$

where T_1 marks the onset of scaling behavior and $\lim_{T\to 0}\mathrm{AF}(T) = 1$. The AF of a randomly selected user from each dataset is shown in Figure 3.5. Linear trends in the tail of each AF series on the log-log plot indicate power law scaling.

3.2.3 Exogenous variables

It is not clear whether the temporal clustering in each dataset is due to user behavior or to exogenous variables. For example, clustering in White House visits can be the result of periods of globally high and low visitation frequency. There could be a political event taking place, in which the number of visitors would be expected to increase, or a visitor ban, in which case the number of visitors would decrease. Visitation frequency is also subject to weekly, monthly, and seasonal rhythms.

A relative clock can be used to eliminate the effect of exogenous variables and determine whether temporal clustering is due to user behavior or external factors. Whereas an absolute clock advances according to the normal concept of time, a relative clock is advanced based on the global state of the system (Zhou et al., 2012). Consider two events that occur at time t_{n-1} and t_n. The absolute time between these events is $\tau_n = t_n - t_{n-1}$. The relative time between the $(n-1)^{\text{th}}$ and n^{th} events is the number of events that occur globally between times t_{n-1} and t_n. The relative clock eliminates exogenous variables by advancing time by the global event count. In periods of global silence (including missing data), time will not advance, and in periods of globally high frequency, time intervals will be proportionally to the global system state.

As motivation for using a relative clock, consider the terrorist activity dataset. Rug plots of the events from each group are shown in Figure 3.6. The GTD underwent several structural changes, resulting in data loss during the 1990s. This can be seen by the gap in events after day 8000 (circa 1993). This gap translates to a period of inactivity for each model using an absolute clock, whereas time advances by 0 events during this period using a relative clock.

The mean CV for each dataset is calculated using a relative clock, with results shown in Table 3.12. Similarly, the AF can be calculated using a relative clock. The relative clock AF values are shown in Figure 3.7 for the same users as in Figure 3.5. Power law scaling, given by linear trends in the AF tail, is more apparent using a relative clock. For the keystroke datasets, each sample is normalized to $t_0 = 0$ before calculating the relative clock times.

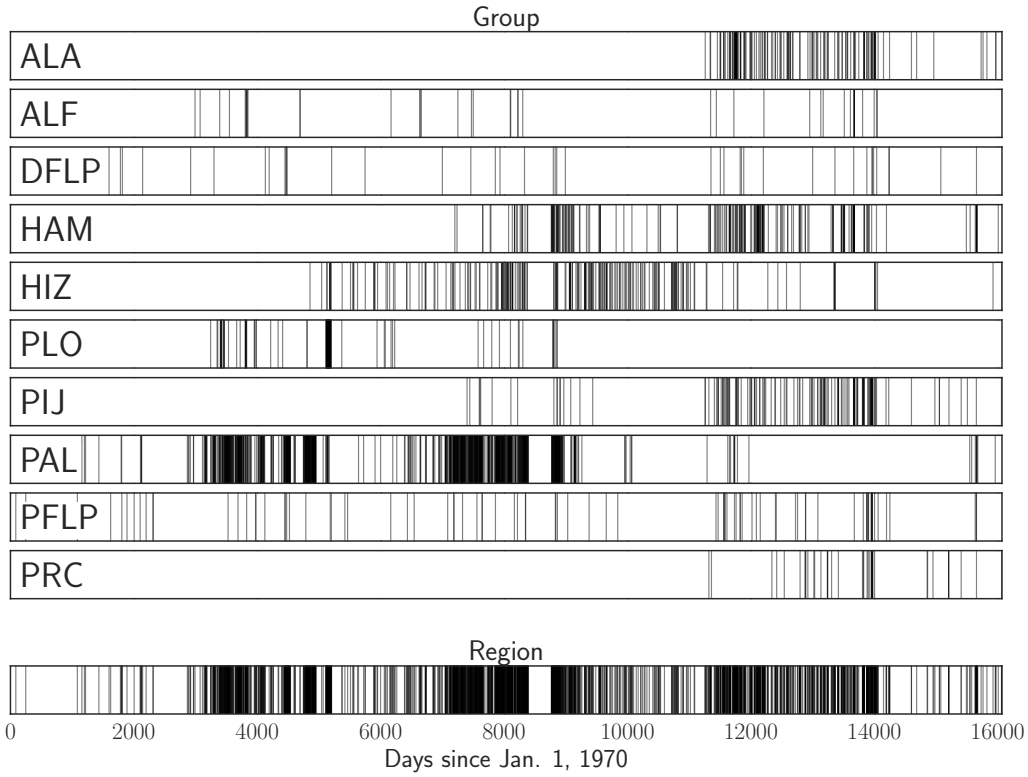


Figure 3.6: Terrorist activity group and global events (original figure).

Dataset	C_v
Keystroke (fixed)	1.31
Keystroke (free)	4.03
Bitcoin	1.64
Kernel commits	3.19
White House visits	1.62
Terrorist activity	1.94

Table 3.12: Mean coefficients of variation using a relative clock.

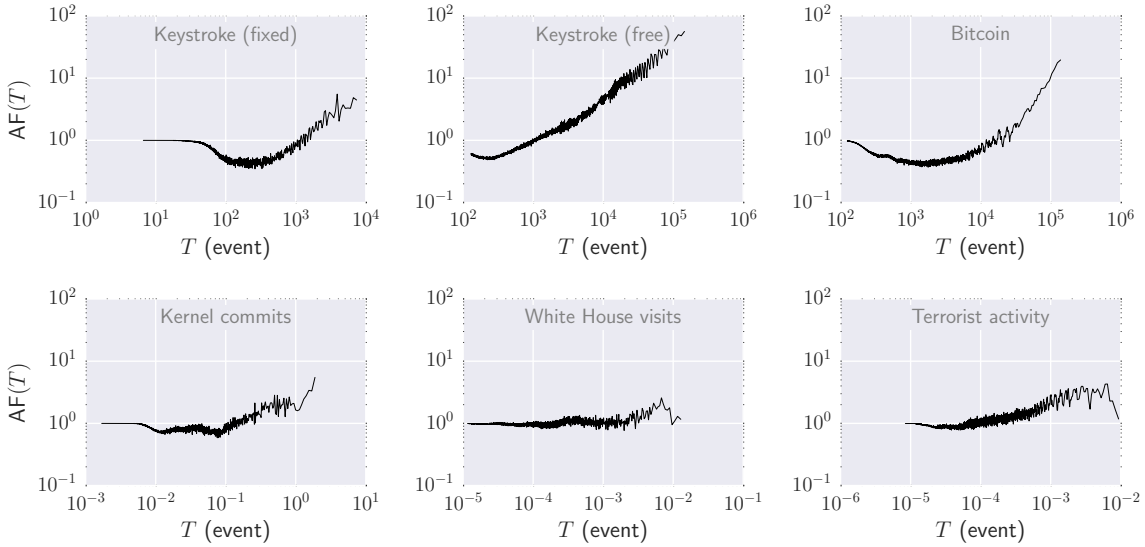
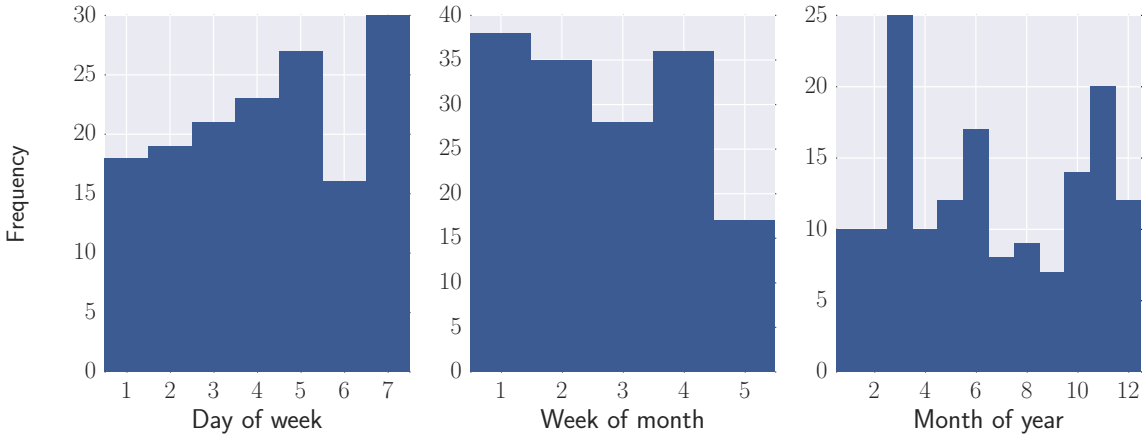


Figure 3.7: Allan Factor using a relative clock of a randomly selected user from each dataset (original figure).

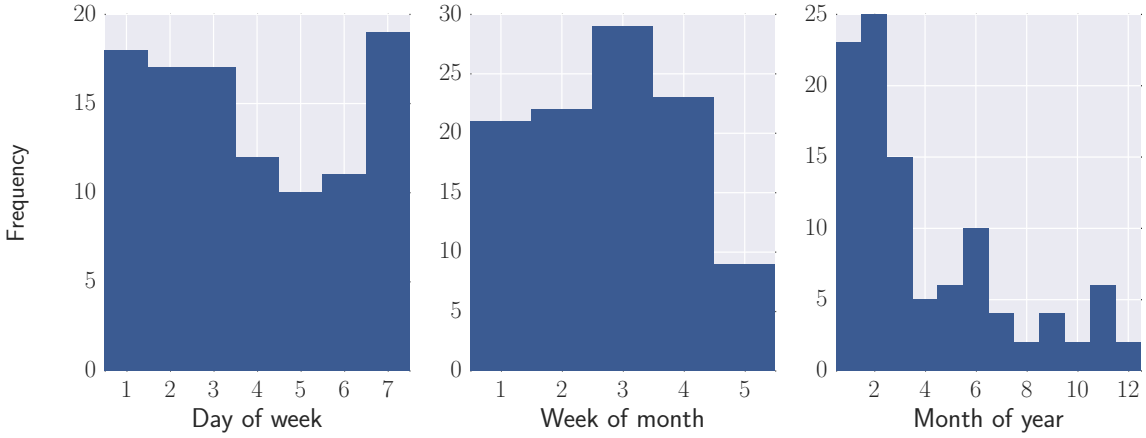
3.3 Time dependence

Time intervals fail to capture one important aspect of timed events, which is the distribution of events within a time window. The events of a random process are expected to be uniformly distributed within a time window T of any length. Specifically, the cumulative distribution function of a random process is $F(t) = \frac{t \mod T}{T}$ for event time t where $0 < t < T$.

Divergence from a uniform distribution indicates a dependence on time. This can be due to circadian, weekly, or seasonal rhythms, where the event probability depends on the time of day, day of the week, or month of the year. As an example, consider the activity of two groups from the terrorist activity dataset. The distributions of events within the week, month, and year are shown in Figure 3.8. The PIJ group shows a tendency of increased activity towards the end of the week with the exception of Saturday (day 6), while the PLO group has decreased activity with the exception of Sunday (day 7). The PLO group is also noticeably more active in the first several months of the year.



(a) PIJ event frequency



(b) PLO event frequency

Figure 3.8: Terrorist activity event distributions (original figure).

3.3.1 Testing for uniformity

A test for uniformity is used to determine whether a user significantly deviates from having uniformly-distributed events within a given interval. The chi-squared (χ^2) test indicates whether observed categorical frequencies differ from the expected frequencies. The test statistic is given by

$$\chi^2 = \sum \frac{(O-E)^2}{E} \tag{3.5}$$

where O and E are the observed and expected frequencies, respectfully, in each category. The test statistic is compared to a chi-squared distribution with *df* degrees of freedom, where *df* is 1 less than the number of categories in the test.

The categories are determined by splitting the time window into equal-sized bins. The observed frequencies are the number of events in each bin, and the expected frequency of each category is the mean observed frequency. For example, in testing whether events are uniformly distribution within each day, the time of day is considered. The day is broken up into 12 bins, from 00:00-02:00, 02:00-04:00, and so on. The event frequency in each bin gives the observed frequencies. The test statistic is compared to a chi-squared distribution with 11 *df* since there are 12 bins .

The chi-squared test for uniformity is performed for each user in each dataset. The proportions of users that reject the null hypothesis are shown in Table 3.13. Only tests for the relevant intervals for each dataset are performed (*i.e.* it doesn't make sense to test whether keystrokes are uniformly distribution over each day since the samples only span several hours. This test would also almost surely be rejected, as keystrokes are subject to circadian rhythm).

The day, week, month, and year time windows are natural choices for low-frequency events, such as those that occur in the rational and social bands of Newell's time scale. For higher frequency events, the event distributions may follow different patterns. Consider the distribution of keystrokes within each word, where words are separated by the Space key. If a user has a tendency to type the beginning of a word quickly, the distribution will be left-skewed, whereas if a user types the end of a word quickly, the distribution will be right

Dataset	Minute	Hour	Day	Week	Month	Year
Keystroke (fixed)	0.98	1.00	-	-	-	-
Keystroke (free)	0.82	1.00	-	-	-	-
Bitcoin	0.03	0.43	0.87	0.85	1.00	-
Kernel commits	-	-	-	0.87	0.87	0.46
White House visits	-	-	-	1.00	0.22	0.39
Terrorist activity	-	-	-	0.00	0.40	0.50

Table 3.13: Proportion of users rejected (0.05 significance) in a chi-squared test for uniformity within the given interval. Minute, hour, and day interval tests are broken up into 12 bins and have 11 *df*, the week interval test has 6 *df*, the month interval test has 3 *df*, and the year interval test has 11 *df*.

skewed. This is due to a higher concentration of events (keystrokes) in the beginning and end of the time window it takes to type the entire word, respectively.

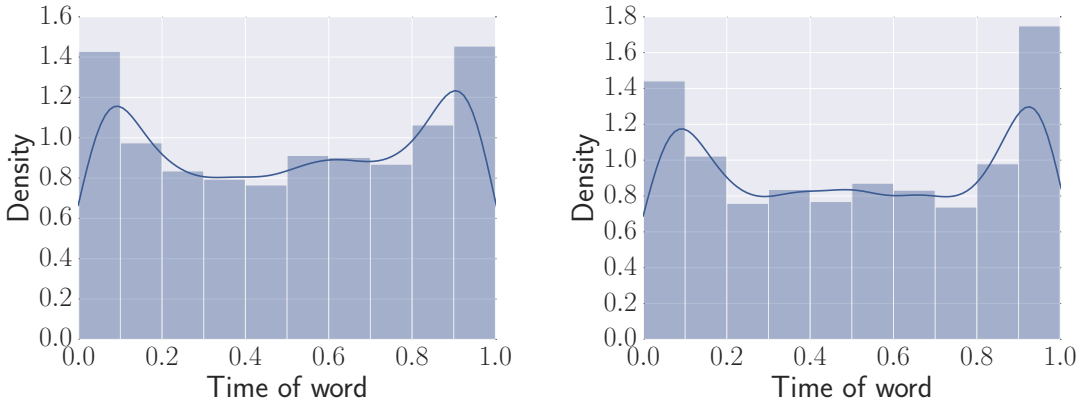
The normalized time of word is given by

$$\frac{t - t_w}{t_s - t_w}$$

where t is the event time (of each key press), t_w is the time of the first key of the word, and t_s is the time of the space key following the word. The normalized time of sentence and time of paragraph are determined similarly, using the Period and Enter keys as delimiters, respectfully.

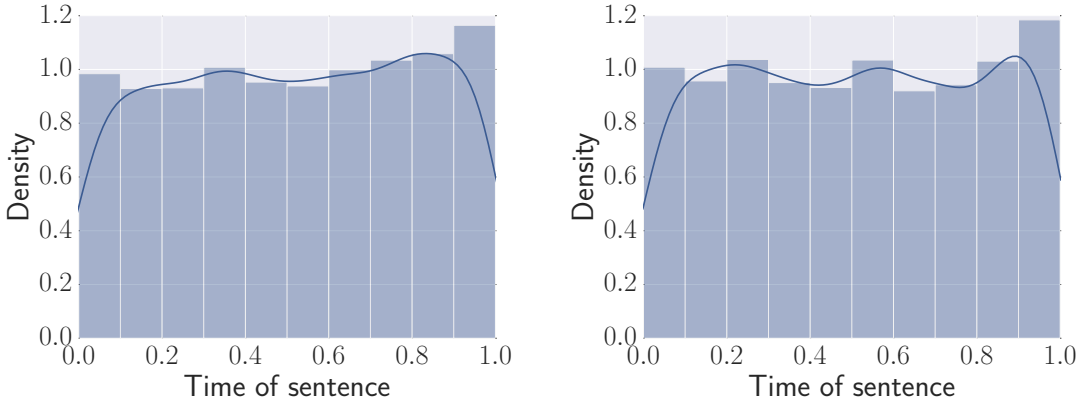
The time of word distributions for two randomly selected users from the free-text keystroke dataset are shown in Figure 3.9. Time of sentence distributions for the same users are shown in Figure 3.10. User A has a slightly higher concentration of events in the second half of each word, an indication that typing speed generally increases towards the end of the word, whereas the mass for User B is roughly the same in the first and second half of each word. Both users have a greater mass at the beginning and end of each word, an indication that typing speed decreases in the middle of words. The time of sentence distributions of both users has a greater mass in the second half of each sentence, an indication that typing speed increases towards the end of each sentence.

The chi-squared test for uniformity is performed for each user in the keystroke datasets, using time of word, time of sentence, and time of paragraph. The results are shown in Table



(a) User A (b) User B

Figure 3.9: Time of word distributions for randomly selected users (original figure).



(a) User A (b) User B

Figure 3.10: Time of sentence distributions for randomly selected users (original figure).

Dataset	Time of word	Time of sentence	Time of paragraph
Keystroke (fixed)	0.95	0.20	0.28
Keystroke (free)	1.00	0.80	0.89

Table 3.14: Proportion of users rejected (0.05 significance) in a chi-squared test for uniformity within the given interval. 11 degrees of freedom.

3.14. Overall, the event distributions for free-text input tend to be less uniform, with more users rejecting the null hypothesis of uniformity within each time window.

3.4 Surrogate data testing

The method of surrogate data can be used to determine whether a time series is nonrandom and/or nonlinear (Theiler et al., 1992). A discriminative test statistic is first calculated for the observed time series. The statistic is then calculated for a number of surrogate time series that exhibit behavior consistent with a null hypothesis (*e.g.* the data is random). The observed statistic is then compared to the surrogate test statistics to determine whether the null hypothesis should be rejected. The tests serve as a statistical proof by contradiction and give evidence for the alternate hypothesis in the case of rejection.

The root mean squared (RMS) nonlinear prediction error (NPE) is used as a test statistic. Computing the NPE requires phase space reconstruction of the observed time series. Embedding parameters are determined using standard techniques as described in Appendix A. Mutual information is used to determine the time lag and the method of false nearest neighbors (FNN) is used to determine the embedding dimension. The NPE is computed from the embedded vectors as described in Appendix B.

The p-value is determined by a rank statistic (Theiler et al., 1992). A biased p-value is used since a large number of tests are being performed, and there is a possibility of increased type I error using an unbiased estimator (Phipson and Smyth, 2010). Let q_0 be the statistic obtained from the original time series, and q be the values of the test statistic obtained from S surrogates. The biased p-value of a lower-tailed test is given by

$$\frac{I(q_0 > q) + 1}{S + 1}$$

where I is the indicator function and $I(q_0 < q)$ is the number of surrogates for which q is greater than q_0 (upper-tailed and two-sided tests are calculated similarly). In all tests, H_0 is rejected at the 0.05 significance level and 100 surrogates are generated for each test. A test is performed for each feature of each sample.

3.4.1 Testing for randomness

Consider the null hypothesis that an observed time series shows no temporal correlation (*i.e.* the time series can be completely described by an independent and identically distributed random variable). The results of this test indicate whether the data has any structure or appears to be uncorrelated noise. Surrogates are generated by random permutations of the original time series. This preserves the empirical distribution of the time series while destroying any temporal correlation, which is consistent with the null hypothesis. Lower-tailed tests are performed since the NPE is expected to increase for random time series.

The proportion of samples for which the null hypothesis is rejected for each feature is shown in Table 3.15. The "time of" features (hour of day, day of week, etc) follow sawtooth patterns and are largely rejected since they increase monotonically within the respective interval. The null hypothesis is rejected in 10% to 35% of time interval samples.

3.4.2 Testing for linearity

Consider the null hypothesis that the data came from a linear stochastic process (*i.e.* the data is completely described by its power spectrum). Surrogate samples are generated using the method of *Amplitude Adjusted Truncated Fourier Transform* (AATFT) (Guarin et al., 2010). This preserves both the power spectrum and amplitude distribution of the original time series, an improvement over the *Amplitude Adjusted Fourier Transform* (AAFT) (Theiler et al., 1992). A cutoff frequency of 0 Hz is used, and all tests are two-sided since it is not obvious whether the surrogate NPE would be strictly less than or greater than that of the original time series. The proportion of samples for which the null hypothesis is rejected for each feature is shown in Table 3.16. A modest proportion of time interval samples in each dataset show evidence of nonlinearity.

	Feature	Rejected
Kfi	Time interval	0.12
	Duration	0.23
Bitcoin	Time interval	0.26
	Minute of hour	0.34
	Hour of day	0.96
	Day of week	1.00
	฿ value	0.27
	Num. outputs	0.20
	Num. inputs	0.36
	Input/output balance	0.26
Visits	Time interval	0.15
	Hour of day	0.08
	Day of week	0.15
	Week of month	0.26
	Duration	0.03
	Tardiness	0.05
	Forecast	0.10
	Num. people	0.21

	Feature	Rejected
Kfr	Time interval	0.21
	Duration	0.41
Commits	Time interval	0.10
	Day of week	0.22
	Week of month	0.55
	Time to commit	0.22
	Num. commits	0.08
	Num. committers	0.23
	Num. characters	0.07
	Num. words	0.06
Terror	Time interval	0.35
	Day of week	0.18
	Week of month	0.24
	Month of year	0.71
	Num. perpetrators	0.00
	Num. killed	0.18
	Num. captured	0.71
	Num. related events	0.59

Table 3.15: Proportion of samples for each feature of each dataset that reject the null hypothesis of being random. Surrogates are generated by shuffling the values in the original time series, one-sided p-values are determined by the NPE rank-statistic, and rejection is at the 0.05 level. Kfi=keystroke-fixed, Kfr=keystroke-free.

	Feature	Rejected
Kfi	Time interval	0.10
	Duration	0.12
Bitcoin	Time interval	0.07
	Minute of hour	0.27
	Hour of day	0.82
	Day of week	0.86
	฿ value	0.11
	Num. outputs	0.05
	Num. inputs	0.26
	Input/output balance	0.06
Visits	Time interval	0.08
	Hour of day	0.05
	Day of week	0.08
	Week of month	0.00
	Duration	0.05
	Tardiness	0.08
	Forecast	0.03
	Num. people	0.08

	Feature	Rejected
Kfr	Time interval	0.09
	Duration	0.25
Commits	Time interval	0.05
	Day of week	0.15
	Week of month	0.20
	Time to commit	0.08
	Num. commits	0.06
	Num. committers	0.19
	Num. characters	0.03
	Num. words	0.04
Terror	Time interval	0.12
	Day of week	0.18
	Week of month	0.12
	Month of year	0.12
	Num. perpetrators	0.00
	Num. killed	0.18
	Num. captured	0.65
	Num. related events	0.47

Table 3.16: Proportion of samples for each feature of each dataset that reject the null hypothesis of being generated by a linear stochastic process. Surrogates are generated using the AATFT algorithm with a cutoff frequency of 0 Hz, two-sided p-values are determined by the NPE rank-statistic, and rejection is at the 0.05 level. Kfi=keystroke-fixed, Kfr=keystroke-free.

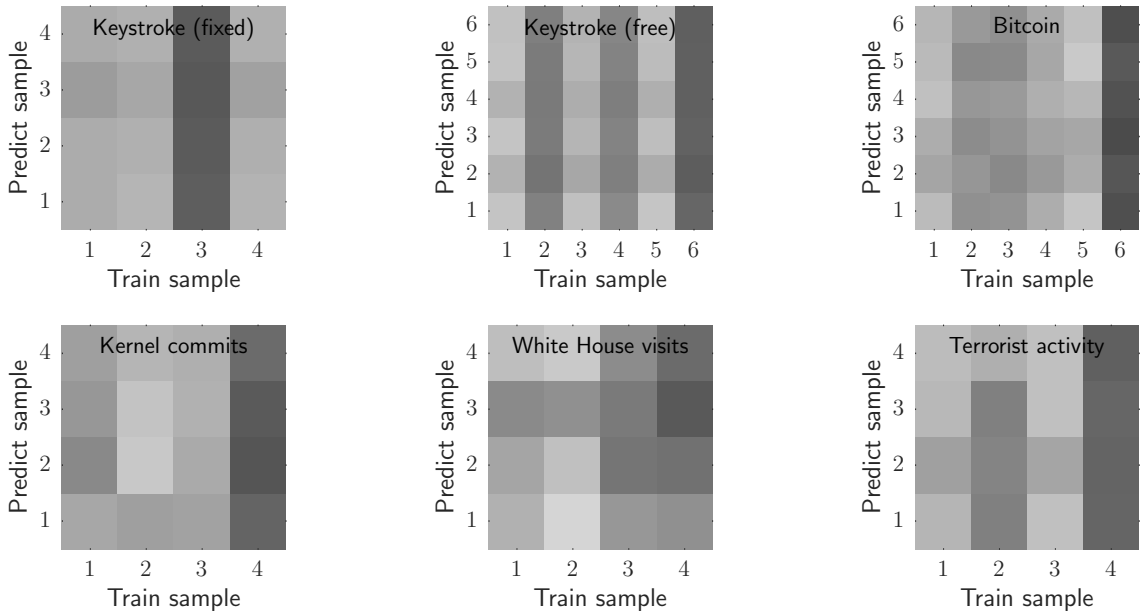


Figure 3.11: Nonlinear prediction error matrices for a randomly selected user from each dataset. In-sample predictions lie along the diagonals, and out-of-sample prediction are everywhere else. Nonstationarity is demonstrated by a change in NPE when moving away from the main diagonal (original figure).

3.5 Stationarity

The NPE can also be used to evaluate time series stationarity. Out-of-sample predictions can be made by using different samples for training and testing, as described in Appendix B. If the user's behavior is stationary, the prediction error should remain relatively constant throughout time. If the prediction error increases or decreases with time between samples, then the user's behavior shows signs of being nonstationarity (Kantz and Schreiber, 2004).

Figure 3.11 shows the time interval NPE matrix for a randomly selected user from each dataset. The PE is calculated for every combination of sample pairs within each user. In-sample prediction errors lie along the diagonal and out-of-sample prediction errors are everywhere else. The color scale is normalized to +/- 4 standard deviations of the NPE. The keystroke, Bitcoin, kernel commit, and terrorist activity users all show one sample that differs from typical behavior, as seen by the darker column. The White House visits show

a change in behavior from the first pair of samples to the second pair, as each row gets progressively darker.

3.6 Summary

The time intervals analyzed in this chapter show evidence of being nonrandom and heavy-tailed. Based on the findings, model development should make two considerations.

Underlying system state. Time intervals from human actions are typically bursty, with many short intervals separated by few long ones. Event times are also generally not uniformly distributed in time. In the simplest realization, a user can be in one of two states corresponding to active and passive states, and the resulting time intervals are dependent on the underlying system state.

The type of event performed. There is typically more than one type of event that can be performed in each dataset. For example, different keys can be pressed on a keyboard, and Bitcoin transactions can be outgoing or incoming. The effort and control over each type of event varies and influences the time to execute the event. Therefore, the resulting time intervals should be dependent on the type of event performed.

Chapter 4

Partially observable hidden Markov model

Often in time interval data, there is some incomplete information about the underlying system state. This usually comes in the form of the type of event that took place, such as which key was pressed on a keyboard. In a two-state model, the probability of being in either an active or passive state may be greater depending on which key was pressed. Certain keys, such as punctuation and the Space key, indicate a greater probability of being in a passive state as the typist often pauses between words and sentences as opposed to between letters in a word (Salthouse, 1986). This reasoning extends to other activities, such as email, where a user might be more likely to pause after sending an email instead of receiving an email, and programming, where a user may fix bugs quicker than making feature additions.

In this chapter, the partially observable hidden Markov model (POHMM) is introduced. The POHMM is a generalization of the hidden Markov model (HMM) in which the underlying system state is partially observable through event metadata at each time step. Whereas in a HMM, the hidden state (h-state) is inferred through the observed values, the hidden state in a POHMM is inferred through both the observed values and a partially observed state (p-state). This chapter is organized as follows. The HMM for human time intervals is first introduced in Section 4.1. The partially observable state is introduced in Section 4.2, followed by the derivation of likelihood, hidden state sequence, and parameter estimation

algorithms for the POHMM in Sections 4.3, 4.4, 4.5, respectively. Prediction is performed using the POHMM in Section 4.6. An example of parameter estimation for the POHMM is carried out in Section 4.7, followed by a demonstration of its consistency in Section 4.8. Finally, the POHMM is compared to similar generalizations of the HMM and concluding remarks are made in Section 4.9.

4.1 Hidden Markov model

The HMM was developed over several decades, dating back to 1960 when the forward-backward procedure was first introduced (Stratonovich, 1960). The theory was later made accessible to non-statisticians in the well-cited tutorial work, Rabiner (1989). The HMM soon became popular for modeling behavioral data, such as speech, handwriting, gesture, and linguistics. More recently, it has been used to model temporal behavior, such as keystroke dynamics and terrorist activity (Rodrigues et al., 2005; Raghavan et al., 2013).

A HMM is a finite-state model in which observed values at time t depend on an underlying hidden state. The hidden state may or may not correspond to some physical state of the system, depending on the application of interest. In speech, the hidden states generally have no physical correspondence, whereas the hidden states in a model of temporal behavior might correspond to the activity levels of the system (Lu and Zeng, 2012; Raghavan et al., 2013).

The system advances in discrete steps, typically with a first-order dependency between the hidden states. At the n^{th} time step t_n, a feature vector $\mathbf{x}_n$ is observed and the system can be in any one of M hidden states. Let $\mathbf{x}_1^N$ be the complete observation sequence from times 0 to T, where N is the total number of observations. The HMM is unsupervised since ground truth about the hidden state is generally not available. The latent variable z_n is introduced to represent the hidden state at time t_n. The structure of the HMM is shown in Figure 4.1.

The model starts in state j at time 0 with probability π_j and transitions from state i to state j with probability a_{ij}. The transition matrix is denoted by $\mathbf{A} = [a_{ij}]$ and starting probability vector by $\pi = [\pi_j]$. The steady state probability of being in state j is given by Π_j, where

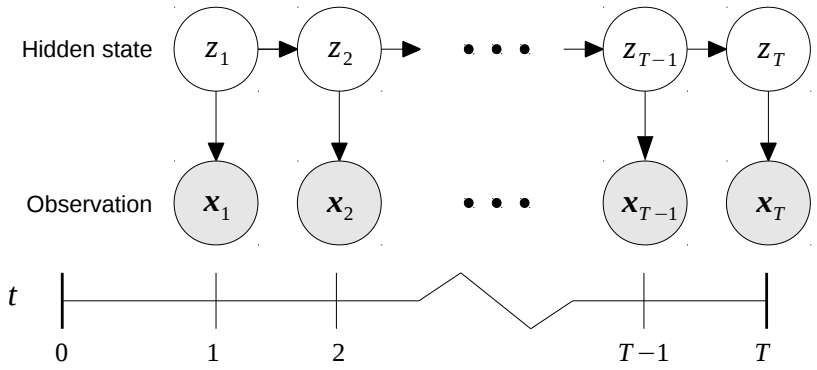


Figure 4.1: Hidden Markov model structure (original figure).

$$\Pi_j = \sum_{1 \le i \le M} \Pi_i a_{ij} \,. \tag{4.1}$$

The steady state vector Π can be determined by taking any row from repeated powers of the transition matrix,

$$\lim_{N \to \infty} \mathbf{A}$$

as the values in column j converge to the stationary probability for state j.

While in state j at time t_n, the system emits an observation vector $\mathbf{x}_n$ distributed according to some density function $f(\cdot; \mathbf{b}_j)$ parametrized by vector $\mathbf{b}_j$. The emission distribution can be either continuous, discrete, or a mix of both. The HMM is completely described by the number of states M, starting probabilities π, transition matrix $\mathbf{A}$, and emission distribution parameters $\mathbf{b}$. The model parameters are given by $\theta = \{\pi, \mathbf{A}, \mathbf{b}\}$.

There are generally three problems associated with the HMM (Rabiner, 1989), and a fourth problem is introduced here.

1. Determine $P(\mathbf{x}_1^N | \theta)$, the likelihood of an observation sequence, given model parameters θ.

2. Determine z_1^N, the maximum likelihood sequence of hidden states, given model parameters θ and an observation sequence.

3. Determine $\arg\max_{\theta\in\Theta} P(\mathbf{x}_1^N|\theta)$, the maximum likelihood parameters θ, given an observation sequence.

4. Determine $\mathsf{E}[\mathbf{x}_{N+1}|\mathbf{x}_1^N,\theta]$, the expected value of the next observation vector, given an observation sequence and model parameters θ.

The first and third problems are necessary for identifying and verifying users, while the second problem is useful in understanding user behavior. The fourth problem, although generally not considered in biometrics, has important applications in behavioral analyses such as resource optimization. The rest of this section deals with the solutions to each of these problems.

4.1.1 Model likelihood

Calculating $P(\mathbf{x}_1^N|\theta)$, the likelihood of an observation sequence $\mathbf{x}_1^N$ for a given model, is necessary for user identification and verification. Let the forward variable $\alpha_j(n)$ be the probability of the partial observation sequence $\mathbf{x}_1^n$ and state j at time t_n, given model parameters θ. This can be computed inductively by Algorithm 4.1.

Algorithm 4.1 HMM forward algorithm.

1. **Initialization**: $\alpha_j(1) = f(\mathbf{x}_1;\mathbf{b}_j)\pi_j$
2. **Induction**: $\alpha_j(n+1) = \left(\sum_{i=1}^M \alpha_i(n)a_{ij}\right) f(\mathbf{x}_{n+1};\mathbf{b}_j)$
3. **Termination**: $P(\mathbf{x}_1^N|\theta) = \sum_{j=1}^M \alpha_j(N)$

There are several ways of handling the underflow errors that will eventually occur as N increases in Algorithm 4.1 due to the floating point values becoming infinitesimally small. Intermediary values may be scaled or calculated in log-space. For long observation sequences, typically the loglikelihood is used. The order of computations required for the forward algorithm is $O(M^2N)$, since it requires M^2 calculations for each observation vector.

4.1.2 Hidden states

It is generally not necessary to know the sequence of hidden states to perform user identification and verification, although this function is useful in other applications. This requires estimating z_1^N, the most likely sequence of hidden states, given the observation sequence $\mathbf{x}_1^N$ and parameters θ. This is accomplished by the Viterbi algorithm, a dynamic programming algorithm that determines the most likely hidden state at each time step t_n.

Similar to the forward variable in the previous section, the backward variable β is introduced here where $\beta_j(n)$ is the probability of the partial observation sequence $\mathbf{x}_{n+1}^N$ and state j at time t_n, given the model parameters θ. Like the forward algorithm, the backward algorithm is $O(M^2N)$, shown in 4.2.

Algorithm 4.2 HMM backward algorithm.

1. **Initialization**: $\beta_j(N) = 1$
2. **Induction**: $\beta_i(n) = \sum_{j=1}^{M} a_{ij} f(\mathbf{x}_{n+1}; \mathbf{b}_j)\beta_j(n+1)$
3. **Termination**: $P(\mathbf{x}_1^N|\theta) = \sum_{j=1}^{M} \beta_j(1)\pi_j$

With both the forward and backward variables, it is straightforward to calculate the posterior probability of being in state j at time t_n, given the observation sequence and model parameters. This is given by the forward-backward variable $\gamma_j(n)$,

$$\gamma_j(n) = \frac{\alpha_j(n)\beta_j(n)}{P(\mathbf{x}_1^N|\theta)} = \frac{\alpha_j(n)\beta_j(n)}{\sum_{i=1}^{M} \alpha_i(n)\beta_i(n)} \tag{4.2}$$

where $1 \leq n \leq N$. The most likely hidden state at time t_n can then be determined by

$$z_n = \arg\max_{1 \leq j \leq M} \gamma_j(n) \; . \tag{4.3}$$

4.1.3 Parameter estimation

Parameter estimation is one of the most important problems associated with the HMM. The goal is to determine $\arg\max_{\theta \in \Theta} P(\mathbf{x}_1^N|\theta)$, the maximum likelihood (ML) parameters,

given observed data $\mathbf{x}_1^N$. This requires estimating the starting probabilities, transition probabilities, and emission distribution parameters. The starting and transition parameters have closed-form solutions, while the formula for the emission distribution parameters depends on the density function. There are closed-form solutions for multinomial, discrete binomial, Poisson, Gaussian, and Gaussian mixture distributions, among others (Couvreur, 1996).

Let $\xi_{ij}(n)$ be the probability of transitioning from state i at time t_n to state j at time t_{n+1}, given the observation sequence and model parameters θ. This is can be determined using the forward and backward variables, transition probability, and emission density, given by

$$\xi_{ij}(n) = \frac{\alpha_i(n)a_{ij}f(\mathbf{x}_{n+1};\mathbf{b}_j)\beta_j(n+1)}{P(\mathbf{x}_1^N|\theta)} = \frac{\alpha_i(n)a_{ij}f(\mathbf{x}_{n+1};\mathbf{b}_j)\beta_j(n+1)}{\sum_{k=1}^{M}\alpha_k(n)\beta_k(n)} \tag{4.4}$$

where $1 \leq n \leq N-1$. Like the other variables, $\xi_{ij}(n)$ can be computed in $O(M^2N)$ time and stored in a $M \times M \times N-1$ matrix. Note that since $\xi_{ij}(n)$ is the probability of transitioning from state i at time t_n to state j at time t_{n+1}, the relation

$$\gamma_i(n) = \sum_{j=1}^{M} \xi_{ij}(n)$$

holds. Summing $\gamma_j(n)$ over t_n gives the expected number of transitions from state j, while summing $\xi_{ij}(n)$ over t_n gives the expected number of transitions from i to j.

The variables $\gamma_j(n)$ and $\xi_{ij}(n)$ can then be used to update the parameters. The new starting probabilities are determined directly from $\gamma_j(1)$,

$$\dot{\pi}_j = \gamma_j(1)\,. \tag{4.5}$$

The transition matrix is updated by the formula

$$\dot{a}_{ij} = \frac{\sum_{n=1}^{N-1}\xi_{ij}(n)}{\sum_{n=1}^{N-1}\gamma_i(n)} \tag{4.6}$$

and the updated steady state probabilities are given by

$$\dot{\Pi}_j = \frac{\sum_{n=1}^{N} \gamma_j(n)}{\sum_{i=1}^{M} \sum_{n=1}^{N} \gamma_i(n)} \,. \tag{4.7}$$

The re-estimates for parameter vectors $\mathbf{b}_j$, $1 \leq j \leq M$, depend on the density function $f(\cdot)$, where

$$\dot{\mathbf{b}}_j = \arg\max_{\mathbf{b} \in \mathscr{B}} \sum_{n=1}^{N} \gamma_j(n) \ln f(\mathbf{x}_n; \mathbf{b}) \tag{4.8}$$

and $\mathscr{B}$ is the parameter space of $f(\cdot)$. The complete parameter estimation procedure, commonly referred to as the Baum-Welch (BW) algorithm, is shown in Algorithm 4.3.

Algorithm 4.3 HMM Baum-Welch algorithm for parameter estimation.

1. **Initialization**
   Choose initial parameters θ^0 and let $\dot{\theta} \leftarrow \theta^0$
2. **Expectation**
   Use $\dot{\theta}$ and $\mathbf{x}_1^N$ to compute $\alpha_j(n)$, $\beta_j(n)$, $\gamma_j(n)$, $\xi_{ij}(n)$, and let $\dot{P} \leftarrow P(\mathbf{x}_1^N|\theta)$
3. **Maximization**
   Update π, $\mathbf{A}$, and $\mathbf{b}$ using the re-estimation formulae and let $\dot{\theta} \leftarrow \{\dot{\pi}, \dot{\mathbf{A}}, \dot{\mathbf{b}}\}$
4. **Termination**
   If $P(\mathbf{x}_1^N|\dot{\theta}) - \dot{P} < \varepsilon$ then terminate and let $\hat{\theta} \leftarrow \dot{\theta}$, otherwise go to step 2

4.1.4 Prediction

An important problem in many behavioral analyses is predicting future actions. For the HMM, the goal is to estimate $\mathsf{E}[\mathbf{x}_{N+1}|\mathbf{x}_1^N, \theta]$, the expected value of the next observation, given an observation sequence and model parameters. When applied to time intervals, this gives a prediction of the next event time.

The observation vector can be predicted by taking the expected observation in each state weighted by the likelihood of being in each state. This is given by

$$\hat{\mathbf{x}}_{N+1} = \sum_{j=1}^{M} P(z_{N+1} = j|\mathbf{x}_1^N) \mathsf{E}[\mathbf{x}_{N+1}|z_{N+1} = j] \tag{4.9}$$

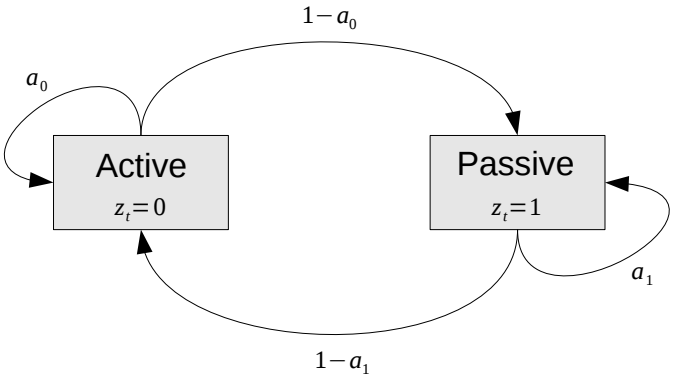


Figure 4.2: Two-state hidden Markov model (original figure).

where

$$P(z_{N+1} = j|\mathbf{x}_1^N) = \frac{\sum_{i=1}^{M} a_{ij}\alpha_i(N)}{\sum_{i=1}^{M} \alpha_i(N)}$$

and $\alpha_i(N)$ is determined 4.1. The expected observation vector in state j, $\mathsf{E}[\mathbf{x}_{N+1}|z_{N+1} = j]$, is determined directly by the density $f(\cdot;\mathbf{b}_j)$.

4.1.5 Hidden Markov model for time intervals

A two-state HMM is defined where states correspond to the active and passive states of the user. The resulting model is shown in Figure 4.2.

As seen in Chapter 3, human-generated time intervals are well described by a log-normal distribution. This serves as motivation for choosing the log-normal as the density function for time intervals τ, given by

$$f(\tau;\eta,\rho) = \frac{1}{\tau\rho\sqrt{2\pi}} \exp\left(\frac{-(\ln\tau - \eta)^2}{2\rho^2}\right) \tag{4.10}$$

where η is the log-mean and ρ is the log-standard deviation. Note that a mixture of log-normals also gives rise to a log-normal. The log-normal parameter re-estimation formulae are given by

$$\dot{\eta}_j = \frac{\sum_{n=1}^{N} \gamma_j(n) \ln\tau_n}{\sum_{n=1}^{N} \gamma_j(n)}$$

and

$$\dot{\rho}_j^2 = \frac{\sum_{n=1}^{N} \gamma_j(n)(\ln \tau_n - \dot{\eta}_j)^2}{\sum_{n=1}^{N} \gamma_j(n)}$$

respectively. A normal distribution is used for additional numerical features, where the normal density is defined as

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \tag{4.11}$$

Parameters μ and σ are re-estimated similarly by

$$\dot{\mu}_j = \frac{\sum_{n=1}^{N} \gamma_j(n) x_n}{\sum_{n=1}^{N} \gamma_j(n)} \tag{4.12}$$

and

$$\dot{\sigma}_j^2 = \frac{\sum_{n=1}^{N} \gamma_j(n)(x_n - \dot{\mu}_j)^2}{\sum_{n=1}^{N} \gamma_j(n)} \tag{4.13}$$

where x_n is a scalar component of $\mathbf{x}_n$. Note that this definition assumes independence between features since covariance terms are not considered.

4.2 Partially observable states

In time interval data, there is often some additional information about the event that gives an indication as to the hidden state. This could be the name of a key that was pressed while typing, the type of commit made to a source code repository, or some other symbol that indicates the type of action that occurred. Certain types of actions, or events, indicate a greater probability of being in a particular hidden state. For example in keystroke dynamics, the probability of being in a passive state should be greater when the Space key is pressed than any letter key. The key name is a *partially observed state*, or *p-state* for short, which partially reveals the hidden state of the system. This phenomenon has been observed in transcription typists (Salthouse, 1986), however the HMM defined in the previous section cannot account for the difference probability due to the type of event that occurred.

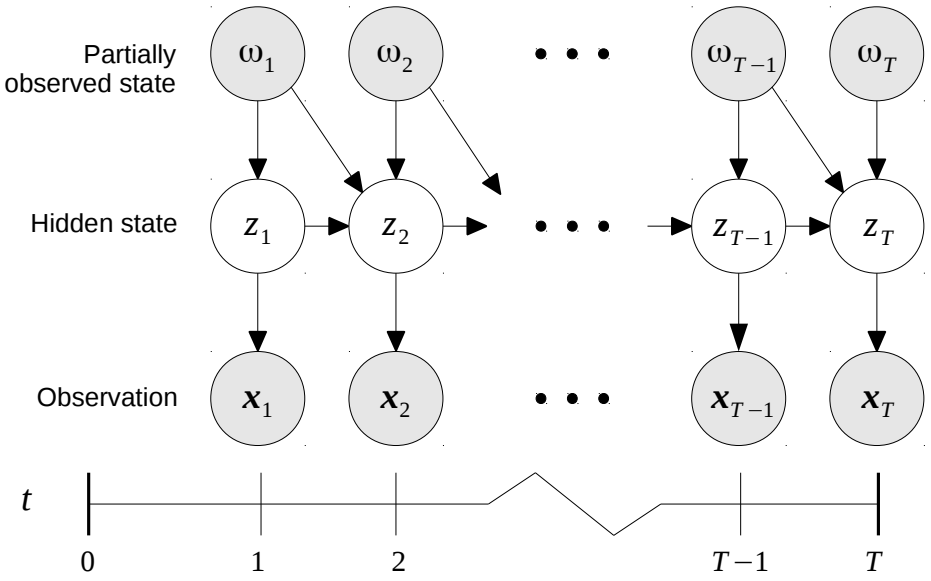


Figure 4.3: Partially observable hidden Markov model structure (original figure).

Let ω be a p-state symbol that comes from a finite alphabet of size m. In the partially observable hidden Markov model, the true system state is a latent variable and depends on both the p-state ω and the previous hidden state. The POHMM dependency structure is shown in Figure 4.3.

The p-state introduces a dependence into the HMM parameters defined in the previous section. Specifically, model parameters include $\pi_{j|\omega}$, the probability of starting in state j, given p-state ω, and $\mathbf{b}_{j|\omega}$ for the parametrized emission distribution $f(\cdot;\mathbf{b}_{j|\omega})$ that depends on p-state ω. Similarly, the probability of transitioning between states i and j, given p-states ω and ψ, is denoted by $a_{ij|\omega,\psi}$. A summary of POHMM parameters and variables is shown in Table 4.1.

Marginal distribution parameters can also be defined, where the p-state is marginalized out. Let $\pi_{j|.}$ and $f(\cdot;\mathbf{b}_{j|.})$ be the marginalized starting probability and emission probability, respectively. Similarly, the parameters $a_{ij|\omega.}$, $a_{ij|.\omega}$, and $a_{ij|..}$ are defined as the transition probabilities after marginalizing out the second, first, and both p-states, respectively. Computation of POHMM marginal distributions is covered in Section 4.5.2.

While the total number of parameters in a HMM is $M + M^2 + MK$, where K is the number of free parameters in the emission distribution $f(\cdot)$, the POHMM contains

Parameter or Variable	Description
$\mathbf{x}_1^N$	Sequence of observed values, where $\mathbf{x}_n$ may be a scalar or a vector
Ω_1^N	Sequence of partially observable states, where the length of Ω_1^N is N and Ω_n is the p-state at time t_n
m	The number of unique p-states in Ω_1^N
ω or ψ	Partially observed state (p-state)
z_n	Hidden state of the system at time t_n
M	The number of hidden states
$a_{ij\|\omega,\psi}$	Probability of transitioning from state i to state j, given observed p-state ω while in state i and ψ in state j
$\pi_{j\|\omega}$	Probability of being in state j at time t_1, given observed p-state ω
$\Pi_{j\|\omega}$	Steady state probability of being in state j, given p-state ω
$\mathbf{b}_{j\|\omega}$	Observation distribution parameters in state j, given p-state ω
$\gamma_{j\|\omega}(n)$	Posterior probability of being in state j, given observations and p-state ω at time t_n
$\xi_{ij\|\omega,\psi}(n)$	Posterior probability of transitioning from state i at time t_n to state j at time t_{n+1}, given p-state ω and ψ at times t_n and t_{n+1}, respectively

Table 4.1: Summary of POHMM parameters and variables.

$m \times (M + mM^2 + MK)$ parameters. (Note that this does not include marginal distribution parameters). After accounting for normalization constraints, the number of degrees of freedom (*dof*) is $m \times (M - 1 + mM(M-1) + MK)$, corresponding to the number of free parameters. This will reduce further after parameter smoothing strategies are applied, as discussed in Section 4.5.3.

4.3 Model likelihood

The POHMM likelihood is computed by a modified forward procedure, similar to the HMM. At each time step t_n, the system emits a partially observable state Ω_n and is in 1 out of M hidden states. The model likelihood is given by $P(\mathbf{x}_1^N|\theta,\Omega_1^N)$, which is the probability of the observation sequence, given both the model parameters θ and the sequence of p-states Ω_1^N. This can be calculated using the conditional model parameters $\pi_{j|\omega}$, $a_{ij|\omega,\psi}$,

and $\mathbf{b}_{j|\omega}$ for hidden states $1 \leq j \leq M$ and p-states $\omega, \psi \in \Omega_1^N$. The modified forward algorithm and backward algorithm for the POHMM are shown in Algorithms 4.4 and 4.5, respectively.

The model likelihood may be obtained by either algorithm upon termination. Note that like the HMM, the modified forward and backward algorithms both take $O(M^2N)$ time to compute. The forward variable $\alpha_{j|\Omega_n}(n)$ is the probability of observing the sequence $\mathbf{x}_1^n$ and being in state j at time t_n, given the model parameters θ and the observed p-state sequence Ω_1^n with p-state Ω_n at time t_n. Similarly, the backward variable $\beta_{j|\Omega_n}(n)$ is the the probability of the observing the sequence $\mathbf{x}_n^N$ and being in state j at time t_n, given the model parameters θ and the observed p-state sequence Ω_n^N with p-state Ω_n at time t_n.

Algorithm 4.4 POHMM forward algorithm.

1. **Initialization**: $\alpha_{j|\Omega_1}(1) = f(\mathbf{x}_1; \mathbf{b}_{j|\Omega_1})\pi_{j|\Omega_1}$
2. **Induction**: $\alpha_{j|\Omega_{n+1}}(n+1) = \left(\sum_{i=1}^{M} \alpha_{i|\Omega_n}(n) a_{ij|\Omega_n,\Omega_{n+1}}\right) f(\mathbf{x}_{n+1}; \mathbf{b}_{j|\Omega_{n+1}})$
3. **Termination**: $P(\mathbf{x}_1^N|\theta, \Omega_1^N) = \sum_{j=1}^{M} \alpha_{j|\Omega_T}(N)$

Algorithm 4.5 POHMM backward algorithm.

1. **Initialization**: $\beta_{j|\Omega_T}(N) = 1$
2. **Induction**: $\beta_{i|\Omega_n}(n) = \sum_{j=1}^{M} a_{ij|\Omega_n,\Omega_{n+1}} f(\mathbf{x}_{n+1}; \mathbf{b}_{j|\Omega_n}) \beta_{j|\Omega_{n+1}}(n+1)$
3. **Termination**: $P(\mathbf{x}_1^N|\theta, \Omega_1^N) = \sum_{j=1}^{M} \beta_{j|\omega}(1)\pi_{j|\omega}$

4.4 Hidden states

Determination of the most likely sequence of hidden states proceeds similar to the HMM, using the p-state dependent parameters. First, the posterior probability of being in state j at time t_n, given observed p-state Ω_n, is defined using the POHMM forward and backward variables

$$\gamma_{j|\Omega_n}(n) = \frac{\alpha_{j|\Omega_n}(n)\beta_{j|\Omega_n}(n)}{P(\mathbf{x}_1^N|\theta,\Omega_1^N)} = \frac{\alpha_{j|\Omega_n}(n)\beta_{j|\Omega_n}(n)}{\sum_{i=1}^{M}\alpha_{i|\Omega_n}(t)\beta_{i|\Omega_n}(n)} \ . \tag{4.14}$$

Hidden states are then taken as the maximum likelihood states at each time step,

$$z_n = \arg\max_{1\leq j\leq M}\gamma_{j|\Omega_n}(n) \ . \tag{4.15}$$

4.5 Parameter estimation

Parameter estimation is similar to the HMM, where the POHMM uses a modified Baum-Welch algorithm. From the modified forward-backward variable given by Equation 4.14, the POHMM starting probabilities are

$$\dot{\pi}_{j|\omega} = \gamma_{j|\Omega_1}(1) \qquad \text{where } \omega = \Omega_1 \ . \tag{4.16}$$

Generally, it may not be possible to estimate $\dot{\pi}_{j|\omega}$ for many ω due to there only being one Ω_1 (or several Ω_1 for multiple observation sequences). To deal with this, parameter smoothing is introduced in Section 4.5.3.

To complete the equations for updating parameters in the modified Baum-Welch algorithm, the POHMM analogue of $\xi_{ij}(t)$ defined for the HMM in Equation 4.4 is needed. Let $\xi_{ij|\Omega_n,\Omega_{n+1}}(n)$ be the probability of transitioning from state i at time t_n to state j at time t_{n+1}, given the observed p-states Ω_n at time t_n and Ω_{n+1} at time t_{n+1} as well as the observation sequence and model parameters θ. Using the POHMM forward and backward variables, this is given by

$$\xi_{ij|\Omega_n,\Omega_{n+1}}(n) = \frac{\alpha_{i|\Omega_n}(n)a_{ij|\Omega_n,\Omega_{n+1}}f(\mathbf{x}_{n+1};\mathbf{b}_{j|\Omega_{n+1}})\beta_{j|\Omega_{n+1}}(n+1)}{P(\mathbf{x}_1^N|\theta,\Omega_1^N)} \qquad 1\leq n\leq N-1 \ . \tag{4.17}$$

Note that computing $\xi_{ij|\omega,\psi}(n)$ for the POHMM can be performed in linear time in the number of observations, $O(M^2N)$, since the p-states are not enumerated at each time step.

Next, the transition probabilities are estimated. In contrast to the HMM, which has M^2 transition probabilities, there are m^2M^2 transition probabilities in the POHMM. Computing the updated transition probabilities can be performed in $O(M^2N)$ time. The equation is

$$\dot{a}_{ij|\omega,\psi} = \frac{\sum_{n\in\mathscr{T}_{\omega,\psi}} \xi_{ij|\Omega_n,\Omega_{n+1}}(n)}{\sum_{n\in\mathscr{T}_{\omega,\psi}} \gamma_{i|\Omega_n}(n)} \qquad \mathscr{T}_{\omega,\psi} = \{n|\Omega_n = \omega, \Omega_{n+1} = \psi\} \tag{4.18}$$

where $\mathscr{T}_{\omega,\psi} = \{n|\Omega_n = \omega, \Omega_{n+1} = \psi\}$ is the set of indexes where the p-state ω is observed at time t_n and ψ is observed at time t_{n+1}. Note that $\dot{a}_{ij|\omega,\psi}$ requires only the transitions between p-states ω and ψ.

Estimating the emission distribution parameters depends on the density function $f(\cdot)$, where the new estimates are given by

$$\dot{\mathbf{b}}_{j|\omega} = \arg\max_{\mathbf{b}\in\mathscr{B}} \sum_{n\in\mathscr{T}_\omega} \gamma_{j|\Omega_n}(n) \ln f(\mathbf{x}_n;\mathbf{b}) \qquad \mathscr{T}_\omega = \{n|\Omega_n = \omega\} \tag{4.19}$$

As an example, the normal emission distribution parameters can also be estimated for the POHMM, conditional on the p-state. Using the forward-backward variable from Equation 4.14,

$$\dot{\mu}_{j|\omega} = \frac{\sum_{n\in\mathscr{T}_\omega} \gamma_{j|\Omega_n}(n)\mathbf{x}_n}{\sum_{n\in\mathscr{T}_\omega} \gamma_{j|\Omega_n}(n)} \qquad \mathscr{T}_\omega = \{n|\Omega_n = \omega\} \tag{4.20}$$

and

$$\dot{\sigma}^2_{j|\omega} = \frac{\sum_{n\in\mathscr{T}_\omega} \gamma_{j|\Omega_n}(n)(\mathbf{x}_n - \dot{\mu}_{j|\omega})^2}{\sum_{n\in\mathscr{T}_\omega} \gamma_{j|\Omega_n}(n)} \qquad \mathscr{T}_\omega = \{n|\Omega_n = \omega\} \tag{4.21}$$

are the updated mean and variance estimates for hidden state j, given p-state ω. Note that the estimates depend only on the elements of $\gamma_{\Omega_n}(n)$ where $\Omega_n = \omega$. The log-normal POHMM emission parameters are re-estimated similarly.

The modified Baum-Welch algorithm, which uses Equations 4.16, 4.18, and 4.19 to update model parameters in each iteration, is shown in Algorithm 4.6. The convergence criterion is a threshold ε on the loglikelihood reduction. The rest of this section deals with

other aspects of parameter estimation, including initialization, marginal distributions, and parameter smoothing.

Algorithm 4.6 POHMM modified Baum-Welch algorithm.

1. **Initialization**
   Choose initial parameters θ^0 and let $\dot{\theta} \leftarrow \theta^0$
2. **Expectation**
   Use $\dot{\theta}$, $\mathbf{x}_1^N$, Ω_1^N to compute $\alpha_{j|\Omega_n}(n)$, $\beta_{j|\Omega_n}(t)$, $\gamma_{j\Omega_n}(n)$, $\xi_{ij\Omega_n,\Omega_{n+1}}(n)$, let $\dot{P} \leftarrow P(\mathbf{x}_1^N|\theta,\Omega_1^N)$
3. **Maximization**
   Update π, $\mathbf{A}$, and $\mathbf{b}$ using the re-estimation formulae and let $\dot{\theta} \leftarrow \{\dot{\pi},\dot{\mathbf{A}},\dot{\mathbf{b}}\}$
4. **Termination**
   If $P(\mathbf{x}_1^N|\dot{\theta},\Omega_1^N) - \dot{P} < \varepsilon$ then terminate and let $\hat{\theta} \leftarrow \dot{\theta}$, otherwise go to step 2

4.5.1 Parameter initialization

Parameter initialization is an important step in the Baum-Welch algorithm and may ultimately determine the quality of the estimated parameters. Initial parameters can either be generated randomly or fixed according to some rules. In this work, a fixed parameter initialization strategy is used, which guarantees reproducible parameter estimates. The problem of initializing a random POHMM is covered in Section 4.8.

The starting and transition probabilities are simply initialized as

$$\pi_{j|\omega} = \frac{1}{M}$$

and

$$a_{ij|\omega,\psi} = \frac{1}{M}$$

for all i, j, ω, and ψ. This reflects the belief of equal probabilities in the absence of any particular ω in Ω_1^N.

Next, the emission density parameters are initialized. The strategy proposed here is to initialize parameters in such a way that there is a correspondence between hidden states from two different models. That is, for any two models A and B, hidden state $j = 1$ corresponds to the active state and $j = 2$ corresponds to the passive state. Using a log-normal emission distribution for time intervals, this is accomplished by spreading the log-mean initial parameters. Let

$$\eta_\omega = \frac{\sum_{n \in \mathscr{T}_\omega} \ln \mathbf{x}_n}{|\mathscr{T}_\omega|} \qquad \mathscr{T}_\omega = \{n | \Omega_n = \omega\}$$

and

$$\rho^2_\omega = \frac{\sum_{n \in \mathscr{T}_\omega} (\ln \mathbf{x}_n - \eta_{j|\omega})^2}{|\mathscr{T}_\omega|} \qquad \mathscr{T}_\omega = \{n | \Omega_n = \omega\}$$

be the log-mean and log-variance of observations $\mathbf{x}_1^N$ conditional on the p-state ω. The model parameters are then initialized as

$$\eta_{j|\omega} = \eta_\omega + \left(\frac{2h(j-1)}{M-1} - h \right) \times \rho_\omega$$

and

$$\rho^2_{j|\omega} = \rho^2_\omega$$

for $1 \leq j \leq M$, where h is a bandwidth parameter that determines the size of the spread. In a two-state model, this ensures that state $j = 1$ corresponds to the active state, *i.e.* the state with the smallest log-mean time interval.

4.5.2 Marginal distributions

In order for the POHMM to handle missing or novel p-states during likelihood calculation, the marginal distributions are calculated. When computing the likelihood of a novel sequence, it is possible that some p-states in the novel sequence were not observed during parameter estimation. For example, this situation can occur when p-states correspond to key names of freely-typed text and novel keys are observed during testing. A fallback

mechanism (sometimes referred to as a "backoff" model) is typically employed to handle missing data during training and novel data during testing, such as those used in keystroke (Monaco et al., 2013; Tappert et al., 2010) and linguistics (Jurafsky et al., 2000). The POHMM marginal distributions, where the p-state is marginalized out, are analogous to a two-level fallback hierarchy where missing or novel p-states fall back to the marginals.

Let the probability of observing p-state ω at time t_1 be π_ω, and the probability of transitioning from p-state ω to ψ be denoted by $a_{\omega,\psi}$. Both of these can be computed directly from the p-state sequence Ω_1^N. The marginal probability $\pi_{j|.}$ is the probability of starting in hidden state j, in which the p-state has been marginalized out, given by

$$\pi_{j|.} = \sum_{\omega \in \Omega} \pi_{j|\omega} \pi_\omega \tag{4.22}$$

where Ω is the set of unique p-states in Ω_1^N.

Marginal transition probabilities can also be calculated. Let $a_{ij|\omega.}$ be the probability of transitioning from hidden state i to hidden state j, given the observed p-state ω while in hidden state i. The second p-state for hidden state j has been marginalized out. This probability is given by

$$a_{ij|\omega.} = \sum_{\psi \in \Omega} a_{ij|\omega,\psi} a_{\omega,\psi} \tag{4.23}$$

The marginal probability $a_{ij|.\omega}$ is defined similarly by

$$a_{ij|.\psi} = \frac{\sum_{\omega \in \Omega} a_{ij|\omega,\psi} a_{\omega,\psi}}{\sum_{\omega \in \Omega} a_{\omega,\psi}} \tag{4.24}$$

Finally, the marginal $a_{ij|..}$ is the probability of transitioning from hidden state i to j,

$$a_{ij|..} = \frac{1}{m} \sum_{\omega \in \Omega} \sum_{\psi \in \Omega} a_{ij|\omega,\psi} a_{\omega,\psi} \tag{4.25}$$

No denominator is needed in Equation 4.23 since the normalization constraints of both transition matrices carry over to the left-hand side. Equation 4.25 is normalized by $\frac{1}{m}$ since $\sum_{\omega \in \Omega} \sum_{\psi \in \Omega} a_{\omega,\psi} = m$ where m is the number of unique p-states in Ω.

The marginal emission distribution is a mixture of the emissions of the partially observed states. For a normal or log-normal emission, the marginal emission is simply a mixture of normals or log-normals, respectively. Let $\mu_{j|.}$ and $\sigma^2_{j|.}$ be the mean and variance of the marginal distribution for hidden state j. The marginal mean is a weighted sum of the p-state distributions, given by

$$\mu_{i|.} = \sum_{\omega \in \Omega} \Pi_\omega \mu_{i|\omega} \tag{4.26}$$

where Π_ω is the stationary probability of observing p-state ω. This can be calculated directly from the p-state sequence Ω_1^T, where

$$\Pi_\omega = \frac{1}{N} \sum_{n=1}^{N} I\left(\Omega_n = \omega\right)$$

and $I\left(\cdot\right)$ is the indicator function. Similarly, the marginal covariance is given by

$$\sigma^2_{j|.} = \sum_{\omega \in \Omega} \Pi_\omega \left[\left(\mu_{j|\omega} - \mu_{j|.}\right)^2 + \sigma^2_{j|\omega} \right] . \tag{4.27}$$

Calculation of the marginalized log-normal distribution parameters is exactly the same.

4.5.3 Parameter smoothing

While the marginal distributions can be used to handle missing or novel data during likelihood calculation, parameter smoothing handles missing or infrequent data during parameter estimation. The purpose of parameter smoothing is twofold. First, it reduces the *dof* in the POHMM to avoid overfitting, a problem often encountered when there is a large number of parameters and small amount of data. Second, parameter smoothing provides superior estimates in the case of missing or infrequent data. For motivation, consider a keystroke letter sequence of length N. There are at most 27 unique keys that can be observed, including the space key, and 27×27 unique digrams (subsequences of length 2). Most of these will rarely, or never, be observed in a typing sample of English text. It is possible that some sequences encountered in the testing phase were not observed during

parameter estimation. The POHMM handles data sparsity by smoothing the parameters dependent on the various p-states for a given hidden state j.

After parameter smoothing, each parameter in the POHMM becomes a weighted average with the corresponding marginal parameters. There are several different weighting strategies that can be used. An inverse frequency strategy uses the p-state frequencies to define the weights as

$$w_\omega = 1 - \frac{1}{1+f(\omega)}$$

where $f(\omega) = \sum_{t=1}^{N} I(\Omega_n = \omega)$, the frequency of p-state ω in the observed sequence Ω_1^N. Alternatively, the stationary probabilities can be used as the weights, where

$$w_\omega = \Pi_\omega \ .$$

There are other possibilities for assigning a weight for each p-state, such as using fixed weights.

The POHMM starting probabilities are smoothed by

$$\pi_{j|\omega} = w_\omega \pi_{j|\omega} + (1 - w_\omega)\pi_{j|.}$$

for each $\omega \in \Omega$. This ensures that the starting probability conditioned on infrequent or missing p-states is estimated using some knowledge from the marginalized starting probability. Similarly, emission parameters are smoothed by

$$\mathbf{b}_{j|\omega} = w_\omega \mathbf{b}_{j|\omega} + (1 - w_\omega)\mathbf{b}_{j|.}$$

The smoothing weights for transition probabilities follow similar formulae. Let $f(\omega, \psi)$ be the frequency of p-state state ω followed by ψ in the observed sequence Ω_1^T. Weights for the conditional and marginal transition probabilities are defined as

$$
\begin{aligned}
w_{\omega.} &= \frac{1}{f(\omega,\psi)+f(\psi)} \\
w_{.\psi} &= \frac{1}{f(\omega,\psi)+f(\omega)} \\
w_{\omega,\psi} &= 1-\left(\frac{1}{f(\omega,\psi)+f(\omega)}+\frac{1}{f(\omega,\psi)+f(\psi)}\right) \\
w_{..} &= 0
\end{aligned}
$$

where $w_{\omega,\psi}+w_{\omega.}+w_{.\psi}+w_{..}=1$. The smoothed parameter transition matrix is given by

$$a_{ij|\omega,\psi} = w_{\omega,\psi}a_{ij|\omega,\psi} + w_{\omega.}a_{ij|\omega.} + w_{.\psi}a_{ij|.\psi} + w_{..}a_{ij|..}$$

In this strategy, the weight for the marginal $a_{ij|..}$ is 0, although in other strategies this could be non-zero. The parameter estimation procedure, including marginal calculations and parameter smoothing, is given by Algorithm 4.7.

Algorithm 4.7 POHMM parameter estimation.

1. **Initialization**
   Choose initial parameters θ^0 and let $\dot{\theta} \leftarrow \theta^0$
2. **Expectation**
   Use $\dot{\theta}$, $\mathbf{x}_1^N$, Ω_1^N to compute $\alpha_{j|\Omega_n}(n)$, $\beta_{j|\Omega_n}(n)$, $\gamma_{j\Omega_n}(n)$, $\xi_{ij\Omega_n,\Omega_{n+1}}(n)$, let $\dot{P} \leftarrow P(\mathbf{x}_1^N|\theta,\Omega_1^N)$
3. **Maximization**
   Update π, $\mathbf{A}$, and $\mathbf{b}$ using the re-estimation formulae and let $\dot{\theta} \leftarrow \{\dot{\pi},\dot{\mathbf{A}},\dot{\mathbf{b}}\}$
4. **Marginal distributions**
   Calculate marginal distributions
5. **Parameter smoothing**
   Calculate smoothing weights and smooth the parameters with marginals
6. **Termination**
   If $P(\mathbf{x}_1^N|\dot{\theta},\Omega_1^N)-\dot{P}<\varepsilon$ then terminate and let $\hat{\theta} \leftarrow \dot{\theta}$, otherwise go to step 2

4.6 Prediction

Predictions can be made by the POHMM when the future p-states are either known or unknown. When the future p-state is known,

$$\hat{\mathbf{x}}_{N+1} = \sum_{j=1}^{M} P(z_{N+1} = j|\mathbf{x}_1^N, \Omega_1^{N+1}) \mathsf{E}[\mathbf{x}_{N+1}|z_{N+1} = j, \Omega_{N+1} = \omega] \tag{4.28}$$

where

$$P(z_{N+1} = j|\mathbf{x}_1^N, \Omega_1^{N+1}) = \frac{\sum_{i=1}^{M} a_{ij|\Omega_N,\Omega_{N+1}} \alpha_{i|\Omega_N}(N)}{\sum_{i=1}^{M} \alpha_{i|\Omega_N}(N)}$$

since ω is known at time $N+1$. Otherwise, when the future p-state is unknown,

$$\hat{\mathbf{x}}_{N+1} = \sum_{\omega \in \Omega} \sum_{j=1}^{M} P(\Omega_{N+1} = \omega, z_{N+1} = j|\mathbf{x}_1^N, \Omega_1^N) \mathsf{E}[\mathbf{x}_{N+1}|z_{N+1} = j, \Omega_{N+1} = \omega] \tag{4.29}$$

where

$$P(\Omega_{N+1} = \omega, z_{N+1} = j|\mathbf{x}_1^N, \Omega_1^N) = \frac{a_{\Omega_N,\omega} \sum_{i=1}^{M} a_{ij|\Omega_N,\omega} \alpha_{i|\Omega_N}(N)}{\sum_{i=1}^{M} \alpha_{i|\Omega_T}(T)} .$$

In both scenarios,

$$\mathsf{E}[\mathbf{x}_{N+1}|z_{N+1} = j, \Omega_{N+1} = \omega] = \mathsf{E}[f(\cdot; \mathbf{b}_{j|\omega})] .$$

4.7 Example

To illustrate the utility of the POHMM, a simple example is developed. This section will demonstrate parameter estimation, the calculation of marginal distributions, parameter smoothing, and likelihood calculation. Consider the univariate observation sequence,

$$\mathbf{x}_1^N = 4 \quad 3 \quad 4 \quad 2 \quad 3 \quad 1$$

where $N = 6$. Assume that the observed values come from a normal distribution and that the system can be in either an active or passive state at each time step. Although the true

state of the system is hidden at each time step, a sequence of partially observed states accompany $\mathbf{x}_1^N$. The p-states correspond to the type of action or event that took place at each time step, given by

$$\Omega_1^N = b \quad a \quad b \quad a \quad c \quad a$$

where the alphabet of partial states is $\Omega = \{a, b, c\}$. Therefore, the parameters for a POHMM will be estimated with normal emission, $M = 2$ hidden states, and $m = 3$ p-states.

4.7.1 Parameter initialization

The first step is parameter initialization. The starting and transition probabilities are initialized as

$$\pi_{i|\omega} = \frac{1}{M} = \frac{1}{2} \quad \text{for } \omega \in \Omega$$

$$a_{ij|\omega,\psi} = \frac{1}{M} = \frac{1}{2} \qquad \text{for } \omega, \psi \in \Omega\,.$$

Using a normal distribution, the emission parameters $\mu_{i|\omega}$ and $\sigma_{i|\omega}$ must be initialized for each hidden state i conditional on each partially observed state ω. The unconditional means μ_ω and standard deviations σ_ω are first determined for each p-state:

$$\mu_a = 2, \quad \mu_b = 4, \quad \mu_c = 3$$

$$\sigma_a = 2, \quad \sigma_b = 4, \quad \sigma_c = 3$$

The conditional parameters, $\mu_{i|\omega}$ and $\sigma_{i|\omega}$, are then initialized using the formula given in Section 4.5.1, where $\mu_{1|a} = \mu_a - \sigma_a$, $\mu_{2|a} = \mu_a + \sigma_a$, etc. This spreads the hidden state means, given each p-state. Marginal distributions for each hidden state can then be determined, as described in Section 4.5.2. The marginals will be used for parameter smoothing, which is performed immediately after parameter initialization.

Parameter	Before smooth	After smooth
$\mu_{1\|a}$	-0.13	0.07
$\mu_{2\|a}$	4.13	4.34
$\mu_{1\|b}$	1.87	1.48
$\mu_{2\|b}$	6.13	5.75
$\mu_{1\|c}$	0.87	0.78
$\mu_{2\|c}$	5.13	5.05

Parameter	Before smooth	After smooth
$\sigma_{1\|a}$	0.82	0.88
$\sigma_{2\|a}$	0.82	0.88
$\sigma_{1\|b}$	0.00	0.36
$\sigma_{2\|b}$	0.00	0.36
$\sigma_{1\|c}$	0.00	0.53
$\sigma_{2\|c}$	0.00	0.53

Table 4.2: POHMM example initial parameters before and after smoothing.

Marginal distributions for the emission parameters are determined by the conditional distribution parameters. These are straightforward to compute as a mixture of normals using Equations 4.26 and 4.27, where

$$\mu_{1|.} = 0.70, \qquad \mu_{2|.} = 4.97$$
$$\sigma_{1|.} = 1.07, \qquad \sigma_{2|.} = 1.07$$

4.7.2 Parameter smoothing

After parameters are initialized and marginal distributions are calculated, smoothing is performed. The smoothing strategy relies on the p-state frequencies. Therefore, the marginal distributions will be weighted more heavily for infrequently occurring p-states than frequent p-states. As an example, consider the standard deviations for hidden state 1, given partially observed states b and c, namely $\sigma_{1|b} = 0$ and $\sigma_{1|c} = 0$. These estimates result in degenerate distributions, a problem encountered when model parameters outnumber the data. Parameter smoothing will avoid overfitting by reducing the *dof* of the model. Let w_ω be the weight given to the parameters for the distribution conditioned on the partially observed state ω, and $(1 - w_\omega)$ be the weight given to the marginal distribution. Using the inverse frequency smoothing strategy, the weights are determined by the p-state inverse frequencies, where $w_b = 1 - \frac{1}{1+f(b)} = \frac{2}{3}$ and $w_c = 1 - \frac{1}{1+f(c)} = \frac{1}{2}$. The conditional parameters

Emission	$\mu_{1\|.}$	2.60		$\sigma_{1\|.}$	1.20		$\mu_{2\|.}$	3.24		$\sigma_{2\|.}$	0.61
	$\mu_{1\|a}$	1.80		$\sigma_{1\|a}$	0.75		$\mu_{2\|a}$	2.91		$\sigma_{2\|a}$	0.45
	$\mu_{1\|b}$	3.53		$\sigma_{1\|b}$	0.40		$\mu_{2\|b}$	3.75		$\sigma_{2\|b}$	0.20
	$\mu_{1\|c}$	2.80		$\sigma_{1\|c}$	0.60		$\mu_{2\|c}$	3.12		$\sigma_{2\|c}$	0.31
Transition	$a_{11\|.}$	0.65		$a_{11\|.a}$	0.97		$a_{11\|.b}$	0.01		$a_{11\|.c}$	0.03
	$a_{11\|a.}$	0.02		$a_{11\|aa}$	0.54		$a_{11\|ab}$	0.01		$a_{11\|ac}$	0.03
	$a_{11\|b.}$	0.94		$a_{11\|ba}$	0.94		$a_{11\|bb}$	0.56		$a_{11\|bc}$	0.56
	$a_{11\|c.}$	1.00		$a_{11\|ca}$	0.99		$a_{11\|cb}$	0.51		$a_{11\|cc}$	0.52
	$a_{22\|.}$	0.51		$a_{22\|.a}$	0.28		$a_{22\|.b}$	0.99		$a_{22\|.c}$	0.97
	$a_{22\|a.}$	0.98		$a_{22\|aa}$	0.55		$a_{22\|ab}$	0.99		$a_{22\|ac}$	0.97
	$a_{22\|b.}$	0.55		$a_{22\|ba}$	0.50		$a_{22\|bb}$	0.69		$a_{22\|bc}$	0.68
	$a_{22\|c.}$	0.00		$a_{22\|ca}$	0.09		$a_{22\|cb}$	0.49		$a_{22\|cc}$	0.48
Starting	$\pi_{1\|.}$	0.00		$\pi_{1\|a}$	0.08		$\pi_{1\|b}$	0.00		$\pi_{1\|c}$	0.02
Stationary	$\Pi_{1\|.}$	0.37		$\Pi_{1\|a}$	0.73		$\Pi_{1\|b}$	0.01		$\Pi_{1\|c}$	0.03

Table 4.3: POHMM example estimated parameters.

are updated to

$$\begin{aligned}\dot{\sigma}_{1|b} &= w_b \times \sigma_{1|b} + (1 - w_b) \times \sigma_{1|.} \\ &= \frac{2}{3} \times 0 + \frac{1}{3} \times 1.07 = 0.36\end{aligned}$$

and

$$\begin{aligned}\dot{\sigma}_{1|c} &= w_c \times \sigma_{1|c} + (1 - w_c) \times \sigma_{1|.} \\ &= \frac{1}{2} \times 0 + \frac{1}{2} \times 1.07 = 0.53\ .\end{aligned}$$

Since c has only been observed one time, more weight is placed on the marginal parameter $\sigma_{1|.}$. In both cases, a degenerate normal distribution is avoided. Table 4.2 shows the initial emission parameters before and after smoothing is performed.

4.7.3 Parameter estimation

After initialization, the modified Baum-Welch algorithm is used to determine parameters, as described in Algorithm 4.7. Each iteration in the modified Baum-Welch algorithm consists of an E step, M step, marginals calculation, and parameter smoothing. Convergence is reached when the difference between the loglikelihood from successive iterations is less than 10^{-6}, which happens after 3 iterations. The final parameters are shown in 4.3, where the sequence above has a loglikelihood of -3.18. The transition, starting, and steady state probabilities are shown for only one hidden state since the other is implied in a two-state model. Figure 4.4 shows part of the fitted model, namely the marginal and event-type-a emission parameters, and the marginal-to-marginal, marginal-to-a, a-to-marginal, and a-to-a transition parameters.

4.7.4 Likelihood calculation

The fitted model can be used to obtain the likelihood of a novel event sequence. Consider the test observation sequence

$$\mathbf{y}_1^N = 2 \quad 1 \quad 6$$

with partially observed states

$$\Psi_1^N = c \quad a \quad d$$

where the symbol d has not been observed during parameter estimation. The log forward lattice as determined by Algorithm 4.4 is shown is Table 4.4. In this example, the a-to-marginal transition probabilities, $a_{11|a.}$, $a_{12|a.}$, $a_{21|a.}$, and $a_{22|a.}$, are used going from $n = 2$ to $n = 3$ since d was not observed during parameter estimation. The resulting loglikelihood of the observation sequence is given by

$$\log \sum_{j=1}^{2} \exp\left[\alpha_j(3)\right] = -15.11 \; .$$

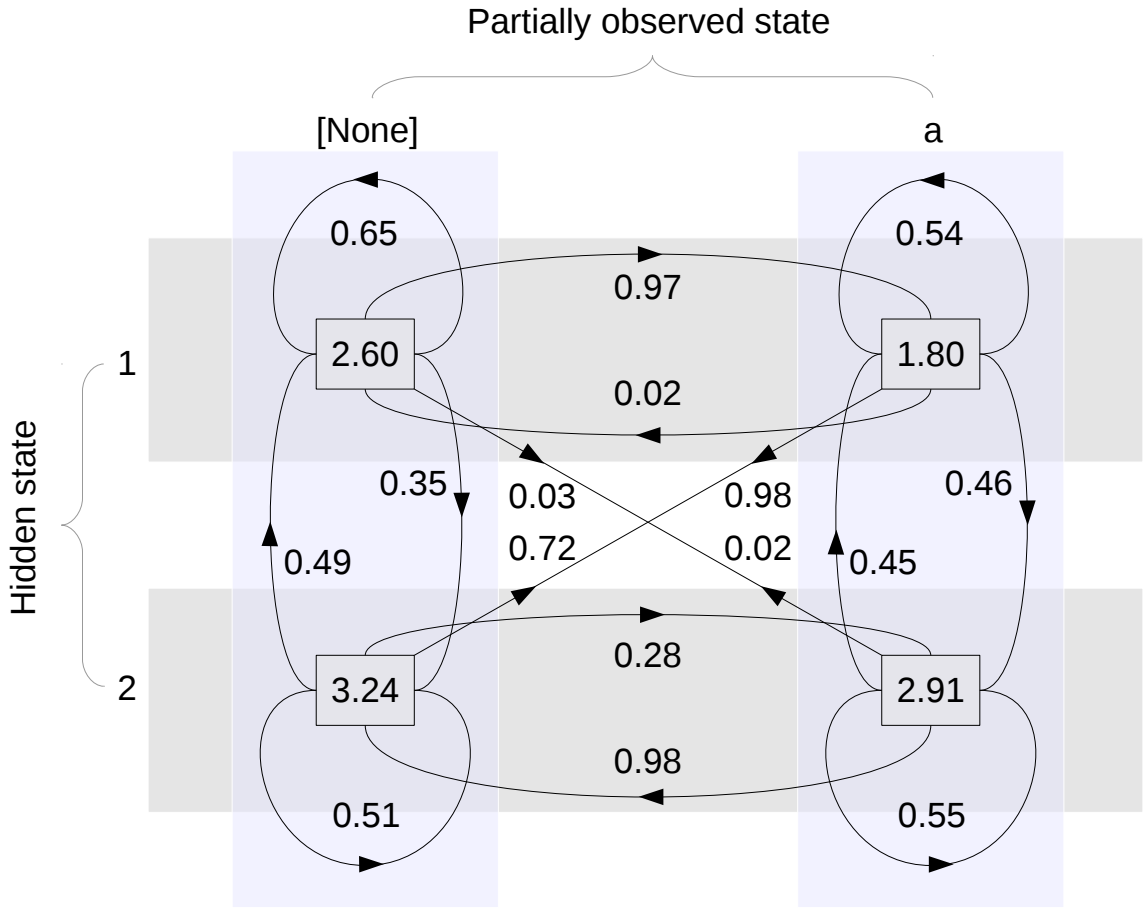


Figure 4.4: POHMM example showing emissions and transitions between hidden states of the marginal distribution and event type a. Emission μ parameters are shown in the small center boxes. Hidden states are along the vertical, and partially observed states are along the horizontal (original figure).

$\alpha_j(n) =$

n				
1	2	3		
-5.42	-6.33	-15.28	1	j
-6.40	-17.63	-16.96	2	

Table 4.4: POHMM example forward lattice.

4.8 Consistency

It is important for model parameter estimation to be consistent (Newey and McFadden, 1994). This requires that parameter estimation be both convergent and asymptotically unbiased. While the BW algorithm for HMM parameter estimation has these properties, it is not clear whether the modified BW for POHMM parameter estimation also does. Specifically, parameter smoothing violates the maximization step in the EM algorithm.

The POHMM consistency is demonstrated is this section using computational methods. First, a model is initialized with parameters θ_0. From this model, S time interval samples are generated, each containing N time intervals. For each time series τ, the best-estimate parameters $\hat{\theta}$ are computed using the modified BW algorithm in Algorithm 4.7. As N increases, with sufficiently large S, the scaled residuals of a consistent model will go to 0. The convergence to 0 should also be insensitive to the choice of θ_0. Let $\hat{\theta}_N$ be the parameters determined by the modified Baum-Welch algorithm for an observed time series of length N generated from a POHMM with true parameters θ^0. The model is consistent if

$$\lim_{N\to\infty} \frac{|\hat{\theta}_N - \theta^0|}{\max_{\hat{\theta}} |\hat{\theta}_N - \theta^0|} = \mathbf{0}\,. \tag{4.30}$$

This procedure requires generating a random POHMM in which certain parameter constraints need to be met. The starting probabilities and transition matrix are first generated for the partially observed states. The conditional transition matrices between the hidden states of each pair of partial states is then generated. Marginals are then determined as described in Section 4.5.2. Rows of the transition matrices should be uniformly distributed and must sum to 1.

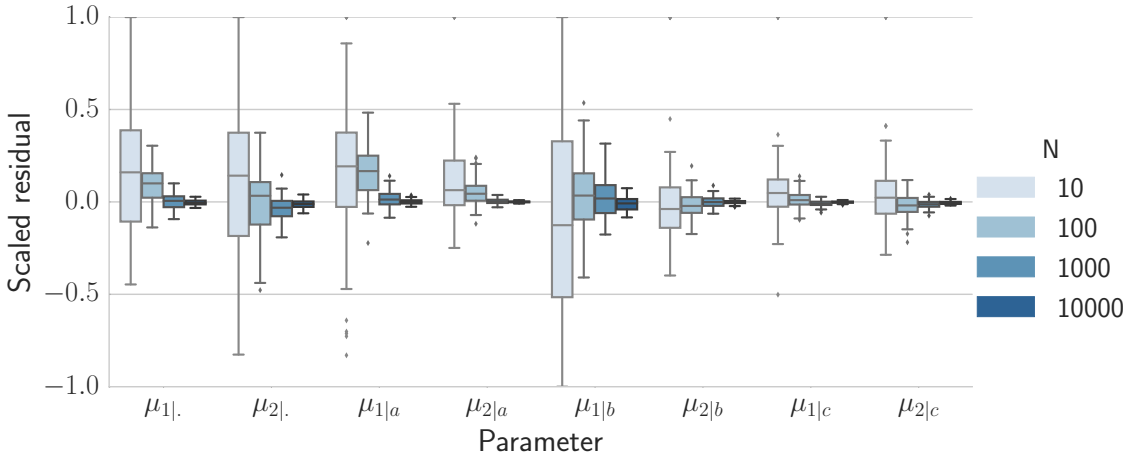
A uniformly-random transition matrix can be generated as follows. Each row must sum to 1 and therefore has $M-1$ *dof*. Generate $M-1$ random numbers between 0 and 1 and then sort the list of numbers with 0 and 1 concatenated. Take the difference between the list of $M+1$ numbers to get M uniformly-distributed numbers that sum to 1. Repeat this M times to get an $M \times M$ transition matrix or just once to get the array of starting probabilities. The algorithm for generating a stochastic matrix is summarized in Algorithm 4.8.

Algorithm 4.8 Random transition matrix generation.

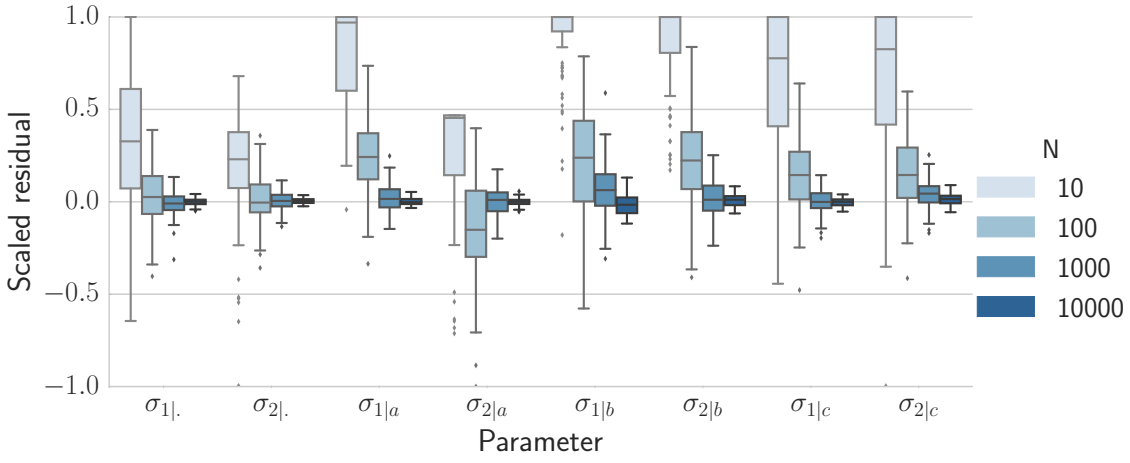
1. Generate $M-1$ numbers between 0 and 1
2. Concatenate 0 to the beginning and 1 to the end of the list of sorted numbers
3. Take the differences between the list of $M+1$ numbers to get M uniformly-random numbers that sum to 1
4. Repeat steps 1-3 M times to generate an $M \times M$ transition matrix.

A POHMM with 3 partially observed states, $\Omega = \{a, b, c\}$ and 2 hidden states for each p-state is generated. A normal distribution is chosen for emissions, with parameters comparable to the key-press latencies in typing. For each value of N, 100 length-N samples are generated. Each sample consists of the observations time series and partially observed states. To generate a sample, first N partially observed states are generated according to the p-state transition matrix. The conditional parameters are then used to generate the sequence of hidden states and observed values.

Figure 4.5 shows the scaled residuals for emission parameters without parameter smoothing. Central lines indicate the median, boxes depict the interquartile range, and whiskers show the 95% confidence interval. Outliers are shown as points outside the whiskers. The residuals go to 0 as N increases, indicating convergent and unbiased estimates. The same test is repeated with a POHMM that uses inverse frequency weights for parameter smoothing. The scaled residuals are shown in Figure 4.6. The results indicate the model is consistent. As N increases, the smoothing strategy weights the marginal distributions less and the conditionals approach those that would be obtained without smoothing. Figure 4.7 shows the sum of absolute residuals for various parameter groups, comparing the smoothed parameter residuals to those obtained without smoothing. The smoothed parameter residuals are generally lower than those without smoothing.

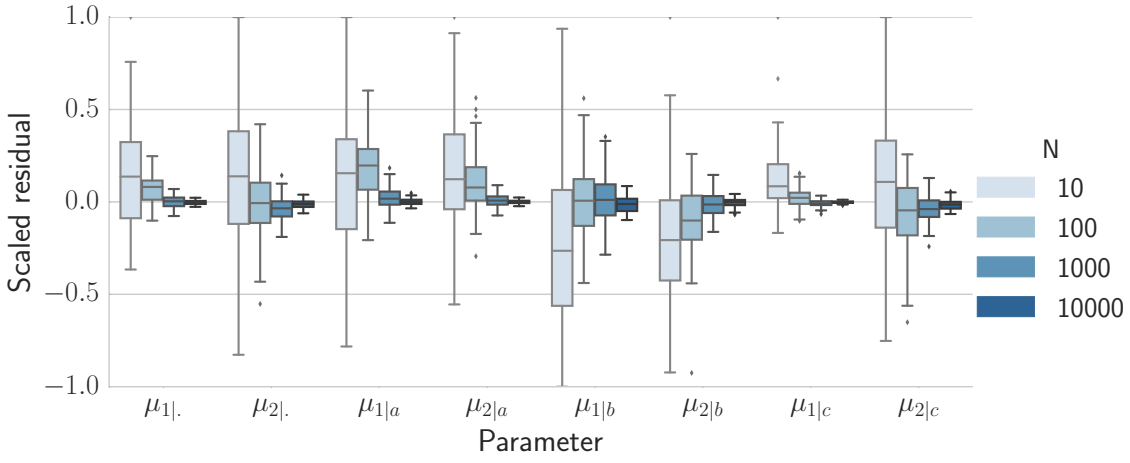


(a) μ

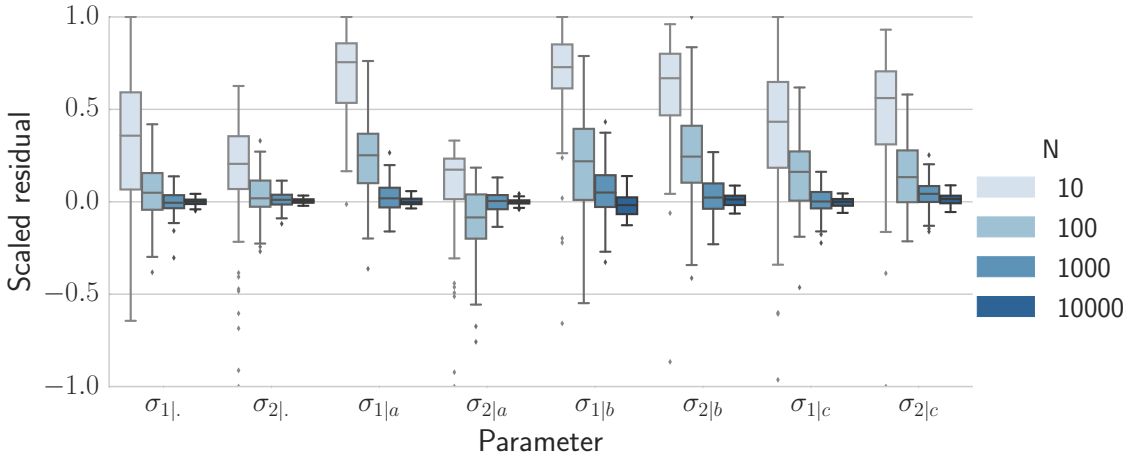


(b) σ

Figure 4.5: POHMM scaled residuals for emission parameters. Central lines indicate the median, boxes depict the interquartile range, and whiskers show the 95% confidence interval. Outliers are shown as points outside the whiskers (original figure).

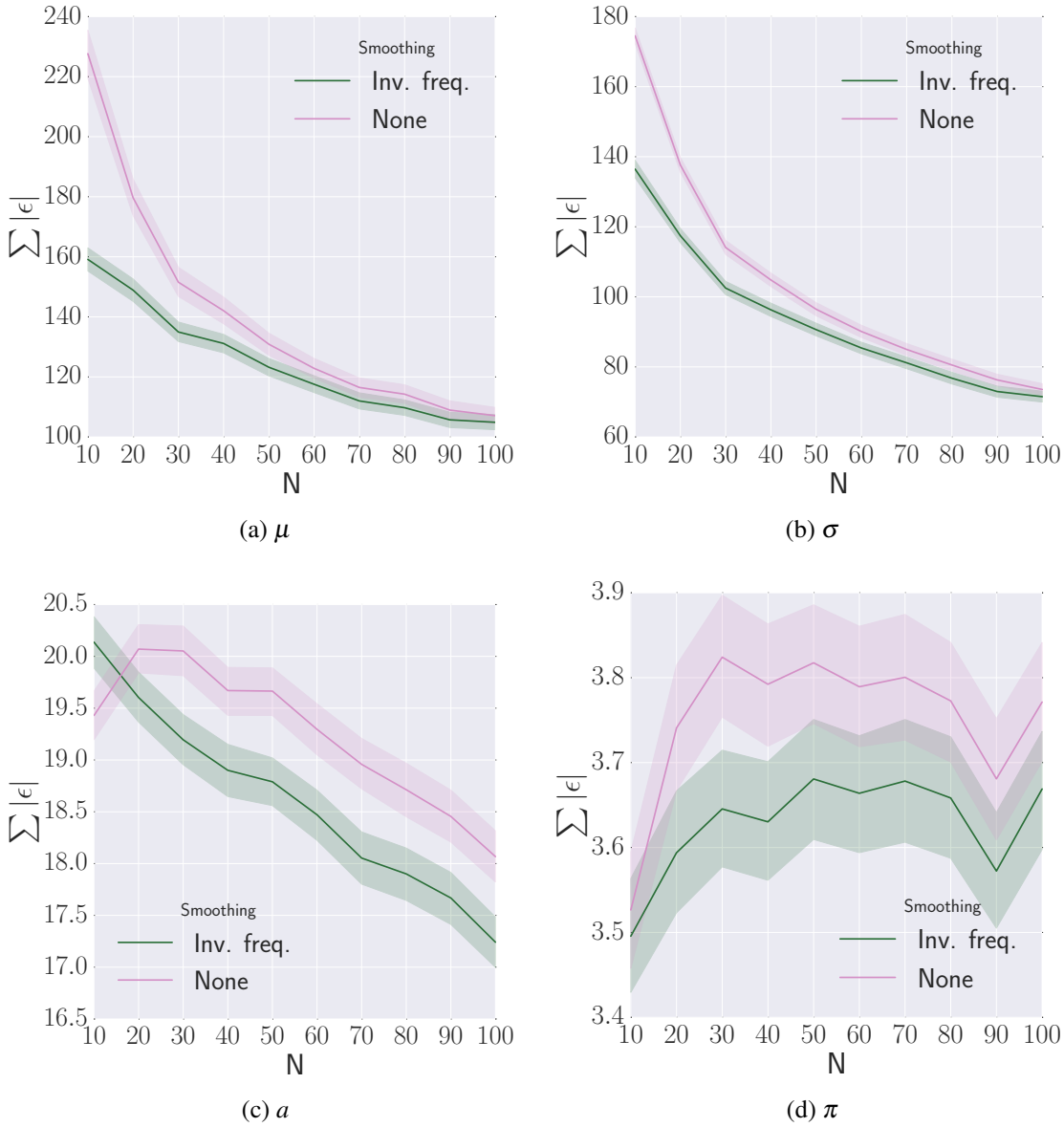


(a) μ



(b) σ

Figure 4.6: POHMM scaled residuals for emission parameters with smoothing. Central lines indicate the median, boxes depict the interquartile range, and whiskers show the 95% confidence interval. Outliers are shown as points outside the whiskers (original figure).



(a) μ (b) σ

(c) a (d) π

Figure 4.7: POHMM parameter residuals with and without smoothing, showing the sum of absolute residuals of various parameter groups as N increases. Smoothing generally results in better estimates, demonstrated by smaller residuals. Bands show the 95% confidence intervals (original figure).

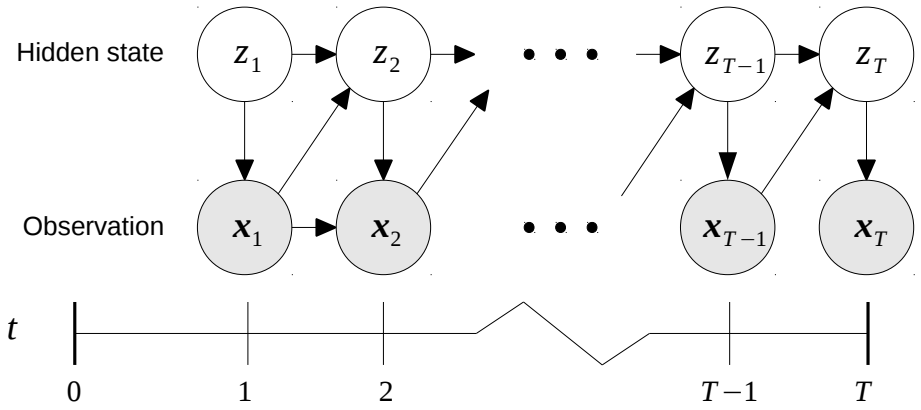


Figure 4.8: Partly hidden Markov model structure (original figure).

4.9 Discussion

There have been various generalizations of the HMM to deal with hidden states that are partially observable in some way. Partial observability may be provided through assumptions about the hidden state, uncertain labels, or contextual information. There is some overlap in the terminology surrounding various models that have been proposed, as explained in this section.

The partly hidden Markov model (partly-HMM) is a second order model in which the first state is hidden and the second state is observable (Kobayashi and Haruyama, 1997). The structure of the partly-HMM is shown in Figure 4.8 where both the state and observation at time t are dependent on the observation at time $t-1$. The partly-HMM can be applied to problems that have a transient underlying process, such as gesture and speech recognition, as opposed to a piecewise stationary process that the HMM assumes (Iobayashi et al., 1999). Parameter estimation can be performed by the EM algorithm, similar to the HMM.

Partially observable states can also come in the form of partial and uncertain ground truth regarding the hidden state at each time step. The partially hidden Markov Model (partially-HMM) deals with this scenario, in which an uncertain hidden state label may observed at each time step (Ozkan et al., 2014). The probability of observing the uncertain label and the probability of the label being correct, were the true hidden state known, are controlled by parameters p_{obs} and p_{true}, respectively. The probability of observing a correct

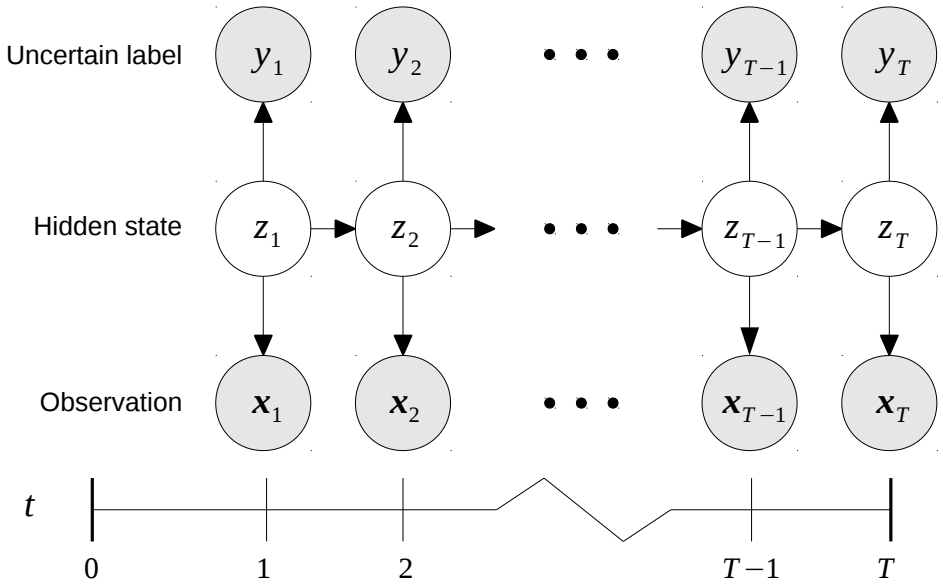


Figure 4.9: Partially hidden Markov model structure (original figure).

label is $p_{obs} \times p_{true}$. The structure of the partially-HMM is shown in Figure 4.9. This type of model is primarily motivated by language modeling applications, in which ground truth state labels can help in parameter estimation although they may be incorrect or missing (Merialdo, 1994). Similar to the HMM, the EM algorithm can be used for estimating the parameters of the partially-HMM (Ozkan et al., 2014).

Previous observations can also provide context for the transition and emission probabilities. Forchhammer and Rissanen (1996) proposed the context hidden Markov model (context-HMM), in which the transition and emission probabilities at time t are conditioned on an observed context function. The context functions are defined as $A_V(x_t) = v_t$ and $B_W(x_{t-1}) = w_t$ for the transition and emission probabilities, respectively, where the context sequences v_t and w_t are functions of the observation sequence. At each time step, the hidden state and observation are dependent on a context from the previous time step, shown in Figure 4.10. The context-HMM has information theoretic motivations, with applications such as image compression Forchhammer and Rasmussen (1999). Used in this way, the neighboring pixels in an image can provide context for the emission and transition probabilities.

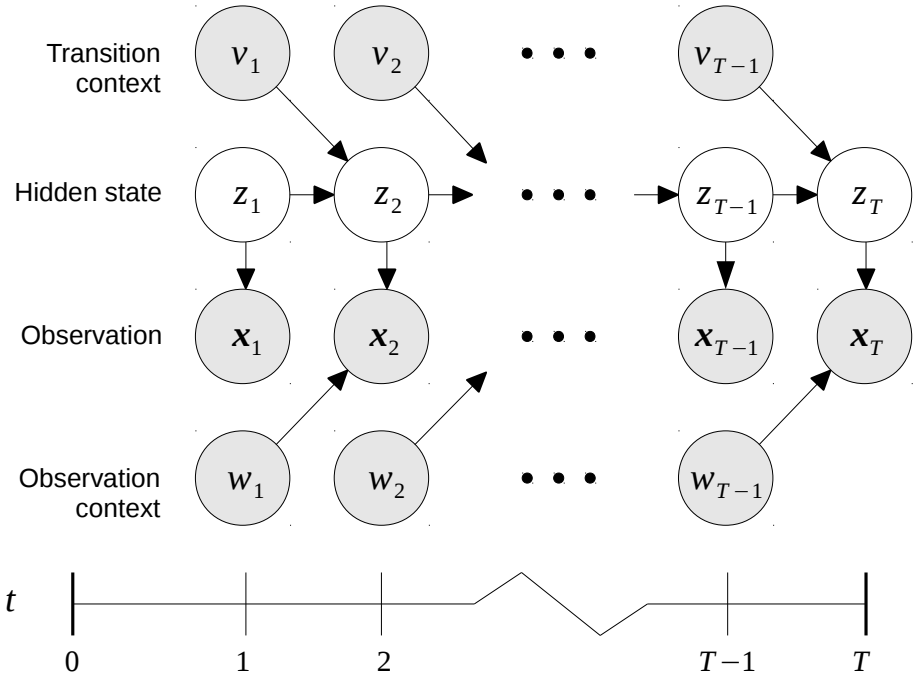


Figure 4.10: Context hidden Markov model structure (original figure).

Related to stochastic processes with partially observable underlying states is the hidden semi-Markov model (HSMM) in which the underlying process is a semi-Markov chain (Ferguson, 1980). Whereas the underlying state of a HMM transitions at each time step, a HSMM remains in a particular hidden state for a variable number of observations. This type of model is also sometimes referred to as an "explicit duration HMM" or "variable duration HMM" (Rabiner, 1989). The hidden state is partially revealed through the passage of time itself, or alternatively, the number of observations produced in a given state. The HSMM is described in detail in Yu (2010).

The POHMM, introduced in this work, is an extension of the HMM in which hidden states are partially observable through event types. A subclass of the POHMM is the HMM in which the event types are all the same or possibly unknown. The POHMM marginal distributions can account for missing data during testing, while parameter smoothing helps to avoid overfitting and accounts for missing data during training. Model consistency was demonstrated empirically, and the estimated parameter residuals were shown to be lower when a parameter smoothing strategy is employed. Computational complexities of the

POHMM parameter estimation and likelihood calculation algorithms are comparable to that of the HMM, which are linear in the number of observations.

The POHMM is different from the partly-HMM, being a first order model, and different from the partially-HMM, since it doesn't assume a partial labeling. In particular, the POHMM makes the following assumptions.

1. There is some additional information associated with each event, such as the event type. This represents the partially observed state.

2. There is a shared underlying stochastic process between events with different types. This represents the hidden state.

3. Some events have missing or unknown partially observed states.

4. Novel partial states are observed during testing.

The POHMM is most similar to the context-HMM in the sense that emission and transition probabilities are conditioned on some observed values. There are several important differences between the POHMM and context-HMM. In the POHMM, the hidden state at time t is dependent on the partially observed state at times t and $t-1$, whereas the hidden state of the context-HMM at time t is dependent only on the context at time $t-1$. Additionally, the context sequences are defined as functions of the observed values, which differ from the partially observed states of the POHMM; the POHMM p-states are composed of metadata that generally capture the event types. For missing p-states, the marginal distributions of the POHMM also act as a fallback mechanism, whereas the context-HMM does not account for missing or undefined context. This is an important distinction since novel p-state sequences may be observed during testing. The marginal distributions of the POHMM are also used in parameter smoothing, which reduces the *dof* of the model and provides superior parameter estimates with small amounts of data.

Chapter 5

Model evaluation

There are numerous evaluation criteria a model can be subjected to. Usually, the choice of criteria depend on the problems the model was originally designed to solve. In physics, models are often evaluated based on their predictive power or plausibility as an underlying generative mechanism. In biometrics, a model is often evaluated based on how well it can identify or verify people. The POHMM proposed in Chapter 4 was designed to solve three problems in biometrics, including identification, verification, and prediction. In this chapter, the criteria for evaluating the discriminative and predictive capabilities of the POHMM are discussed. Most of the criteria are standard in the biometrics and machine learning literature.

5.1 Goodness of fit

To determine whether the proposed model is consistent with observed data, a goodness of fit test is performed through Monte Carlo hypothesis testing. The test proceeds as follows. For each time interval sample, the model parameters, $\hat{\theta}_m$ are determined. The area test statistic between the model and empirical distribution is then taken. The area test statistic is a compromise between the Kolmogorov-Smirnov (KS) test and Cramer-von Mises test Malmgren et al. (2008), given by

$$A = \int |P_D(\tau) - P_M(\tau|\hat{\theta}_m)| d\tau$$

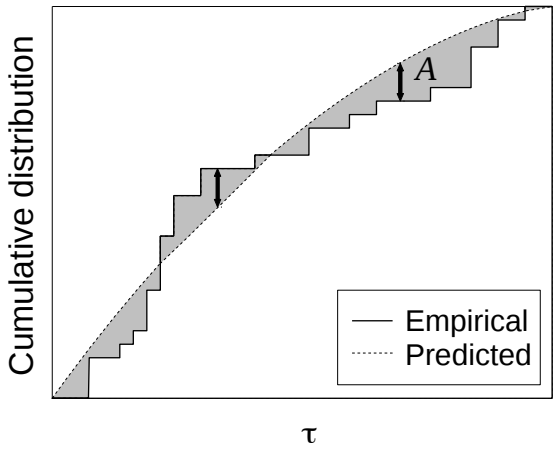


Figure 5.1: Area test statistic given by the shaded region (original figure).

where P_D is the empirical cumulative distribution and P_M is the model cumulative distribution. An example of calculating A is shown in Figure 5.1.

The marginal density of the model is given by

$$g(\mathbf{x};\theta) = \sum_{j=0}^{1} \Pi_j f(\mathbf{x};\mathbf{b}_j) \tag{5.1}$$

where Π_j is the steady state probability of being in state j and f is the emission distribution. Using the fitted model with parameters $\hat{\theta}_m$, a surrogate data sample the same size as the empirical data is generated. The surrogate data is then treated similarly to the empirical data, where estimated parameters $\hat{\theta}_s$ are determined. The area test statistic between the surrogate-data-trained model and surrogate data is computed, given by A_s. This process repeats until enough surrogate statistics have accumulated to reliably determine $Pr(|A_s - \langle A_s \rangle| > |A - \langle A_s \rangle|)$. The biased P value is given by

$$\frac{I(|A_s - \langle A_s \rangle| > |A_m - \langle A_s \rangle|) + 1}{m+1} \tag{5.2}$$

where $I(\cdot)$ is the indicator function. The testing procedure is summarized in Figure 5.2. The null hypothesis (that the model is consistent with the data) is tested for each sample in each dataset. Each test requires fitting $S+1$ models (1 empirical and S surrogate samples).

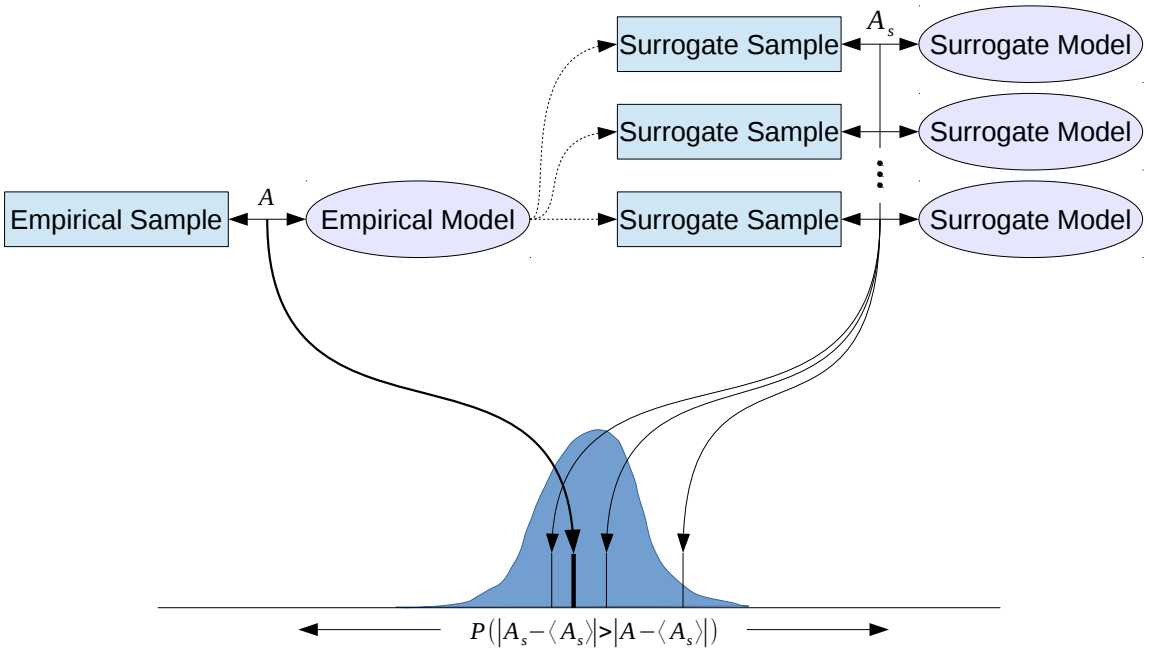


Figure 5.2: Monte Carlo Monte Carlo goodness of fit testing procedure testing procedure (original figure).

5.2 Identification

Behavioral biometric identification utilizes individual differences in behavior to identify users. Given a sample from an unknown user, identification requires a choice of 1 out of U known users. Typically, the proportion of correctly identified samples is used as an empirical measure of classification accuracy (ACC), where random choice has an asymptotic ACC of $\frac{1}{U}$.

Identification is performed by the POHMM as follows. A system of U models is created, with one model trained for each user. The likelihood of an unknown sample is determined for each model. The model with the highest likelihood is chosen as the class label for the unknown sample. This scenario assumes that the correct class label is contained in the system, which may or may not be true in practice. Identification accuracy depends on many factors, including the number of samples used for training, the size of samples, and number of users in the system.

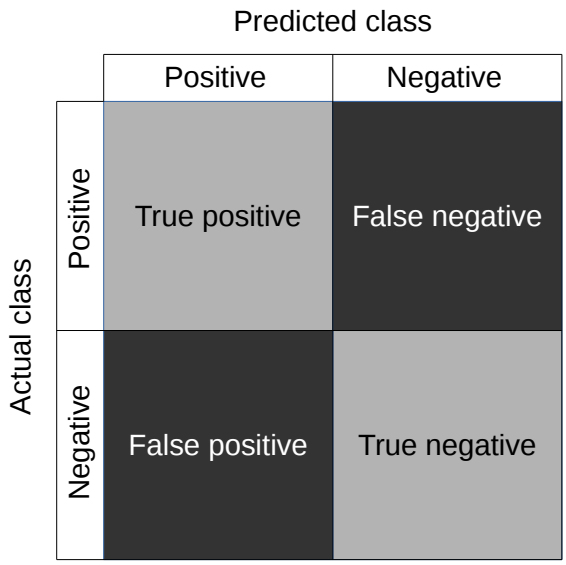


Figure 5.3: Binary classification errors (original figure).

5.3 Verification

Verification is a binary decision problem in which a biometric sample with claimed ownership is labeled as either being *genuine* or an *impostor*. There are two types of errors that can occur, as demonstrated by the two-class confusion matrix in Figure 5.3. A false negative, or *false rejection,* occurs when a genuine user is rejected by the system, and a false positive, or *false acceptance,* occurs when an impostor fails to be rejected by the system. The two types of error correspond to type I and type II errors in statistics and are typically measured by the empirical *false rejection rate* (FRR) and *false acceptance rate* (FAR), respectively.

There is generally an incentive for an impostor to make a false claim in access control applications. The usual scenario is an impostor attempts to bypass a biometric access control system and gain access to sensitive information. There are also situations where it would be desirable to keep one's identity hidden. This is more common in forensic applications, in which verification must be performed on some data that has been collected in the past with a suspected identity linked to the data. In this scenario, there is incentive for the genuine user to conceal their true identity.

In any biometric verification system, there is a direct tradeoff between FRR and FAR. In practice, this corresponds to a tradeoff between convenience and security where higher

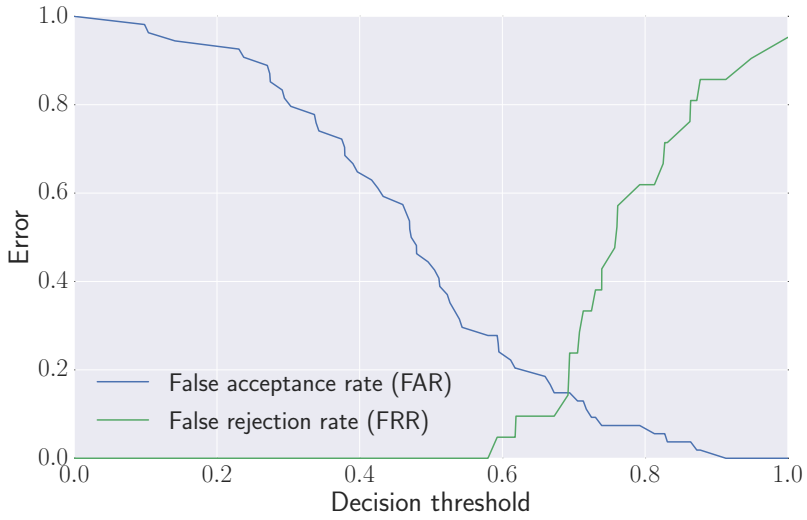


Figure 5.4: Convenience-security tradeoff controlled by a decision threshold (original figure).

FRR inhibits the normal operation of the system and higher FAR introduces greater risk of an impostor going undetected. This tradeoff is usually determined by a single model parameter. With a statistical model such as the POHMM, a decision threshold on the model likelihood can be used make a verification decision. The likelihood of a sample under the claimed model is obtained and a decision is made such that above the decision threshold, the sample is labeled as genuine, and below the decision threshold, the sample is labeled as an impostor. As the threshold varies, the tradeoff between FRR and FAR is obtained. An example is shown in Figure 5.4, where FRR increases and FAR decreases as the decision threshold increases.

The receiver operating characteristic (ROC) curve graphically captures the tradeoff between FAR and FRR and can be used to evaluate system performance. The ROC curve is simply the FRR plotted against the FAR. An example ROC curve is shown in Figure 5.5, which is obtained directly from the data in Figure 5.4. The point on the ROC curve where FAR and FRR are equal is the equal error rate (ERR). The EER summarizes the performance of the system in a single number so that various systems can be compared. As an alternative to the EER, the total area under the ROC curve is calculated, given by the area under curve (AUC). The AUC is sometimes preferable since it summarizes the performance

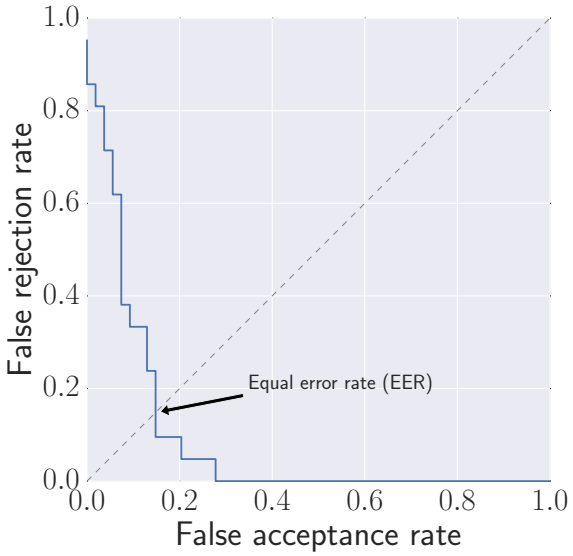


Figure 5.5: Example ROC curve (original figure).

of the system over the entire curve and not just at a single point. A perfect classifier would have an AUC of 0, and a random classifier would have an AUC of 0.5. Note that a system that generates an ROC curve with an EER and AUC of 1 also has perfect classification since the decisions could be inverted to obtain an EER and AUC of 0.[1]

A verification decision using the POHMM proceeds as follows. The likelihood of a query sample is obtained under the model corresponding to the identity of the claimed user. This score is then normalized by the minimum and maximum likelihoods under every other model in the system, *i.e.* the models of every other user. The resulting normalized score is compared to a decision threshold, which varies between 0 and 1. If the score is less than the threshold, then the claim is rejected. The aggregation of verifications using all sample/identity combinations allows the ROC curve to be constructed.

[1] Some authors use the true positive rate (TPR) instead of the FRR. In this case, a perfect classifier would have an AUC of 1 and a random classifier an AUC of 0.5.

5.3.1 Continuous verification

When dealing with behavioral biometrics it makes sense to consider the performance of a system operating continuously. This is especially true for modalities that produce a continuous stream of events, such as Bitcoin transactions or terrorist activity[2]. One approach that has often been used is to split a stream of events into sessions by grouping events within a time window and then performing static verification, *i.e.* the entire session is either verified or rejected (Bours and Mondal, 2015). As the size of the window approaches 1 event, the granularity of verification increases although the information between sessions is not utilized.

In this work, continuous verification is enforced through a penalty function in which each event incurs a non-negative penalty within a rolling window. When the cumulative penalty incurred within the sliding window exceeds a given threshold, the user is rejected from the system. The penalty at any given time can be thought of as the inverse of trust. The sliding window ensures that the maximum achievable penalty is bounded, and may actually decrease as behavior becomes more consistent with a template. In Bours (2012), a different approach is taken with a penalty-and-reward function. The approach introduced here avoids having to determine distance threshold and reward parameters. The sliding window replaces the reward, since penalties outside the window do not contribute towards the cumulative penalty.

The penalty of each event is determined as follows. The marginal probability of each new event, given the preceding events, can be obtained from the forward lattice, α, given by

$$P(\mathbf{x}_{t+1}|\mathbf{x}_1^t) = P(\mathbf{x}_1^{t+1}) - P(\mathbf{x}_1^t)$$

When a new event is observed, the likelihood is obtained under each model in the system. The likelihoods are ranked, with the highest given a rank of 0, and the lowest a rank of $U-1$, with U models in the system. The penalty is the rank of the claimed user's model. Thus, if a single event is correctly identified, a penalty of 0 is incurred. If it scores the second highest likelihood, a penalty of 1 is incurred, and so on. The rank penalty is added

[2]This also depends on the application. Keystrokes collected during an online exam could be considered as belonging to one session, whereas the keystrokes collected by a system-wide logger that is permanently installed on a machine do not necessarily form a session.

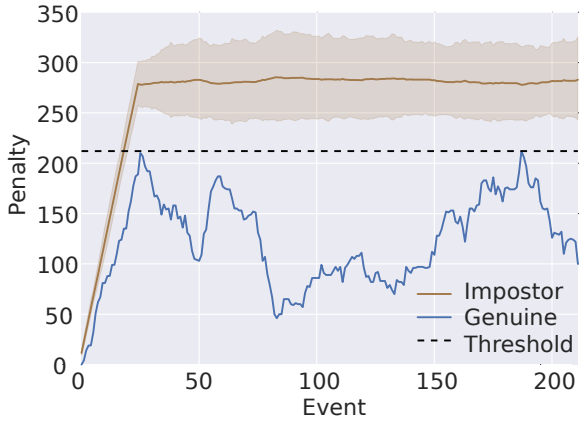


Figure 5.6: Continuous verification example (original figure).

to the cumulative penalty in the sliding window, while penalties outside the window are discarded.

The continuous verification performance is reported as the number of events that can occur before an impostor is detected. This is determined by adjusting the penalty threshold so that the genuine user is never locked out of the system. Since the genuine user's penalty is always below the threshold, this is the maximum number of events that an impostor can execute before being rejected by the system, given the genuine user is not rejected.

An example of the penalty function for genuine and impostor users is shown in Figure 5.6. The decision threshold is set to the maximum penalty incurred by the genuine user so that a false rejection does not occur. The average penalty for impostor users, with 95% CI, is shown. In this particular example, the impostor penalties exceed the decision threshold after about 20 keystrokes.

For each sample, the maximum rejection time (MRT) is determined for each impostor. This procedure is summarized as follows.

1. Determine the maximum penalty threshold such that the genuine user is not rejected.

2. Determine the number of keystrokes it takes each impostor to be rejected.

3. Repeat Steps 1 - 2 to obtain the MRT for each sample.

The average MRT (AMRT) is the average time it takes an impostor to be rejected. Since the MRT is obtained for each sample, a bootstrapped confidence interval for the MRT can be obtained.

5.4 Prediction

Prediction is important when an application requires the estimation of resource consumption, such as mobile call routing, or cost minimization, in the case of terrorist events. Forecasting human behavior has countless other applications. The task can be stated as follows. Given time intervals τ_1^N, predict τ_{N+1}, or the time of the next event. The symmetric mean absolute percentage error (SMAPE) is used to evaluate prediction accuracy. Let the prediction for each time interval τ_n be $\hat{\tau}_n$. The SMAPE is given by

$$\text{SMAPE} = \frac{1}{N}\sum_{n=1}^{N}\left|\frac{\tau_n - \hat{\tau}_n}{\tau_n + \hat{\tau}_n}\right|. \tag{5.3}$$

This value ranges between 0 and 1, with 0 indicating a perfect prediction.

There are two scenarios to consider when making predictions using the POHMM. The future p-states can either be known or unknown. For example, in a sequence of keystrokes typed freely, the future p-state (*i.e.* the key that will be pressed) is generally not known. For a password, or any other fixed text, the future p-states will be known. With known future p-state, the POHMM can be used to predict the time of the $(n+1)^{\text{th}}$ event by determining the probability of being in each hidden state at time $n+1$ and the expected time interval in each state. This is given by

$$\hat{\tau}_{n+1} = \sum_{i=0}^{1} P(z_{n+1} = i|\tau_1^n)\mathsf{E}[\tau_{n+1}|z_{n+1} = i]$$

where z_n is the state of the system at time n. Using the variables calculated from the forward procedure, this translates to

$$\hat{\tau}_{n+1} = \sum_{i=0}^{1} \beta_i \mathsf{E}[\tau_{n+1}|z_{n+1} = i]$$

where β_i is the posterior probability of being in state i at time n,

$$\beta_i = \frac{\sum_j \alpha_n(j) P(z_{n+1}|z_n = j)}{\sum_j \alpha_n(j)}$$

and $\alpha_n(j)$ is the probability of time intervals τ_1^n and state $z_n = j$ at time n. This can be efficiently computed using the forward procedure.

With unknown future p-state, the expectation of observing each p-state must be accounted for. In this case, the predicted time interval is given by

$$\hat{\tau}_{n+1} = \sum_{\psi \in \Omega} \sum_{i=0}^{1} P(\omega_{n+1} = \psi | \Omega_1^n) P(z_{n+1} = i | \tau_1^n, \Omega_1^n) E[\tau_{n+1} | z_{n+1} = i, \Omega_{n+1} = \psi]$$

For comparison, a simple baseline predictor is defined as

$$\hat{\tau}_{n+1|\text{Baseline}} = \frac{1}{n} \sum_{i=1}^{n} \tau_i \tag{5.4}$$

which is the mean time interval up to time n.

5.5 Confidence intervals on evaluation metrics

The evaluation metrics described in this chapter are meant to provide empirical estimates of model performance in identification and verification scenarios. Experimental results are obtained by dividing the dataset into *reference* and *query* sets, where the reference set with associated class labels is used to train the models and the query set is used to determine the evaluation metric. When this process is repeated several times, a confidence interval can be obtained based on the variation of the metric.

A stratified k-fold cross-validation (SCV) is used to obtain the confidence intervals on the metrics described. The SCV, depicted in Figure 5.7, splits the dataset up into k different *folds*, or subsets. The evaluation metrics are obtained for each fold, using every other fold as the reference set. The proportion of samples from each class remains the same across

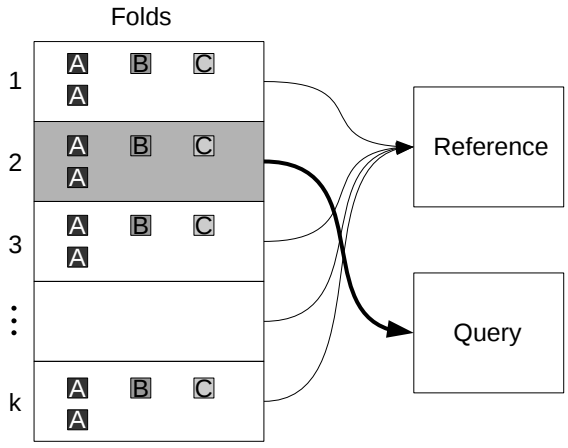


Figure 5.7: Stratified k-fold cross-validation (original figure).

the folds. This ensures that no single user is over- or under-represented in the reference or query set..

Chapter 6

Experimental results

In this chapter, experimental results are obtained for each dataset using the evaluation criteria described in Chapter 5. All models are fit using a loglikelihood reduction threshold of 10^{-6} or maximum of 1000 iterations, whichever is reached first. The time interval emission parameters, log-mean η and log-standard deviation ρ, are initialized using $h = 2$, as described in Section 4.5.1. This separates the initial log-means by 2 log-standard deviations. Log-normal distributions are used for time interval and duration features, while all other features are modeled by normal distributions. Starting probabilities are fixed between the partially observable states (p-states), a frequency smoothing strategy is used for μ, η, and transition parameters, and a steady state smoothing strategy is used for σ and η parameters. Smoothing strategies are described in Section 4.5.3.

The 95% confidence interval (CI) is shown in parentheses next to each evaluation metric. The CI is obtained using the stratified k-fold cross-validation described in Section 5.5 where the number of folds is equal to the number of samples per user in the dataset. This leaves one sample per user in the query set and $k-1$ samples per user in the reference set for parameter estimation. Each dataset contains approximately 50 users, with the exception of White House visits and terrorist activity. The size and features of each dataset are described in detail in Chapter 3. Each experiment compares the POHMM to the HMM, in which event types are not accounted for. For the keystroke datasets, benchmark results are also obtained by an algorithm with previously published results. The general procedure for each experiment is shown in Algorithm 6.1.

Algorithm 6.1 Experimental protocol.

1. **Preprocess**
   Extract features from the raw data.
2. **Split**
   Randomly split the data into k stratified folds.
3. **Train/test**: For each fold,
   (a) Train one model per user with $k-1$ samples.
   (b) Test each model using the query samples.
   (c) Determine evaluation metrics (ACC, EER, AUC, AMRT).
4. **Aggregate**
   Aggregate the evaluation metrics from the folds to obtain the mean and 95% CI.

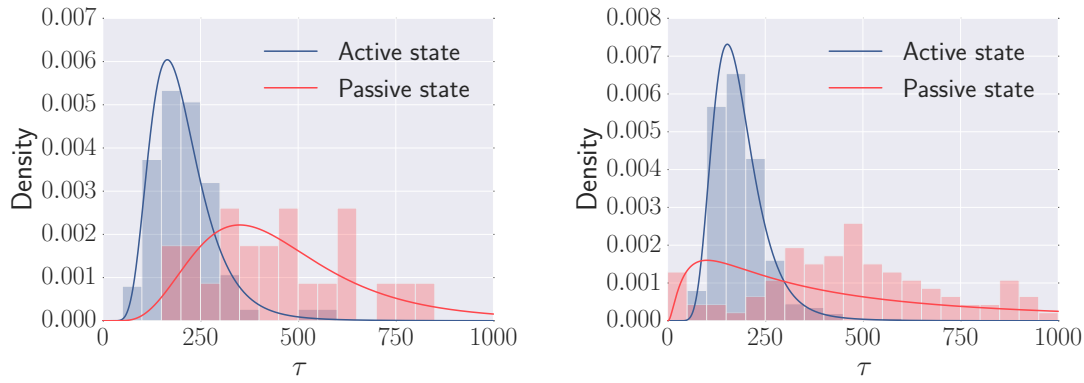
6.1 Keystrokes

In a two-state model, in which the user can be in either a passive or active state while typing, the key names partially reveal the underlying state of the user. The POHMM can take advantage of this by using the key names as p-states. For example, a typist is more likely to pause between words, after the space key is pressed, or between sentences, after the period key is pressed. Therefore, the probability of being in a passive state should be higher when the space or period keys are press compared to when letter keys are pressed.

There are several ways to define the p-states for keystroke data. For short fixed-text, a partially observable state can be created for each key on the keyboard and for long text, the keys can be grouped into several categories. This avoid an unnecessarily large number of p-states for long text. The two types of partially observable states are summarized as follows.

Key type Keys are grouped as: left letters, right letters, Space, Shift, Period, Comma, and other. Left letters include "qwertasdfgzxcvb" and right letters include "yuiophjklnm". Each key group is a p-state. This strategy is used for free-text.

Key name Each individual key is a p-state. This strategy is used for fixed-text.



(a) Fixed-text (b) Free-text

Figure 6.1: Keystroke time interval distribution examples showing the model time interval distribution in each state and empirical densities of the classified events as determined by the modified Viterbi algorithm (original figure).

An example, the log-normal time interval distributions of the fitted POHMM are shown in Figure 6.1 for both fixed-text and free-text keystroke samples. The passive state in the free-text example has a heavier tail than the fixed-text, while the active state distributions in both examples are comparable.

The POHMM is fit to each sample in both keystroke datasets and the mean model parameters are shown in Table 6.1. The mean proportion of events in each state as determined by the modified Viterbi algorithm is also shown. On average, 60% of keystrokes in fixed text and 75% of keystrokes in free text are in an active state. In both datasets, left-hand keys take longer to press and are held down longer than right-hand keys. Longer time intervals for the Shift, Period, and Comma keys are observed in both states. In both states and both datasets, the duration of the Shift key is about twice as long as other keys since it is typically used as a modifier (*i.e.* for upper case letters and special symbols).

6.1.1 Goodness of fit

A Monte Carlo goodness of fit test is performed for time intervals only, as described in Section 5.1. The results indicate that the POHMM is more consistent with the data compared to a HMM, in which key names are ignored. The proportion of samples in each dataset for

	Active (60%)						
	[Left]	[Right]	Space	Shift	Period	Comma	[Other]
Time interval	133.94	130.65	130.75	143.47	236.13	262.96	208.84
Duration	100.48	90.02	97.51	184.93	91.84	93.69	86.49

	Passive (40%)						
	[Left]	[Right]	Space	Shift	Period	Comma	[Other]
Time interval	287.36	283.28	283.94	347.49	361.67	455.26	537.51
Duration	101.49	88.23	96.54	214.86	93.69	97.51	90.92

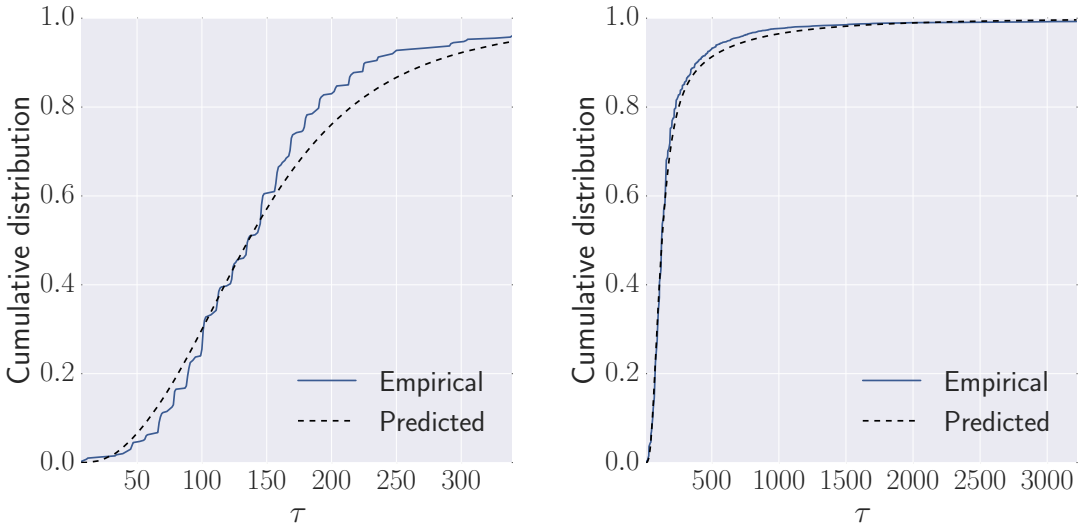
(a) Fixed-text

	Active (75%)						
	[Left]	[Right]	Space	Shift	Period	Comma	[Other]
Time interval	151.61	147.25	142.00	182.91	283.96	259.92	186.87
Duration	100.48	90.02	100.48	212.72	85.63	91.84	79.84

	Passive (25%)						
	[Left]	[Right]	Space	Shift	Period	Comma	[Other]
Time interval	329.8	326.11	365.58	690.52	699.45	618.54	604.69
Duration	95.58	88.23	97.51	307.97	83.10	92.76	87.36

(b) Free-text

Table 6.1: Keystroke average estimated emission parameters.



(a) Fixed-text

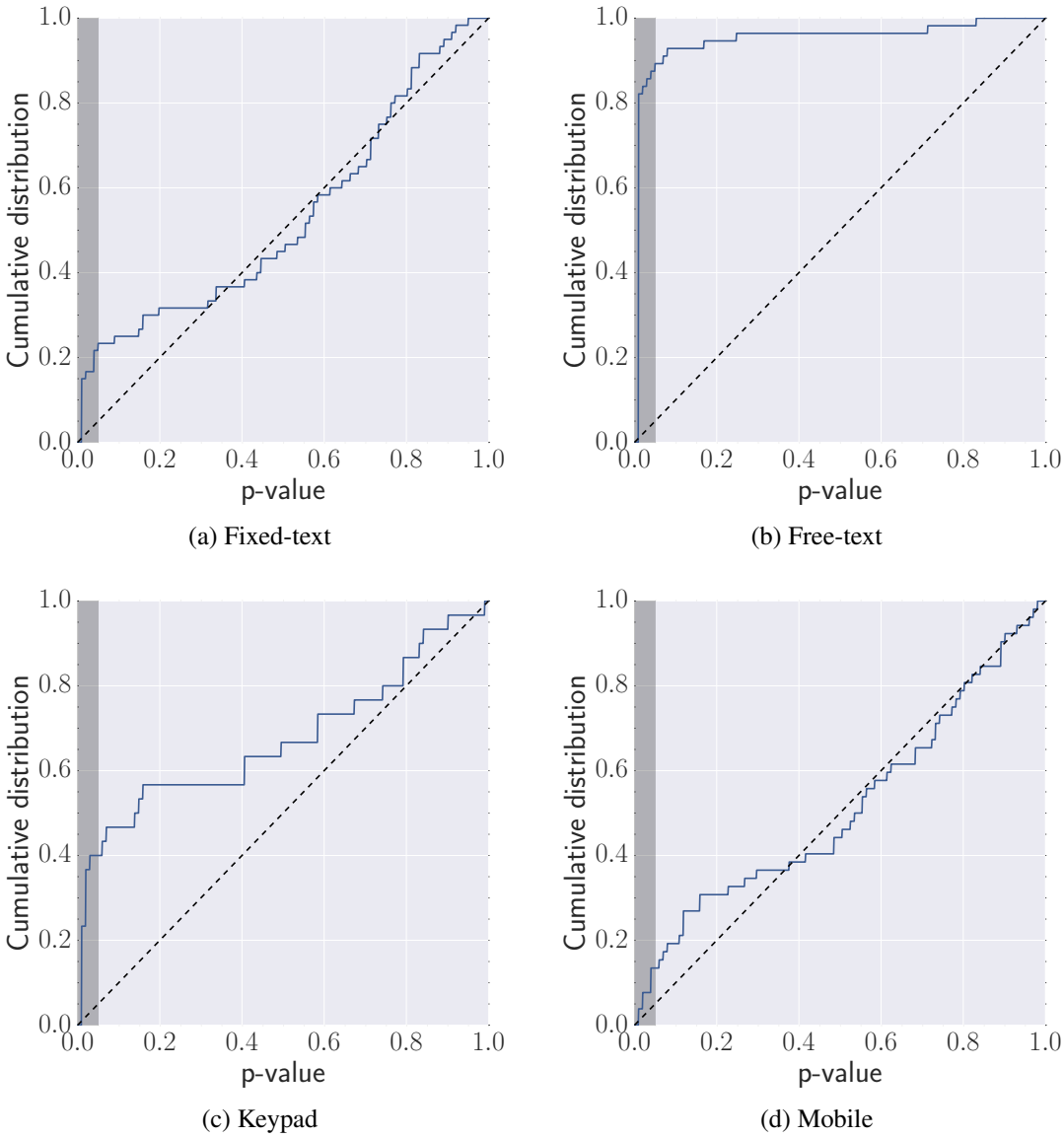
(b) Free-text

Figure 6.2: Keystroke time interval empirical and predicted marginal distribution examples (original figure).

	HMM	POHMM
Keystroke (fixed)	0.32	0.23
Keystroke (free)	0.98	0.89
Keypad	0.77	0.40
Mobile	0.56	0.13

Table 6.2: Keystroke goodness of fit test results, showing the proportion of users that reject the null hypothesis that the model is consistent with the data.

which the model is rejected is shown in Table 6.2. The p-value distributions indicate that the POHMM is better suited for fixed-text, keypad, and mobile input, while both the HMM and POHMM are largely rejected for long free-text input. The results suggests that typing behavior is different for long free-text than for long fixed-text. An example of the predicted and empirical time interval distributions for two randomly selected users is shown in Figure 6.2. For fixed-text, the model overestimates time intervals in the 50-100 ms range and underestimates time intervals in the 150-250 ms range. The opposite is true for the free-text example, where the model overestimates the tail of the distribution.



(a) Fixed-text

(b) Free-text

(c) Keypad

(d) Mobile

Figure 6.3: Keystroke goodness of fit p-value distributions testing the null hypothesis that the model is consistent with the data. A uniform distribution is given by the dashed black line (original figure).

	Folds	Dichotomy	POHMM	p-value
Nursery rhymes	4	0.11 (0.04)	**0.00** (0.01)	0.003
Keystroke (fixed)	4	0.13 (0.02)	**0.08** (0.04)	0.041
Keystroke (free)	6	**0.02** (0.01)	0.06 (0.01)	8.9×10^{-5}
Keypad	20	0.11 (0.03)	**0.05** (0.02)	1.3×10^{-8}
Mobile (w/o sensors)	20	0.20 (0.03)	**0.10** (0.02)	2.7×10^{-14}
Mobile (w/ sensors)	20	0.01 (0.01)	0.01 (0.01)	0.500

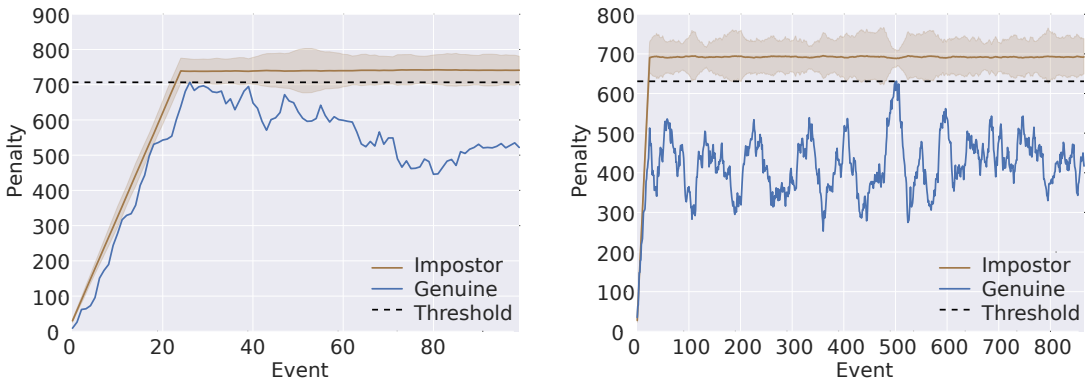
Table 6.3: Keystroke benchmark verification results comparing the EER of the POHMM to the dichotomy classifier (Monaco et al., 2013). Bold EER indicates a significant result at the 0.05 level.

6.1.2 Identification and verification

A benchmark experiment compares the POHMM to a discriminative model with previously published results. Verification performance is obtained for the POHMM, as described in Section 5.3, and the dichotomy classifier with a standard set of keystroke features, as described in Monaco et al. (2013). Results are obtained using the same stratified k-fold cross validation for both classifiers. Standard deviations of the EER are estimated from the performance of each fold, and the number of folds in each dataset is equal to the number of samples for each user.

The POHMM significantly outperforms the dichotomy model in 4 out of 6 datasets. These results are consistent with previous work that indicates discriminative models have better asymptotic performance while generative models perform well with smaller amounts of data (Jordan, 2002). The dichotomy classifier significantly outperforms the POHMM for the free-text dataset only. The performance of both classifiers is comparable for the mobile dataset with all sensors.

Continuous verification results are obtained as described in Section 5.3.1. The penalty functions for two randomly selected users are shown in Figure 6.4. In both examples, impostors are detected after about 20 keystrokes. The penalty function for the free-text example demonstrates behavior seen in a bounded random walk (Nicolau, 2002). Typing behavior deviates from an expected penalty of approximately 300-500 and is consistently pulled back into that range.



(a) Fixed-text (b) Free-text

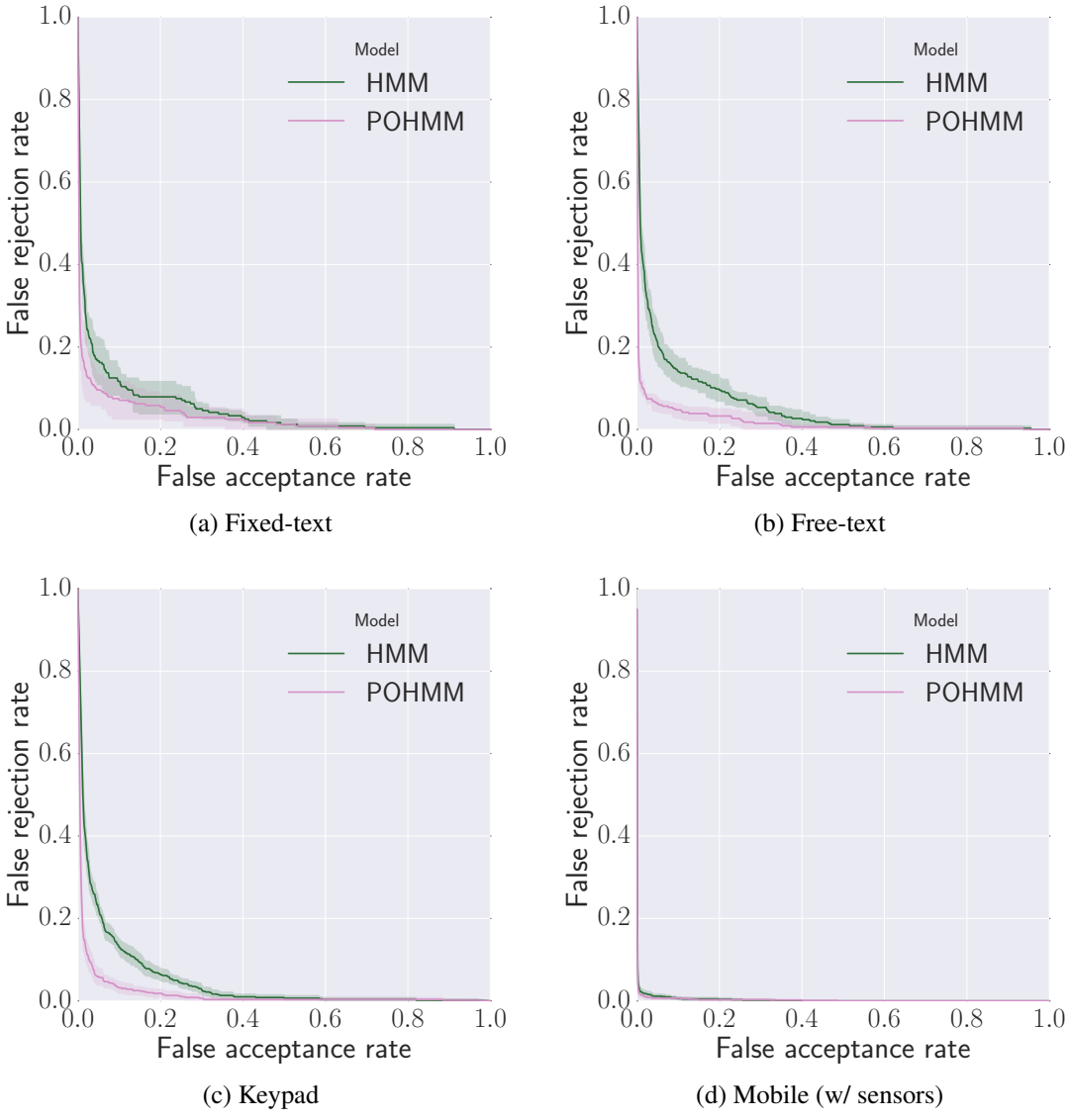
Figure 6.4: Keystroke continuous verification examples (original figure).

Figure 6.5 shows the ROC curves obtained for each dataset using the HMM and POHMM. Table 6.4 summarizes identification and verification performance results for each keystroke dataset using both the HMM and POHMM. Results for a subset of the fixed-text keystroke dataset are included here, in which users had to copy a nursery rhyme containing 100-150 characters. The POHMM outperforms the HMM for most criteria. Continuous verification results are comparable to those obtained on different keystroke datasets (Bours, 2012). For fixed-text tasks, impostors are locked out of the system after 25 keystrokes on average. For free-text input, it takes about 4 times as many keystrokes for an intruder to be detected. On mobile devices an intruder can typically be detected after only 1 keystroke, which corresponds to 0 lockout time (since the single keystroke results in a penalty score above the lockout threshold).

6.2 Bitcoin transactions

Bitcoin transactions can either be incoming or outgoing. An incoming transaction is one in which the user receives bitcoin from another user and an outgoing transaction is one in which the user sends bitcoin to another user. The direction of the transaction is used as the partially observable state in the POHMM.

Figure 6.6 shows the fitted marginal time interval distributions in each state for two randomly selected users. In this example, the passive states have a heavier tail than the active



(a) Fixed-text

(b) Free-text

(c) Keypad

(d) Mobile (w/ sensors)

Figure 6.5: Keystrokes ROC curves comparing the POHMM to the HMM (original figure).

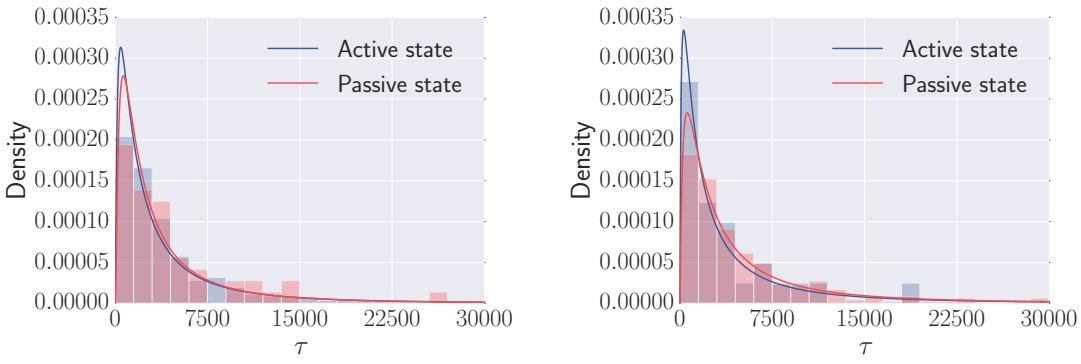
	Model	ACC	EER	AUC	AMRT
Nursery rhymes	HMM	0.88 (0.05)	0.03 (0.02)	0.01 (0.00)	34.79 (19.91)
	POHMM	0.99 (0.02)	0.00 (0.01)	0.00 (0.00)	25.27 (13.04)
Keystroke (fixed)	HMM	0.56 (0.04)	0.10 (0.03)	0.05 (0.02)	30.41 (14.88)
	POHMM	0.75 (0.07)	0.08 (0.04)	0.03 (0.02)	24.55 (13.54)
Keystroke (free)	HMM	0.53 (0.07)	0.13 (0.03)	0.05 (0.02)	162.0 (128.5)
	POHMM	0.82 (0.05)	0.06 (0.01)	0.02 (0.01)	103.2 (112.5)
Keypad	HMM	0.58 (0.08)	0.12 (0.02)	0.05 (0.01)	4.24 (1.83)
	POHMM	0.77 (0.06)	0.05 (0.02)	0.02 (0.01)	3.21 (1.98)
Mobile (w/o sensors)	HMM	0.42 (0.05)	0.15 (0.02)	0.07 (0.01)	5.23 (2.02)
	POHMM	0.59 (0.05)	0.10 (0.02)	0.04 (0.01)	4.16 (2.15)
Mobile (w/ sensors)	HMM	0.93 (0.04)	0.02 (0.01)	0.00 (0.00)	0.16 (0.68)
	POHMM	0.94 (0.02)	0.01 (0.01)	0.00 (0.00)	0.15 (0.64)

Table 6.4: Keystroke identification and verification results.

	Active (49%)		Passive (51%)	
	In	Out	In	Out
Time interval	1574.02	1784.42	2593.81	2695.49
Minute of hour	28.66	28.64	29.18	28.84
Hour of day	11.63	11.62	11.55	11.59
Day of week	2.98	2.92	2.90	2.87
฿value	38.96	-99.94	27.24	-80.75
Num. inputs	0.13	7.27	0.10	6.01
Num. outputs	1.77	0.84	1.70	0.75
Input/output balance	1.64	-6.43	1.60	-5.26

Table 6.5: Bitcoin average estimated emission parameters.

states. The average model parameters from models fit to all Bitcoin samples are shown in Table 6.5. The incoming transactions have shorter time intervals than outgoing transactions in both states. In the active state, higher bitcoin values are sent and received compared to the passive state. The transactions in the active state also have more connections on average.



(a) User A

(b) User B

Figure 6.6: Bitcoin time interval distribution examples showing the model time interval distribution in each state and empirical densities of the classified events as determined by the modified Viterbi algorithm (original figure).

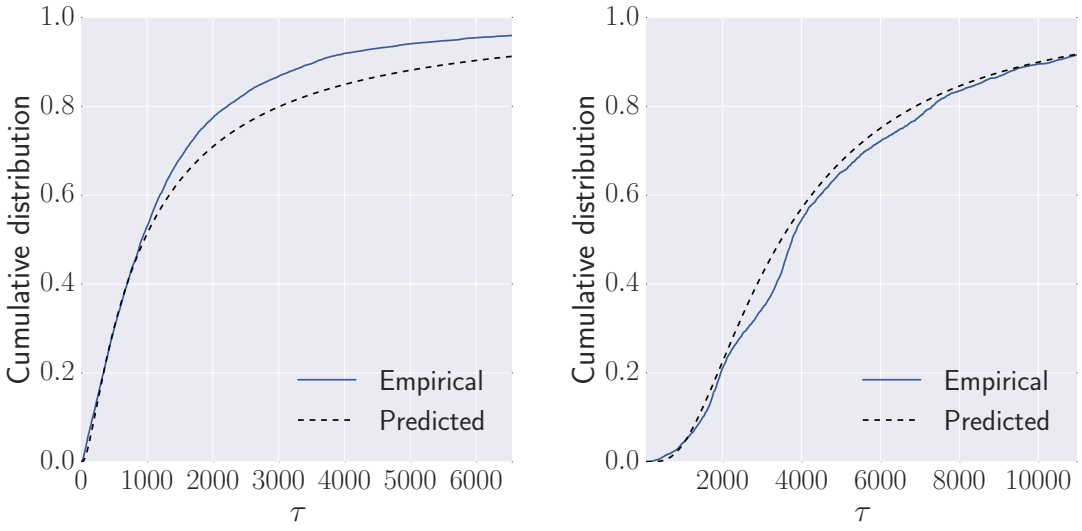
6.2.1 Goodness of fit

In a Monte Carlo goodness of fit test, the POHMM is rejected for 69% of samples and the HMM for 67% of samples. This indicates that the POHMM and HMM are roughly equivalent in terms of explaining the data. Examples of the predicted and empirical CDF are shown in Figure 6.7. In the example, the tail distribution of time intervals for User A is underestimated. The circadian and weekly rhythms are apparent in the empirical distribution of User B. The p-value distributions of the HMM and POHMM are shown in Figure 6.8.

6.2.2 Identification and verification

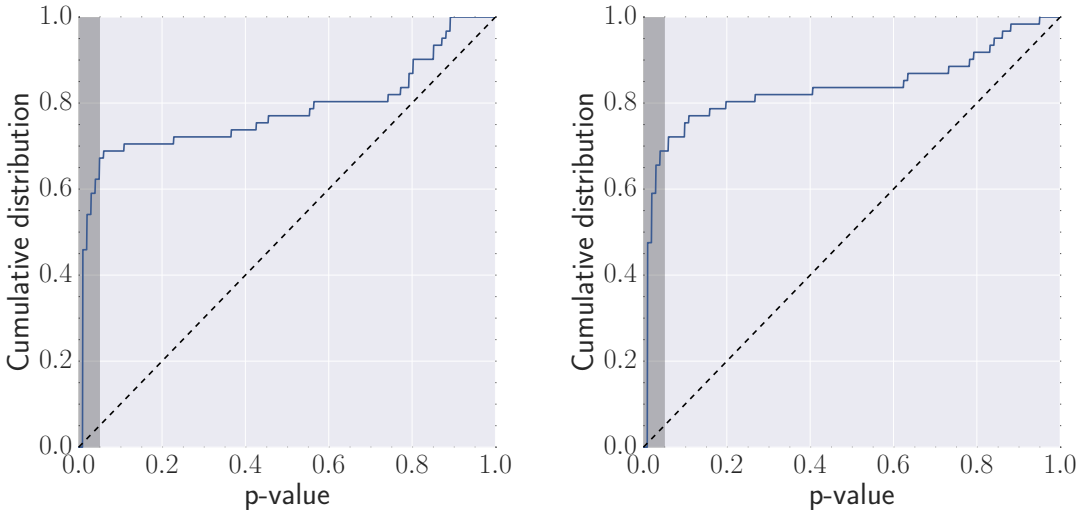
Verification results are obtained for the POHMM using three feature groups: time intervals only, timing features, and all features. The timing features consist of time intervals, minute of hour, hour of day, and day of week. The ROC curves for the three feature groups are shown in Figure 6.9. The timing features perform worse than time intervals only, and using all features improves performance.

Identification and verification results for the HMM and POHMM using the three feature groups are summarized in Table 6.6. The POHMM consistently outperforms the HMM. On



(a) User A (b) User B

Figure 6.7: Bitcoin time interval empirical and predicted marginal distribution examples (original figure).



(a) HMM (rejected 67%) (b) POHMM (rejected 69%)

Figure 6.8: Bitcoin goodness of fit p-value distributions testing the null hypothesis that the model is consistent with the data. A uniform distribution is given by the dashed black line (original figure).

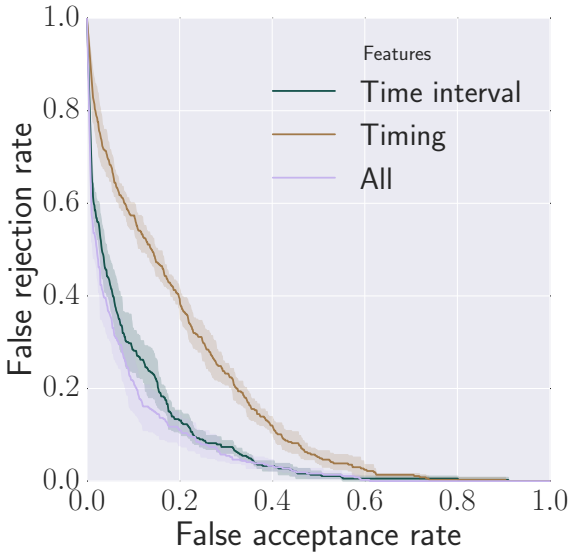


Figure 6.9: Bitcoin ROC curves comparing features three feature groups (original figure).

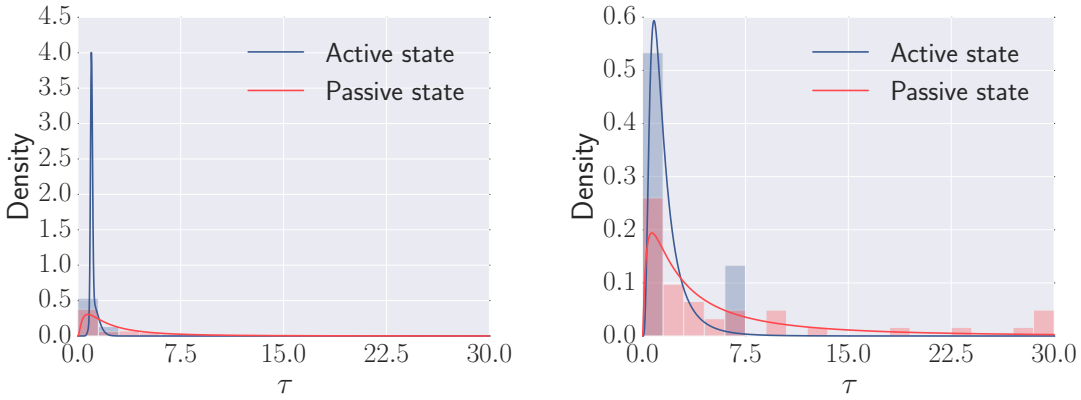
Features	Model	ACC	EER	AUC	AMRT
Time interval	HMM	0.26 (0.04)	0.21 (0.02)	0.13 (0.01)	225.02 (217.91)
	POHMM	0.36 (0.04)	0.17 (0.02)	0.09 (0.01)	201.04 (194.56)
All timing	HMM	0.16 (0.07)	0.32 (0.02)	0.23 (0.01)	232.06 (211.59)
	POHMM	0.18 (0.07)	0.27 (0.02)	0.18 (0.01)	203.60 (178.74)
All	HMM	0.32 (0.04)	0.22 (0.01)	0.12 (0.00)	163.00 (221.83)
	POHMM	0.42 (0.06)	0.14 (0.03)	0.07 (0.02)	138.85 (214.03)

Table 6.6: Bitcoin identification and verification results.

average, impostors are detected after 140 transactions using the POHMM with all features. The timing features generally perform worse than time intervals alone, an indication that minute of hour, hour of day, and day of week are poor indications of Bitcoin transaction behavior.

6.3 Linux kernel commits

The p-states for Linux kernel commits are defined as follows. Commits that contain at least one of the words {"bug", "patch", "error", "fix"} in the description are labeled as bug fixes,



(a) User A (b) User B

Figure 6.10: Linux kernel commit time interval distribution examples showing the model time interval distribution in each state and empirical densities of the classified events as determined the modified Viterbi algorithm (original figure).

and commits that do not contain any of those words are labeled as feature additions. The type of commit, either bug fix or feature addition, is used as the p-state.

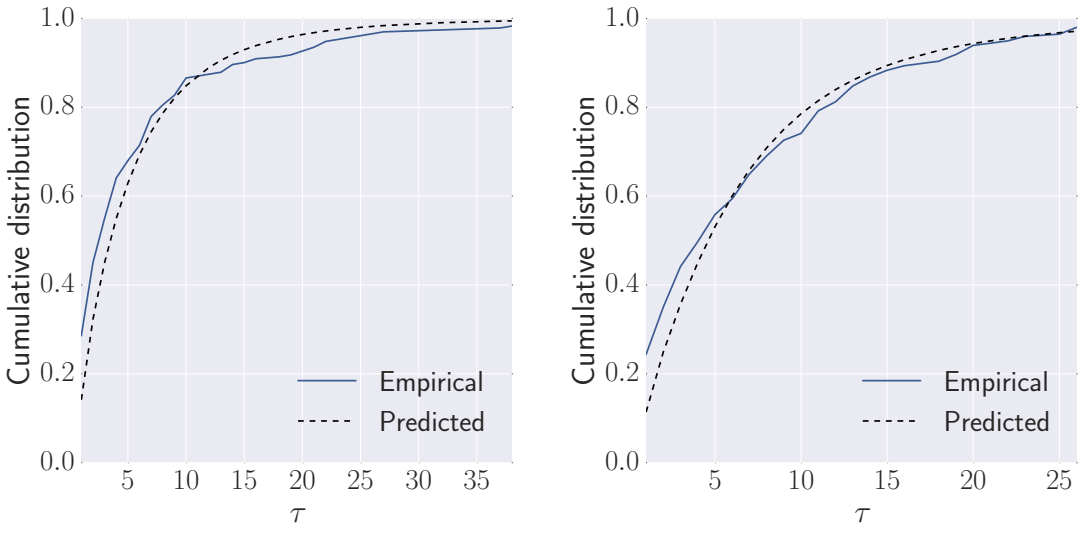
Figure 6.10 shows the fitted time interval distributions for two randomly selected users. Both examples show a clear separation between active and passive states. The average model parameters for all samples are shown in Table 6.7. In both states, bug fixes have slightly longer time intervals than feature additions. This indicates that features additions are generally authored quicker than bug fixes. However, the bug fixes have a lower time to commit in both states, indicating that the integrations of bug fixes have higher priority than new features. Descriptions also tend to be longer in the active state, and shorter for feature additions in both states.

6.3.1 Goodness of fit

In a Monte Carlo goodness of fit test using time intervals only, the HMM is rejected for 98% of samples and POHMM for 85% of samples. This indicates that neither model adequately explains the data, and that the POHMM provides a better fit than the HMM. An example of the empirical and predicted distributions for two randomly selected users is shown in Figure 6.11.

	Active (47%)		Passive (53%)	
	Bug fix	Feature	Bug fix	Feature
Time interval	2.06	1.90	4.30	4.09
Day of week	2.46	2.49	2.32	2.37
Week of month	1.85	1.77	1.81	1.83
Time to commit	14.98	21.28	13.81	20.73
Num. edits	4.65	2.99	4.45	2.44
Num. committers	1.39	1.22	1.27	1.13
Description chars	2017.57	909.75	1697.27	764.52
Description words	133.40	64.09	116.81	59.79

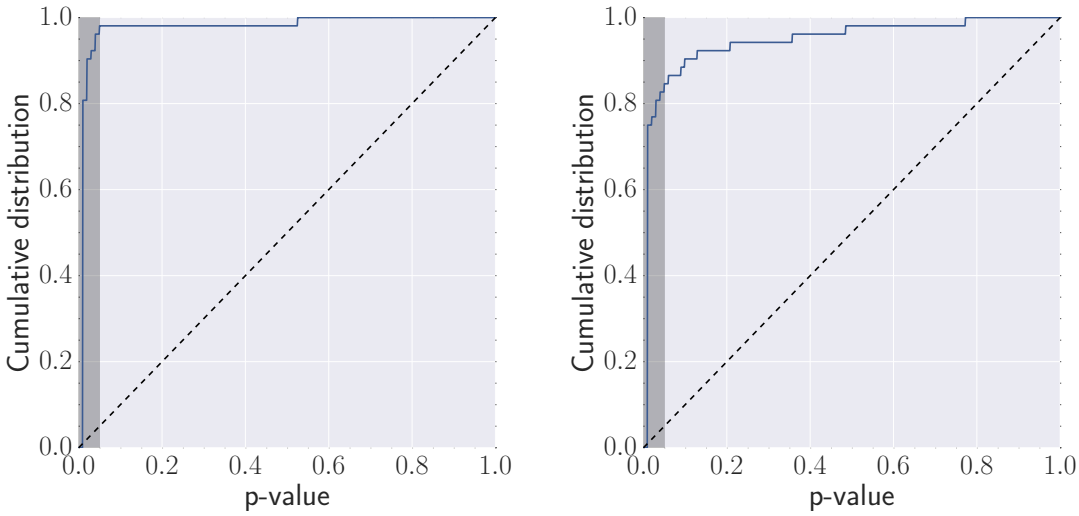
Table 6.7: Linux kernel commit average estimated emission parameters.



(a) User A

(b) User B

Figure 6.11: Linux kernel commit time interval empirical and predicted marginal distribution examples (original figure).



(a) HMM (rejected 98%)

(b) POHMM (rejected 85%)

Figure 6.12: Linux kernel commit goodness of fit p-value distributions testing the null hypothesis that the model is consistent with the data. A uniform distribution is given by the dashed black line (original figure).

6.3.2 Identification and verification

The ROC curves for three features groups are shown in Figure 6.12. Similar to Bitcoin, the time intervals alone perform better than timing features and the best results are obtained using all features. Verification and identification results are summarized in Table 6.8, where the HMM and POHMM performance is roughly equivalent.

Features	Model	ACC	EER	AUC	AMRT
Time interval	HMM	0.04 (0.03)	0.39 (0.02)	0.34 (0.03)	44.02 (17.23)
	POHMM	0.03 (0.01)	0.41 (0.02)	0.36 (0.02)	44.84 (17.43)
All timing	HMM	0.06 (0.02)	0.44 (0.02)	0.38 (0.03)	41.23 (23.06)
	POHMM	0.06 (0.02)	0.43 (0.02)	0.39 (0.02)	40.14 (22.43)
All	HMM	0.20 (0.08)	0.34 (0.03)	0.29 (0.02)	40.37 (32.71)
	POHMM	0.17 (0.06)	0.36 (0.01)	0.30 (0.01)	40.81 (31.45)

Table 6.8: Linux kernel commit identification and verification results.

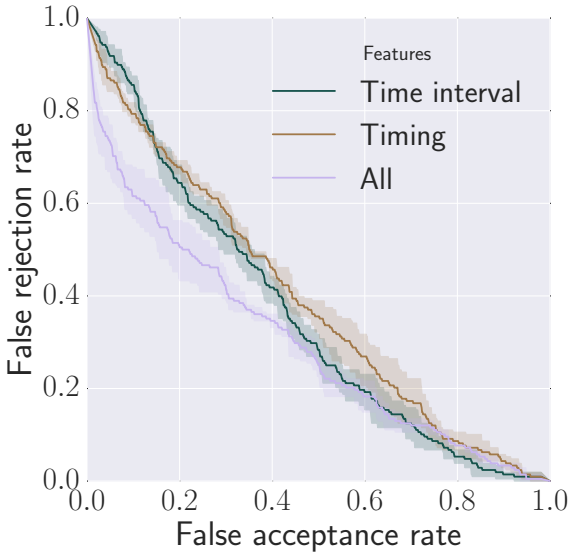


Figure 6.13: Linux kernel commit ROC curves comparing features three feature groups (original figure).

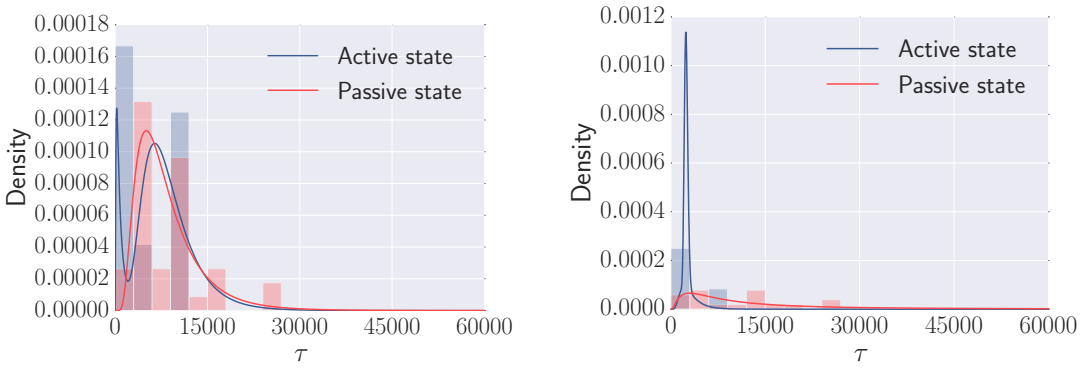
6.4 White House visits

The group size is used as the p-state for White House visits. Visits with greater than 10 people are labeled as large groups and 10 people or less are labeled as small groups. The time interval distributions and fitted models for two randomly selected users are shown in Figure 6.14. In both examples, the passive state has a heavier tail than the active state.

The average estimated model parameters over all samples are shown in Table 6.9. In both the active and passive states, large groups tend to have longer time intervals and are scheduled earlier in the day. Small groups tend to arrive earlier in both states. The appointments made for large groups in the passive state are usually farther in advance than the active state, shown by the larger mean forecast parameter.

6.4.1 Goodness of fit

Two examples of the predicted and empirical time interval distributions are shown in Figure 6.15 and results of a Monte Carlo goodness of fit test in Figure 6.16. The examples demonstrate the predicted distributions visually agree with the empirical distributions with



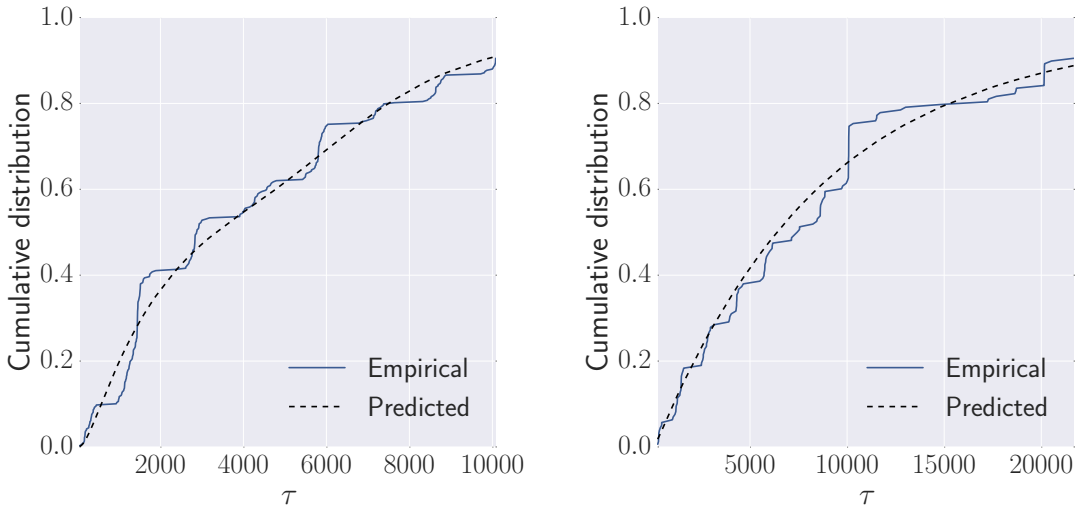
(a) User A

(b) User B

Figure 6.14: White House visit time interval distribution examples showing the model time interval distribution in each state and empirical densities of the classified events as determined the modified Viterbi algorithm (original figure).

	Active (49%)		Passive (51%)	
	Small	Large	Small	Large
Time interval	3058.90	3962.98	13389.47	14216.55
Hour of day	12.42	11.69	12.38	11.47
Day of week	2.19	2.36	1.91	2.14
Week of month	1.71	1.79	1.82	1.79
Duration	102.93	126.22	115.27	112.29
Tardiness	-5.09	-3.81	-6.47	-3.58
Forecast	5013.59	4072.79	5110.75	8430.40
Num. people	6.67	39.41	6.69	40.07

Table 6.9: White House visit average estimated emission parameters.



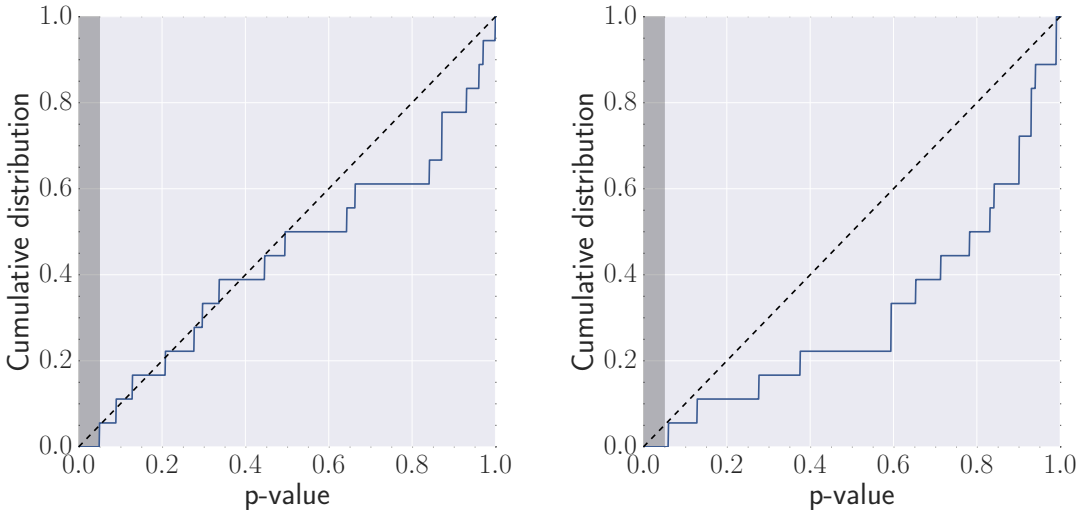
(a) User A (b) User B

Figure 6.15: White House visit time interval empirical and predicted marginal distribution examples (original figure).

apparent circadian and weekly rhythms. The p-value distributions indicate the POHMM is consistent with the data, with a 0% rejection rate, compared to the HMM, which has a 6% rejection rate.

6.4.2 Identification and verification

Verification results are summarized in Figure 6.17, which shows the ROC curves obtained using the POHMM with three different feature groups. Unlike Bitcoin and kernel commits, the inclusion of all timing features yield better verification performance than time intervals alone. Identification and verification results are summarized in Table 6.10. The POHMM generally performs better than the HMM. Using all features, an impostor is detected after 18 visits on average.



(a) HMM (rejected 6%)

(b) POHMM (rejected 0%)

Figure 6.16: White House visit goodness of fit p-value distributions testing the null hypothesis that the model is consistent with the data. A uniform distribution is given by the dashed black line (original figure).

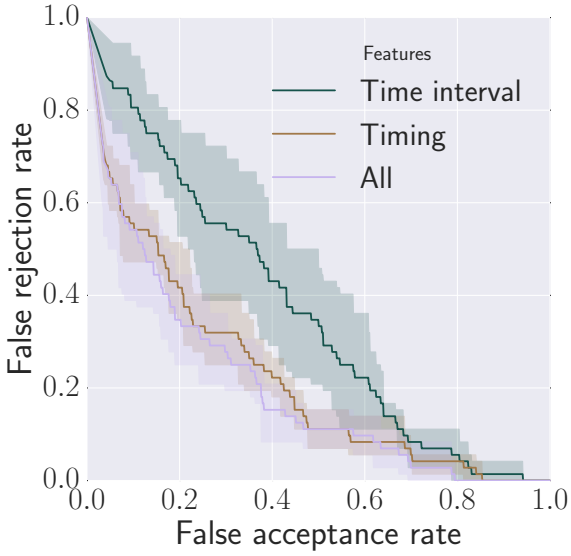


Figure 6.17: White House visit ROC curves comparing features three feature groups (original figure).

Features	Model	ACC	EER	AUC	AMRT
Time interval	HMM	0.08 (0.07)	0.41 (0.05)	0.40 (0.06)	26.32 (7.67)
	POHMM	0.14 (0.11)	0.42 (0.07)	0.36 (0.09)	25.25 (7.84)
All timing	HMM	0.28 (0.08)	0.32 (0.05)	0.26 (0.02)	21.34 (11.31)
	POHMM	0.32 (0.05)	0.31 (0.05)	0.22 (0.04)	21.06 (9.76)
All	HMM	0.40 (0.13)	0.27 (0.02)	0.18 (0.02)	18.11 (9.95)
	POHMM	0.31 (0.13)	0.28 (0.05)	0.20 (0.05)	18.59 (10.43)

Table 6.10: White House visit identification and verification results.

Criterion	Description
P	Political, economic, religious, or social motivations
I	Intent to coerce or intimidate a large audience
A	Actions that violate international humanitarian law

Table 6.11: GTD event criteria.

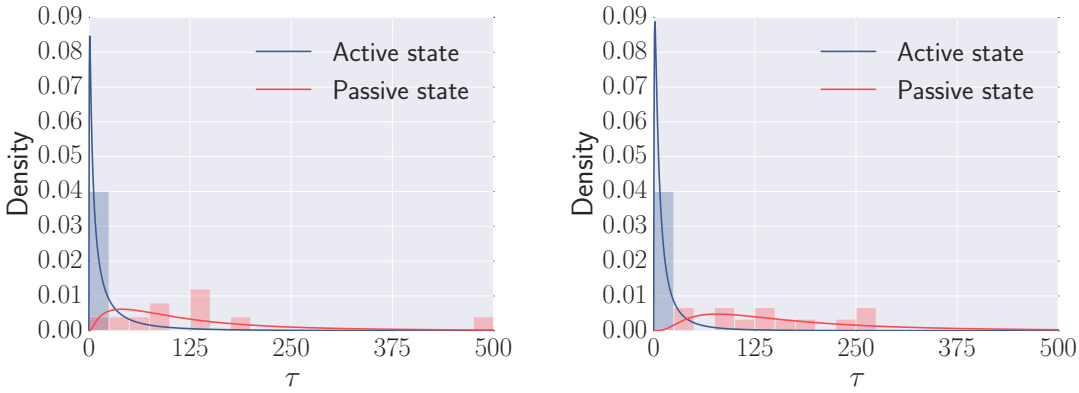
6.5 Terrorist activity

Terrorist events in the GTD are labeled according to 3 different criteria, summarized in Table 6.11. Each event must satisfy at least 2 out of the 3 criteria, therefore there are 4 different possible combinations of criteria for each event. The combination of criteria (PI, PA, IA, or PIA) is used as the p-state in the POHMM.

The time interval distributions and fitted models for two randomly selected groups are shown in Figure 6.18. There is a clear separation between the active and passive states, demonstrated by the heavy tail for passive states. The average model parameters from all samples are shown in Table 6.12. In both the active and passive states, events that intend to intimidate (I criterion) and violate humanitarian law (A criterion) generally have shorter time intervals, and in the active state, the IA events involve more perpetrators.

6.5.1 Goodness of fit

A Monte Carlo goodness of fit test demonstrates both the HMM and POHMM are consistent with the data, shown by the p-value distributions in Figure 6.20. An example of the predicted and empirical time interval distributions for two randomly selected groups is shown in Figure 6.19.



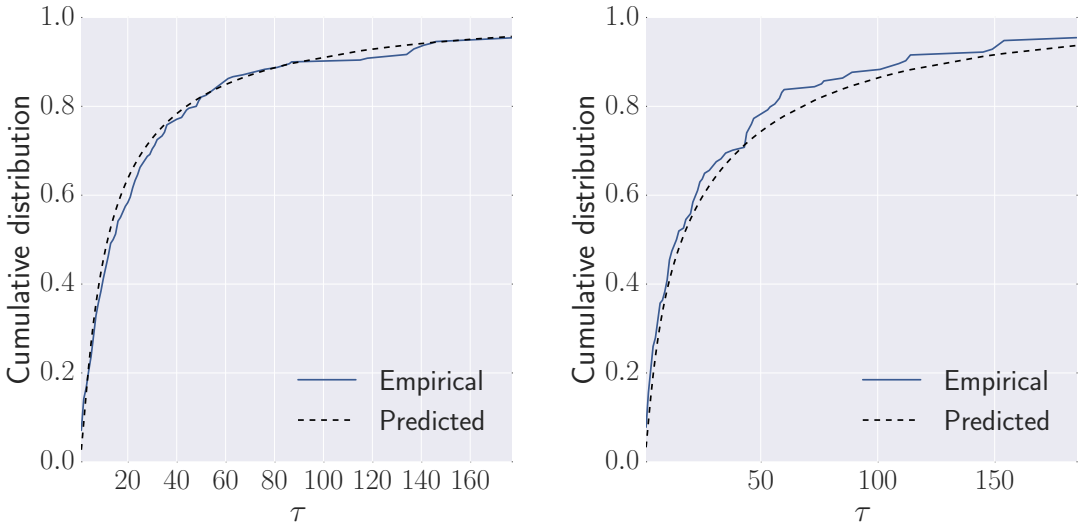
(a) User A

(b) User B

Figure 6.18: Terrorist activity time interval distribution examples showing the model time interval distribution in each state and empirical densities of the classified events as determined by the modified Viterbi algorithm (original figure).

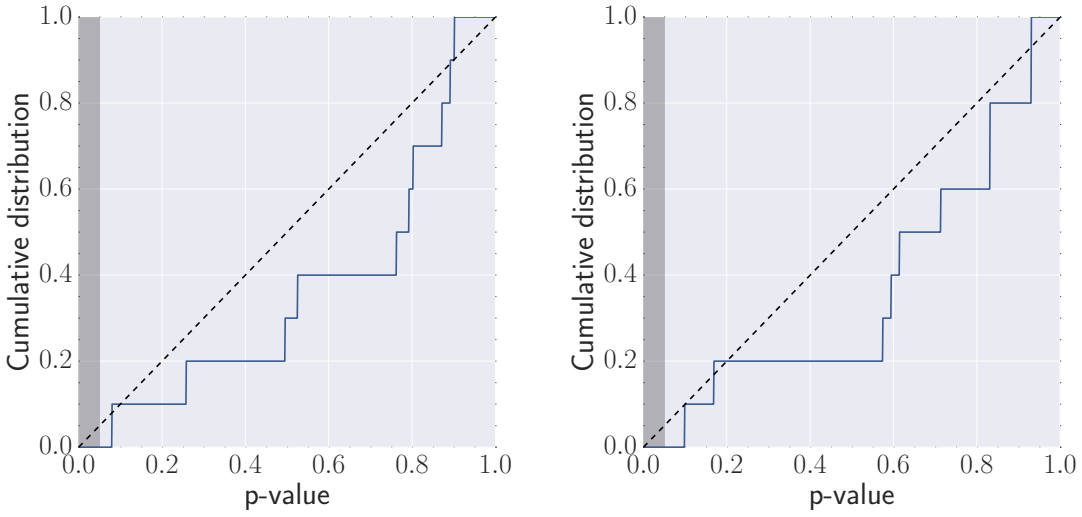
	Active (54%)				Passive (46%)			
	IA	PA	PI	PIA	IA	PA	PI	PIA
Time interval	2.95	13.37	8.12	9.67	2.49	16.83	45.72	65.27
Day of week	3.08	2.66	2.83	2.99	0.82	1.71	3.12	2.95
Week of month	1.42	1.39	1.34	1.68	0.49	0.97	1.79	1.64
Month of year	6.69	5.56	5.76	5.93	2.75	5.21	5.39	6.63
Num. perpetrators	4.06	2.09	1.52	2.66	1.33	1.15	1.19	1.89
Num. killed	3.87	2.03	1.56	1.92	0.15	1.97	4.48	2.84
Num. captured	0.00	0.66	0.02	0.09	0.46	1.54	0.17	0.32
Num. related events	0.95	0.53	0.86	0.93	6.18	1.12	0.17	1.08

Table 6.12: Terrorist activity average estimated emission parameters.



(a) User A (b) User B

Figure 6.19: Terrorist activity time interval empirical and predicted marginal distribution examples (original figure).



(a) HMM (rejected 0%) (b) POHMM (rejected 0%)

Figure 6.20: Terrorist activity goodness of fit p-value distributions testing the null hypothesis that the model is consistent with the data. A uniform distribution is given by the dashed black line (original figure).

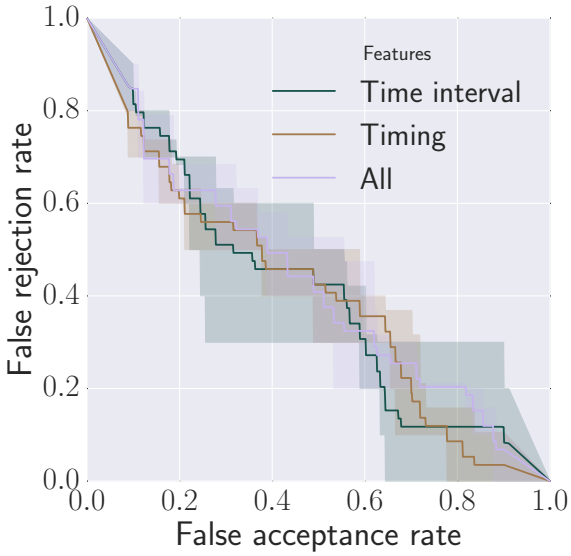


Figure 6.21: Terrorist activity ROC curves comparing features three feature groups (original figure).

Features	Model	ACC	EER	AUC	AMRT
Time interval	HMM	0.19 (0.10)	0.47 (0.07)	0.45 (0.02)	25.75 (26.22)
	POHMM	0.15 (0.05)	0.42 (0.11)	0.41 (0.01)	25.36 (26.08)
All timing	HMM	0.19 (0.10)	0.45 (0.05)	0.41 (0.03)	22.41 (17.11)
	POHMM	0.20 (0.01)	0.46 (0.05)	0.40 (0.03)	23.05 (18.78)
All	HMM	0.17 (0.06)	0.42 (0.04)	0.42 (0.01)	33.59 (67.75)
	POHMM	0.15 (0.05)	0.45 (0.07)	0.43 (0.05)	33.65 (67.30)

Table 6.13: Terrorist activity identification and verification results.

6.5.2 Identification and verification

Verification results are obtained using the three feature groups and summarized in Figure 6.21. The ROC curves indicate that no feature group outperforms any other. This is also confirmed in the summary of identification and verification results in Table 6.13. Performance in roughly equivalent for the HMM and POHMM using any of the three feature groups.

	Absolute clock		Relative clock	
	Baseline	POHMM	Baseline	POHMM
Keystroke (fixed)	0.26	0.26	0.35	0.32
Keystroke (free)	0.32	0.29	0.33	0.30
Bitcoin	0.44	0.44	0.45	0.44
Commits	0.44	0.42	0.37	0.36
Visits	0.43	0.42	0.34	0.34
Terror	0.62	0.56	0.40	0.35

Table 6.14: Time interval prediction results using an absolute and relative clock.

6.6 Prediction

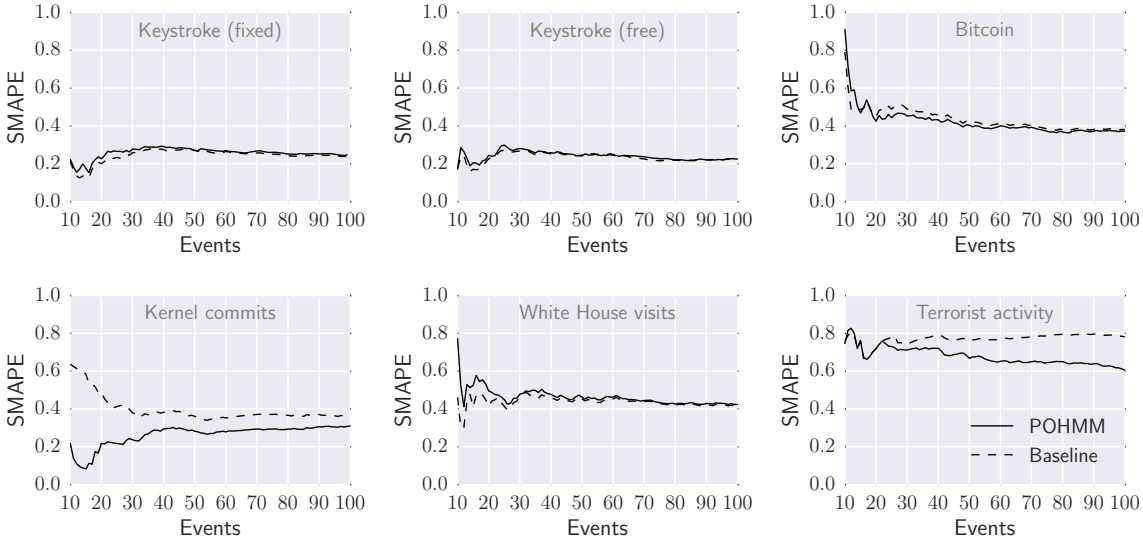
The fitted POHMM is used to make time interval predictions for each dataset, as described in Section 5.4. Baseline predictions are also obtained using the sample mean. Predictions are made by iteratively training the model on N events and predicting the time to the next event. This is performed using both an absolute and relative clock. With a relative clock, time passes as the number of global events[1].

Examples of the SMAPE for a randomly selected user in each dataset is shown in Figure 6.22. In the examples, the POHMM is at least as powerful as the baseline in making predictions. The average SMAPE results are shown in Table 6.14. The greatest improvement over baseline accuracy is obtained for the terrorist activity dataset, where the POHMM provides a 10% improvement over baseline.

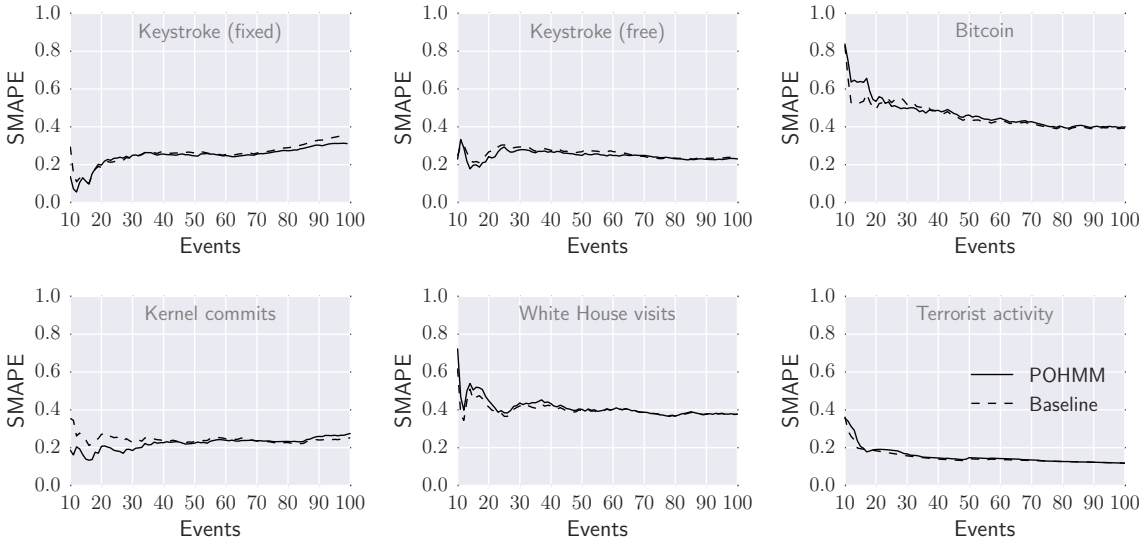
6.7 Summary

Identification and verification results for all datasets are summarized in Table 6.15. The POHMM outperforms the HMM for most datasets, particularly for higher-frequency events including keystroke and Bitcoin transactions. Performance generally increases when all available features are utilized.

[1]Since time as measured by a relative clock is given by the event count, time interval predictions take on a different meaning. In predicting the time to the next event, the relative clock time prediction indicates the number of global events that are expected to occur before the next event.



(a) Absolute clock



(b) Relative clock

Figure 6.22: Time interval SMAPE examples for a randomly selected user from each dataset (original figure).

	Model	ACC	EER	AUC	AMRT
Keyst. (fixed)	HMM	0.35 (0.04)	0.15 (0.02)	0.08 (0.02)	45.06 (17.83)
	POHMM	0.59 (0.10)	0.09 (0.02)	0.04 (0.02)	36.91 (16.55)
Keyst. (free)	HMM	0.59 (0.07)	0.09 (0.01)	0.04 (0.02)	267.19 (168.61)
	POHMM	0.85 (0.03)	0.05 (0.02)	0.02 (0.02)	201.70 (162.39)
Bitcoin	HMM	0.26 (0.04)	0.21 (0.02)	0.13 (0.01)	225.02 (217.91)
	POHMM	0.36 (0.04)	0.17 (0.02)	0.09 (0.01)	201.04 (194.56)
Commits	HMM	0.04 (0.03)	0.39 (0.02)	0.34 (0.03)	44.02 (17.23)
	POHMM	0.03 (0.01)	0.41 (0.02)	0.36 (0.02)	44.84 (17.43)
Visits	HMM	0.08 (0.07)	0.41 (0.05)	0.40 (0.06)	26.32 (7.67)
	POHMM	0.14 (0.11)	0.42 (0.07)	0.36 (0.09)	25.25 (7.84)
Terror	HMM	0.19 (0.10)	0.47 (0.07)	0.45 (0.02)	25.75 (26.22)
	POHMM	0.15 (0.05)	0.42 (0.11)	0.41 (0.01)	25.36 (26.08)

(a) Time intervals only.

	Model	ACC	EER	AUC	AMRT
Keyst. (fixed)	HMM	0.56 (0.04)	0.10 (0.03)	0.05 (0.02)	30.41 (14.88)
	POHMM	0.75 (0.07)	0.08 (0.04)	0.03 (0.02)	24.55 (13.54)
Keyst. (free)	HMM	0.53 (0.07)	0.13 (0.03)	0.05 (0.02)	162.03 (128.52)
	POHMM	0.82 (0.05)	0.06 (0.01)	0.02 (0.01)	103.25 (112.53)
Bitcoin	HMM	0.32 (0.04)	0.22 (0.01)	0.12 (0.00)	163.00 (221.83)
	POHMM	0.42 (0.06)	0.14 (0.03)	0.07 (0.02)	138.85 (214.03)
Commits	HMM	0.20 (0.08)	0.34 (0.03)	0.29 (0.02)	40.37 (32.71)
	POHMM	0.17 (0.06)	0.36 (0.01)	0.30 (0.01)	40.81 (31.45)
Visits	HMM	0.40 (0.13)	0.27 (0.02)	0.18 (0.02)	18.11 (9.95)
	POHMM	0.31 (0.13)	0.28 (0.05)	0.20 (0.05)	18.59 (10.43)
Terror	HMM	0.17 (0.06)	0.42 (0.04)	0.42 (0.01)	33.59 (67.75)
	POHMM	0.15 (0.05)	0.45 (0.07)	0.43 (0.05)	33.65 (67.30)

(b) All features.

Table 6.15: Identification and verification results summary.

Chapter 7

Privacy issues

The issue of privacy in behavioral monitoring has recently started to gain attention. It is not clear how many organizations routinely track user behavior and to what extent this information is being used. On the web, the situation is further complicated by the prevalence of third-party content which allows identifying attributes to be seen by a number of websites as a result of a user visiting only a single page (Mayer and Mitchell, 2012). Behavior tracking is motivated by targeted advertising, analytics, and law enforcement agencies. The laws and regulations currently surrounding behavior tracking are relatively lax, although a majority of users negatively view behavioral targeting. In 2012, a study by Pew Research Group found 68% of respondents to be "not okay" with behavioral targeted advertising, and a 2010 USA Today/Gallup poll showed 67% of respondents felt that behavioral targeting should not be legal (Center, 2012; Gallup, 2010).

Behavioral patterns and identity may be unintentionally leaked through event timestamps alone, as demonstrated in Chapter 6. The simplicity and ubiquitous nature of timestamped events makes this attractive as a biometric modality. It is not difficult to capture the key-press times from an interactive applicative over a network or the financial transaction times from a peer-to-peer digital currency. Masking spatial information, *e.g.* through TOR (Tor, 2015), does not guarantee anonymity when temporal behavior can be used for identification. Over time, a temporal behavior profile can be established to identify and predict the behavior of a user. This chapter introduces some techniques that can be used to prevent identification and prediction based on event timestamps.

7.1 Masking temporal behavior

There are primarily two goals in masking temporal behavior. The first is to limit an adversary's capability to identify. Under this goal, the user wishes to remain anonymous within a pool of U users. Anonymity is achieved when the probability of correctly identifying the user is no greater than $\frac{1}{U}$. With cooperation from all the users in the pool, it is relatively easy to obtain perfect anonymity for every user. Each user needs only to behave in some predefined way agreed upon by the pool. If the behavior of every user is exactly the same, the best strategy for identification is simply a random guess out of the U users.

The second goal is to limit an adversary's capability to predict behavior. This goal is quite different, in that the user wishes for the time of a future event to be unpredictable after having generated N events. Perfect unpredictability is achieved when the expected SMAPE of the time to the next event goes to 1. The SMAPE can go to 1 when either the actual time interval, τ_{N+1}, or predicted time interval, $\hat{\tau}_{N+1}$, is infinite. The actual time interval is infinite when the user plans on waiting indefinitely before generating another event. The predicted time interval is infinite if the adversary's model predicts an infinite estimated delay until the next event. Neither of these situations are practical and thus perfect unpredictability can generally not be achieved. Despite this, even a moderate increase in the SMAPE can be effective in practice.

Both goals in masking temporal behavior call for different masking strategies. Consider two extreme strategies. In the first, every user decides to generate events with exactly the same frequency. That is, the time between events, τ, is the same for every user. Using this strategy, perfect anonymity is achieved. However, each user's actions can also be predicted exactly and the masking strategy fails to address the predictability criterion. Consider a different strategy, in which each user generates events according to a Poisson process. Let the users have rates $\lambda_1 \ll \cdots \ll \lambda_U$. On a per-user basis, predictive accuracy is relatively low, as the time intervals are independent and identically distributed, following an exponential distribution. However, users can be easily identified over time by their expected time interval, or event frequency.

A more effective strategy would be to let each user generate events according to a Poisson process with rate λ. With cooperation from every user, perfect anonymity is achieved

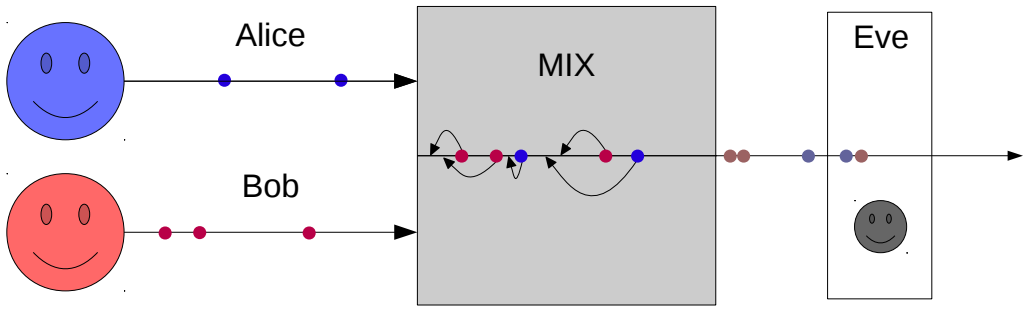


Figure 7.1: Chaum mix (original figure).

as the Poisson process is memoryless and the resulting distributions of time intervals from each user will appear to be the same. Unfortunately, this strategy is generally not practical since it requires the cooperation from every user. Some users who naturally generate events at a rate much lower than λ would be forced to increase their speed. Conversely, users who naturally generate events at a rate much higher than λ would be forced to slow down.

Chaum mixes can be used to provide anonymity to users behind a router, in which cooperation is assumed (Chaum, 1981). This scenario is shown in Figure 7.1, where users Alice and Bob are behind a router and wish to transmit a series of packets. The *mix* reorders the packets by introducing a random delay to each packet. An eavesdropper, Eve, observes the sequence of reordered packets, however cannot discern whether any packet came from Alice or Bob. The identity and temporal behavior of Alice and Bob are protected by the mix.

The anonymity of the mixing strategy is given by the expected entropy of the observed packets,

$$\mathscr{A} = \lim_{N\to\infty} \frac{1}{N} \mathsf{E}\left[H\left(U_1, \ldots, U_N\right)\right]$$

where U_n is the identity of the n^{th} observed packet and H is the entropy function. Without any constraints, the optimal strategy that maximizes anonymity can be shown to require an infinite delay (Mishra and Venkitasubramaniam, 2011). This is due to the mix waiting for all packets from both Alice and Bob and then randomly selecting one of the $\binom{2N}{N}$

permutations for the packet reordering. This strategy is not practical for a number of reasons, one being that it would require an infinite buffer size.With a finite buffer size, the upper bound on the maximum-achievable anonymity decreases. Despite this, a reasonable level of anonymity can be achieved with practical constraints (Venkitasubramaniam and Anantharam, 2008).

Time interval mix

Since the Chaum mix requires the cooperation of all the users, *i.e.* every user must be on the same network, it is a global mixing strategy, or *global mix*. If the assumption of cooperation is relaxed, then a mixing strategy without cooperation must be developed. This type of mixing strategy is a *user mix*. A user mix is appropriate when there is no cooperation between users, or the events from different users are separated by both space and time[1].

Instead of a global mix, the events from each user pass through a user mix, which operates exclusively on the events from that user. The mix for each user is atomic, in the sense that it operates independently of every other user. There can be different mixing strategies employed by different users and some users with no mix at all. The mix separates the generating process (the sequence of events from the perspective of the user) from the arrival process (the sequence of events as seen by the application). The goal of the mix is to provide anonymity and unpredictability at the arrival process.

This scenario is depicted in Figure 7.2, where Alice and Bob both generate events and wish to remain anonymous and unpredictable. An eavesdropper, Eve, is able to observe the sequence of events from each user after passing through the mix and knows that there are two users in the system. From the perspective of Eve, it is not clear which sequence belongs to each user, as she can only observe the arrival processes. The technical conditions of the user mix are summarized as follows.

1. Alice and Bob have zero knowledge of each other's behavior and don't cooperate.

[1]For example, consider keystrokes recorded by a web application. The keystrokes from two different sessions have been recorded days apart and come from entirely different network locations.

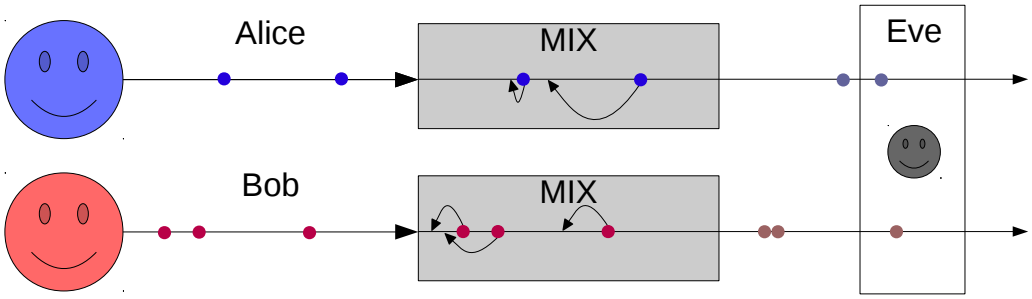


Figure 7.2: Time interval mix scenario (original figure).

2. Eve can see multiple arrival processes, but cannot discern which process belongs to each user

3. Each mix has a source of randomness unknown to Eve.

4. Event order is preserved by the mix.

A time interval mix is a type of user mix that sits between the user and application with the purpose of introducing noise to the time intervals. The mix delays events by temporarily storing them in a buffer before releasing them to the application. It is necessary that the order of events is preserved, and the only action that can be performed is to delay an event.

There are a few caveats in masking temporal behavior. Since the only action is to delay, a lag is introduced between the generation of an event by the user and the observation of the event in the application. Thus, it may not be possible to use a mix in some real-time systems. For many human-computer interaction applications, a small lag is acceptable and would not be noticed by the user. As the lag increases, the movement time and error rate on behalf of the user also increase (MacKenzie and Ware, 1993).

A user mix should possess several properties, summarized in Table 7.1. The expected lag between the generated events and observed events should be finite. The size of the event buffer ultimately determines how large the lag can grow to. If the tolerated lag is unbounded, then the number of events that need to be stored will eventually exceed the size of the buffer. A user mix should also protect the user from being identified out of

Finite	The expected delay between the user and the arrival process should not grow unbounded.
Anonymous	The mix should make it difficult to identify the user.
Unpredictable	The mix should make it difficult to predict future behavior.

Table 7.1: Time interval mix properties.

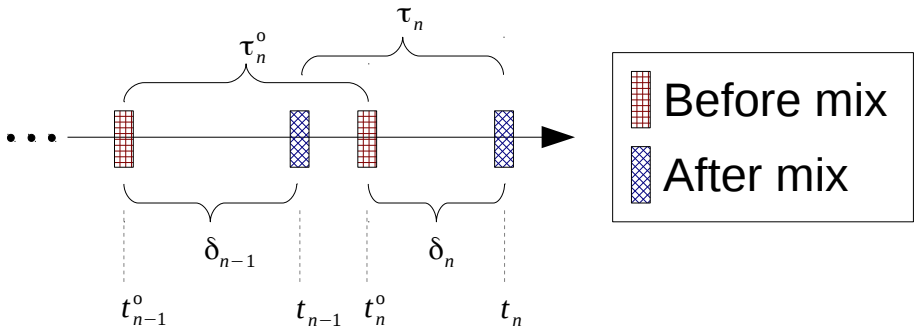


Figure 7.3: Time interval mix example (original figure).

a population of users. This condition is more difficult to satisfy, as it requires at least some global knowledge of other users in the system. A mix that provides anonymity will emulate the temporal behavior of the "typical" user. Finally, a mix should make it difficult to reproduce and predict the temporal behavior of a user.

There is a direct tradeoff between time lag and the masking capability of the mix. As the lag decreases, the dependence between the arrival process and the generating process increase. As the lag increases, the potential to reduce dependence increases. It is desirable to have a mix that maximally reduces the dependence between the generating process and the arrival process for a given lag.

Let t_n° and τ_n° be the true time and time interval of the n^{th} event generated by the user, and t_n and τ_n be the observed time and time interval of the n^{th} event after passing through the mix. The sequence of t_n, or alternatively τ_n, constitute the arrival process. The lag, or delay, between the generated event and observed event is given by δ_n. Since event order must be preserved by the mix, it is necessary that $t_{n-1} + \delta_{n-1} < t_n + \delta_n$. Figure 7.3 summarizes the variables of the time interval mix.

Predictability can be measured by the dependence between τ° and τ. Anonymity is given by the dependence between τ^A and τ^B from two random instantiations of the mix,

A and B. Dependence is measured by the mutual information between two continuous random variables, given by

$$MI(X,Y) = \int_Y \int_X f(x,y) \log \left[\frac{f(x,y)}{f(x) f(y)} \right] dxdy$$

where $f(x,y)$ is the joint probability density function of x and y. For empirical data, the mutual information can be estimated using a Gaussian kernel density estimator (Moon et al., 1995).

A mix with that provides perfect unpredictability has $MI(\tau^\circ, \tau) = 0$. Perfect anonymity is achieved when $MI\left(\tau^A, \tau^B\right) = 0$. Both can be achieved with an unlimited buffer size and delay. To see this, consider a mix that samples τ from a uniform distribution, *i.e.* $\tau \sim \mathscr{U}(0,k)$. Let τ_n be the time interval of the arrival process. If $\tau^\circ_{n+1} < \tau_n$, then the lag between the user and the arrival process must increase by at least $\tau_n - \tau^\circ_{n+1}$ and as $n \to \infty$, the lag increases unbounded. In this case, $MI(\tau^\circ, \tau) = 0$ since $f(\tau^\circ, \tau) = f(\tau^\circ) f(\tau)$, *i.e.* the observed time intervals are independent from the actual time intervals. If $\tau^\circ_{n+1} > \tau_n$, then the time interval on the arrival process is no longer independent from the actual time intervals since $\tau_{n+1} > \tau^\circ_{n+1} - \tau_n$. The next sample will have $\tau_{n+1} \sim \mathscr{U}\left(\tau^\circ_{n+1} - \tau_n, \tau^\circ_{n+1} - \tau_n + 1\right)$, or alternatively $P(\tau_{n+1} = s) = \frac{1}{k}$, $\tau - \tau_n < s < \tau^\circ_{n+1} - \tau_n + k$. Consequently, the dependence guarantees that $f(\tau^\circ, \tau) > f(\tau^\circ) f(\tau)$ and $MI(\tau^\circ, \tau) > 0$. Therefore, in practice it is necessary to introduce a lag constraint in minimizing the dependence between τ° and τ.

There are primarily two ways a time interval mix can add noise to the time intervals of the arrival process. Let δ be the lag between τ° and τ. The first is by generating a random delay between the actual events that occur at time t° resulting in the time $t^\circ + \delta$ at the arrival process. This type of mix is a *delay mix*. The second way is to generate τ directly, referred to as an *interval mix*.

7.2 Delay mix

The delay mix adds noise to the time intervals of the arrival process by randomly delaying each event. For the n^{th} event at time t°_n, the delay mix generates a random delay δ_n to produce the event time at the arrival process, $t_n = t^\circ_n + \delta_n$. Time intervals of the arrival

process are given by $\tau_n = (t_n^\circ + \delta_n) - \left(t_{n-1}^\circ + \delta_{n+1}\right)$. A uniform distribution can be used to generate delays. Let $\delta_n \sim \mathscr{U}\left(\max\left(\delta_{n-1} - \tau_n^\circ, 0\right), \Delta\right)$, where Δ is the upper bound on the delay. The lower bound of $\max\left(\delta_{n-1} - \tau_n^\circ, 0\right)$ is needed to ensure the event order is preserved. Without this constraint, events will become permuted if $\tau_n^\circ + \delta_n < \delta_{n-1}$. The process is summarized in Algorithm 7.1.

Algorithm 7.1 Delay mix.

1. **Initialize**
   Let $\tau_0^\circ = \infty$ and $t_0^\circ = 0$
2. **Generate**
   Let $l_n = \max\left(\delta_{n-1} - \tau_n^\circ, 0\right)$ be the lower bound of the n^{th} delay and $\delta_n \sim \mathscr{U}\left(l, \Delta\right)$ be a random delay. The event time on the arrival process is given by $t_n = t_n^\circ + \delta_n$

The expected delay is bounded above by Δ. To see this, consider the two scenarios: $\tau_n^\circ < \delta_{n-1}$ and $\tau_n^\circ > \delta_{n-1}$. If $\tau_n^\circ < \delta_{i-1}$, then the expected lag is simply $\mathrm{E}\left[\delta_n\right] = \frac{\Delta}{2}$ since the lower bound will always be 0. If $\tau_n^\circ > \delta_{n-1}$, then $\mathrm{E}\left[\delta_i\right] = \frac{\Delta + \delta_{n-1} - \tau_n^\circ}{2}$ which is bounded above by Δ. Given $\tau_0 = \infty$, it is true that $0 < \delta_1 < \Delta$. By induction, it is also true that $\delta_{n-1} < \Delta$ since $\tau_n^\circ > 0$.

7.2.1 Example

As an example, consider events occurring at times $t^\circ = [0, 5, 7, 11, 14]$ passing through a delay mix with $\Delta = 7$. The actual time intervals are $\tau^\circ = [\infty, 5, 2, 4, 3]$. The trace of each variable is shown in Table 7.2.

7.2.2 Experimental results

The effect of the delay mix can be evaluated empirically. Using each dataset, performance results are obtained before and after applying the delay mix. The relative change in identification accuracy, verification EER and AUC, and SMAPE are obtained for increasing values of Δ. The mean lag, $\bar{\delta}$, and loglikelihood ratio of the model trained on data after passing through the delay mix to the model trained on the original data, lr, are obtained. Table 7.3 contains the performance results of all datasets for various values of Δ.

Event	t°	τ°	l	δ	t	τ
0	0	∞	0	3	3	∞
1	5	5	0	6	11	8
2	7	2	4	5	12	1
3	11	4	1	5	16	4
4	14	3	6	6	20	4

Table 7.2: Delay mix example where t° and τ° are the actual time and time interval of the n^{th} event, l is the lower bound of the delay, δ is the random delay, and t and τ are the time and time interval of the arrival process.

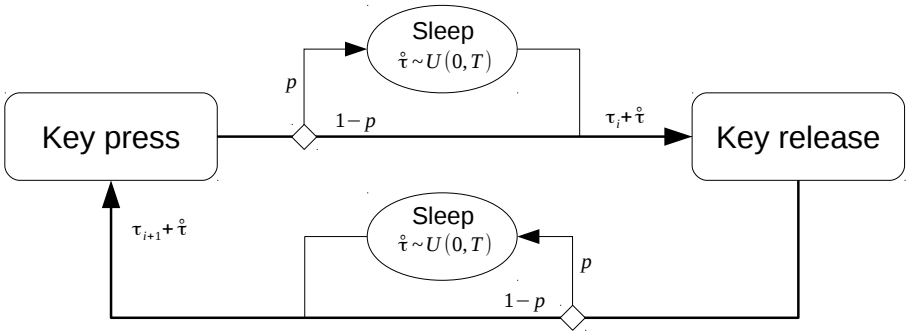


Figure 7.4: KeyboardPrivacy browser extension (original figure).

7.2.3 Case study

The *KeyboardPrivacy* plugin was developed to resist identification via typing behavior. The source code is publicly available, and the strategy employed, shown in Figure 7.4, is a variation of the delay mix (Moore, 2015). The delay mechanism introduces an artificial lag into the time intervals between key-press and key-release events. With probability p, an artificial lag $\delta_d \sim \mathscr{U}(0,\Delta_d)$ is added to the dwell time and $\delta_f \sim \mathscr{U}\left(0,\Delta_f\right)$ flight, respectively. The Δ parameters determine the distribution from which the artificial lags are sampled. This has the effect of masking the true temporal behavior. One problem with the KeyboardPrivacy mix is that the resulting lags can become unbounded. This happens when the event rate exceeds the artificial lags.

	Δ	$\bar{\delta}$	lr	ACC	EER	AUC	$AMRT$	$SMAPE$
Keyst. (fixed)	50 ms	25.22	-13.30	0.94	1.30	1.32	1.04	1.03
	100 ms	50.73	-38.75	0.73	1.49	1.74	1.13	1.11
	200 ms	106.32	-67.38	0.51	1.68	2.30	1.24	1.26
	500 ms	326.27	-84.69	0.30	2.39	3.88	1.35	1.44
	1 second	775.16	-90.84	0.19	2.82	4.96	1.46	1.48
Keyst. (free)	50 ms	25.08	-107.49	0.95	1.20	1.15	1.09	1.03
	100 ms	50.49	-307.51	0.83	1.42	1.55	1.21	1.10
	200 ms	105.64	-539.39	0.68	2.29	2.62	1.35	1.25
	500 ms	324.72	-669.75	0.52	2.58	3.41	1.55	1.39
	1 second	775.85	-702.42	0.43	2.89	4.27	1.58	1.42
Bitcoin	1 hour	2159.22	-37.19	0.84	1.12	1.30	1.07	1.02
	12 hours	37536.81	-74.18	0.65	1.24	1.55	1.08	1.03
	1 day	79598.00	-84.47	0.64	1.22	1.55	1.08	1.05
	7 days	595039.10	-124.32	0.62	1.28	1.62	1.07	1.16
	30 days	2574777.17	-164.41	0.57	1.33	1.65	1.08	1.27
Commits	1 day	0.50	-154.58	2.00	0.95	0.94	1.01	1.02
	3 days	1.66	-207.03	1.00	0.92	0.98	1.00	1.07
	7 day	4.47	-211.62	1.83	0.98	1.00	1.02	1.09
	14 days	10.25	-215.47	1.00	1.02	1.03	1.02	1.09
	30 days	25.01	-223.62	1.17	0.97	1.07	1.03	1.09
Visits	1 hour	30.32	-0.34	0.70	1.00	1.02	0.99	1.00
	12 hours	367.95	-3.94	0.90	0.96	1.00	1.01	1.00
	1 day	741.44	-4.45	0.90	0.95	0.99	1.00	0.99
	7 days	5810.91	-10.21	0.80	0.91	0.96	1.01	1.01
	30 days	31791.11	-17.84	0.70	0.95	1.02	1.03	1.04
Terror	1 day	0.52	-19.47	1.44	1.00	0.92	1.07	1.01
	3 days	1.60	-39.02	1.44	1.05	0.97	1.05	1.03
	7 days	3.92	-39.37	1.66	1.06	0.97	1.10	1.03
	30 days	18.94	-44.53	1.10	1.04	1.06	1.03	1.01
	90 days	63.49	-54.22	0.89	0.94	1.01	0.98	1.00

Table 7.3: Delay mix performance results.

7.3 Interval mix

Instead of generating delays, the time intervals of the arrival process can be modeled explicitly. The goal of the mix is to minimize the dependence between τ° and τ, and this strategy will give the mix greater control over the resulting time intervals. While the delay mix generates δ_n at each time step to produce arrival process time $t_n = t_n^\circ + \delta_n$, the interval mix generates τ_n to produce $t_n = t_{n-1} + \tau_n$. With an infinite delay, this allows the τ_n to be generated independently of τ_n°. With a finite delay, the τ_n are constrained by the event rate of the user.

At each time step t_n, let the desired time interval be $\dot{\tau}_n \sim \mathscr{U}(0, u_{n-1})$ where u_{n-1} is an upper bound parameter. If $t_{n-1} + \dot{\tau}_n > t_n^\circ$, then the n^{th} event is delayed by $\delta_n = t_{n-1} + \dot{\tau}_n - t_n^\circ$ to get the arrival process time $t_n = t_{n-1} + \dot{\tau}_n$. If $t_{n-1} + \dot{\tau}_n \leq t_n^\circ$, then $\delta_n = 0$ and the event is released immediately. The upper bound u is updated as $u_n = \max\{u_{n-1} + b[t_n - (t_{n-1} + \dot{\tau}_n)], \varepsilon\}$, where b is a parameter that controls the rate at which u_n can change. This update moves the expected time interval in the direction of the rate of the user. This process is summarized in Algorithm 7.2.

Algorithm 7.2 Interval mix.

1. **Initialize**
   Let $t_0^\circ = t_0 = 0$, $\tau_0^\circ = \tau_0 = \infty$, and $u_1 > 0$.
2. **Generate**
   Let $\dot{\tau}_n \sim \mathscr{U}(0, u_n)$ be the desired time interval and $\dot{t}_n = t_{n-1} + \dot{\tau}_n$ be the desired time of the arrival process. The arrival process time is given by $t_n = \max(\dot{t}_n, t_n^\circ)$.
3. **Update**
   Let $u_{n+1} = \max[u_n + b(t_n^\circ - \dot{t}_n), \varepsilon]$

7.3.1 Example

An example is used to demonstrate the interval mix with parameter $b = 1$. Similar as before, the actual event times are given by $t^\circ = [0, 5, 7, 11, 14]$ with time intervals $\tau^\circ = [\infty, 5, 2, 4, 3]$. At the first event, there is no delay between the actual event time and the arrival process,

Event	t°	τ°	u	$\dot{t}$	$\dot{\tau}$	δ	t	τ
0	0	∞	-	-	-	0	0	∞
1	5	5	7	3	3	0	5	5
2	7	2	9	11	6	4	11	6
3	11	4	5	15	4	4	15	4
4	14	3	1	16	1	2	16	1

Table 7.4: Interval mix example where $t^\circ$is the actual event time, τ° is the actual time interval, u is the interval distribution parameter, $\dot{t}$ is the desired event time, t is the arrival process event time, and τ is the arrival process time interval.

therefore $t_0 = t_0^\circ = 0$. The parameter updates and time intervals of the arrival process are shown in Table 7.4. For simplicity, values are sampled from a discrete uniform distribution. The starting value of u is chosen to be 7 to demonstrate the adaptability of the mix to the user's behavior in several iterations. The starting value u_1 can be any positive value and will quickly converge to an appropriate range.

7.3.2 Experimental results

Performance results are obtained for each dataset for various mix parameters. The average lag and loglikelihood ratio of the model trained on the interval mix data to the model trained on the original data are determined. The loglikelihood ratio usually favors the model trained on the data after passing through the interval mix, contrary to the results of the interval mix. This is primarily due to the resulting time interval distribution of the interval mix at the arrival process. The interval mix produces time intervals which are more clustered than the delay mix, leading to a stronger separation between active and passive states in the trained model.

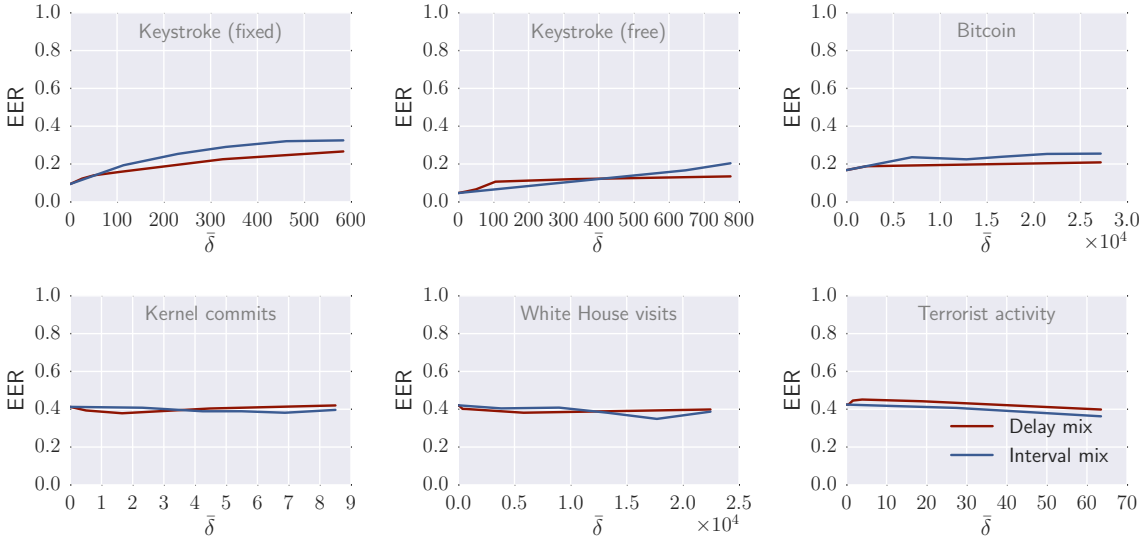
7.4 Summary

The identification and verification performance results of each mix as a function of the mean lag $\bar{\delta}$ are summarized in Figure 7.5. The interval mix generally provides greater

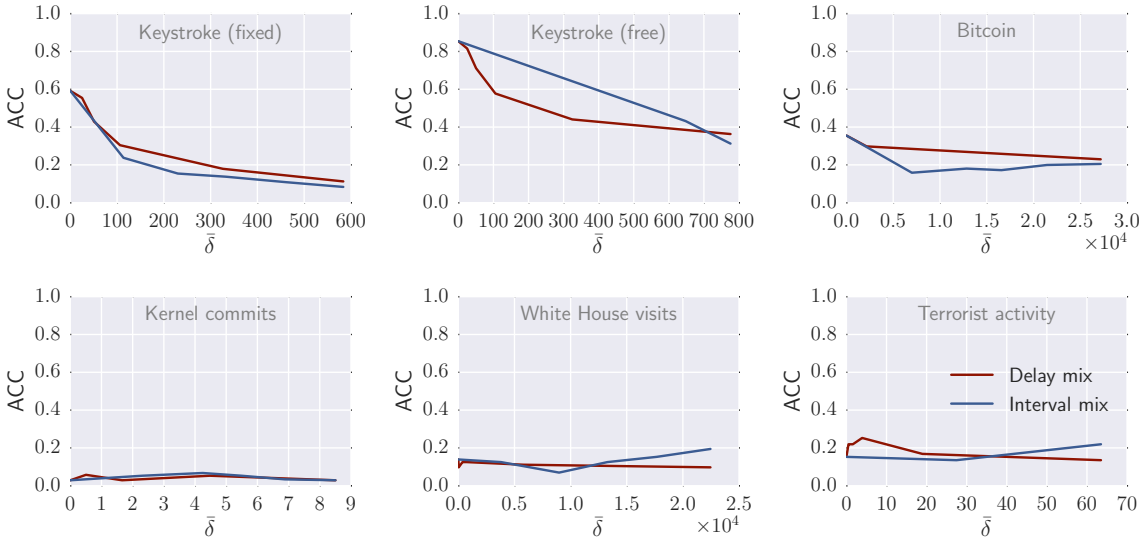
	b	$\bar{\delta}$	lr	ACC	EER	AUC	$AMRT$	$SMAPE$
Keyst. (fixed)	0.1	113.65	-38.95	0.40	2.05	3.08	1.22	1.10
	0.5	229.89	-26.82	0.26	2.68	4.62	1.30	1.45
	1.0	332.38	17.07	0.23	3.07	5.70	1.36	1.71
	1.5	461.97	66.05	0.18	3.40	6.74	1.41	1.91
	2.0	583.70	126.34	0.14	3.44	7.00	1.46	2.10
Keyst. (free)	0.1	647.93	-286.97	0.51	3.58	5.16	1.49	1.19
	0.5	2896.62	218.41	0.37	4.39	7.69	1.69	1.56
	1.0	5352.91	781.14	0.36	4.26	7.70	1.73	1.78
	1.5	7333.97	1297.78	0.33	4.19	7.94	1.79	1.96
	2.0	9728.61	1832.20	0.33	4.36	7.99	1.84	2.12
Bitcoin	0.1	6998.06	128.43	0.45	1.41	1.80	1.12	0.87
	0.5	12842.49	740.03	0.51	1.34	1.71	1.19	1.10
	1.0	16570.40	1463.71	0.48	1.42	1.82	1.24	1.25
	1.5	21441.46	2178.37	0.56	1.51	1.96	1.31	1.36
	2.0	27184.57	2834.23	0.58	1.52	2.03	1.36	1.45
Commits	0.1	2.31	-135.83	1.83	0.99	1.02	1.03	0.93
	0.5	4.25	-35.50	2.33	0.94	0.99	1.04	1.08
	1.0	5.49	61.22	1.83	0.94	0.96	1.04	1.21
	1.5	6.88	128.60	1.17	0.92	0.96	1.08	1.28
	2.0	8.50	189.73	1.00	0.96	1.00	1.10	1.35
Visits	0.1	3805.90	75.94	0.90	0.96	1.03	1.02	0.98
	0.5	8975.00	349.44	0.50	0.97	1.02	1.03	1.25
	1.0	13277.57	624.49	0.90	0.91	0.92	1.05	1.34
	1.5	17686.12	851.85	1.10	0.83	0.86	1.08	1.46
	2.0	22458.98	1074.81	1.40	0.92	0.97	1.07	1.56
Terror	0.1	27.44	-27.08	0.89	0.96	0.92	1.05	0.94
	0.5	86.61	97.27	1.44	0.85	0.79	0.99	1.08
	1.0	125.26	202.96	1.66	0.90	0.82	1.03	1.21
	1.5	163.36	281.75	1.22	0.80	0.74	1.18	1.23
	2.0	203.81	349.81	2.21	0.80	0.71	1.28	1.24

Table 7.5: Interval mix performance results.

masking capabilities than the delay mix for the same lag. For low-frequency events, using a mix has the opposite effect and actually increases performance results.



(a) Time lag vs EER.



(b) Time lag vs ACC.

Figure 7.5: Time lag vs mix performance (original figure).

Chapter 8

Conclusions

Time interval biometrics is multidisciplinary, borrowing ideas from various fields including biology, statistics, machine learning, computer science, and psychology. The aim is to capture the fundamental principles underlying human temporal behavior and utilize these for the purpose of biometric identification and verification. A wide range of human actions were considered in this work to demonstrate the utility of time interval biometrics across several modalities.

8.1 Discussion

Humans exhibit individually unique temporal behavior for a variety of activities, and this behavior can be used to identify and verify. High-frequency actions, such as keystroke and Bitcoin transactions, tend to be more stable than those that are low-frequency, such as source code authorships and terrorist events. This may be due partly to the greater influence of exogenous variables on low-frequency events. For example, the timing between keystrokes is driven by learned motor behavior, while the timing between Linux kernel commits is the result of task difficulty, priority, and programmer schedule. It is unlikely that time intervals alone provide sufficient means of identification and verification, however greater than chance accuracies indicate individualized temporal behavior in numerous scenarios.

For time intervals, the POHMM better captures the underlying structure of the data than the HMM. The increase in power comes from a dependence on event types. The underlying system state, which remains hidden, is partially observed through the type of event that occurs. This offers greater identification and verification capability using the POHMM over the HMM.

The proposed strategies for masking temporal behavior offer some solace to those who wish to remain truly anonymous. Despite this, a limitation of any masking strategy is the lag introduced between the user and the application. With a 100 ms delay for keystrokes, which would be hardly noticeable to the user, the identification and verification accuracies are approximately halved. Allowing a 1 day lag for Bitcoin transactions achieves a similar effect. Results indicate that the biggest gains in terms of identification and verification accuracy reduction occur within a lag proportional to the expected time interval of the user, and beyond a certain point the reduction in accuracy is incremental. Interestingly, even with a lag of 1 s, free-text keystroke users are identifiable with close to 50% accuracy.

8.2 Future work

There are several reasons to believe that time interval biometrics will become increasingly popular over the next several decades. With increased privacy and security concerns, temporal behavior remains largely ignored. This is likely to change, as timestamps are ubiquitous, perhaps more so than any other biometric. Future work should further investigate the mechanisms for masking temporal behavior and the range of modalities that are available.

Researchers Buhusi and Meck note, “space is gradually losing its value in a world of computer networks ... and time is becoming the essence of our times” (Buhusi and Meck, 2005). Timestamps will likely prevail in the age of privacy. The encryption of transmitted messages may be resilient to standard techniques of identification that rely on payload analysis, however they remain susceptible to the methods developed in this work if the time of transmission can be observed. Identification and verification methods relying on temporal information can still be improved considerably, and the utilization of temporal behavior for biometric purposes will likely remain an ongoing area of research.

Appendix A

Phase space reconstruction

Phase space reconstruction is a process by which a univariate time series is embedded in a higher dimension through time delay embedding. Takens' Theorem states that a proper time delay embedding of a univariate time series may reveal the structure of the underlying dynamical system (Takens, 1981). Time delay embedding requires the estimation of an embedding dimension d_e and time lag τ_e. For a given d_e and τ_e, the embedding procedure transforms the original time series, y, into a series of embedded vectors $\mathbf{y}$ where

$$\mathbf{y}_n = \left[y_n, y_{n-\tau_e}, y_{n-2\tau_e}, \ldots, y_{n-(d_e-1)\tau_e}\right].$$

There are several heuristics that can be used to determine the time lag and embedding dimension (Abarbanel, 1996; Small, 2005). The time lag is determined first since its heuristic does not depend on the embedding dimension. In this work, both embedding parameters are determined for each feature in each dataset.

A.1 Time lag

The time lag is determined using the mutual information between lagged components of y,

$$\mathscr{I}(T) = \sum_{n=1}^{N-T} P(y_n, y_{n+T}) \log_2 \frac{P(y_n, y_{n+T})}{P(y_n)P(y_{n+T})} \tag{A.1}$$

where y has N components. The time lag at which the first local minimum occurs is chosen as the embedding parameter. This ensures that the lagged data are somewhat correlated (Small, 2005). As suggested in Abarbanel (1996), when there is no local minimum up to some T_{max}, τ_e is taken to be T where $\mathscr{I}(T)/\mathscr{I}(0) \approx \frac{1}{5}$. A maximum lag of $T_{max} = 5$ is used in this work.

A.2 Embedding dimension

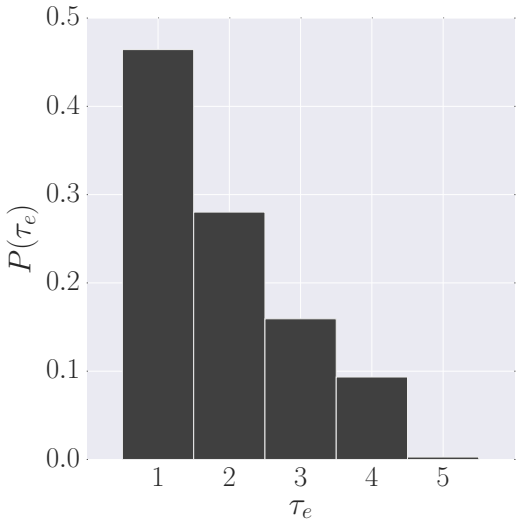
The method of false nearest neighbors (FNN) is used to estimate embedding dimension d_e. Two embedded vectors $\mathbf{y}_n$ and $\mathbf{y}_m$ are FNN if they are neighboring in dimension d_e and distant in dimension $d_e + 1$. The embedding dimension can be estimated by increasing d_e until the proportion of FNN falls below some threshold. This ensures that the underlying dynamical system is unfolded and not projected onto a dimension which is too small, causing an unnecessarily large number of FNN. To classify two vectors as FNN, consider the normalized increase in distance when going from d_e to $d_e + 1$,

$$\mathscr{R} = \frac{|y_{n-(d_e+1)\tau} - y_{n^{NN}-(d_e+1)\tau}|}{\| \mathbf{y}_n - \mathbf{y}_n^{NN} \|} \tag{A.2}$$

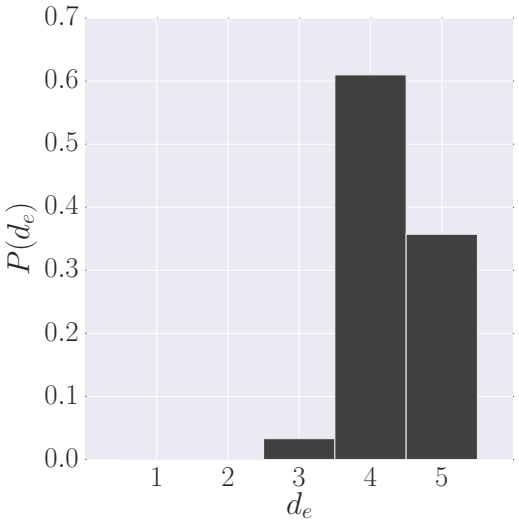
where $\mathbf{y}_n^{NN}$ is the nearest neighbor to $\mathbf{y}_n$, and n^{NN} is the index of $\mathbf{y}_n^{NN}$. When $\mathscr{R}$ is above some threshold, $\mathscr{R}_{thresh}$, then $\mathbf{y}_n$ and $\mathbf{y}_n^{NN}$ are considered to be FNN. This work uses the threshold $\mathscr{R}_{thresh} = 15$, a value suggested to give reasonable results in many situations (Small, 2005). The embedding dimension is chosen to be the smallest d_e that results in less than 5% FNN. If the proportion of FNN does not fall below 5% up to $d_{e\,max} = 5$, then d_e is take to be $d_{e\,max}$. An upper bound on the embedding dimension is necessary since noisy samples are prone to high embedding dimensions (Small, 2005).

A.3 Embedding parameters

The embedding parameters for each sample are determined and the global embedding parameters taken to be the parameters with the highest frequency, *i.e.* the mode, over all samples. As an example, the densities of d_e and τ_e for the 366 Bitcoin time interval samples



(a) Distribution of τ_e. For each sample, τ_e is selected where the first minimum of mutual information occurs.



(b) Distribution of d_e. For each sample, d_e is selected where the proportion of FNN drops below 5%.

Figure A.1: Bitcoin time interval embedding parameter distributions (original figure).

are shown in Figure A.1. The high density at $d_e = 5$ indicates there are many samples that desire a greater embedding dimension due to either noise or an inherently high-dimensional underlying system. Embedding parameters for each feature are shown in Table A.1.

	Feature	Dim.	Lag
Keys. (free)	Time interval	5	1
	Duration	5	1
	Time of word	5	1
	Time of sentence	3	5
	Time of paragraph	1	5
Bitcoin	Time interval	4	1
	Minute of hour	4	2
	Hour of day	5	2
	Day of week	5	5
	฿ value	5	1
	Num. outputs	5	1
	Num. inputs	5	1
	Input/output balance	5	1
Visits	Time interval	3	2
	Hour of day	5	1
	Day of week	5	1
	Week of month	5	2
	Duration	3	1
	Tardiness	3	1
	Forecast	5	1
	Num. people	4	2

	Feature	Dim.	Lag
Keys. (fixed)	Time interval	4	1
	Duration	4	2
	Time of word	4	3
	Time of sentence	2	3
	Time of paragraph	1	5
Commits	Time interval	5	1
	Day of week	4	2
	Week of month	5	2
	Time to commit	5	1
	Num. commits	5	1
	Num. committers	5	1
	Description chars	3	1
	Description words	3	1
Terror	Time interval	3	1
	Day of week	4	1
	Week of month	5	3
	Month of year	5	4
	Num. perpetrators	5	1
	Num. killed	5	1
	Num. captured	1	1
	Num. related events	1	1

Table A.1: Embedding dimension and time-lag determined for each feature of each dataset using the mutual information and method of false nearest neighbors.

Appendix B

Nonlinear prediction error

Given a time series y_n and predictions $\hat{y}_n$, the root mean squared (RMS) prediction error is

$$e = \frac{1}{N}\sqrt{\sum_{n=1}^{N}(y_n - \widehat{y}_n)}$$

where N is the length of the time series. The system history of a deterministic nonlinear dynamical system can be used to predict future states and obtain the nonlinear prediction error (NPE) (Lorenz, 1969). Nonlinear time series predictions are made after an appropriate embedding is determined for y. To predict y_{n+1}, consider the embedded vector $\mathbf{y}_n$ in $\mathbf{R}^{d_e}$. Let the k nearest neighbors to $\mathbf{y}_n$ be $\mathbf{y}_n^{NN}$, and let $\mathbf{y}_{n+1}^{NN}$ be the successors to each of the neighbors $\mathbf{y}_n^{NN}$. The prediction $\widehat{y}_{n+1}$ is calculated as a weighted sum of y_{n+1}^{NN}, the scalar components of the nearest neighbor successors $\mathbf{y}_{n+1}^{NN}$.

In this work, predictions are made by assigning the y_{n+1}^{NN} linear decreasing weights where $\mathbf{y}_n^{NN}$ are ordered by their distance from $\mathbf{y}_n$. The first neighbor is assigned a weight of $\frac{k}{k(k+1)/2}$, the second a weight of $\frac{k-1}{k(k+1)/2}$, and so on, where the k^{th} neighbor has a weight of $\frac{1}{k(k+1)/2}$. This places more emphasis on close neighbors, while retaining some influence from distance neighbors. Other weighting strategies can be used, such as uniform or exponentially decreasing weights. The out-of-sample prediction error between two time series is obtained by taking the nearest neighbors from another sample. To make out-of-sample predictions, take $\mathbf{x}_n^{NN}$ (and consequently $\mathbf{x}_{n+1}^{NN}$) from a different sample x_n, making predictions similarly as above.

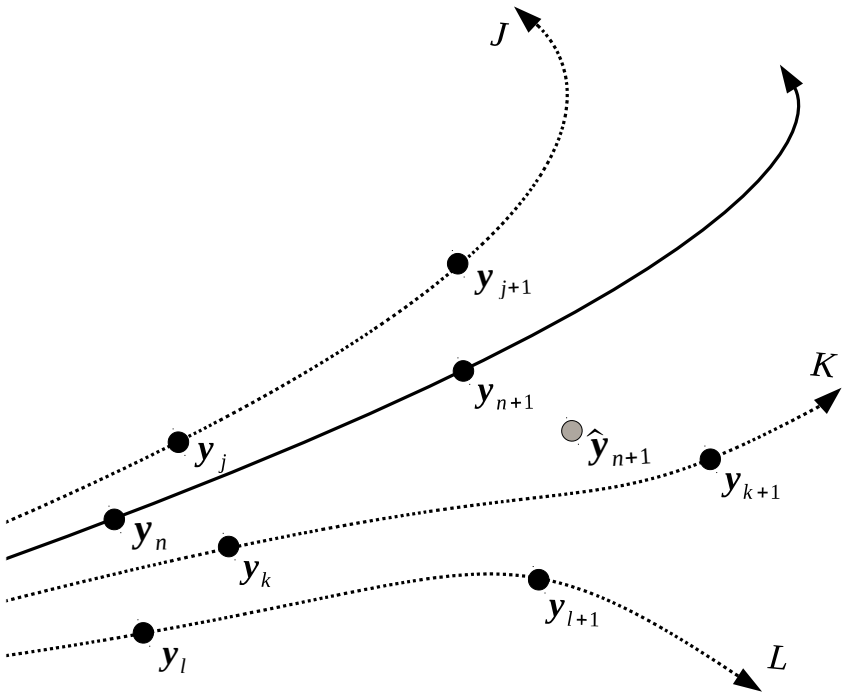


Figure B.1: Nonlinear prediction example (original figure).

An example is shown in Figure B.1 in which $\mathbf{y}_{n+1}$ is predicted. The 3 nearest neighbors to $\mathbf{y}_n$ are determined and their successors used to make the prediction. In this example, uniform weights are used and $\mathbf{y}_{n+1}$ is predicted from the average of $\mathbf{y}_{j+1}$, $\mathbf{y}_{k+1}$, and $\mathbf{y}_{l+1}$, the successors to the 3 nearest neighbors of $\mathbf{y}_n$. The appropriate scalar component of $\widehat{\mathbf{y}}_{n+1}$ can be taken to obtain the prediction $\widehat{y}_{n+1}$.

Appendix C

List of abbreviations and symbols

Abbreviations

AAFT	amplitude adjusted Fourier transform
AATFT	amplitude adjusted truncated Fourier transform
ACC	accuracy
AMRT	average maximum rejection time
AUC	area under curve
AF	Allan Factor
AP	access point
BW	Baum-Welch
CI	confidence interval
CV	coefficient of variation
context-HMM	context hidden Markov model
DCHMM	double chain hidden Markov model

dof	degrees of freedom
ECG	electrocardiogram
EM	expectation maximization
FNN	false nearest neighbors
GTD	Global Terrorism Database
h-state	hidden state
HMM	hidden Markov model
HSMM	hidden semi-Markov model
kNN	k-nearest neighbor
MCVSVM	minimum class variance support vector machine
MRT	maximum rejection time
ML	maximum likelihood
MOOC	massively open online course
MST	minimum spanning tree
MWW	multivariate Wald-Wolfowitz
NPE	nonlinear prediction error
NPHMM	nonhomogeneous Poisson hidden Markov model
p-state	partially observed state
partly-HMM	partly hidden Markov model
partially-HMM	partially hidden Markov Model
POHMM	partially observable hidden Markov model

RMS	root mean squared
ROC	receiver operating characteristic
SCV	stratified k-fold cross-validation
SMAPE	symmetric mean absolute percentage error
SVM	support vector machine

Symbols

$a_{ij\|\omega,\psi}$	Probability of transitioning from state i to state j, given observed p-state ω while in state i and ψ in state j
A	area test statistic
$\mathscr{A}$	anonymity
C_v	coefficient of variation
$\mathbf{b}_{j\|\omega}$	Observation distribution parameters in state j, given p-state ω
d_e	embedding dimension
e	root mean squared error
$f(\cdot)$	probability density function
$I(\cdot)$	indicator function
$\mathscr{I}(\cdot)$	mutual information between lagged time series components
m	number of unique p-states
M	number of hidden states
$MI(\cdot)$	mutual information between two random variables

n	observation index
N	number of observations
$N(T)$	number of events in a length T time window
$P(\cdot)$	probability function
q	discriminative test statistic
$\mathscr{R}$	normalized increase in distance
S	number of surrogate samples
t	time
T	time window
U	number of users
$\mathscr{U}$	uniform distribution
$\mathbf{x}_1^N$	sequence of observed values
y	univariate time series
$\mathbf{y}$	series of embedded vectors
z_n	hidden state at time t_n
τ	time interval
τ_e	embedding time lag
δ	time lag between a mix input and output
θ	model parameters
ω or ψ	p-state
Ω_1^N	sequence of p-states

$\gamma_{j\|\omega}(n)$	Posterior probability of being in state j, given observations and p-state ω at time t_n
$\xi_{ij\|\omega,\psi}(n)$	Posterior probability of transitioning from state i at time t_n to state j at time t_{n+1}, given p-state ω and ψ at times t_n and t_{n+1}, respectively
$\pi_{j\|\omega}$	Probability of being in state j at time t_1, given observed p-state ω
$\Pi_{j\|\omega}$	Steady state probability of being in state j, given p-state ω
ε	loglikelihood reduction convergence criterion
μ	normal distribution mean
σ	normal distribution standard deviation
η	log-normal distribution log-mean
ρ	log-normal distribution log-standard deviation
$\widehat{x}$	parameter estimate of x
$\dot{x}$	parameter re-estimate of x

Bibliography

H. Abarbanel. *Analysis of observed chaotic data*. Springer, 1996.

N. Alotaibi, R. Barilla, F. Betances, A. Chohan, A. Gazarov, M. Reid, A. Scolaro, and V. Monaco. Biometric system design for handheld devices. *Proc. Research Day, CSIS, Pace University*, 2014.

L. C. Araujo, L. H. Sucupira Jr, M. G. Lizarraga, L. L. Ling, and J. B. T. Yabu-Uti. User authentication through typing biometrics features. *Signal Processing, IEEE Transactions on*, 53(2):851–855, 2005.

S. Asmussen. *Applied probability and queues*, volume 51. Springer Science & Business Media, 2008.

T. Auld, A. W. Moore, and S. F. Gull. Bayesian neural networks for internet traffic classification. *Neural Networks, IEEE Transactions on*, 18(1):223–239, 2007.

S. P. Banerjee and D. L. Woodard. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research*, 7(1):116–139, 2012.

A.-L. Barabási. The origin of bursts and heavy tails in human dynamics. *Nature*, 435 (7039):207–211, 2005.

A.-L. Barabási, K.-I. Goh, and A. Vazquez. Reply to comment on "the origin of bursts and heavy tails in human dynamics". *arXiv preprint physics/0511186*, 2005.

J. Barnes and D. Allan. A statistical model of flicker noise. *Proceedings of the IEEE*, 54 (2):176–178, 1966.

A. Berchtold. The double chain markov model. *Communications in Statistics-Theory and Methods*, 28(11):2569–2589, 1999.

P. Bours. Continuous keystroke dynamics: A different perspective towards biometric evaluation. *Information Security Technical Report*, 17(1):36–43, 2012.

P. Bours and S. Mondal. Performance evaluation of continuous authentication systems. *IET Biometrics*, 2015.

D. G. Brizan, A. Goodkind, P. Koch, K. Balagani, V. V. Phoha, and A. Rosenberg. Utilizing linguistically-enhanced keystroke dynamics to predict typist cognition and demographics. *International Journal of Human-Computer Studies*, 2015.

C. V. Buhusi and W. H. Meck. What makes us tick? functional and neural mechanisms of interval timing. *Nature Reviews Neuroscience*, 6(10):755–765, 2005.

D. V. Buonomano. The biology of time across different scales. *Nature chemical biology*, 3 (10):594–597, 2007.

C. E. Carr. Processing of temporal information in the brain. *Annual review of neuroscience*, 16(1):223–243, 1993.

P. R. Center. Search engine use 2012, 2012. URL `http://www.pewinternet.org/2012/03/09/search-engine-use-2012/`. Accessed August 2015.

D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

A. Clauset, M. Young, and K. S. Gleditsch. On the frequency of severe terrorist events. *Journal of Conflict Resolution*, 51(1):58–87, 2007.

A. Clauset, C. R. Shalizi, and M. E. Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

A. Clauset, R. Woodard, et al. Estimating the historical and future probabilities of large terrorist events. *The Annals of Applied Statistics*, 7(4):1838–1865, 2013.

W. E. Cooper. *Cognitive aspects of skilled typewriting*. Springer Science & Business Media, 1983.

C. Couvreur. *Hidden Markov models and their mixtures*. DEA Report, Dep. of Mathematics, Université catholique de Louvain, 1996.

D. R. Cox. Some statistical methods connected with series of events. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 129–164, 1955.

D. R. Cox and V. Isham. *Point processes*, volume 12. CRC Press, 1980.

D. J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*, volume 1. Springer Science & Business Media, 2003.

D. J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes: Volume II: General Theory and Structure*, volume 2. Springer Science & Business Media, 2007.

T. Dunstone and N. Yager. *Biometric system and data analysis: Design, evaluation, and data mining*. Springer Science & Business Media, 2008.

J. D. Ferguson. Variable duration models for speech. In *Proceedings of the Symposium on the Application of HMMs to Text and Speech*, pages 143–179, 1980.

S. Forchhammer and T. S. Rasmussen. Adaptive partially hidden markov models with application to bilevel image coding. *Image Processing, IEEE Transactions on*, 8(11): 1516–1526, 1999.

S. O. Forchhammer and J. Rissanen. Partially hidden markov models. *IEEE Transactions on Information Theory*, 42(4):1253–1256, 1996.

J. H. Friedman and L. C. Rafsky. Multivariate generalizations of the wald-wolfowitz and smirnov two-sample tests. *The Annals of Statistics*, pages 697–717, 1979.

R. S. Gaines, W. Lisowski, S. J. Press, and N. Shapiro. Authentication by keystroke timing: Some preliminary results. Technical report, DTIC Document, 1980.

Gallup. Usa today/gallup poll, 2010. URL `http://www.gallup.com/file/poll/145334/Internet_Ads_Dec_21_2010.pdf`. Accessed August 2015.

A. Gavaldà-Miralles, J. S. Otto, F. E. Bustamante, L. A. Amaral, J. Duch, and R. Guimerà. User behavior and change: File-sharers and copyright laws. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 319–324. ACM, 2014.

C. Gentry et al. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.

D. Guarin, A. Orozco, and E. Delgado. A new surrogate data method for nonstationary time series. *arXiv preprint arXiv:1008.1804*, 2010.

R. Hidalgo and A. César. Conditions for the emergence of scaling in the inter-event time of uncorrelated and seasonal systems. *Physica A: Statistical Mechanics and its Applications*, 369(2):877–883, 2006.

A. J. Hopwood, C. Hurth, J. Yang, Z. Cai, N. Moran, J. G. Lee-Edghill, A. Nordquist, R. Lenigk, M. D. Estes, J. P. Haley, et al. Integrated microfluidic system for rapid forensic dna analysis: sample collection to dna profile. *Analytical chemistry*, 82(16):6991–6999, 2010.

D. Hudson. White house visitor records, 2015. URL `http://www.whitehouse.gov/goodgovernment/tools/visitor-records`. Accessed August 2015.

T. Iobayashi, J. Furuyama, and K. Mas. Partly hidden markov model and its application to speech recognition. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 1, pages 121–124. IEEE, 1999.

S. A. Israel, J. M. Irvine, A. Cheng, M. D. Wiederhold, and B. K. Wiederhold. Ecg to identify individuals. *Pattern recognition*, 38(1):133–142, 2005.

A. K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):4–20, 2004.

L. Jain, J. Monaco, M. Coakley, and C. Tappert. Passcode keystroke biometric performance on smartphone touchscreens is superior to that on hardware keyboards. *International Journal of Research in Computer Applications & Information Technology*, 2(4):29–33, 2014.

S. Jana and S. K. Kasera. On fast and accurate detection of unauthorized wireless access points using clock skews. *Mobile Computing, IEEE Transactions on*, 9(3):449–462, 2010.

A. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14:841, 2002.

D. Jurafsky, J. H. Martin, A. Kehler, K. Vander Linden, and N. Ward. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*, volume 2. MIT Press, 2000.

H. Kantz and T. Schreiber. *Nonlinear time series analysis*, volume 7. Cambridge university press, 2004.

K. Killourhy and R. Maxion. The effect of clock resolution on keystroke dynamics. In *Recent Advances in Intrusion Detection*, pages 331–350. Springer, 2008.

K. S. Killourhy. A scientific understanding of keystroke dynamics. Technical report, DTIC Document, 2012.

K. S. Killourhy and R. A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on*, pages 125–134. IEEE, 2009.

T. Kobayashi and S. Haruyama. Partly-hidden markov model and its application to gesture recognition. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 4, pages 3081–3084. IEEE, 1997.

D. Kondor, M. Pósfai, I. Csabai, and G. Vattay. Do the rich get richer? an empirical analysis of the bitcoin transaction network. *PloS one*, 9(2):e86197, 2014.

M. Kyoso and A. Uchiyama. Development of an ecg identification system. In *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, volume 4, pages 3721–3723. IEEE, 2001.

G. LaFree and L. Dugan. Introducing the global terrorism database. *Terrorism and Political Violence*, 19(2):181–204, 2007.

G. LaFree and L. Dugan. Global terrorism database codebook: Inclusion criteria and variables. *National Consortium for the Study of Terrorism and Responses to Terrorism and the Center for Terrorism and Intelligence Studies, University of Maryland*, 2014.

N. A. Laskaris, S. P. Zafeiriou, and L. Garefa. Use of random time-intervals (RTIs) generation for biometric verification. *Pattern Recognition*, 42(11):2787–2796, 2009.

E. N. Lorenz. Atmospheric predictability as revealed by naturally occurring analogues. *Journal of the Atmospheric sciences*, 26(4):636–646, 1969.

S. B. Lowen and M. C. Teich. *Fractal-based point processes*, volume 366. John Wiley & Sons, 2005.

Y. Lu and L. Zeng. A nonhomogeneous poisson hidden markov model for claim counts. *Astin Bulletin*, 42(01):181–202, 2012.

A. Maas, C. Heather, C. T. Do, R. Brandman, D. Koller, and A. Ng. Offering verified credentials in massive open online courses: Moocs and technology to advance learning and learning research (ubiquity symposium). *Ubiquity*, 2014(May):2, 2014.

I. S. MacKenzie and C. Ware. Lag as a determinant of human performance in interactive systems. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 488–493. ACM, 1993.

R. D. Malmgren, D. B. Stouffer, A. E. Motter, and L. A. Amaral. A poissonian explanation for heavy tails in e-mail communication. *Proceedings of the National Academy of Sciences*, 105(47):18153–18158, 2008.

R. D. Malmgren, J. M. Hofman, L. A. Amaral, and D. J. Watts. Characterizing individual communication patterns. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 607–616. ACM, 2009.

J. R. Mayer and J. C. Mitchell. Third-party web tracking: Policy and technology. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 413–427. IEEE, 2012.

S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140. ACM, 2013.

B. Merialdo. Tagging english text with a probabilistic model. *Computational linguistics*, 20(2):155–171, 1994.

A. Mishra and P. Venkitasubramaniam. The anonymity of an almost fair chaum mix. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pages 693–700. IEEE, 2011.

M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet mathematics*, 1(2):226–251, 2004.

J. V. Monaco. Classification and authentication of one-dimensional behavioral biometrics. In *International Joint Conference on Biometrics (IJCB)*. IEEE,IAPR, 2014.

J. V. Monaco. Identifying bitcoin users by transaction behavior. In *DSS Biometric and Surveillance Technology for Human and Activity Identification XII*. SPIE, 2015.

J. V. Monaco, N. Bakelman, S.-H. Cha, and C. C. Tappert. Recent advances in the development of a long-text-input keystroke biometric authentication system for arbitrary text input. In *European Intelligence and Security Informatics Conference (EISIC)*, pages 60–66. IEEE, 2013.

J. V. Monaco, C. C. Tappert, and M. L. Ali. Spoofing key-press latencies with a generative keystroke dynamics model. In *Biometrics: Theory, Applications and Systems (BTAS)*. IEEE, 2015.

Y.-I. Moon, B. Rajagopalan, and U. Lall. Estimation of mutual information using kernel density estimators. *Physical Review E*, 52(3):2318–2321, 1995.

P. Moore. Behavioral profiling: The password you can't change, 2015. URL `https://paul.reviews/behavioral-profiling-the-password-you-cant-change/`. Accessed July 2015.

S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28, 2008.

A. Newell. *Unified theories of cognition*. Harvard University Press, 1994.

W. K. Newey and D. McFadden. Large sample estimation and hypothesis testing. *Handbook of econometrics*, 4:2111–2245, 1994.

M. E. Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351, 2005.

J. Nicolau. Stationary processes that look like random walks - the bounded random walk process in discrete and continuous time. *Econometric Theory*, 18(01):99–118, 2002.

H. Ozkan, A. Akman, and S. S. Kozat. A novel and robust parameter training approach for hmms under noisy and partial access to states. *Signal Processing*, 94:490–497, 2014.

L. T. Passos and K. Czarnecki. A dataset of feature additions and feature removals from the linux kernel. In *MSR*, pages 376–379, 2014.

R. Perline. Strong, weak and false inverse power laws. *Statistical Science*, pages 68–88, 2005.

P. J. Phillips and A. J. O'toole. Comparison of human and computer performance across face recognition experiments. *Image and Vision Computing*, 32(1):74–85, 2014.

B. Phipson and G. K. Smyth. Permutation p-values should never be zero: calculating exact p-values when permutations are randomly drawn. *Statistical applications in genetics and molecular biology*, 9(1), 2010.

H. Pohl, M. Krause, and M. Rohs. One-button recognizer: Exploiting button pressing behavior for user differentiation. In *Proc. UbiComp*, volume 15, 2015.

M. D. Porter, G. White, et al. Self-exciting hurdle models for terrorist activity. *The Annals of Applied Statistics*, 6(1):106–124, 2012.

P. Praamstra, D. Kourtis, H. F. Kwok, and R. Oostenveld. Neurophysiology of implicit timing in serial choice reaction-time performance. *The Journal of neuroscience*, 26(20): 5448–5455, 2006.

L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

S. V. Radhakrishnan, A. S. Uluagac, and R. Beyah. Gtid: A technique for physical device and device type fingerprinting. 2014.

V. Raghavan, A. Galstyan, A. G. Tartakovsky, et al. Hidden markov models for the activity profile of terrorist groups. *The Annals of Applied Statistics*, 7(4):2402–2430, 2013.

K. A. Rahman, K. S. Balagani, and V. V. Phoha. Snoop-forge-replay attacks on continuous verification with keystrokes. *IEEE Transactions on Information Forensics and Security*, 8(3):528–541, 2013.

D. Read. Monetary incentives, what are they good for? *Journal of Economic Methodology*, 12(2):265–276, 2005.

F. Reid and M. Harrigan. *An analysis of anonymity in the bitcoin system*. Springer, 2013.

R. N. Rodrigues, G. F. Yared, C. R. d. N. Costa, J. B. Yabu-Uti, F. Violaro, and L. L. Ling. Biometric access control through numerical keyboards based on keystroke dynamics. In *Advances in Biometrics*, pages 640–646. Springer, 2005.

T. A. Salthouse. Perceptual, cognitive, and motoric aspects of transcription typing. *Psychological bulletin*, 99(3):303, 1986.

S. A. Schuckers. Spoofing and anti-spoofing measures. *Information Security technical report*, 7(4):56–62, 2002.

M.-A. Schulz, B. Schmalbach, P. Brugger, and K. Witt. Analysing humanly generated random number sequences: a pattern-based approach. *PloS one*, 7(7):e41531, 2012.

S. L. Scott. Detecting network intrusion using a markov modulated nonhomogeneous poisson process. *Submitted to the Journal of the American Statistical Association*, 2001.

A. Serwadda and V. V. Phoha. Examining a large keystroke biometrics dataset for statistical-attack openings. *ACM Transactions on Information and System Security (TISSEC)*, 16(2):8, 2013.

I. S. Sheehan. Assessing and comparing data sources for terrorism research. In *Evidence-based counterterrorism policy*, pages 13–40. Springer, 2012.

M. Small. *Applied nonlinear time series analysis: applications in physics, physiology and finance*, volume 52. World Scientific, 2005.

D. X. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on ssh. In *USENIX Security Symposium*, volume 2001, 2001.

R. Spillane. Keyboard apparatus for personal identification. *IBM Technical Disclosure Bulletin*, 17(3346):3346, 1975.

D. Stouffer, R. Malmgren, and L. Amaral. Comments on "the origin of bursts and heavy tails in human dynamics". *arXiv preprint physics/0510216*, 2005.

R. Stratonovich. Conditional markov processes. *Theory of Probability & Its Applications*, 5(2):156–178, 1960.

F. Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pages 366–381. Springer, 1981.

C. C. Tappert, M. Villani, and S.-H. Cha. Keystroke biometric identification and authentication on long-text input. *Behavioral biometrics for human identification: Intelligent applications*, pages 342–367, 2009.

C. C. Tappert, S.-H. Cha, M. Villani, and R. S. Zack. A keystroke biometric system for long-text input. *International Journal of Information Security and Privacy (IJISP)*, 4(1): 32–60, 2010.

L. Telesca and M. Lovallo. Are global terrorist attacks time-correlated? *Physica A: Statistical Mechanics and its Applications*, 362(2):480–484, 2006.

C. M. Tey, P. Gupta, D. Gao, and Y. Zhang. Keystroke timing analysis of on-the-fly web apps. In *Applied Cryptography and Network Security*, pages 405–413. Springer, 2013.

J. Theiler, S. Eubank, A. Longtin, B. Galdrikian, and J. Doyne Farmer. Testing for nonlinearity in time series: the method of surrogate data. *Physica D: Nonlinear Phenomena*, 58(1):77–94, 1992.

Tor. The tor project: Anonymity online, 2015. URL `https://www.torproject.org`. Accessed August 2015.

A. S. Uluagac, S. V. Radhakrishnan, C. Corbett, A. Baca, and R. Beyah. A passive technique for fingerprinting wireless devices with wired-side observations. In *Communications and Network Security (CNS), 2013 IEEE Conference on*, pages 305–313. IEEE, 2013.

J. R. Vacca. *Biometric technologies and verification systems*. Butterworth-Heinemann, 2007.

A. Vázquez, J. G. Oliveira, Z. Dezsö, K.-I. Goh, I. Kondor, and A.-L. Barabási. Modeling bursts and heavy tails in human dynamics. *Physical Review E*, 73(3):036127, 2006.

P. Venkitasubramaniam and V. Anantharam. On the anonymity of chaum mixes. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 534–538. IEEE, 2008.

M. Villani, C. Tappert, G. Ngo, J. Simone, H. S. Fort, and S.-H. Cha. Keystroke biometric recognition studies on long-text input under ideal and application-oriented conditions. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pages 39–39. IEEE, 2006.

Q. H. Vuong. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica: Journal of the Econometric Society*, pages 307–333, 1989.

J. L. Wayman. Biometric verification/identification/authentication/recognition: The terminology. In *Encyclopedia of Biometrics*, pages 153–157. Springer, 2009.

D. White, P. J. Phillips, C. A. Hahn, M. Hill, and A. J. O'Toole. Perceptual expertise in forensic facial image comparison. In *Proc. R. Soc. B*, volume 282, page 20151292. The Royal Society, 2015.

M. Wiener, M. S. Matell, and H. B. Coslett. Multiple mechanisms for temporal processing. *Frontiers in integrative neuroscience*, 5, 2011.

S.-Z. Yu. Hidden semi-markov models. *Artificial Intelligence*, 174(2):215–243, 2010.

H. N. Zelaznik, R. Spencer, and R. B. Ivry. Dissociation of explicit and implicit timing in repetitive tapping and drawing movements. *Journal of Experimental Psychology: Human Perception and Performance*, 28(3):575, 2002.

T. Zhou, Z.-D. Zhao, Z. Yang, and C. Zhou. Relative clock verifies endogenous bursts of human dynamics. *EPL (Europhysics Letters)*, 97(1):18006, 2012.