Pattern Recognition 76 (2018) 449–462

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

The partially observable hidden Markov model and its application to keystroke dynamics

John V. Monaco[a,*], Charles C. Tappert[b]

[a] U.S. Army Research Laboratory, Aberdeen Proving Ground, MD 21005, USA
[b] Pace University, Pleasantville, NY 10570, USA

ARTICLE INFO

Article history:
Received 2 December 2016
Revised 3 November 2017
Accepted 16 November 2017
Available online 21 November 2017

Keywords:
Hidden Markov model
Keystroke biometrics
Behavioral biometrics
Time intervals
Anomaly detection

ABSTRACT

The partially observable hidden Markov model is an extension of the hidden Markov Model in which the hidden state is conditioned on an independent Markov chain. This structure is motivated by the presence of discrete metadata, such as an event type, that may partially reveal the hidden state but itself emanates from a separate process. Such a scenario is encountered in keystroke dynamics whereby a user's typing behavior is dependent on the text that is typed. Under the assumption that the user can be in either an active or passive state of typing, the keyboard key names are event types that partially reveal the hidden state due to the presence of relatively longer time intervals between words and sentences than between letters of a word. Using five public datasets, the proposed model is shown to consistently outperform other anomaly detectors, including the standard HMM, in biometric identification and verification tasks and is generally preferred over the HMM in a Monte Carlo goodness of fit test.

Published by Elsevier Ltd.

1. Introduction

The hidden Markov model (HMM), which dates back over 50 years [1], has seen numerous applications in the recognition of human behavior, such as speech [2], gesture [3], and handwriting [4]. Recent successes have leveraged the expressive power of connectionist models by combining the HMM with feed-forward deep neural networks, which are used to estimate emission probabilities [5–7]. Despite the increasing interest in sequential deep learning techniques, e.g., recurrent neural networks, HMMs remain tried-and-true for time series analyses. The popularity and endurance of the HMM can be at least partially attributed to the tractability of core problems (parameter estimation and likelihood calculation), ability to be combined with other methods, and the level of insight it provides to the data.

At least part its success can also be attributed to its flexibility, with many HMM variants having been developed for specific applications. This usually involves introducing a dependence, whether it be on time [8], previous observations [9], or a semantic context [10]. The motivation for doing so is often to better reflect the structure of the underlying problem. Although many of these variations have increased complexity and number of parameters over the standard HMM, their estimation remains tractable.

In this work, we introduce the partially observable hidden Markov model (POHMM), an extension of the HMM intended for keystroke dynamics. We are interested in modeling the temporal behavior of a user typing on a keyboard, and note that certain keyboard keys are thought to influence typing speed. Non-letter keys, such as punctuation and the Space key, indicate a greater probability of being in a *passive* state of typing, as opposed to an *active* state, since the typist often pauses between words and sentences as opposed to between letters in a word [11]. The POHMM reflects this scenario by introducing a dependency on the key names which are observed alongside the time intervals, and in this way, the keys provide a context for the time intervals.

The idea of introducing a context upon which some behavior depends is not new. Often, an observation depends not only on a latent variable but on the observations that preceded it. For example, the neighboring elements in a protein secondary structure can provide context for the element under consideration, which is thought to depend on both the previous element and a hidden state [9]; nearby phonemes can aid in the recognition of phonemes [12]; and the recognition of human activities can be achieved with greater accuracy by considering both a spatial context (e.g., where the activity occurred) and temporal context (e.g., the duration of the activity) [13].

Handwriting recognition has generally seen increased performance with models that consider the surrounding context of a

* Corresponding author.
E-mail address: john.v.monaco2.civ@mail.mil (J.V. Monaco).
URL: http://www.vmonaco.com (J.V. Monaco)

https://doi.org/10.1016/j.patcog.2017.11.021
0031-3203/Published by Elsevier Ltd.

handwritten character. The rationale for such an approach is that a character may be written with different style or strokes depending on its neighboring characters in the sequence. Under this assumption, the neighboring pixels or feature vectors of neighboring characters can provide additional context for the character under consideration. Alternatively, a separate model can be trained for each context in which the character appears, e.g., "t" followed by "e" versus "t" followed by "h" [10]. This same principle motivates the development of the POHMM, with the difference being that the context is provided not by the observations themselves, but by a separate sequence.

We apply the POHMM to address the problems of user identification, verification, and continuous verification, leveraging keystroke dynamics as a behavioral biometric. Each of these problems requires estimating the POHMM parameters for each individual user. Identification is performed with the maximum a posteriori (MAP) approach, choosing the model with maximum a posterior probability; verification, a binary classification problem, is achieved by using the model log-likelihood as a biometric score; and continuous verification is achieved by accumulating the scores within a sliding window over the sequence. Evaluated on five public datasets, the proposed model is shown to consistently outperform other leading anomaly detectors, including the standard HMM, in biometric identification and verification tasks and is generally preferred over the HMM in a Monte Carlo goodness of fit test.

All of the core HMM problems remain tractable for the POHMM, including parameter estimation, hidden state prediction, and likelihood calculation. However, the dependency on event types introduces many more parameters to the POHMM than its HMM counterpart. Therefore, we address the problem of parameter smoothing, which acts as a kind of regularization and avoids overfitting [14]. In doing so, we derive explicit marginal distributions, with event type marginalized out, and demonstrate the equivalence between the marginalized POHMM and the standard HMM. The marginal distributions conveniently act as a kind of backoff, or fallback, mechanism in case of missing data, a technique rooted in linguistics [15].

The rest of this article is organized as follows. Section 2 briefly describes keystroke dynamics as a behavioral biometric. Section 3 introduces the POHMM, followed by a simulation study in Section 4 and a case study of the POHMM applied to keystroke dynamics in Section 5. Section 6 reviews previous modeling efforts for latent processes with partial observability and contains a discussion. Finally, Section 7 concludes the article. The POHMM is implemented in the `pohmm` Python package and source code is publicly available.[1]

2. Keystroke dynamics

Keystroke dynamics refers to the way a person types. Prominently, this includes the timings of key press and release events, where each keystroke is comprised of a press time t_n and a duration d_n. The time interval between key presses, $\tau_n = t_n - t_{n-1}$, is of interest. Compared to random time intervals (RTIs) in which a user presses only a single key [16], key press time intervals occur between different keys and are thought to be dependent on key distance [11]. A user's keystroke dynamics is also thought to be relatively unique to the user, which enable biometric applications, such as user identification and verification [17].

As a behavioral biometric, keystroke dynamics enables low-cost and non-intrusive user identification and verification. Keystroke dynamics-based verification can be deployed remotely, often as a second factor to username-password verification. Some of the same attributes that make keystroke dynamics attractive as a behavioral biometric also present privacy concerns [18], as there exist numerous methods of detecting keystrokes without running software on the victim's computer. Recently, it has been demonstrated that keystrokes can be detected through a wide range of modalities including motion [19], acoustics [20], network traffic [21], and even WiFi signal distortion [22].

Due to the keyboard being one of the primary human-computer interfaces, it is also natural to consider keystroke dynamics as a modality for *continuous verification* in which a verification decision is made upon each key pressed throughout a session [23]. Continuous verification holds the promise of greater security, as users are verified continuously throughout a session beyond the initial login, which is considered a form of *static verification*. Being a sequential model, the POHMM is straightforward to use for continuous verification in addition to identification and static verification.

Keystroke time intervals emanate from a combination of physiology (e.g., age, gender, and handedness [24]), motor behavior (e.g., typing skill [11]), and higher-level cognitive processes [25], highlighting the difficulty in capturing a user's typing behavior in a succinct model. Typing behavior generally evolves over time, with highly-practiced sequences able to be typed much quicker [26]. In biometrics, this is referred to as *template aging*. A user's keystroke dynamics is also generally dependent on the typing task. For example, the time intervals observed during password entry are much different than those observed during email composition.

3. Partially observable hidden Markov model

The POHMM is intended for applications in which a sequence of *event types* provides context for an observed sequence of *time intervals*. This reasoning extends to activities other than keystroke dynamics, such as email, in which a user might be more likely to take an extended break after sending an email instead of receiving an email, and programming, in which a user may fix bugs quicker than making feature additions. The events types form an independent Markov chain and are observed alongside the sequence of time intervals. This is in contrast to HMM variants where the neighboring observations themselves provide a context, such as the adjacent characters in a handwritten segment [10]. Instead, the event types are independent of the dynamics of the model.

With this structure, a distinction can be made between user *behavior* and *task*: the time intervals comprise the *behavior*, and the sequence of event types, (e.g., the keys pressed) comprise the *task*. While the time intervals reflect *how* the user behaves, the sequence of events characterize *what* the user is doing. This distinction is appropriate for keystroke dynamics, in which the aim is to capture typing behavior but not the text itself which may be more appropriately modeled by linguistic analysis. Alternatively, in case the user transcribes a sequence, such as in typing a password, the task is clearly defined, i.e. the user is instructed to type a particular sequence of characters. The POHMM aims to capture the temporal behavior, which depends on the task.

3.1. Description

The HMM is a finite-state model in which observed values at time t depend on an underlying latent process [2]. At the nth time step t_n, a feature vector $\mathbf{x}_n$ is emitted and the system can be in any one of M hidden states, z_n. Let $\mathbf{x}_1^N$ be the sequence of observed emission vectors and z_1^N the sequence of hidden states, where N is the total number of observations. The basic HMM is defined by the recurrence relation,

$$P\left(\mathbf{x}_1^{n+1}, z_1^{n+1}\right) = P(\mathbf{x}_1^n, z_1^n) P(\mathbf{x}_{n+1}|z_{n+1}) P(z_{n+1}|z_n) \ . \quad (1)$$

[1] Available at https://github.com/vmonaco/pohmm and through PyPI.