12th IFAC/IEEE Workshop on Discrete Event Systems
Cachan, France. May 14-16, 2014

Pattern Mining for Predicting Critical Events from Sequential Event Data Log

Jun Chen * and **Ratnesh Kumar** *

* *Department of Electrical and Computer Engineering*
Iowa State University, Ames, IA 50011 USA
e-mail: {junchen,rkumar}@iastate.edu.

Abstract: This paper studies the mining of patterns for predicting critical events from observed ordered event data, where the observations can contain interleaving from non-predictor and other predictor event sequences. These are characteristics of many practical applications such as monitoring in power systems or telecommunication networks, as well as computational biology. For settings where system behaviors are affected by noise, a critical event can sometimes occur without its predictor executed prior to it, and we propose algorithm to recursively compute the frequency that a predictor candidate precedes the critical event. This we use for identifying a predictor, and study the performance of such a scheme. We also consider the noise-free settings, in which a critical event occurs only after the execution of its predictor, and propose an algorithm to recursively compute the set of maximal predictors for each critical event.

Keywords: Pattern mining, finite automata, alarm monitoring, common subsequence, algorithm.

1. INTRODUCTION

For certain monitoring applications such as failure diagnosis (Chen and Kumar (2013a,b,c)) and failure prediction (Kumar and Takai (2010); Chen and Kumar (2014b)), knowledge of system model is essential for any detection algorithm. However, in many scenarios, such as business processes, offline determination of a detailed model describing a certain process is challenging and many times impossible since the system may be overly complex, e.g., vast legacy software. The task is further complicated by the fact that, there exists discrepancy between a *prescriptive* model describing how process is expected to work and a *driving* model according to which the system evolves. Thus even the prescriptive models may not be used (while driving models are overly complex to obtain). Therefore, instead of performing offline process/workflow modeling, there have been efforts towards online process/workflow mining, where the idea is to develop a model describing the underlying process, given the workflow logs in terms of ordered event data. See the next section for literature information on workflow/process mining.

The task of mining workflow/system model is computationally hard (Hsu et al. (2012)), and instead, the simpler problem of pattern mining intends to find certain sequence patterns that precede certain critical events. Such predictor pattern identification problem has many real-world applications such as alarm log processing in power systems and telecommunication networks, and genetic motif discovery in computational biology (e.g., Liu et al. (1995), Chudova and Smyth (2002) and Bailey and Elkan (1995)). The authors Agrawal and Srikant (1995); Pei et al. (2004); Han et al. (1999) and Cheng et al. (2005) study a type of pattern mining problem where a pattern is simply a repetitive sequence of events, not necessarily associated with a critical event.

In this paper, we study the mining of patterns for predicting critical events from observed ordered event data, where the observations can contain interleaving from non-predictor and other predictor event sequences. Two different scenarios, noisy and noise-free, are considered. In the settings where system behaviors are affected by noise, there is a non-zero probability that a critical event can occur without being preceded by its predictor. We propose an algorithm to recursively compute the frequency that a predictor candidate precedes the critical event. A candidate is deemed to be a predictor for certain critical event if its frequency of occurrence preceding to that critical event is above a user-specified confidence level and it is "maximal", i.e., none of its *supersequence* is a predictor candidate.

We also consider the noise-free settings, in which a critical event can occur only after the execution of its predictor, and the predictor mining reduces to finding the maximal common subsequences, which has been studied by Hunt and Szymanski (1977); Hirschberg (1975); Allison and Dix (1986). We propose another recursive algorithm to compute the set of maximal predictors for each critical event. The related work by Wang and Johnson (2007) considers a similar problem of pattern mining from event sequence log, where the patterns are assumed to be contiguous event sequences of the system model. In this paper, we relax such assumption about contiguity by allowing interleaving among the predictors as well as the non-predictor sequences.

⋆ The research was supported in part by the National Science Foundation under the grants, NSF-ECCS-0801763, NSF-ECCS-0926029, and NSF-CCF-1331390.

The contributions of this paper are summarized as follows:

- Compared to prior works, we allow the observed ordered event data to contain interleaving from non-predictor and other predictor event sequences;
- We propose recursive algorithms for mining predictor patterns for both noisy and noise-free settings;
- For noisy setting, we provide the existence of a bound for the length of event log that guarantees the proposed algorithm to output correct predictor within a desired error probability.

The rest of this paper is organized as follows: Section 2 briefly describes some related work on process mining. The notation and some preliminaries are presented in Section 3. The formulation of the pattern mining problem is presented in Section 4, which also provides the recursive algorithm for the computing the predictors in setting of noisy observations. Section 5 considers the noise-free observations and provides an algorithm to recursively compute the set of maximal predictors for each critical event. The paper is concluded in Section 6 with directions for future work.

2. RELATED WORK ON PROCESS MINING

While pattern mining attempts to discover the signatures for critical events, process mining or model identification (Cabasino et al. (2011)) is a more ambitious endeavor that tries to uncover the entire process model. Being again driven by the observed ordered event data, it is a related topic of research, and here we provide a short summary for interested readers. An algorithm to extract a process model in form of a Petri net from event sequence log was introduced in van der Aalst et al. (2003, 2004), where the set of underlying model that can be extracted from event sequence log is also given. Cook and Wolf (1998) investigated the process mining problem in the context of software engineering processes, under the framework of *grammar inference* by Gold (1967, 1978) and Angluin (1987). Cook and Wolf (1998) presented several approaches: i) *RNet*, which is based on neural network; ii) *Ktail*, which outputs a finite state automaton whose state space is given by the set of equivalence classes of traces that have the same k-step extensions; and iii) *Markov* method, which assumes that the underlying model is Markovian with order at most 2, whose dependencies are statistically deduced when the occurrence frequencies of contiguity for event pairs is above a user-specified threshold.

As indicated by Cook and Wolf (1998), the process mining can be cast into a grammar inference problem as in Gold (1967, 1978) and Angluin (1987). An algorithm for grammar inference, called L^* algorithm, was proposed by Angluin (1987) which requires a membership oracle as well as an equivalence oracle, the former of which identifies the membership of a given trace while the latter can confirm the correctness of a postulated grammar and return a counterexample if the postulate is false. The stochastic grammar identification was addressed in Carrasco and Oncina (1994) by merging the equivalent nodes in a complete prefix tree, where two nodes are deemed equivalent if they have equivalent successors and the distance between their distributions over the set of successors is within a tolerance. The identification of stochastic grammar can also be addressed in the framework of source identification. Cybenko and Crespi (2011) consider the identification of a hidden Markov model from the observed symbols sequence using *nonnegative matrix factorization*. The proposed algorithm in Cybenko and Crespi (2011) computes a frequency matrix by computing, for each pair of sequence of observed symbols, the occurrence frequency that the first sequence of the pair is followed immediately by the second sequence of that pair. The order of the hidden Markov model as well as the state transition matrix are then estimated from this frequency matrix, by computing its *positive rank* and nonnegative matrix factorization, which in general is approximated by optimizing an objective function consisting of a type of divergence. It turns out that (see Hsu et al. (2012)) identification of hidden Markov models from data is computationally hard. Hence the literature also explores the simpler problem of order estimation of a Markov model. Merhav et al. (1989) studied the estimation of the order of a fully observable Markov process, whereas Liu and Narayan (1994) investigated the order estimation of hidden Markov model. The order estimation problem examines the dependency of data in the observed sequence with its preceding history, and is relevant for data compression (source coding) for storage and communication.

3. NOTATIONS AND PRELIMINARIES

For an event set Σ, define $\overline{\Sigma} := \Sigma \cup \{\epsilon\}$, where ϵ denotes "no-event". The set of all finite length event sequences over Σ, including ϵ, is denoted as Σ^*. A *trace* is a member of Σ^*, i.e., a trace is a sequence of events, and a *language* is a subset of Σ^*. We use $s \leq t$ to denote that $s \in \Sigma^*$ is a prefix of $t \in \Sigma^*$, $pr(s)$ to denote the set of all prefixes of s, and $|s|$ to denote the length of s or the number of events in s. For $\sigma \in \Sigma$ and $s \in \Sigma^*$, we use $\sigma \in s$ to denote that the event σ is an element of the trace s. For $\sim \in \{<, \leq, >, \geq, =\}$ and $n \in \mathbb{N}$, where $\mathbb{N}$ denotes the set of all nonnegative integers, define $\Sigma^{\sim n} := \{s \in \Sigma^* \mid |s| \sim n\}$ and denote $\Sigma^{=n}$ as Σ^n for simplicity. For any $s = \sigma_1 \dots \sigma_{|s|} \in \Sigma^*$, denote as $s^{-1} := \sigma_{|s|} \dots \sigma_1$ for the *reverse* of s. $t \in \Sigma^{\leq |s|}$ is said to be a *subsequence* of s, denoted $t \ll s$, if there exists indices $1 \leq i_1 \leq \dots \leq i_{|t|} \leq |s|$ such that $t = \sigma_{i_1} \dots \sigma_{i_{|t|}}$, and in this case we also call s as a *supersequence* of t. t is said to be a common subsequence of s_1 and s_2 if $t \ll s_1$ and $t \ll s_2$. The interleaving product of a pair of traces $s, t \in \Sigma^*$, denoted $s \bowtie t$, is defined recursively as follows: $\epsilon \bowtie \epsilon := \epsilon$; $\forall s, s' \in \Sigma^*, \sigma, \sigma' \in \Sigma : s\sigma \bowtie s'\sigma' := (s\sigma \bowtie s').\sigma' + (s \bowtie s'\sigma')\sigma$. The interleaving product of two languages L_1 and L_2 is given by, $L_1 \bowtie L_2 := \{s \bowtie t \mid s \in L_1, t \in L_2\}$.

A *finite state automaton* is a tuple $G = (X, \Sigma, \alpha, X_0)$, where X is the set of states, Σ is the set of events, $\alpha : X \times \overline{\Sigma} \to 2^X$ is the transition function, and $X_0 \subseteq X$ is the set of initial states. G is deterministic if $|X_0| = 1$, i.e., a unique initial state, and $\forall x \in X, \sigma \in \Sigma, |\alpha(x, \sigma)| \leq 1$ and $|\alpha(x, \epsilon)| = 0$, i.e., each state has at most one transition on each event and no transition on "no-event"; otherwise G is called nondeterministic. A path of G is a sequence of transitions $(x_1, \sigma_1, x_2, \dots, \sigma_{n-1}, x_n)$ such that $\sigma_i \in \overline{\Sigma}$ and $x_{i+1} \in \alpha(x_i, \sigma_i)$ for each $1 \leq i \leq n-1$. A path is called a cycle if $x_n = x_1$. For any $x \in X$, define the ϵ-closure of x, denoted as $\epsilon^*_G(x)$, as the set of all states that can be