Predicting Rare Events In Temporal Domains

Ricardo Vilalta
Department of Computer Science
University of Houston
Houston TX, 77204-3010, USA
vilalta@cs.uh.edu

Sheng Ma
T.J. Watson Center
IBM Research
19 Skyline Dr., Hawthorne N.Y., 10532 USA
shengma@us.ibm.com

Abstract

Temporal data mining aims at finding patterns in historical data. Our work proposes an approach to extract temporal patterns from data to predict the occurrence of target events, *such as computer attacks on host networks, or fraudulent transactions in financial institutions. Our problem formulation exhibits two major challenges: 1) we assume events being characterized by categorical features and displaying uneven inter-arrival times; such an assumption falls outside the scope of classical time-series analysis, 2) we assume target events are highly infrequent; predictive techniques must deal with the class-imbalance problem. We propose an efficient algorithm that tackles the challenges above by transforming the event prediction problem into a search for all frequent eventsets preceding target events. The class imbalance problem is overcome by a search for patterns on the minority class exclusively; the discrimination power of patterns is then validated against other classes. Patterns are then combined into a rule-based model for prediction. Our experimental analysis indicates the types of event sequences where target events can be accurately predicted.*

1 INTRODUCTION

Learning to predict infrequent but highly correlated subsequences of events, such as an attack in a computer network, or a fraudulent transaction in a financial institution, is a difficult problem. The difficulty in learning to recognize rare events may stem from several sources: few examples support the target class; events are described by categorical features that display uneven inter-arrival times; and time recordings only approximate the true arrival times, such as occurs in computer-network logs, transaction logs, speech signals, etc.

In this paper we describe an approach to predict rare events, called *target events*, in event sequences with categorical features, uneven inter-arrival times, and noisy signals. Our approach differs from previous work in the learning strategy. Most learning algorithms assume even class distributions and adopt a discriminant-description strategy: they search for a separator that best discriminates examples of different class. Under skewed distributions, however, separating away the under-represented class is difficult. We show that a better approach is to adopt a characteristic-description strategy: we specify common properties of examples of the same class before validating those properties against other classes. The new strategy helps improve efficiency, accuracy, and interpretability.

Our prediction strategy first looks for common properties among the target events alone (i.e., the under-represented class). Common properties, or patterns, are then validated against the non-target-event class. Finally, patterns are combined into a rule-based model for prediction. More specifically, our approach divides into the following steps:

1. We characterize target events by finding the types of events frequently preceding target events within a fixed time window.

2. We validate that these event types uniquely characterize target events, and do not occur often far from the time arrival of target events.

3. We combine validated event types to build a rule-based system for prediction.

Since our final rules are obtained by combining association rule mining and classification techniques, they receive the name of associative classification rules (Liu, Hsu, & Ma, 1998; Pijls & Potharst, 2000; Li, Han, & Pei, 2001; Dong, Zhang, Wong, & Li, 1999). Most previous work has focused on exploring this type of integration without any concern for the time dimension. We show how the same integration on event sequences requires special consideration.

We test our approach in the domain of service problems in computer networks, where predicting rare events brings multiple benefits. First, detecting system failures on a few

servers can prevent widespread dissemination of those failures over the entire network. For example, low response time on a server may gradually escalate to technical difficulties on all nodes attempting to communicate with that server. Second, prediction can be used to ensure continuous provision of network services through the automatic implementation of corrective actions. For example, prediction of high CPU demand on a server can initiate a process to balance the CPU load by re-routing new demands to a back-up server.

Using both artificial and real production-network data, we conduct an empirical evaluation of our rule-based model. Our results show low error rates when the probability of a pattern occurring by chance is low, and when the size of the time window preceding target events is sufficiently large (Section 7).

Our contributions can be enumerated as follows:

- We propose an approach to transform an event sequence described by categorical features and uneven inter-arrival times into a prediction problem where the goal is to anticipate a specific class of events called target events. Our approach shows how a characteristic description approach provides an advantage under skewed distributions where the positive class corresponds to the minority class.

- We introduce an efficient algorithm that conducts a search for all patterns preceding the occurrence of target events within memory. Furthermore, the algorithm requires only two passes over the original event sequence. On each pass, only those events within a sliding time window are kept in memory. As a result, we are able to conduct an exhaustive search for rules in short time.

- Our approach combines association-rule mining with classification in temporal domains. Most previous work has focused on exploring this type of integration disregarding the role of time. We show how the integration requires a unique approach because of the time dimension and the class-imbalance problem.

- We perform an analytical study of learning rare events in temporal domains. In summary, our study shows how the probability of a pattern being generated randomly decreases with 1) long patterns, 2) large number of event types, and 3) few events occurring within the time intervals of interest.

The paper organization is described next. Section 2 introduces background information and our problem statement. Section 3 presents the logic to find all frequent and accurate eventsets preceding target events. Section 4 describes how to combine the frequent and accurate set of eventsets into a rule-based model. Section 5 analyzes the computational efficiency of our approach. Section 6 investigates the learnability of rare events in temporal domains. Section 7 reports an empirical assessment of our approach. Section 8 reviews related work. Finally, Section 9 states our summary and conclusions.

2 THE EVENT PREDICTION PROBLEM

In our problem formulation the fundamental unit of study is an **event**. Events belong to sequences, and can be identified by their time of occurrence and type (Mannila, Toivonen, & Verkamo, 1995).

Definition 1. A sequence of events is an ordered collection of events $D = (d_i)$, where each event d_i is a pair $d_i = (e_i, t_i)$. The first element of the pair, e_i, indicates the event type. The second element of the pair, t_i, indicates its occurrence time.

We assume event types come from a finite alphabet $\mathcal{E}$, $(\forall i)(e_i \in \mathcal{E})$, and that all events are ordered along time, $(\forall i, j)(t_i \leq t_j \rightarrow i < j)$.

For example, events obtained when monitoring a computer network can be classified based on severity level (e.g., normal, critical, fatal), or by the kind of disruption (e.g., node-down, link-failed, printer-failed, etc.). In a sequence of events, ((node-down, t_1), (link-failed, t_2), etc)), each t_i specifies date and time. If multiple attributes are necessary to describe an event, one may think of event type e_i as an attribute vector.

We are interested in predicting certain kinds of events that occur in sequence D. We refer to this subset of events, $D_{\text{target}} \subset D$, as *target events*. We assume the proportion of target events with respect to all events in sequence D is low; target events do not represent a *global* property of D, such as periodicity or constant trend, but rather a *local* property of D, such as a computer attack on a host network.

We assume the user specifies a target event type $e_{\text{target}} \in \mathcal{E}$, that characterizes D_{target} as,

$$D_{\text{target}} = \{d_i \in D | e_i = e_{\text{target}}\} \tag{1}$$

For example, if severity level defines the type of event, target events could be all events sharing one kind of severity level (e.g., all fatal events). The definition of target events, however, can be made more general. We can use an arbitrary function over the features characterizing each event (e.g., all events occurring in the time interval $[t_1, t_2]$).

2.1 PROBLEM STATEMENT

Our framework assumes a dataset D of size n, containing a sequence of events (Definition 1). Event types take on

categorical values from an alphabet $\mathcal{E}$. We also assume we have identified a set of events, $D_{\text{target}} \subset D$, of size m, $m \ll n$, as our set of target events.

In this paper we describe an approach to capture patterns characterizing the conditions preceding each target event. We wish to know what types of events frequently precede a target event for the purpose of prediction. In particular we look at those events occurring within a time window of fixed size W before a target event, as illustrated in Figure 1. We are interested in finding sets of event types, called *eventsets*, which occur frequently before a target event within a window of size W.

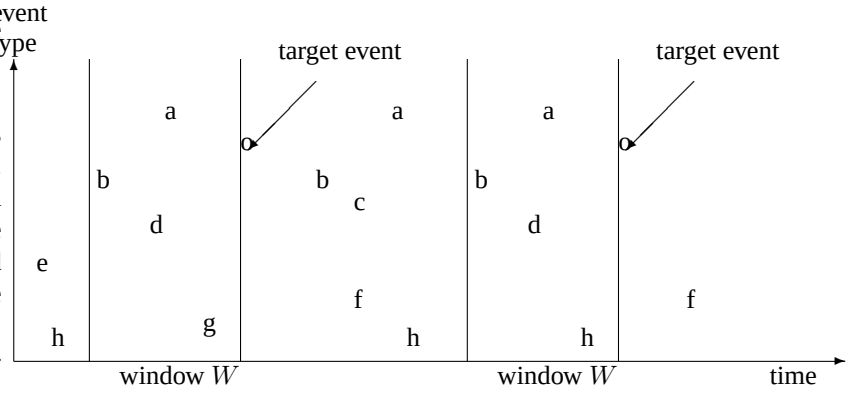

Figure 1. A plot of different event types vs. time. A time window W before each target event enables us to identify frequent sets of event types preceding each target event.

Definition 2. An eventset Z is a set of event types $\{e_i\}$. Eventset Z matches the set of events in window W, if every event type $e_i \in Z$ is found in W.

For example, in Figure 1, eventset $\{a, b, d\}$ matches the sets of events within the two time windows preceding the target events.

Definition 3. Eventset Z has *support* s in D if $s\%$ of all windows of size W preceding target events are matched by Z. Eventset Z is *frequent* if s is above a minimum user-defined threshold.

Definition 4. Eventset Z has *confidence* c in D if $c\%$ of all time windows of size W matched by Z precede a target event. Eventset Z is *accurate* if c is above a minimum user-defined threshold.

Notice our definition of support is not simply an extension of that used in non-temporal domains: it is restricted to eventsets preceding target events. A periodic eventset in D may not qualify as frequent if it fails to appear before target events. In Figure 1, for example, eventset $\{a, b, d\}$ has support 100%, since it always appears before a target event; its confidence is also 100%. Eventset $\{a, b\}$, on the other hand, has support 100%, but since it also appears within a time window of size W that does not precede a target event, its confidence is 67%.

Our goals can now be enumerated as follows: 1) to find all frequent and accurate eventsets, and 2) to combine eventsets into a rule-based model for prediction. We describe each step in turn.

3 SEARCHING FOR EVENTSETS

This section presents the logic to find all frequent and accurate eventsets from input sequence D, given a set of target events D_{target}. We proceed by first finding all frequent eventsets.

3.1 FREQUENT EVENTSETS

Figure 2 (Algorithm 1) illustrates the logic to find all frequent eventsets. The general idea is to maintain in memory all events within a sliding window of size W. On every occurrence of a target event, all event types within the sliding window are simply stored as a new transaction. Once all events have been analyzed, it is straightforward to apply the A-priori algorithm to find all eventsets above a minimum user-defined support.

Specifically, our algorithm makes one pass through the sequence of events in D (lines 2-11), which we assume to be in increasing order along time. With each new event, the current time is updated (line 3); the algorithm keeps in memory only those events within a time window of size W from the current time (lines 5-8). If the current event is a target event, the set of event types contained in the most recent time window become a new transaction in database B (lines 9-10). Finally, we invoke the A-priori algorithm to find all eventsets above a minimum user-defined support over B (lines 12-14).

As an example, the result of applying Algorithm 1 over the data shown in Figure 1 yields the following database of event types: $B = \{\{a, b, d, g\}, \{a, b, d, h\}\}$. Although not shown in Figure 1, it is admissible for consecutive target events to generate time windows that overlap. If the minimum support is 1.0, then eventset $\{a, b, d\}$ is the only frequent eventset.

Note that both the ordering of events and the inter-arrival times between events within each time window are not relevant. This is useful when an eventset occurs under different permutations, and when inter-arrival times exhibit high variation (i.e., signals are noisy). These characteristics are

Algorithm 1: Finding Frequent Eventsets
Input: event sequence D, window size W, minimum support $s\%$, target-event type e^*
Output: frequent eventsets $\mathcal{F}$

FREQUENTEVENTSETS(D,W,s,e^*)
(1) $B = \emptyset$; $T = \emptyset$
(2) **foreach** event $d_i = (e_i, t_i) \in D$
(3) currentTime $= t_i$
(4) **foreach** event $d_j = (e_j, t_j) \in T$
(5) **if** (currentTime $- t_j) > W$
(6) Remove d_j from T
(7) **end**
(8) **if** d_i is a target event (i.e., $e_i = e^*$)
(9) $B = B \cup \{e_j \mid (e_j, \cdot) \in T\}$
(10) $T = T \cup d_i$
(11) **end**
(12) Use A-priori over B to find all frequent
(13) eventsets with minimum support $s\%$.
(14) Let $\mathcal{F}$ be the set of all frequent eventsets.
(15) **return** $\mathcal{F}$

Figure 2. Logic to find all frequent eventsets.

present in many domains, including the real production network used for our experiments. For example, we observed a printer-network problem may generate a set of events under different permutations, and with inter-arrival-time variation in the order of seconds. Our approach to overcome these uncertainties is to collect all event types falling inside the time windows preceding target events, which can then be simply treated as database transactions.

3.2 ACCURATE EVENTSETS

Once the set of frequent eventsets is available, we proceed to filter out those eventsets below a minimum degree of confidence. Figure 3 (Algorithm 2), illustrates the filtering mechanism. The general idea is to look at the number of times each of the frequent eventsets occurs outside the time windows preceding target events. Such information enables us to compute the confidence of each frequent eventset and to eliminate those below a minimum threshold.

Specifically, the first part of the algorithm (lines 1-10) assumes as input a set of intervals corresponding to all time windows of size W that do not overlap with the time windows preceding target events. These are the true negative time windows[1]. We proceed to capture all event types within each window (line 6) to construct a new database of eventsets, B'. This database contains all eventsets not preceding target events. The second part of the algorithm uses our two eventset databases, B and B', to compute the confidence of each frequent eventset Z (lines 11-16). Let x_1 and x_2 be the number of transactions in B and B' respectively matched by eventset Z. Then the confidence of Z is defined as follows:

$$\text{confidence}(Z, B, B') = x_1/(x_1 + x_2) \quad (2)$$

Confidence is an estimation of the conditional probability of Z belonging to a time window that precedes a target event, given that Z matches the event types in that same time window.

Our filtering mechanism performs one more test to validate an eventset (line 14). The reason is that confidence alone is not sufficient to guarantee that the probability of finding an eventset Z within database B is significantly higher than the corresponding probability in B'; confidence does not check for negative correlations (Brin, Motwani, & Silverstein, 1997). Thus, we add a validation step described as follows.

Let $P(Z|B)$ denote the probability of Z occurring within database B, and $P(Z|B')$ the corresponding probability within B'. Eventset Z is validated if we can reject the null hypothesis:

$$H_0 : P(Z|B) = P(Z|B') \quad (3)$$

Algorithm 2: Finding Confident Eventsets
Input: event sequence D, minimum confidence $c\%$, time intervals I, frequent eventsets $\mathcal{F}$, database eventsets B
Output: confident eventsets $\mathcal{F}'$

CONFIDENTEVENTSETS(D,c,I,$\mathcal{F}$,B)
(1) $T = \emptyset$; [a,b] = next interval from I
(2) **foreach** event $d_i = (e_i, t_i) \in D$
(3) **if** $t_i \in [a, b]$
(4) $T = T \cup d_i$
(5) **if** $t_i > b$
(6) $B' = B' \cup \{e_j \mid (e_j, \cdot) \in T\}$
(7) $T = \emptyset$; $[a, b]$ = next interval from I
(8) **if** $t_i \in [a, b]$
(9) $T = T \cup d_i$
(10) **end**
(11) $\mathcal{F}' = \emptyset$
(12) **foreach** eventset Z in $\mathcal{F}$
(13) **if** confidence(Z,B,B')$> c$ AND
(14) $P(Z|B) > P(Z|B')$
(15) $\mathcal{F}' = \mathcal{F}' \cup Z$
(16) **end**
(17) **return** $\mathcal{F}'$

Figure 3. Logic to find all confident eventsets.

[1]These intervals can be easily obtained as a side output from Algorithm 1.

with high confidence. If the number of events is large, one can assume a Gaussian distribution and reject the null hypothesis in favor of the alternative hypothesis:

$$H_1 : P(Z|B) > P(Z|B') \quad (4)$$

if, for a given confidence level α, the difference between the two probabilities (normalized to obtain a standard normal variate) is significant. In such case we reject H_0. The probability of this happening when H_0 is actually true is α. By choosing a small α we can be almost certain that Z is related to the occurrence of target events.

In summary, our validation phase ensures that the probability of an eventset Z appearing before a target event is significantly larger than the probability of Z not appearing before target events. The validation phase discards any negative correlation between Z and the occurrence of target events. In addition, this phase serves as a filtering step to reduce the number of candidate patterns used to build a rule-based model for prediction.

Together, Algorithms 1 and 2 enable us to attain our goal of finding the set of all frequent and accurate eventsets. We now turn to our goal of finding a model for prediction.

4 BUILDING A RULE-BASED MODEL

In this section we describe how to combine the frequent and accurate set of eventsets into a rule-based model. Before describing our algorithm we provide definitions that are common in the construction of associative classification rules (Liu et al., 1998).

Definition 5. Eventset Z_i is said to be more specific than eventset Z_j, if $Z_j \subset Z_i$.

For example, eventset $\{a, b, c\}$ is more specific than eventset $\{a, b\}$.

Definition 6. Eventset Z_i is said to have higher rank over eventset Z_j, represented as $Z_i \succ Z_j$, if any of the following conditions is true:

1. The confidence of Z_i is greater than that of Z_j.
2. The confidence of Z_i equals that of Z_j, but the support of Z_i is greater than the support of Z_j
3. The confidence and support of Z_i equal that of Z_j, but Z_i is more specific than Z_j.

Definition 6 imposes a partial ordering over the space of eventsets. Note condition 3 favors eventsets that represent maximal characteristic descriptions of the target class (i..e, specific descriptions). Most approaches prefer minimal discriminant descriptions (i.e., general descriptions). The difference stems from our approach to classification: instead of discriminating among classes, we first characterize the target class, and then validate our descriptions. This is convenient when the a priori probability of the target class is small. Otherwise we must face the class imbalance problem in classification (Japkowicz, 2000).

Algorithm 3: Building Rule-Based Model
Input: eventsets $\mathcal{F}'$
Output: Set of rules $\mathcal{R}$
RULE-BASED-EVENTSETS($\mathcal{F}'$)
(1) $\mathcal{R} = \emptyset$
(2) Sort $\mathcal{F}'$ in decreasing order by rank
(3) **while** $\mathcal{F}'$ is not empty
(4) Let Z_i be the first eventset in $\mathcal{F}'$
(5) if $Z_j \subset Z_i$, $i \neq j$, remove Z_j from $\mathcal{F}'$
(6) Make a new rule $r : Z_i \rightarrow \mathrm{targetevent}$
(7) $\mathcal{R} = \mathcal{R} \cup r$
(8) Remove Z_i from $\mathcal{F}'$
(9) **end**
(10) **return** $\mathcal{R}$

Figure 4. Logic describing the process to build a rule-based system from eventsets.

The rationale behind our rule-based system is to find the most accurate and specific rules first (Michalski, 1983). Our assumption of having a large number of available eventsets and few positive examples obviates ensuring each example is covered by a rule. Specifically, let $\mathcal{F}'$ be the set of large and validated eventsets. Figure 4 (Algorithm 3) illustrates our approach. The first step (line 2) sorts all eventsets according to definition 6. In general other metrics can be used to replace confidence, such as information gain, gini, or χ^2 (White & Liu, 1994). In the next step (lines 4-6), our algorithm selects the next best eventset Z_i and removes all other eventsets Z_j in $\mathcal{F}'$ more general than Z_i. This step eliminates eventsets that refer to the same pattern as Z_i but are overly general. The resulting rule is of the form $Z_i \rightarrow \mathrm{targetevent}$. The search then continues for all eventsets capturing different patterns preceding the occurrence of target events. The final rule-based system $\mathcal{R}$ can be used for prediction by checking for the occurrence of any of the eventsets in $\mathcal{R}$ along the event sequence used for testing. The model predicts finding a target event within a time window of size W after any such eventset is detected. Section 8 compares our approach with other rule-based methods.

5 COMPUTATIONAL EFFICIENCY

We now analyze the efficiency of our approach. Finding all frequent eventsets requires a single pass over D. Look-

ing for all frequent eventsets is in the worst case exponential in the number m of target events, and not in the size n of the input sequence. Algorithm 1 has a worst-case running time of $O(n + 2^m)$. Since $m \ll n$, we are able to keep in memory the database B of eventsets preceding target events. As a result, finding all frequent eventsets is inexpensive in both memory and time.

Filtering out frequent eventsets below a minimum confidence takes time $O(mf + f)$, where f is the number of frequent eventsets, and requires a single pass over D. Algorithms 1 and 2 obviate keeping the whole event sequence in memory; we keep in memory only those events within a time window of size W.

The algorithm to build a rule-based model for prediction (Figure 4) runs in (worst-case) time $O(f' log_2 f' + f'^2)$, where f' is the number of validated and frequent eventsets. We find the running time to be within a few seconds for large enough minimum support values (Figure 6), which shows our approach makes efficient use of memory and disk space.

6 BEHIND EVENT PREDICTION

In this section we investigate the learnability of rare events in temporal domains. Specifically, we wish to identify the conditions under which a *pattern* in an event sequence can be distinguished from random noise. In our context, *pattern* will mean a frequent and accurate eventset. Our goal is to have a conceptual understanding of the conditions that simplify the task of identifying true patterns.

6.1 PATTERNS OR NOISE?

We begin our analysis with a simple scenario consisting of a single time window. Let $k = |\mathcal{E}|$ be the number of possible event types. We pose the following question: when can we confidently know that a pattern is not the result of pure chance?

Let us make the (strong) assumption that events are independently and identically distributed (i.i.d.) in a fixed time interval. Let $P(e_i)$ be the prior probability of event type e_i occurring within the time window, and let $P'(e_i) = 1 - P(e_i)$. Assume our time window holds exactly x events and that an eventset Z of size z is claimed to be a (previously known) pattern in the time window. We ask the question: what is the probability of Z occurring by chance within the time window? If $z = 1$, the probability that the single event type in Z occurs one or more times in a time window by chance, denoted as $P^+(e_i)$, is one minus the probability that Z misses all x events. If $Z = \{e_i\}$, then this probability is

$$P^+(e_i) = 1 - (P'(e_i))^x \tag{5}$$

The probability of all z event types in Z ($z > 1$) happening within the time window by chance is the product of conditional probabilities. If $Z = \{e_1, e_2...., e_z\}$, then the probability of Z occurring by chance is

$$P^+(e_1) \cdot P^+(e_2|e_1) \cdot P^+(e_3|e_1e_2) \cdot ... \cdot P^+(e_z|e_1e_2 \cdot e_{z-1}) \tag{6}$$

where $P^+(e_i)$ is the probability of event type e_i occurring one or more times within our time window, and $P^+(e_j|A)$ is the equivalent probability for event type e_j given that A already occurred one or more times within the time window. Assuming conditional independence, equation 6 can be upper bounded as:

$$\prod_{i=1}^{z} P^+(e_i) \tag{7}$$

Equation 7 gives an upper bound of the probability of Z occurring by chance. To visualize this equation let us make a further simplification and assume a uniform distribution over the prior probabilities of all event types. If k is the total number of possible event types, then for all e_i:

$$P(e_i) = \frac{1}{k} \quad \text{and} \quad P'(e_i) = \frac{k-1}{k} \tag{8}$$

Under the assumption of a uniform distribution, Equation 7 takes the following form:

$$\left(1 - \left(\frac{k-1}{k}\right)^x\right)^z \tag{9}$$

Figure 5 illustrates the behavior of Equation 9 for different parameter settings. The probability of pattern Z occurring by chance decreases as its size z increases. For $k = 50$ (Figure 5(left)), the number of events x in a time window can produce very different results. For $z = 3$ and $x = 90$, the probability is close to 0.6. But a value of $x = 10$ ($z = 3$) makes this probability negligible. An increase in the number of event types reduces this probability significantly ($k = 200$, Figure 5 (right)).

In summary, our study shows that under certain assumptions, the probability of a pattern Z occurring by chance reduces considerably under the following conditions:

1. long patterns
2. many possible event types
3. few events occurring within the time intervals of interest.

Any form of event prediction must take these conditions into account. Short patterns and few event types increase the possibility of a pattern occurring by chance significantly. Also, a densely populated time window increases the chance of random patterns.

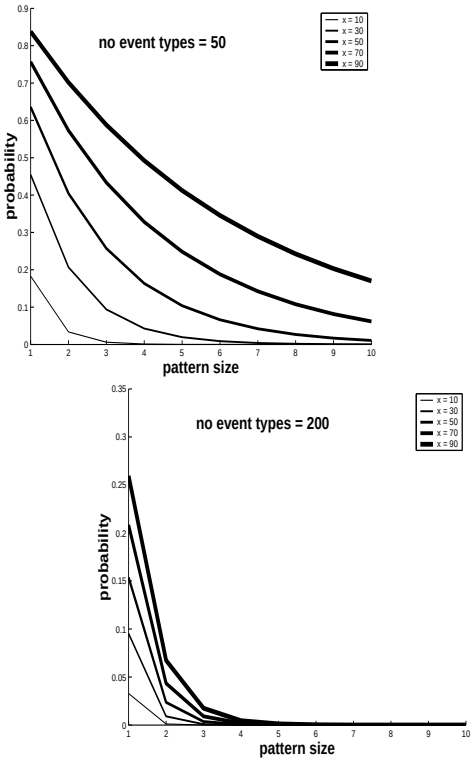

Figure 5. Probability of pattern Z occuring by chance vs size z of Z; the number x of events in the time window varies in the range $[10, 90]$. (left) the number of even types k is set to 50; (right) k is set to 200.

7 EXPERIMENTS

We now describe an empirical assessment of our approach. We use both artificial and real production data to evaluate the performance of our method.

7.1 ARTIFICIAL DATA

To assess the performance of our algorithm we first use artificial domains. Our data generator outputs a sequence of events uniformly distributed over a fixed time interval. By default we use a time interval of 1 week; the total number of event types is set to $k = 50$; we use a time window of size $W = 5$ minutes to capture patterns preceding target events; the number of target events is set to 50; the minimum-support level while searching for large eventsets is set to $s = 0.1$; the significance level for hypothesis testing while validating eventsets is set to $\alpha = 0.01$.

For each event sequence, the first 50% events serve for training and the other 50% serve for testing. Each point in the graphs is the average over 30 runs (i.e., 30 different input sequences) using 95% confidence intervals. Runs were performed on a RISC/6000 IBM model 7043-140.

Error is computed on the testing set only as follows. Starting at the beginning of the sequence, non-overlapping time windows of size W that do not intersect the set of time windows preceding target events are considered negative examples; all time windows preceding target events are considered positive examples. Error is defined as the fraction of examples incorrectly classified by the rule-based model.

ASSESSING ERROR

Our first experiment measures the error of the rule-based model vs. the size of the pattern. Figure 6 (left) shows our results. Since we guarantee that the pattern will always occur within 5-minutes preceding a target event, any form of error stems from the pattern occurring by chance outside the 5-minute window intervals (i.e., errors are false positives). The number x of events per time takes values in the set $\{10, 30, 50\}$. Figure 6 (left) shows results when the number of event types is set to 50. The results agree with our study (Section 6.1) in which the probability of a pattern occurring by chance decreases as the size of the pattern increases. The different lines show how the probability decreases when fewer events populate each time window.

ASSESSING CPU TIME

Our next experiment compares CPU time vs different levels of minimum support f. Figure 6 (right) shows our results. CPU time rarely exceeds 20 seconds when $f > 0.10$, but it grows considerably higher when $f < 0.05$. The value of f imposes a trade-off between computational cost and performance. Setting the value of f too high increases the chances of missing relevant eventsets, while setting the value too low increases the computational cost. In our experiments, a value of $f = 0.10$ appears to provide a good balance between both factors.

7.2 REAL PRODUCTION DATA

We now report results obtained from a production computer network. Data was obtained by monitoring systems active during one month on a network having 750 hosts. One month of continuous monitoring generated over 26,000 events, with 165 different types of events. Our analysis concentrates on two types of target events labeled as *critical* by domain experts. The first type, EPP Event, indicates that end-to-end response time to a host generated by a probing mechanism is above a critical threshold. The second type,

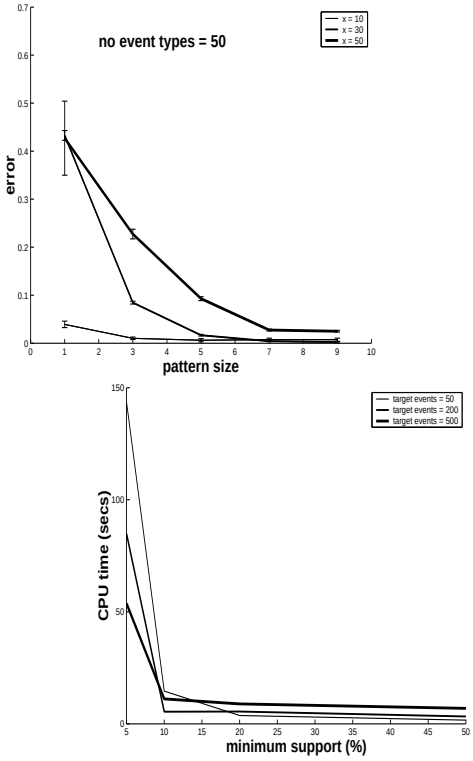

Figure 6. Artificial Data. (left) Error rate vs pattern size when the number of event types k is set to 50; (right) CPU time vs minimum support.

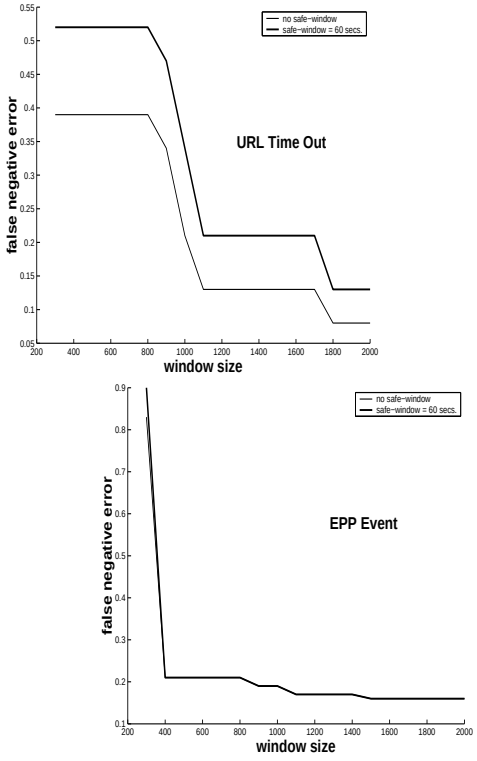

Figure 7. Real Production Data. Error vs window size with and without a safe window. (left) target events is URL Time Out; (right) target event is EPP Event (note that the curves with and without safe window are identical).

URL Time-Out, indicates a web site is unaccessible. Detecting the cause of these event types and providing a mechanism to predict their occurrence is important to guarantee continuous network operation and compliance with service level agreements.

In our production computer network an event is characterized by three features: time, event type, and host. We merge event-type and host into a single feature to identify the nature of the event. A target event is, therefore, the combination of type (EPP Event or URL Time-Out) and a particular host of interest. Our parameters take the same default values as with synthetic data.

VARYING TIME WINDOW

We investigate the effect of varying the time window preceding target events on the error of the rule-based model. In all our experiments, the error corresponding to false positives is small (< 0.1) and does not vary significantly while increasing the time windows. We focus on the false negative error rate defined as the proportion of times the rule-based model fails to predict a true target event. Figure 7-left shows our results when the target event corresponds to URL Time-Out on a particular host. With a time window of 300 seconds, the error is 0.39 $(9/23)$. But as the time window increases, the error decreases significantly. Evidently, larger time windows enable us to capture more information preceding target events. The number of false positives (false alarms) is close to $35/800$.

We also investigate the effect of having a *safe-window* before each target event in case the rule-based model were used in a real-time scenario with a need for corrective actions to take place. In this case, the algorithm does not capture any events within the safe-window while characterizing the conditions preceding target events. Figure 7-left shows our results for a safe-window of 60 seconds. Our results show a degradation of performance when the safe-window is incorporated, albeit to a small degree.

Figure 7-right shows our results with a different target event: EPP Event on a particular host. With a time window of 300 seconds the error is as high as 0.83 $(9/62)$. Increasing the window to 2000 seconds brings the error rate down to 0.16. Our results highlight the importance of the size of

the time window preceding target events: a low value may fail to capture relevant patterns. In this case adding a safe-window causes no degradation of performance.

Not all our experiments show positive results. In some cases the false negative error rate is as high as 95%; increasing the time window does not reduce the error beyond 80%. An inspection of the data reveals no pattern (i.e., sets of event types) preceding target events. Thus, the success of our algorithm is contingent on the existence of patterns preceding target events.

In terms of the significance of the discovered rules, our findings reveal interesting patterns. In the production-network data, for example, a URL Time-Out on a particular host, set as the target event, reveals the frequent presence of a URL Time-Out on a remote host within a 15-minute interval. Such information helps establish a correlation between the two hosts, and may prove crucial in finding the nature of the underlying problem.

8 RELATED WORK

Our approach is different from other methods for classification. Traditional classification methods like C4.5 (Quinlan, 1994) recursively split the instance space until each region is class uniform (i.e., they follow a discriminant-description strategy). Our approach differs not only in the temporal nature of the data but in the prediction strategy. Since in our case target events are highly infrequent, we first characterize the positive class and then validate those descriptions against the negative class (i.e., we follow a characteristic-description strategy). There are two major benefits from our approach. First, infrequent patterns are isolated from the negative class and thus can be more easily identified. Otherwise one must face the class-imbalance problem in classification (Japkowicz, 2000). Second, the interpretability of the derived rules takes on a different nature. Algorithms like C4.5 provide rules that show how to discriminate among classes, whereas our method derives rules that characterize the positive class. In many real-world applications it is the latter type of rules that have more practical value. For example, in characterizing a computer attack on a host network it is of vital importance to detect the commonalities of event sequences corresponding to each attack. Knowing that a printer error is uncorrelated with the attack to discriminate between the negative and positive class bears little practical value.

Other studies have been performed attempting to integrate classification and association-rule mining (Ali, Manganaris, & Sikant, 1997; Bayardo, 1997; Meretakis & Wuthrich, 1999; Pijls & Potharst, 2000; Li et al., 2001; Dong et al., 1999; Liu et al., 1998). Associative classification methods like CBA (Liu et al., 1998) find all frequent and accurate rules for each class, whereas we only concentrate on the positive class. CBA selects the rule with highest confidence for each example while ensuring all examples are covered, as opposed to other methods that vote among several rules (Li et al., 2001). We also select the highest confidence rule; our assumption of having a large number of available eventsets and few positive examples obviates ensuring each example is covered by a rule. Compared to methods like CBA, our strategy of mining the positive class only enables us to keep the transaction database of positive examples in memory, which allows for efficient mining of frequent and accurate eventsets (Section 3).

Our work has the same goal as that of Weiss and Hirsh (1998) and Vilalta, Ma, and Hellerstein (2001), namely to predict the occurrence of target events along an event sequence. In both references a classification technique is employed to distinguish between time windows preceding target events and time windows not preceding target events; mining for eventsets is not done. As mentioned above, mining for frequent and accurate eventsets on the positive class provides an advantage under skewed class distributions, and enables us make efficient use of memory.

Finally, our work is also related to the area of sequential mining (Agrawal & Srikant, 1995; Mannila et al., 1995; Srikant & Agrawal, 1996; Zaki, 2001), in which traditional data mining is extended to search for frequent subsequences. Our work is different in that we first detect those events preceding a target event within a (fixed) time window. A data mining step is then applied without any further consideration of the temporal distribution of events within the time window.

9 CONCLUSIONS

This paper describes an approach to detect patterns in event sequences. In particular, we assume the existence of specific kinds of events of interest along the sequence, called *target events*. We describe a method to find patterns frequently occurring before target events using association-rule mining techniques. Patterns are then combined into a rule-based model for prediction.

Our paper investigates the conditions required to distinguish patterns from noise. In general, patterns are easier to discern as the size of the pattern increases, as the number of event types increases, and as the density of events (per time window) decreases. Results on artificial domains show the effect of these conditions on error rate. Our results also show how setting the level of minimum support high enough decreases the amount of CPU time.

Results on a real-world production network show how the size of the time window preceding target events is crucial to the effectiveness of our approach. Experiments on two different combinations of event-type and host of interest show how the false negative error rate decreases signif-

icantly as the time window increases. Our approach, however, is not universally applicable. Clearly the method proposed is contingent on sets of events frequently occurring before target events.

One may argue that the characterization of the conditions preceding target events ignores the temporal distribution of events within the specified time window. We find this in fact convenient for real applications like the production network data described in Section 7.2. The data displays some variation in the difference between the occurrence of a pattern and the occurrence of a target event (i.e., signals are noisy). In addition, a pattern may repeat under different event-type permutations. Hence, taking the temporal distribution of events deeply into account may harm accuracy performance.

Future work will consider alternative ways to formulate the classification model. A rule-based approach has the advantage of providing output amenable to interpretation. Our main goal, however, is to find highly accurate models. We believe a promising research avenue is to adapt the predictive model according to the characteristics of the sequence under analysis. The problem is related to the field of meta-learning: we want to learn to match the right bias for each different event sequence.

ACKNOWLEDGMENTS

This work was funded by IBM Research.

References

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. In Yu, P. S., & Chen, A. L. P. (Eds.), *Proc. 11th Int. Conf. Data Engineering, ICDE*, pp. 3–14. IEEE Press.

Ali, K., Manganaris, S., & Sikant, R. (1997). Partial classification using association rules. In *ACM SIGMOD Management of Data*, pp. 115–118.

Bayardo, R. (1997). Brute-force mining of high confidence classification rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pp. 123–126.

Brin, S., Motwani, R., & Silverstein, C. (1997). Beyond market baskets: generalizing association rules to correlations. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pp. 265–276.

Dong, G., Zhang, X., Wong, L., & Li, J. (1999). Caep: Classification by aggregating emerging patterns. In *Proceedings of the International Conference on Discovery Science*.

Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. In *Proceedings of the International Conference on Artificial Intelligence*, pp. 111–117.

Li, W., Han, J., & Pei, J. (2001). Cmar: Accurate and efficient classification based on multiple class-association rules. In *IEEE International Conference on Data Mining*, pp. 369–376.

Liu, B., Hsu, W., & Ma, Y. (1998). Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pp. 80–86. New York, USA.

Mannila, H., Toivonen, H., & Verkamo, A. I. (1995). Discovering frequent episodes in sequences.. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD-95)*. Montreal, Canada.

Meretakis, D., & Wuthrich, B. (1999). Classification as minig and use of labeled itemsets. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD-99)*. Philadelphia, USA.

Michalski, R. S. (1983). A theory and methodology of inductive learning. In *Machine Learning: An Artificial Intelligence Approach*, chap. 1, pp. 83–134. Tioga Publishing Co., Palo Alto, CA.

Pijls, W., & Potharst, R. (2000). Classification and target group selection based upon frequent patterns. In *Proceedings of the Twelfth Belgium-Netherlands Artificial Intelligence Conference (BNAIC 00)*, pp. 125–132.

Quinlan, J. R. (1994). *C4.5: Programs for Machine Learning*. Mogan Kaufmann.

Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In Apers, P. M. G., Bouzeghoub, M., & Gardarin, G. (Eds.), *Proceedings of the 5th International Conference Extending Database Technology, EDBT*, Vol. 1057, pp. 3–17. Springer-Verlag.

Vilalta, R., Ma, S., & Hellerstein, J. (2001). Rule induction of computer events. In *Proceedings of the 12th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management*. Springer Verlag, Lecture Notes in Computer Science.

Weiss, G., & Hirsh, H. (1998). Learning to predict rare events in event sequences. In *Knowledge Discovery and Data Mining*, pp. 359–363.

White, A., & Liu, W. (1994). Bias in information-based measures in decision tree induction. *Machine Learning*, *15*, 321–329.

Zaki, M. J. (2001). Sequence mining in categorical domains. In Sun, R., & Giles, G. L. (Eds.), *Sequence Learning: Paradigms, Algorithms, and Applications*, pp. 162–187. Springer Verlag, Lecture Notes in Computer Science.