Rule Induction of Computer Events

Ricardo Vilalta, Sheng Ma, and Joseph Hellerstein

IBM T.J. Watson Research Center
30 Saw Mill River Rd., Hawthorne N.Y., 10592 USA

Monitoring systems are able to capture thousands of events from a computer network. Some of those events may be particularly informative to ensure the correct operation of an application. We assume the user is interested in a specific class of events, called *target events* (e.g., communication link is down). We propose a system that generates a set of rules correlating each target event with events occurring previous to the target event within a specified time window interval. Such rules can be extremely helpful in elucidating the origin of target events. We conduct experiments to assess the accuracy of the induced rules for different types of target events in a real-world network environment. Our results show the accuracy of the induced rules generally above 80% when the time window interval is at least 20 minutes wide. Such results give strong empirical support to the validity of our approach.

Keywords: machine learning, rule induction, events

1 Introduction

Monitoring systems are able to capture an assortment of different events from a network environment. A single event may correspond to one host, e.g., "cpu utilization is above a critical threshold", or the network, e.g., "communication link is down". Monitoring systems can capture thousands of events in a short time period; applying data analysis techniques (e.g., machine-learning, data-mining) on those events may reveal useful patterns characterizing a network problem. Data analysis techniques have proved useful in the area of network fault management.

In this paper we consider the following scenario. A computer network is under continuous monitoring; a user is interested in identifying what triggers a specific kind of events, which we refer to as *target events*. The user would like to know what events correlate to each target event within a fixed time window. We describe a data-mining technique that takes as input all events gathered by the monitoring system, and outputs a list of empirical rules, where each rule shows a form of correlation to a target event, for example, within 5 minutes previous to the occurrence of a target event, event A often occurs 3 times and event B occurs 2 times. Such rules can be extremely helpful in elucidating the origin of target events. In a practical scenario, the rules can be used off-line for problem determination, or on-line for automatic problem detection.

Our study reports on data generated by monitoring systems active during one consecutive month on a computer network having 750 hosts. Our analysis concentrates on target events labeled as *critical* by domain experts. One month of continuous monitoring generated over 26,000 events, with 165 different types of events. Such high volume of records defies any form of manual analysis, rendering the use of algorithms for pattern analysis indispensable.

Our approach to inducing rules combines the strength of association-rule mining [AMS+96] with a systematic search for rules under strong pruning techniques [Rym93, Web93, Web95, Web00]. Our analysis is similar to the general framework for rule induction proposed by [Web00], but with important modifications (Section 4). We show how specific settings for the modified framework provide a principled approach to the construction of accurate rules correlating computer events.

Our experiments show the effects of varying several parameters on the quality of the induced rules. We study the impact of varying the window size and the number of rules. In addition we analyze the feasibility of adding a safe-window to allow for corrective actions to take place before each target event actually

O. Festor and A. Pras (Eds.) : 12th International Worshop on Distributed Systems: Operations and Management
DSOM'2001 Nancy France, October 15-17, 2001.

Tab. 1: A description of event data for a particular domain of application.

Features	Examples of Possible Values	No. of Values or Range of Values
Time	2.27.2001.900, 3.7.2001.856	February 25 2001 9:00 PM - March 26 2001 12:00 PM
Event Type	Interface-Down, high-cpu-usage, high-disk-usage, disk-error	165
Host Name	ibm1.xxx.com, ibm2.xxx.com	750
Severity	harmless, minor, warning, critical, and fatal	5

occurs. Our results show accuracy levels above 80% when the time window interval is at least 20 minutes wide. Such results give strong empirical support to the validity of our approach.

This paper is organized as follows. Section 2 provides background information on the nature of the event data. Section 3 describes how the event data is transformed into a classification problem. Section 4 details on the methodology for rule induction. Section 5 shows our experimental results. Finally, Section 6 gives a summary and conclusions.

2 Background Information

We assume a network environment under continuous monitoring from which multiple events are generated. We define an event, $\tilde{\mathbf{X}} = (F_1, F_2, \cdots, F_n)$, as a feature vector, where each F_i describes a condition under which the event took place. Possible features are the time at which the event occurred (F_{time}), the host generating the event (F_{host}), the type of event(F_{type}), the severity level (F_{severity}), etc. Consider the following example:

$$(1.1.2000.900,\ \text{cpubusy},\ \text{ibm.xxx.com},\ \text{critical}) \tag{1}$$

The event above can be interpreted as occurring on January 1, 2000 at 9:00 AM on host ibm.xxx.com when the level of cpu utilization exceeded a maximum threshold; a domain expert qualified such event as critical. Table 1 shows the set of features considered for our domain of application, examples of possible values, and the number of different values for each feature.

The set of all m events generated by the network on a period of time defines an event dataset $T_{\text{events}} : \{\tilde{\mathbf{X}}_\mathbf{i}\}_{i=1}^{m}$. In our study we examine closely a subset of events in T_{events} that share the same event type of interest to the user. We refer to that subset of events as $T_{\text{target}} \subset T_{\text{events}}$. If the user specifies a target event type V, then we characterize T_{target} as:

$$T_{\text{target}} = \{\tilde{\mathbf{X}}_\mathbf{i} \in T_{\text{events}} | F_{\text{type}} = V\} \tag{2}$$

where V could be any possible event type (e.g., Interface-Down).

3 Transforming The Event Dataset

Our goal is to induce a set of rules that show some form of correlation with the set of target events T_{target}. We want to assess the degree of correlation between sets of events for the purpose of prediction. The idea is to detect how many times each different type of event occurred before a target event, and to induce from those counts correlation rules that can provide insight into the cause of target events. Our general approach divides in two steps: feature extraction and classification.

1. **Feature Extraction**. This process transforms data set T_{events} into a classification problem by creating a positive example for every event in T_{target}, and a negative example for some of the events in the complement set $\overline{T_{\text{target}}}$.

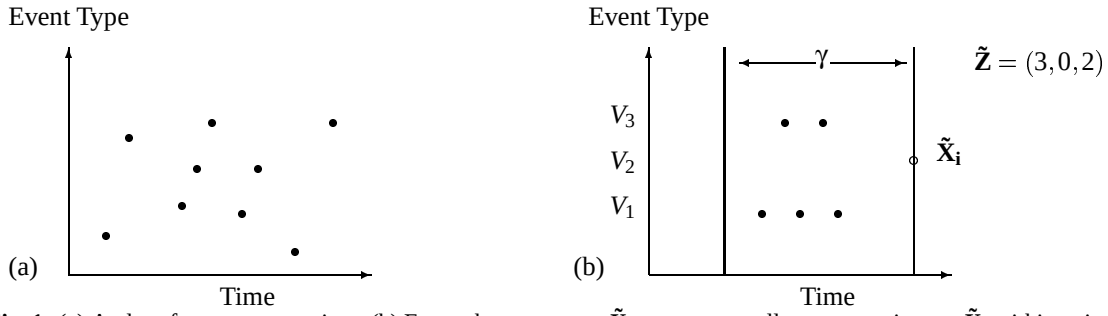

Fig. 1: (a) A plot of event type vs time. (b) For each target event $\tilde{\mathbf{X}}_\mathbf{i}$, we capture all events previous to $\tilde{\mathbf{X}}_\mathbf{i}$ within a time window of size γ. The result is a count vector indicating the number of instances of each event type.

2. **Classification**. This process uses the transformed dataset to generate a set of rules for classification. Each induced rule can potentially be used to classify an event in one of two categories: target or non-target.

Each event $\tilde{\mathbf{X}}_\mathbf{i} \in T_{\text{events}}$ is characterized by a set of features, two of which are the temporal feature F_{time}, and the type of event F_{type}. A plot of F_{type} vs. F_{time} shows how different event types distribute through time, as exemplified in Figure 1(a). The problem we address is how to identify the conditions that serve to differentiate between different types of events (e.g., target and non-target) through time.

3.1 Feature Extraction: Characterizing Target and Non-Target Events

Our characterization of target events is described in Figure 2. For each target event, we create a time window of size γ right before it. We then count the number of times each event type occurs within such time window. In Figure 1(b), for example, target event $\tilde{\mathbf{X}}_\mathbf{i}$ (of type V_2), is preceded by three occurrences of events of type V_1, and two occurrences of events of type V_3. Assuming only three possible types of events, the result is a count vector $\tilde{\mathbf{Z}} = (3,0,2)$. Running the process described above over all target events produces a set of count vectors $Z_{\text{target}} = \{\tilde{\mathbf{Z}}_\mathbf{i}\}$ of size equal to the number of target events (i.e., equal to $|T\text{target}|$). The set Z_{target} captures the conditions under which a target event occurs. A similar process is conducted to characterize the conditions under which a target event *does not* occur, resulting in a set of count vectors $Z_{\text{non-target}}$. The process is very similar to Figure 2 and can be described as follows. For every two consecutive target events in T_{target}, $\tilde{\mathbf{X}}_\mathbf{i}$ and $\tilde{\mathbf{X}}_\mathbf{j}$, we look for an event in the complement set $\overline{T_{\text{target}}}$ that stands closest to the middle point between $\tilde{\mathbf{X}}_\mathbf{i}$ and $\tilde{\mathbf{X}}_\mathbf{j}$. That is, we look for an event $\tilde{\mathbf{X}}_\mathbf{k}$ with a time of occurrence close to $\frac{F_{\text{time}}(\tilde{\mathbf{X}}_\mathbf{i})+F_{\text{time}}(\tilde{\mathbf{X}}_\mathbf{j})}{2}$. Once $\tilde{\mathbf{X}}_\mathbf{k}$ is located we proceed similarly as before by generating a window of size γ previous to the occurrence of $\tilde{\mathbf{X}}_\mathbf{k}$, and by generating a count vector of event types. The rationale behind the mechanism described above is that non-target events are characterized by a set of conditions lying as far as possible from target events to facilitate the search for rules discriminating between both kinds of events.

The two sets Z_{target} and $Z_{\text{non-target}}$ are forms of characterizing target and non-target events. This study is based on the hypothesis that both sets contain enough information to differentiate the situations in which the two types of events take place. We assume the number of event types previous to an event can be used to form correlation rules able to discriminate among event types. Our experimental results provide evidence supporting this hypothesis (Section 5).

3.2 Formulating a Classification Problem

We cast our problem as a classification problem in a straightforward manner. We generate a new dataset T_{train} by labeling each count vector of a target event as positive and conversely by labeling each count vector of a non-target event as negative. T_{train} is made of pairs $\{(\tilde{\mathbf{Z}}_\mathbf{i}, c_i)\}$, where $\tilde{\mathbf{Z}}_\mathbf{i}$ is a count vector and c_i is the class to which the count vector belongs, $c_i \in \{+,-\}$.

T_{train} can serve as input to a classification problem [WK90]. In this problem, T_{train} is said to be made of examples, where each example is assigned a class. The problem is to learn how to assign the correct class to an example not previously seen before. In our case, we wish to learn how to classify a new event as either a target or non-target event. One approach to classification is to generate a set of rules that show

Algorithm 1: Characterizing Target Events
Input: Event Set T_{events}, Window Size γ, Event Type V
Output: Set of count vectors $Z_{\text{target}} = \{\tilde{\mathbf{Z}}_\mathbf{i}\}$
CHARACTERIZE(T_{events},γ,V)
(1) Generate set of target events T_{target} according to V.
(2) **foreach** $(\tilde{\mathbf{X}}_\mathbf{i} \in T_{\text{target}})$
(3) Let T_{window} be the set of events within window γ
(4) previous to the ocurrence of $\tilde{\mathbf{X}}_\mathbf{i}$
(5) Initialize $\tilde{\mathbf{Z}}_\mathbf{i} \leftarrow 0$
(6) **foreach** $(\tilde{\mathbf{X}}_\mathbf{j} \in T_{\text{window}})$
(7) Let V_k be the event type of $\tilde{\mathbf{X}}_\mathbf{j}$
(8) Update count vector: z_k^i++
(9) **end for**
(10) **end for**
(11) **return** $\{\tilde{\mathbf{Z}}_\mathbf{i}\}$

Fig. 2: A characterization of target events.

strong correlation to the class. Another approach is to generate a full classification model such as a decision tree, a neural network, or a support vector machine. We choose to generate a set of rules rather than a full classification model because our interest lies primarily in helping the user elucidate the conditions preceding target events through a mechanism that is fast (in case corrective actions need to take place) and able to produce output amenable to interpretation. We describe our own approach to the rule-generation process next.

4 Generating Rules For Target Events

We explain how to induce rules of the form $A \rightarrow \{+,-\}$, where the antecedent A is a conjunction of feature values that implies either a target event $(+)$ or a non-target event $(-)$. As an example, let's assume three event types $\{V_1,V_2,V_3\}$; An induced rule could be represented as follows:

$$z_1 \in [1,5] \text{ and } z_2 \in [2,4] \rightarrow + \tag{3}$$

where z_j stands for the number of occurrences of event type V_j. The rules states that within time window of size γ of an event $\tilde{\mathbf{X}}_\mathbf{i}$, if event type V_1 occurs between one and 5 times, and event type V_2 occurs between 2 and 4 times, then $\tilde{\mathbf{X}}_\mathbf{i}$ is a target event. Each rule antecedent is then a logical expression formed by the conjunction of event-type intervals. We expect the user to specify the number K of rules output by the system. We will show how K has a direct bearing on the amount of pruning available over the the search space: the smaller K the stronger the amount of pruning available.

Previous to the rule-generation process, the range of counts z_j for each event type V_j in T_{train} needs to be divided into a set of meaningful intervals. We follow a discretization method that minimizes the entropy between the interval and the class [Cat91]. The method looks for intervals that facilitate discriminating target events from non-target events. Each resulting interval can be considered as a Boolean variable (either an event falls into the interval or not). As an example, consider again three possible event types $\{V_1,V_2,V_3\}$. If we create two intervals for each event type V_j, I_{j1},I_{j2}, then we transform each count vector $\tilde{\mathbf{Z}}_\mathbf{i}$ in T_{train} into a Boolean vector $B_i = (b^i_{11},b^i_{12},b^i_{21},b^i_{22},b^i_{31},b^i_{32})$. For example, boolean variable $b^i_{11} = 1$ (i.e., is true) if the number of times event-type V_1 occurred within a time window is within interval I_{11}.

In general, the new dataset $T'_{\text{train}} = \{(B_i,c_i)\}$ indicates the interval into which each event-type count falls for the set of events occurring before a target event ($c_i = +$) or a non-target event ($c_i = -$).

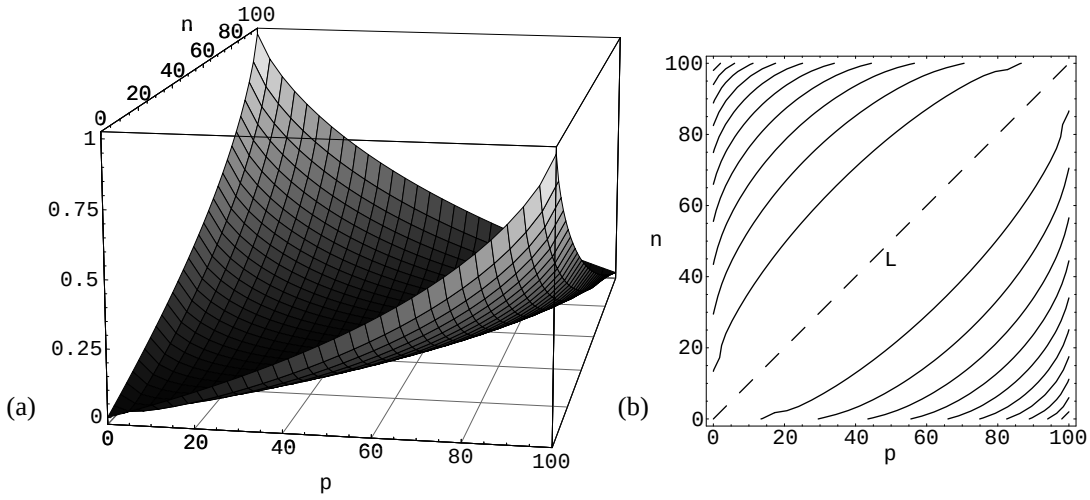

Fig. 3: (a) Information Gain as a function of the possible coverage (number of positive and negative events) of a monomial. (b) A contour plot of (a) showing isometric lines over the coverage plane.

4.1 Evaluating Monomials

We first describe the criterion used to compare monomials (i.e., the metric used over the search space). A monomial M is said to *cover* an event if the conjunction is true for that event. As an example, the monomial in the antecedent of the rule in equation 3 covers an event if it is true that event type V_1 occurs between one and five times, and event type V_2 occurs between two and four times. Events that are *not* covered by M are covered by the complement $\overline{M}$.

A monomial M shows perfect correlation with the class if it covers all target events, and the complement $\overline{M}$ covers all non-target events. Such monomial gets a maximum score. But in the general case, a monomial covers both positive and negative events, and a metric is necessary to assign a score. We use a common metric called information-gain IG [Qui94] explained next.

Intuitively a monomial M is good if it is able to reduce the amount of *chaos* derived by a mixture of target and non-target events in T'_{train}. We say chaos is minimized if the proportion of target events existing in T'_{train} increases within M and decreases within $\overline{M}$. Information gain is defined as the reduction of entropy (i.e., chaos) induced by M. Let p be the proportion of target events in T'_{train}. Let p_m be the proportion of target events covered by monomial M, and let $\overline{p_m}$ be the proportion of target events covered by the complement $\overline{M}$. Information gain is defined as follows:

$$IG(M) = H(p) - (W_1 H(p_m) + W_2 H(\overline{p_m})) \tag{4}$$

where W_1 and W_2 are the proportion of events covered by M and the complement $\overline{M}$ respectively. H measures the degree of entropy or chaos over probability q as follows:

$$H(q) = -[q\log_2 q + (1-q)\log_2(1-q)] \tag{5}$$

Information gain measures the benefit that comes from partitioning the example space using monomial M; it quantifies the degree of class uniformity of the sets covered by M and $\overline{M}$ and compares it to the degree of class uniformity over the whole training set T'_{train}. The result is a measure of the improvement produced by M and $\overline{M}$ in grouping together examples of the same class.

Figure 3(a) plots information gain over a plane that counts the number of positive and negative events covered by monomial M [VO00], also known as the *coverage plane*. The maximum scores are attained at the extreme points where M covers all positive events and no negative events and vice versa. Figure 3(b) shows contour lines obtained by projecting information gain over the coverage plane. The function grows monotonically from axis-line L to the extreme points. Other metrics other than Information Gain have been used in the machine-learning literature [WL94], such as Gini, G statistic, Laplace, χ^2. All of them show a similar monotonic shape over the coverage plane [VO00]. We choose Information Gain because it has

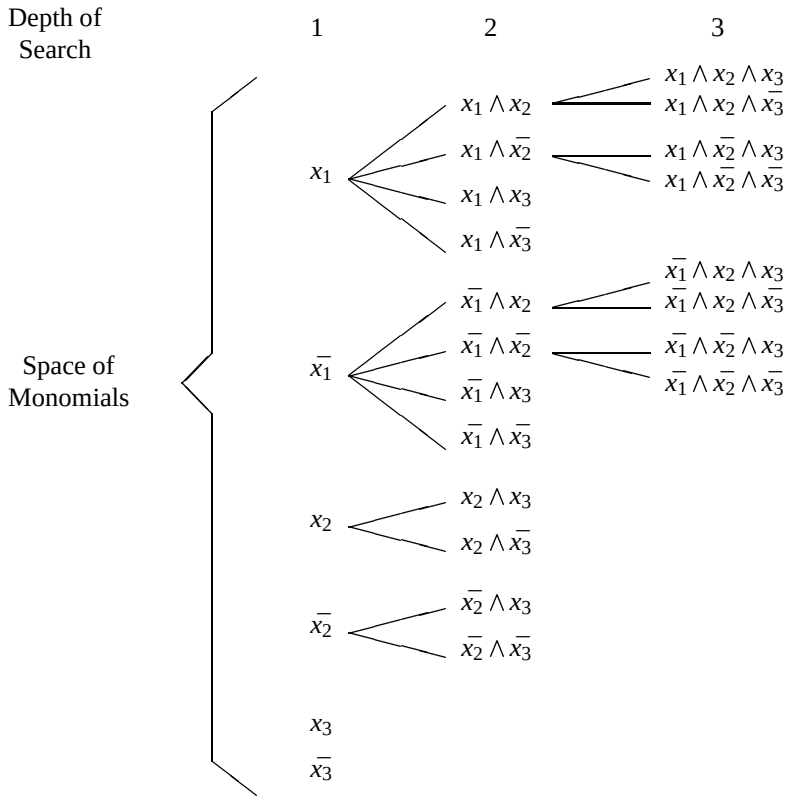

Fig. 4: A systematic search in the space of monomials.

proved effective in previous studies [Qui94], but as any form of induction one must be aware that any fixed form of bias is prone to perform well in some domains and poor in others [Wol96].

4.2 Searching the Space of Monomials

Our problem formulation leads us to a search for the best K rules over the space of all possible conjunctions of Boolean features or *monomials*. In our case each Boolean feature is an interval over the range of event-type counts. An example of the space of all possible monomials on three Boolean features is illustrated in Figure 4.

Our approach consists in carrying out a search over the space of monomials while keeping track of the best K states. We then output those K states as our set of induced rules. Specifically, we conduct a beam search over the space of all monomials. A beam search explores states of size d before exploring states of size $d+1$. The search is not exhaustive but rather keeps only the best α candidates at level d to generate candidates at level $d+1$. In our case, the mechanism works by adding one literal (i.e., Boolean feature or its complement) at a time to the α best previously retained monomials in the beam (the beam starts with the best α single literals). Adding literals extends the depth of the search; retaining the best α monomials limits the width of the search. The process continues while keeping track of the best K global monomials. Figure 5 describes the logic behind our search mechanism. A distinction must be made between the role of parameters α and K. The former keeps track of the best local monomials at each level of search, while the latter keeps track of the best monomials across all levels of search.

To avoid exploring all possible states, the size of the search space can be constrained with two major operations: a systematic search, and a pruning mechanism. We explain each operation next.

Algorithm 2: Search Mechanism for New Monomials
Input: Training set T'_{train}, beam width α
Output: Best K monomials
SEARCH(T'_{train})
(1) Let L_{literals} be the list of literals
(2) (i.e., boolean features and their complements)
(3) $L_{\text{beam}} \leftarrow$ best α literals in L_{literals}
(4) **while** (**true**)
(5) $L_{\text{new}} \leftarrow$ Systematically form the conjunction
(6) of every $M_i \in L_{\text{beam}}$ with every $M_j \in L_{\text{literals}}$
(7) Apply pruning into L_{new}
(8) **if** $L_{\text{new}} = \emptyset$
(9) **return** K best global monomials
(10) $L_{\text{beam}} \leftarrow$ best α combinations in L_{new}
(11) **end while**

Fig. 5: The search mechanism outputs the K best logical monomials.

A Systematic Search

A systematic search is necessary to avoid redundant monomials [Rym93]. Each monomial conjoins several boolean features (or their complements). Because conjunction is commutative, the search space is defined by avoiding any state that is identical to an existing state except for the order in which features appear (Figure 4).

The number of all possible new monomials grows exponentially with the depth of the search, but is limited by the number of combinations that remain still feasible as deeper subspaces are explored. In general, the number of possible monomials at depth d is $2^d \times \binom{n}{d}$, where n is the number of features.

The Pruning Mechanism

We are now in a position to explain our two pruning techniques: information-gain and minimum support.

Information-gain pruning. Let (c_p, c_n) be the number of positive and negative events covered by monomial M, expressed as a pair of coordinates in the coverage plane. Let $(M_1, M_2, \cdots, M_k)$ be the list of the best K global monomials found up to this point in the search space. We test to see if M can improve over the worst of the best K monomials, M_k. Now, the best M can do is to conjoin with other literals until it covers positive events only (cp, o), or negative events only $(0, c_n)$. Let's call M_{optim1} and M_{optim2} the two optimal cases respectively. If the information gain scored by M_k is better than M_{optim1} and M_{optim1} then M can be safely pruned away because it can never be part of the list of K best monomials. This can be derived from the monotonic nature of information-gain over the coverage plane (Figure 3(b)).

In information-gain pruning, the value of K directly affects the number of monomials amenable to elimination. In the case where K is too large, the search becomes longer. A small value of K increases the efficiency of the search process but at the expense of having less rules output by the system.

Minimum-support pruning. Our second technique simply eliminates a monomial M when the number of events covered by M falls below a minimum threshold Π as specified by the user. Let (c_p, c_n) be the number of positive and negative events covered by monomial M. We eliminate M if $c_p + c_n < \Pi$ where Π is a constant. This technique borrows from the area of association-rule mining in data mining [AMS+96]. It enables us to focus the search on those rules that have a minimum degree of support.

5 Experiments

Our experiments test the quality of the rules induced from a particular domain of application. The computer network under analysis comprises 750 hosts (Table 1). Monitoring systems active during 1 month generated

over 26,000 events, with 165 different types of events. We focus our analysis on 4 target events labeled as *critical* by domain experts. The target events are listed as follows:

- *Node-Down* indicates that a server or router cannot be reached.
- *CRT URL Time-Out* indicates a URL page unaccessible by a probing mechanism.
- *Email server* indicates the server is been started.
- *EPP Event* indicates that end-to-end response time generated by a probing mechanism is above a critical threshold.

For each target event and time window, the first step creates a training set for classification as described in Section 3. The resulting training set is then used for rule-generation (Section 4).

5.1 Methodology

We report on the predictive accuracy of the best rule and the predictive accuracy averaged over all K rules. Assessing the quality of a rule is not trivial and requires more than one indicator. For example, the approach adopted by most association-rule mining algorithms [AMS$^+$96] is to evaluate a rule based on its support (proportion of examples covered by the rule), and confidence (accuracy of the rule on the examples covered by the rule exclusively). This approach ignores the accuracy of the rule over all examples (examples covered and not covered by the rule) and may carry little information on the correlation between the rule and the class [BMS99]. For a correct assessment we use three different types of predictive accuracy:

- The overall predictive accuracy of the rule (Acc) defined as the fraction of examples correctly classified by the rule.
- The accuracy of a rule over the examples covered by the rule (Acc+) .
- The accuracy of a rule over the examples *not* covered by the rule (Acc-).

In the following we will take the accuracy over all examples (Acc) as the main performance indicator and use the rest to complement our analysis.

By default we set the number of rules $K = 5$. The size of the time window capturing all events previous to a target event will take one of the following values: $\gamma \in \{1\text{min.}, 5\text{min.}, 10\text{min.}, 20\text{min.}\}$. Predictive accuracy is estimated using stratified 10-fold cross-validation [Koh95]. For each run, 10% of the examples are used for testing, 60% for training, and 20% for validation to avoid statistical errors from multiple comparisons [JC99]. Runs were performed on a RISC/6000 IBM model 7043-140.

5.2 Results on Predictive Accuracy

Our experimental results for the predictive accuracy of the best rule are depicted in Table 2. In addition to reporting on the 3 types of accuracy described above we also include the rule support.

Table 2 shows an increase in accuracy as the time window increases from 1 minute to 20 minutes. For Node-Down accuracy reaches the 99% level with a support of around 50%. CRT URL Time-Out starts with low accuracy, but it increases up to the 87.5% level as the time window grows; notice, however, that the accuracy in the examples covered by the rule (Acc+) is low (60.4%). Email-Server may seem at first glance displaying high accuracy but the low accuracy for the rule complement set (Acc-) points to the existence of false negatives. EPP starts with 61.6% accuracy, ending with 80% accuracy at a time window of 20 minutes.

In general, the accuracy for the best rule in a time window of 20 minutes is above 80%; we take this as encouraging evidence that counting event types before the occurrence of a target event results in relevant features for classification.

The fact that longer time windows result in better accuracy may seem counterintuitive. Normally, any form of time-series analysis gives less weight to events happening farther away in the past. But notice that

Tab. 2: Accuracy of best rule ($\times 100\%$) using different time windows.

Event Type	1 min.				5 min.			
	Acc	Acc(+)	Acc(-)	Support	**Acc**	Acc(+)	Acc(-)	Support
Node-Down	**82.4**	52.4	89.1	36.4	**97.2**	95.3	87.6	49.3
CRT URL Time-Out	**59.7**	60.2	52.6	92.4	**65.2**	62.7	71.6	91.7
Email Sever	**96.0**	92.0	10.0	92.0	**97.2**	88.1	10.0	90.9
EPP Event	**61.6**	63.3	31.2	85.8	**73.2**	52.2	67.2	58.3
	10 min.				20 min.			
	Acc	Acc(+)	Acc(-)	Support	**Acc**	Acc(+)	Acc(-)	Support
Node-Down	**96.5**	86.6	92.9	34.2	**99.0**	94.5	92.5	49.7
CRT URL Time-Out	**60.9**	55.4	68.9	74.1	**87.5**	60.4	44.9	58.3
Email Server	**97.2**	88.1	10.0	90.9	**94.5**	98.1	0.0	96.3
EPP Event	**76.6**	53.3	65.6	47.7	**80.5**	62.2	85.2	54.4

Tab. 3: Average accuracy ($\times 100\%$) using different values for K ($\gamma = 20$ min.)

Event Type	Average Accuracy			
	$K = 5$	$K = 10$	$K = 15$	$K = 20$
Node-Down	98.6	98.4	97.7	97.5
CRT URL Time-Out	78.8	78.6	78.6	78.8
Email Sever	94.5	94.5	88.3	87.6
EPP Event	76.3	78.3	79.1	79.5

the size of the time window plays a role in capturing more information about what triggers a target event. Our results simply indicate that those events showing some form of correlation to a target event are expected to occur 20 minutes or more before the occurrence of the target event; events too close to the target event represent only a subset of those events relevant for prediction. In other words, longer time windows result in improved accuracy because there is more information contained on each window about the set of conditions preceding target events.

5.3 Results Varying the Number of Rules

Our next experiments measure the average accuracy of all K rules output by the system. Results are depicted in Table 3. We report on the first type of accuracy only (Acc)[†]. We keep the size of the time window fixed at $\gamma = 20$ min. and vary the number of rules as follows: $K \in \{5, 10, 15, 20\}$.

Table 3 shows little variation in the average accuracy of the best rules as K increases. Node-Down looses about one percent point. CRT URL Time-Out remains at about the same level (78.6%-78.8%), while Email-Server looses about 7%. EPP shows, surprisingly, an increase in average accuracy of about 3%. This indicates that the beam-search strategy described in Section 4 may miss relevant monomials. But the advantage obtained with a beam-search is a means to overcome the exponential growth of the search space as the depth of the search increases (Figure 4).

We conclude that the quality of the set of rules produced by our system shows little variation when K is kept in the range $[1, 20]$. For most practical applications, this range of values seems appropriate to provide evidence about the conditions preceding a target event.

5.4 Results After Creating a Safe-Window

Our last experiments test the following idea. If the best rule output by our system were to be used for prediction of target events, could we create a safe-window between the prediction and the occurrence of a target event to allow corrective actions to take place? To answer this question we modified our algorithm for transforming the event set into a classification problem (Section 3) by adding a new parameter β. This parameter indicates the size of a time window placed right before a target event in which no event counts are made (to allow for corrective actions). Let t_i be the time at which target event $\tilde{\mathbf{X}}_\mathbf{i}$ occurs ($t_i = F_{\text{time}}(\tilde{\mathbf{X}}_\mathbf{i})$). The modified algorithm counts all event types occurring before $\tilde{\mathbf{X}}_\mathbf{i}$, starting at time $t_i - \gamma$ and ending at time

[†] The other two types of accuracy, Acc(+) and Acc(-), are meaningful when evaluating a single rule.

Tab. 4: Accuracy of best rule ($\times 100\%$) using different safe windows ($\gamma = 20$ min.)

Event Type	$\beta = 1$ min.			$\beta = 5$ min.			$\beta = 10$ min.		
	Acc	Acc(+)	Acc(-)	**Acc**	Acc(+)	Acc(-)	**Acc**	Acc(+)	Acc(-)
Node-Down	**85.1**	18.8	89.2	**75.0**	42.3	61.3	**67.2**	50.7	49.7
CRT URL Time-Out	**80.0**	67.9	39.5	**79.5**	75.5	22.2	**90.4**	85.5	19.5
Email Sever	**94.5**	98.1	00.0	**98.1**	88.1	10.0	**98.1**	98.1	00.0
EPP Event	**83.0**	66.3	65.6	**78.8**	41.5	99.2	**73.6**	44.8	72.0

$t_i - \beta$. The result is a safe-window of size β before $\tilde{\mathbf{X}}_\mathbf{i}$. The goal of our experiments is to measure the degree of accuracy loss stemming from the introduction of this safe-window.

Our results are shown in Table 4. We keep the size of the time window fixed at $\gamma = 20$ minutes and vary the safe window $\beta \in \{1\text{min.}, 5\text{min.}, 10\text{min.}\}$. For Node-Down, a safe-window of $\beta = 1$ minute lowers the accuracy from 99.0% to 85.1%; a further decrease is observed when $\beta = 10$ minutes. The same effect is observed on EPP, although the accuracy when $\beta = 1$ minute is higher that the case when no safe-window exists (Table 1). CRT URL Time-Out shows a similar decrease in accuracy but a sudden increase happens when $\beta = 10$ minutes. This may simply indicate that the relevant features correlating to the target event are located between 10-20 minutes before the target event; the safe-window helps to isolate those features. A similar decrease and increase in accuracy is observed on Email-Server.

Except for the case in which a safe-window becomes useful at isolating relevant features for the prediction of a target event, one should expect a decrease in accuracy as the size of β grows higher. In some cases, however, it is clear that a safe-window can be implemented without loss of accuracy (and even with gains in accuracy). Thus, the implementation of a safe-window must be done according to the target event under analysis.

6 Summary and Conclusions

We propose a system that generates a set of rules correlating a (user-defined) target event with those events occurring before the target event within a specified time window interval. Our approach divides in two steps: 1) transform the event dataset into a classification problem by labeling vectors characterizing target events as positive and non-target events as negative (Section 3); 2) generate a set of rules by conducting a beam-search over the space of rule antecedents or monomials using strong pruning techniques (Section 4). We report on a series of experiments designed to assess the quality of the rules induced by our system (Section 5). In terms of predictive accuracy, the best rule is generally above 80% when the time window is 20 minutes wide. Our results show little variation in terms of predictive accuracy when the number of rules varies in the range $[5-20]$. In addition, we show the feasibility of adding a safe-window before the occurrence of a target event; this allows for corrective actions to take place. A safe-window is expected to generate a loss in predictive accuracy unless it is able to isolate those features that are relevant to the target event.

Our approach aims at generating a set of rules to facilitate the understanding of the conditions preceding the occurrence of a target event. The induced set of rules *do not* constitute a model for prediction. Even though our experimental results show cases where predictive accuracy for the best single rule can be as high as 99% (Section 5), a single rule can hardly qualify as a full classification model. One line of future research is to take the rules produced by our current system and use them to generate a full classification model. A possibility is to select the best induced rule and place it at each node of a decision tree [VBR97]. Another possibility is to construct a rule-based system from our set of induced rules [BHM98].

Our characterization of the conditions preceding a target event (Section 3) looks at the time and type of the events occurring before the target event. We ignore other features, such as the host generating the event, that may help to further increase the quality of the rules due to the improved granularity of the data. Assume the characterization of an event using event-type and host. For large computer networks, the resulting number of possible (event-type, host) pairs may be unmanageable. Future work will explore possible mechanisms for characterizing events using multiple features.

References

[AMS+96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pp. 307–328. AAAI Press, Menlo Park, CA., 1996.

[BHM98] Liu Bing, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Proceeding of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 80–96, New York, NY, 1998. AAAI Press.

[BMS99] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Data Mining and Knowledge Discovery*, volume 2, pp. 39–68, 1999.

[Cat91] J. Catlett. On changing continuous attributes into ordered discrete attributes. In *European Workshop on Machine Learning*, pp. 164–178. Springer-Verlag, 1991.

[JC99] David Jensen and Paul Cohen. Multiple comparisons in induction algorithms. In *Machine Learning (in press)*. Boston, MA: Kluwer, 1999.

[Koh95] R. Kohavi. A study of cross validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1137–1143. Morgan Kaufmann, 1995.

[Qui94] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc., 1994.

[Rym93] R. Rymon. An SE-tree based characterization of the induction problem. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 268–275. San Francisco: Morgan Kaufmann, 1993.

[VBR97] R. Vilalta, G. Blix, and L. A. Rendell. Global data analysis and the fragmentation problem in decision tree induction. In *9th European Conference on Machine Learning*, pp. 312–326. Lecture Notes in Artificial Intelligence, Vol. XXX, Springer-Verlag, Heidelberg, Available: http://www.research.ibm.com/people/v/vilalta, 1997.

[VO00] R. Vilalta and D. Oblinger. A quantification of distance-bias between evaluation metrics in classification. In *Proceedings of the 17th International Conference on Machine Learning*, pp. 1087–1094. Morgan Kaufman, 2000.

[Web93] G. I. Webb. Systematic search for categorical attribute-value data-driven machine learning. In N. Foo and C. Rowles, editors, *Proceedings of the Sixth Australian Joint Artificial Intelligence Conference*, pp. 342–347, Singapore, 1993. World Scientific.

[Web95] G. I. Webb. Opus: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, 3:431–435, 1995.

[Web00] G. I. Webb. Efficient search for association rules. In *In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 99–107, 2000.

[WK90] S. M. Weiss and C. A. Kulikowski. *Computer Systems That Learn*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[WL94] A.P. White and W.Z. Liu. Bias in information-based measures in decision tree induction. *Machine Learning*, 15:321–329, 1994.

[Wol96] D. Wolpert. The lack of a priori distinctions between learning algorithms and the existence of a priori distinctions between learning algorithms. *Neural Computation*, 8:1341–142, 1996.