A Classification Approach for Prediction of Target Events in Temporal Sequences

Carlotta Domeniconi[1], Chang-shing Perng[2], Ricardo Vilalta[2], and Sheng Ma[2]

[1] Department of Computer Science, University of California, Riverside, CA 92521 USA
{carlotta@cs.ucr.edu}
[2] IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, N.Y. 10532 USA
{perng,vilalta,shengma}@us.ibm.com

Abstract. Learning to predict significant events from sequences of data with categorical features is an important problem in many application areas. We focus on events for system management, and formulate the problem of prediction as a classification problem. We perform co-occurrence analysis of events by means of Singular Value Decomposition (SVD) of the examples constructed from the data. This process is combined with Support Vector Machine (SVM) classification, to obtain efficient and accurate predictions. We conduct an analysis of statistical properties of event data, which explains why SVM classification is suitable for such data, and perform an empirical study using real data.

1 Introduction

Many real-life scenarios involve massive sequences of data described in terms of categorical and numerical features. Learning to predict significant events from such sequences of data is an important problem useful in many application areas.

For the purpose of this study, we will focus on system management events. In a production-network, the ability of predicting specific harmful events can be applied for automatic real-time problem detection. In our scenario, a computer network is under continuous monitoring. Our data reveals that one month of monitoring of a computer network with 750 hosts can generate over 26,000 events, with 164 different types of events. Such high volume of data makes necessary the design of efficient and effective algorithms for pattern analysis.

We take a classification approach to address the problem of event data prediction. The historical sequence of data provides examples that serve as input to the learning process. Our settings allow to capture temporal information through the use of adaptive-length monitor windows. In this scenario, the main challenge consists in constructing examples that capture information that is relevant for the associated learning system. Our approach to address this issue has its foundations in the information retrieval domain.

Latent Semantic Indexing (LSI) [5] is a method for selecting informative subspaces of feature spaces. It was developed for information retrieval to reveal

semantic information from co-occurrences of terms in documents. In this paper we demostrate how this method can be used for pattern discovery through feature selection for making predictions with temporal sequences. The idea is to start with an initial rich set of features, and cluster them based on feature correlation. The finding of correlated features is carried out from the given set of data by means of SVD. We combine this process with SVM, to obtain efficient and accurate predictions. The resulting classifier, in fact, is expressed in terms of a reduced number of examples, which lie in the feature space constructed through feature selection. Thereby, predictions can be performed efficiently. Besides performing comparative studies, we also take a more theoretical perspective to motivate why SVM learning method is suitable for our problem. Following studies conducted for text data [8], we discover that the success of SVM in predicting events has its foundations on statistical properties of event data.

2 Problem Settings

We assume that a computer network is under continuous monitoring. Such monitoring process produces a sequence of events, where each event is a timestamped observation described by a fixed set of categorical and numerical features. Specifically, an event is characterized by four components: the time at which the event occurred, the type of the event, the host that generated the event, and the severity level. The severity component can assume five different levels: {*harmless*, *warning*, *minor*, *critical*, *fatal*}.

We are interested in predicting events with severity either critical or fatal, which we call *target events*. Such events are rare, and therefore their occurrence is sparse in the sequence generated by the monitoring process. Our goal is then to identify situations that lead to the occurrence of target events. Given the current position in time, by observing the monitoring history within a certain time interval (monitor window), we want to predict if a given target event will occur after a warning interval.

In our formulation of the problem, as we shall see, events will be characterized by their timestamp and type components. In this study, the host component is ignored (some hosts generate too few data). Therefore, we will denote events as two dimensional vectors $\mathbf{e}$ =(timestamp, type). We will use capital letter $\mathbf{T}$ to denote each target event, which is again a two dimensional vector. We assume that the severity level of target events is either critical or fatal.

3 Related Work

Classical time series analysis is a well studied field that involves identifying patterns (trend analysis), identifying seasonal changes, and forecasting [2]. There exist fundamental differences between time series and event data prediction that render techniques for time series analysis unappropriate in our case. A time series is, in fact, a sequence of real numbers representing measurements of a variable

taken at equal time intervals. Techniques developed for time series require numerical features, and do not support predicting specific target events within a time frame. The nature of event data is fundamentally different. Events are characterized by categorical features. Moreover, events are recorded as they occur, and show different inter-arrival times. Correlations among events are certainly local in time, and not necessarily periodic. New models that capture the temporal nature of event data are needed to properly address the prediction problem in this context.

The problem of mining sequences of events with categorical features has been studied by several researchers [10, 16]. Here the focus is on finding frequent subsequences. [16] systematically searches the sequence lattice spanned by the subsequence relation, from the most general to the most specific frequent sequences. The minimum support is a user defined parameter. Temporal information can be considered through the handling of a time window.

[10] focuses on finding all frequent episodes (from a given class of episodes), i.e., collections of events occurring frequently close to each other. Episodes are viewed as partially ordered sets of events. The width of the time window within which the episode must occur is user defined. The user also specifies the number of times an event must occur to qualify as frequent. Once episodes are detected, rules for prediction can be obtained. Clearly, the identified rules depend on the initial class of episodes initially considered, and on the user defined parameters.

Our view of target events and monitor periods is akin to the approach taken in [14], and in [15]. [14] adopts a classification approach to generate a set of rules to capture patterns correlated to classes. The authors conduct a search over the space of monomials defined over boolean vectors. To make the search systematic, pruning mechanisms are employed, which necessarily involve parameter and threshold tuning.

Similarly, [15] sets the objective of constructing predictive rules. Here, the search for prediction patterns is conducted by means of a genetic algorithm (GA), followed by a greedy procedure to screen the set of pattern candidates returned by the GA.

In contrast, in this work, we fully exploit the classification model to solve the prediction problem. We embed patterns in examples through co-occurrence analysis of events in our data; by doing so we avoid having to search in a space of possible solutions, and to conduct post-processing screening procedures. We are able to conduct the classification approach in a principled manner that has its foundations on statistical properties of event data.

4 Prediction as Classification

The overall idea of our approach is as follows. We start with an initial rich set of features which encodes information relative to each single event type; we then cluster correlated components to derive information at pattern level. The resulting feature vectors are used to train an SVM. The choice of conducting SVM classification is not merely supported by our empirical study, but finds its

Input: Sequence of events $\{\mathbf{e}\}$ (m different event types occur in $\{\mathbf{e}\}$); target event $\mathbf{T}$.
Feature Construction:
(1) Let $n = 2*$(number of occurrences of $\mathbf{T}$ in $\{\mathbf{e}\}$);
(2) Construct a training set $S = \{\mathbf{l}_i, y_i\}_{i=1}^{n}$, with $\mathbf{l}_i = (l_{i1}, l_{i2}, \ldots, l_{im})^t \in \Re^m$, for $1 \leq i \leq n$, and $y_i \in \{-1, 1\}$ ($\mathbf{l}_i$ is a column vector).
Feature Selection:
(1) Consider the matrix $D = (\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_n)$;
(2) Decompose D into the product $D = U\Sigma V^t$;
(3) Let $\bar{\sigma}$ be the average value of singular values σ_i;
(4) Set $k =$ number of singular values above $\bar{\sigma}$;
(5) Construct the projection operator $P = U_k^t$, where U_k is the matrix consisting of the k columns of U corresponding to the k largest singular values;
(6) $\forall\, \mathbf{l}_i \in S$ compute $\hat{\mathbf{l}}_i = (P\mathbf{l}_i) \in \Re^k$; we obtain the new training set: $\hat{S} = \{\hat{\mathbf{l}}_i, y_i\}_{i=1}^{n}$.
Classification:
(1) Train an SVM using the training set $\hat{S}$.
Output: $f(\mathbf{x}) = \sum_i \alpha_i y_i K(\hat{\mathbf{l}}_i, \mathbf{x}) - b$.

Fig. 1. Summary of the SVD-SVM algorithm

foundation on the structure and on the distribution properties of our data, as we will discuss in section 6. In figure 1 we summarize the whole algorithm, which we call SVD-SVM.

4.1 Feature Construction

We generate a training set of positive and negative examples for each target event $\mathbf{T}$. Specifically, we generate a positive example for each occurrence of $\mathbf{T}$. We do so by observing the monitoring history within a certain time interval, which we call *monitor window*, preceding the occurrence of $\mathbf{T}$, and preceding a *warning window*. The warning window represents the leading time useful to take actions for preventing the target event from happening during the on-line prediction process.

We proceed similarly for the construction of negative examples, monitoring the history of events within windows which are far from occurrences of the target event along the time axis. The rationale behind this choice is to try to minimize the overlapping of features between positive and negative examples. This strategy is a heuristic, and other approaches may be valid as well.

Our first step toward feature selection involves the mapping of temporal sequences (i.e., monitor windows) into vectors $\mathbf{l} \in \Re^m$, whose dimensionality m is given by the number of event types in the data. Each component l_i of $\mathbf{l}$ encodes the information relative to event e_i with respect to the monitor window under consideration. In general, the value for l_i could be a function of the timestamps of e_i. Alternatively, l_i could simply encode the number of occurrences, or just the existence of the corresponding event type e_i, within the monitor window.

4.2 Feature Selection

Let us assume we have m different event types, and we take into consideration n monitor windows to generate both positive and negative examples. Then, the feature construction step gives a training set of n m-dimensional vectors: $\mathbf{l}_1$, $\mathbf{l}_2$, ..., $\mathbf{l}_n$, with $\mathbf{l}_i \in \Re^m$ for $1 \leq i \leq n$. We denote each $\mathbf{l}_i$ as a column vector: $\mathbf{l}_i = (l_{i1}, l_{i2}, \ldots, l_{im})^t$. We can then represent the vectors in the training set as a matrix: $D = (\mathbf{l}_1\ \mathbf{l}_2\ \ldots \mathbf{l}_n)$, whose rows are indexed by the event types, and whose columns are indexed by the training vectors. We call D the *event-by-window* matrix. Its dimensions are $m \times n$.

We extract relevant information from the given training set by performing the SVD of the event-by-window matrix. The vectors are projected into the subspace spanned by the first k singular vectors of the feature space. Hence, the dimension of the feature space is reduced to k, and we can control this dimension by varying k. This process allows us to obtain a vector space in which distances reflect pattern-context information.

Using SVD, we decompose the event-by-window matrix D into the product $D = U\Sigma V^t$, where U and V are square orthogonal matrices, and Σ has the same dimensions as D, but is only non-zero on the leading diagonal. The diagonal contains the (positive) singular values in decreasing order, $\sigma_1 \geq \sigma_2 \ldots \geq \sigma_k \geq \ldots \geq 0$ (we denote with $\bar{\sigma}$ their average value). The first k columns of U span a subspace of the space of event vectors which maximizes the variation of the examples in the training set. By using this subspace as a feature space, we force the co-occurring event types to share similar directions.

The number of features is reduced; the level of grouping controls the performance, and is determined by the choice of k. In our experiments we exploit the pattern similarity information that characterizes the data by setting k equal to the number of singular values which are above the average $\bar{\sigma}$, in correspondence of the monitor window length that minimizes the error rate (see section 7).

The projection operator onto the first k dimensions is given by $P = U_k^t$, where U_k is the matrix consisting of the first k columns of U. We can then project the vectors $\mathbf{l}_i$ into the selected k dimensions by computing $\hat{\mathbf{l}}_i = (P\mathbf{l}_i) \in \Re^k$. This gives us the new k-dimensional vectors $\hat{\mathbf{l}}_i$, for $1 \leq i \leq n$. Assuming we are interested in predicting c target events, the feature selection process produces c training sets: $\{\hat{\mathbf{l}}_i, y_i\}_{i=1}^n$. We use each of these sets to train an SVM, obtaining c classifiers SVM_1, SVM_2, ..., SVM_c.

The meaning of feature selection is twofold. Correlated dimensions are explicitely embedded in the induced space; they represent relevant features for the successive classification step. Thereby, higher prediction accuracy can be achieved. Furthermore, since the dimensionality of the induced feature space is reduced, this phase makes classification (and prediction) more efficient. Of course, the SVD process itself has a cost, but it is performed only once and off-line. For on-line updates, incremental techniques have been developed for computing SVD in linear time in the number of vectors [4,6]. [9] reduces this linear dependence by using a much smaller aggregate data set to update SVD. Furthermore, new optimization approaches that specifically exploit the struc-

ture of the SVM have been introduced for scaling up the learning process [3, 12]. Techniques have been developed to also extend the SVM learning algorithm in an incremental fashion [11, 13].

5 SVMs for text classification

The rationale for using SVMs relies on the fact that event and text data share important statistical properties, that can be tied to the performance of SVMs. Here we discuss such properties for text data, and then show that similar ones hold for event data also.

SVMs have been successfully applied for text classification. In [8], a theoretical learning model that connects SVMs with the statistical properties of text classification tasks has been presented. This model explains why and when SVMs perform well for text classification. The result is an upper bound connecting the expected generalization error of an SVM with the statistical properties of a particular text classification task.

The most common representation for text classification is the bag-of-words model, in which each word of the language corresponds to an attribute. The attribute value is the number of occurrences of that word in a document. It has been shown [8] that such text classification tasks are characterized by the following properties: *High Dimensional Feature Space.* Many text classification tasks have more than 30,000 features. *Heterogenous Use of Words.* There is generally not a small set of words or a single word that sufficiently describes all documents with respect to the classification task. *High Level of Redundancy.* Most documents contain not only one, but multiple features indicating its class. *Sparse Document Vectors.* From the large feature space, only a few words occur in a single document. *Frequency Distribution of Words follows Zipf's Law.* This implies that there is a small number of words that occurs very frequently while most words occur very infrequently [17]. It is possible to connect these properties of text classification tasks with the generalization performance of an SVM [8]. In particular, the listed properties necessarily lead to large margin separation. Moreover, large margin, combined with low training error, is a sufficient condition for good generalization accuracy.

6 Statistical Properties of Event Data

Here we pose the following question: do properties similar to those discussed in section 5 hold for event data also? In order to answer it, we analyzed the frequency distributions of event types for positive and negative examples of training sets relative to different target events. For lack of space, we report the results obtained for only one target event: CRT_URL_Timeout (coded as target event 2), which indicates that a web site is unaccessible. We have obtained similar results for the other target events. The data used are real event data from a production computer network. The data shows 164 different event types, numbered from 1 to 164. Therefore, each example is a 164-dimensional feature vector, with one

component per event type. Each component encodes the existence of the corresponding event type. The training set for event type 2 contains 460 examples, with roughly the same number of positive and negative instances.

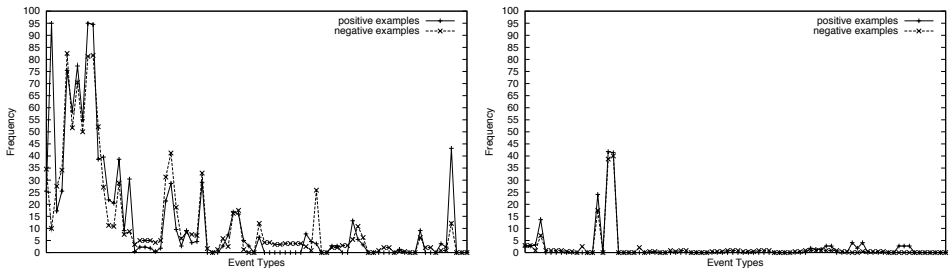

Fig. 2. Frequency of Event Types 1-82 (*left*) and 83-164 (*right*) in positive and negative examples for prediction of Target Event 2

In figure 2, we plot the frequency value of each event type for both positive and negative examples in the training set relative to target event 2. We observe that many event types have very low or zero frequency value, indicating that the 164-dimensional examples are sparse feature vectors. Furthermore, we observe a significant gap in frequency levels between positive and negative examples in correspondence of multiple event types. This shows positive evidence for redundancy of features indicating the class of examples. Also, given the frequency levels of some relevant features, it is highly likely that a significant number of positive examples does not share any of these event types, showing a heterogeneous presence of event types. In order to test the Zipf's law distribution property, in figure 3 we show the rank-frequency plot for the positive examples relative to target event 2. A similar plot was obtained for the negative examples. The plot shows the occurrence frequency versus the rank, in logarithmic scales. We observe a Zipf-like skewed behavior, very similar to the one obtained for rank-frequency plots of words [1].

We observe that the Zipf distribution does not perfectly fit the plot in figure 3. In fact, in log-log scales, the Zipf distribution gives a straight line, whereas our plot shows a top concavity. It is interesting to point out that the same "parabola" phenomenon has been observed with text data also [1]. In [8], the assumption that term frequencies obey Zipf's law is used to show that the Euclidean length of document vectors is small for text-classification tasks. This result in turns contributes to bound the expected generalization error tightly. The characteristic that feature vectors are short still holds under the Zipf-like skewed behavior observed for our data. These results provide experimental evidence that statistical properties that hold for text data are valid for event data also. As a consequence, similar theoretical results [8] derived for text data also apply for event data. This establishes the foundations for conducting SVM classification.

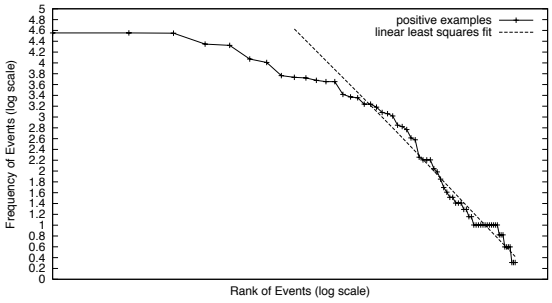

Fig. 3. Rank-frequency plot in logarithmic-logarithmic scales, positive examples for prediction of Target Event 2

7 Experiments on Real Data

In the following we compare different classification methods using real data. We compare the following approaches: (1) **SVD-SVM** classifier. We used SVM^{light} [7] with radial basis kernels to build the SVM classifier. We optimized the value of γ in $K(\mathbf{x}_i, \mathbf{x}) = e^{-\gamma||\mathbf{x}_i - \mathbf{x}||^2}$, as well as the value of C for the soft-margin classifier, via cross-validation over the training data. (2) **SVM** classifier in original feature space. Again we used SVM^{light} with radial basis kernels, and optimized the values of γ and C via cross-validation over the training data. (3) **C4.5** decision tree method in original and reduced feature spaces.

We used real event data from a production computer network. The monitoring process samples events at equal time intervals of one minute length. The characteristics of the data are as follows: the time span covered is of 30 days; the total number of events is 26,554; the number of events identified as critical is 692; the number of events identified as fatal is 16; the number of hosts is 750; the number of event types is 164.

We focus on the prediction of two critical event types: CRT_URL_Timeout (coded as type 2), which indicates that a web site is unaccessible, and OV_Node_Down (coded as type 94), which indicates that a managed node is down. This choice has been simply determined by the fact that the data contain a reasonable number of occurrences for these two critical event types. In particular, we have 220 occurrences of event type 2, and 352 occurrences of event type 94. We have generated, roughly, an equal number of positive and negative examples for each of the two event types. Specifically, we have constructed 460 examples (220 positive and 240 negative) for event type 2, and 702 examples (352 positive and 350 negative) for event type 94. We have performed 10 2-fold cross-validation to compute error rates.

Feature construction processes. We have first conducted an experiment to compare different feature construction processes: (1) **existence**: encodes the existence of each event type; (2) **count**: encodes the number of occurrences of each event type; (3) **temporal**: encodes times of occurrences of each event type.

Table 1. Prediction of Event Type 2 using SVD-SVM. Performance results for three different feature construction processes.

	ERROR(%)	STD DEV	SEL.DIM.
EXISTENCE	10.6	0.3	61
COUNT	14.7	0.4	40
TEMPORAL	15.1	0.4	66

To encode times of occurrences, we partition the monitor window into time slots. Then, for each event type, we generate a binary string with one digit for each time slot: the digit value is one if the event type occurs within the corresponding time slot; otherwise it is zero. We then translate the resulting binary sequence into a decimal number, that uniquely encodes the timestamps (time slots) of the correspondent event type. The collection of the resulting m numbers gives the feature vector.

The results shown in table 1 have been obtained applying the SVD-SVM technique for prediction of event type 2, using a 30 minutes length monitor window and a 5 minutes length warning window. The three columns show: error rate, standard deviation, and number of selected dimensions. The best performance has been obtained when existence is used as feature construction process. The same trend has been observed for target event 94 also. The fact that ignoring the temporal distribution of events within the specified monitor window gives better results, while surprising at first, may be due to patterns that repeat under different event type permutations. Furthermore, patterns may show some variability in number of occurrences of some event types. This explains the superiority of the existence approach versus the count technique. Based on these results, we have adopted the existence feature construction process, and all the results presented in the following make use of such scheme.

Monitor window length. Second, we have performed an experiment to determine the proper length of monitor windows. The objective of this experiment is to determine to which extent a target event is temporally correlated to previous events.

In figure 4 (left) we plot the average error rate as a function of the monitor window length, for target events 2 and 94. We have tested monitor windows of length that range from 5 up to 100 minutes (at intervals of 5 minutes). We have used a 5 minutes length warning window. We observe the same trend for both target events, but the extent of correlation with the monitoring history is different for the two. The error rate for event type 2 shows a significant drop for monitor windows of length up to 30 minutes. Then, the error rate slowly decreases, and reaches a minimum at 95 minutes. For event type 94 the drop occurs more rapidly within 15 minutes; then, the error keeps decreasing, and reaches a minimum at 45 minutes.

Interestingly, figure 4 (right) shows an almost mirrored behavior. Here we plot the number of selected dimensions as a function of the monitor window length. In our experiments, the number of selected dimensions corresponds to

the number of singular values above the average $\bar{\sigma}$. We observe that the number of such dimensions grows as the length of the monitor window increases, and reaches a stable point when the error rate reaches a minimum (95 minutes for event type 2, and 45 minutes for event type 94).

These results show a nice correlation between the error rate, the number of selected dimensions, and the length of monitor windows: by customizing the length of the monitor window, we are able to capture the useful information, expressed in terms of selected dimensions, i.e., patterns that predict the target, to minimize the prediction error. We emphasize that, although the choice of setting k equal to the number of singular values above the average $\bar{\sigma}$ (figure 1) is a heuristic, we are still able of estimating the intrinsic dimensionality of the data by letting k grow to the value that gives the optimal error rate.

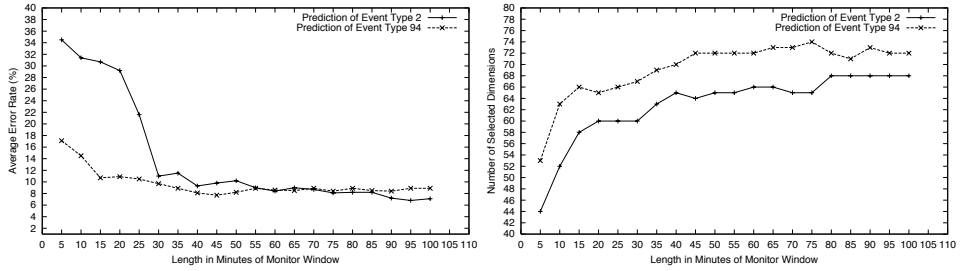

Fig. 4. Prediction of Event Types 2 and 94 using the SVD-SVM technique: (*left*) average error rate (*right*) number of selected dimensions as a function of the length of the monitor window

Off-line and on-line prediction. In table 2 we report the results obtained for the three methods we are comparing. The four columns show: error rate (with standard deviation), error rates for positive and negative examples, and number of selected dimensions. We report the results obtained with C4.5 over the whole feature space, omitting the results obtained over the reduced space, since C4.5 always performed poorly in the second case. For the monitor window length, we have used the optimal values determined in the previous experiment. SVD-SVM and SVM show similar results in both cases, with SVD-SVM selecting 68 (out of 164) dimensions in the first case and 72 in the second. C4.5 is the worst performer in both cases.

We present next the results we have obtained for prediction in an on-line setting. We consider the data preceding a certain timestamp to train a classifier; then we use such classifier for on-line prediction over the remaining time span. To simulate an on-line prediction setting, we consider sliding windows (positioned at each event) of length 95 minutes for event type 2, and of length 45 for event type 94. A warning window of 5 minutes is considered for training in both cases. Therefore, the positive examples for on-line testing are those with an occurrence

Table 2. Prediction of Event Types 2 (top 3 rows) and 94.

	ERROR	ERROR+	ERROR-	SEL.DIM.
SVD-SVM	6.8 ± 0.2	4.9	8.3	68
C4.5	7.7 ± 1.0	4.8	10.2	164
SVM	7.0 ± 0.2	4.9	8.9	164
SVD-SVM	7.7 ± 0.3	8.0	7.3	72
C4.5	9.3 ± 1.0	9.8	8.2	164
SVM	7.6 ± 0.3	8.4	6.8	164

Table 3. On-line prediction of Event Types 2 (top 3 rows) and 94.

	ERROR	ERROR+	ERROR-
SVD-SVM	8.6	12.5	8.5
C4.5	8.4	12.5	8.4
SVM	9.8	12.5	9.8
SVD-SVM	7.2	3.0	7.2
C4.5	34.9	2.4	35.1
SVM	6.6	3.0	6.7

of the target event within the fifth and sixth minute following the end of the monitor window.

Table 3 shows the results. The number of positive and negative examples used for training is 124 and 160, respectively, for event type 2, and 179, 222 for event type 94. The number of positive and negative examples used for on-line testing is 64 and 9491, respectively, for event type 2, and 165, 19655 for event type 94. Clearly, since target events are rare, the number of tested negative examples is much larger than the positives. We observe that a trivial classifier that always predicts no flaw will make a smaller number of mistakes than SVD-SVM, but it is useless since its recall will always be zero.

On target event 2, all three methods show a similar performance, with SVM being slightly worst (due to a larger number of false positives). On target event 94, SVD-SVM and SVM show a similar performance, whereas C4.5 performs poorly in this case, due to a large number of false positives. By analyzing the tree produced, we see that C4.5 has correctly chosen event type 94 as predictor at the top of the tree (in fact, we did observe that event type 94 is the most relevant predictor of itself). On the other hand, the subtree following the arc labelled 0 contains event types which are not discriminant; they cause the false positives.

8 Conclusions

We have presented a framework to fully exploit the classification model for event prediction. The accuracy achieved by SVD-SVM throughout our experiments

validate the effectiveness of selected features. We have also established the foundations for conducting SVM classification based on statistical properties of event data.

References

1. Bi, Z., Faloutsos, C., Korn, F. (2001). The "DGX" Distribution for Mining Massive, Skewed Data. *International Conference on Knowledge Discovery and Data Mining.*
2. Brockwell, P.J., Davis, R. (1996). *Introduction to Time-Series and Forecasting*, Springer-Verlag.
3. Cauwenberghs, G., Poggio, T. (2000). Incremental and Decremental Support Vector Machine Learning. *NIPS.*
4. Chandrasekharan, S., Manjunath, B.S., Wang, Y. F., Winkeler, J., Zhang, H.(1997). An eigenspace update algorithm for image analysis. *Journal of graphical models and image processing*, 321-332, 59(5).
5. Deerwester, S., Dumais, S.T., Furnas, G. W., Landauer, T.K., Harshman, R.A.(1990). Indexing by latent semantic analysis, *Journal of the American Society for Information Science*, 391-407, 41(6).
6. Degroat, R., Roberts, R.(1990). Efficient numerically stabilized rank-one eigenstructure updating. *IEEE transactions on acoustic and signal processing*, 38(2).
7. Joachims, T.(1999). Making large-scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning*, MIT-Press.
8. Joachims, T.(2000). *The maximum margin approach to learning text classifiers: Methods, theory, and algorithms.* Doctoral dissertation, Universität Dortmund, Informatik, LS VIII.
9. Kanth, K.V.R., Agrawal, D., Singh, A.(1998). Dimensionality Reduction for Similarity Searching in Dynamic Databases. *ACM SIGMOD.*
10. Mannila, H., Toivonen, H., Verkamo, A.I.(1995). Discovering frequent episodes in sequences. *International Conference on Knowledge Discovery and Data Mining.*
11. Mitra, P., Murthy, C.A., Pal, S.K. (2000). Data Condensation in Large Databases by Incremental Learning with Support Vector Machines. *International Conference on Pattern Recognition.*
12. Platt, J.C.(1999). Fast Training of Support Vector Machines using Sequential Minimal Optimization. *Advances in Kernel Methods*, MIT Press, 185-208.
13. Syed, N.A., Liu, H., Sung, K.K.(1999). Incremental Learning with Support Vector Machines. *International Joint Conference on Artificial Intelligence.*
14. Vilalta, R., Ma, S., Hellerstein, J.(2001). Rule induction of computer events. *IEEE International Workshop on Distributed Systems: Operations & Management*, Springer Verlag, Lecture Notes in Computer Science.
15. Weiss, G., Hirsh, H.(1998). Learning to predict rare events in event sequences. *International Conference on Knowledge Discovery and Data Mining.*
16. Zaki, M.J.(2001). Sequence mining in categorical domains. *Sequence Learning: Paradigms, Algorithms, and Applications*, pp. 162-187, Springer Verlag, Lecture Notes in Computer Science.
17. Zipf, G.K.(1949). *Human behavior and the principle of least effort: An introduction to human ecology.* Addison-Wesley Press.