Filtering Techniques for Rapid User Classification

Terran Lane
School of Electrical and Computer Engineering
Purdue University, West Lafayette, IN 47907-1287
terran@ecn.purdue.edu

Abstract

In the computer security task of anomaly detection, we wish to measure not only the classification accuracy of a detector but also the average time to detection. This quantity represents either the average time between false alarms (for a valid user) or the average time until a hostile user is detected. We examine the use of noise suppression filters as componants of a learning classification system for this domain. We empirically evalute the behaviors of a trailing window mean value filter and a trailing window median value filter in terms of both accuracy and time to detection. We find that the median filter is generally to be preferred for this domain.

Keywords: Noise reduction, time to classification, application.

Introduction

In this paper we examine methods for learning to classify temporal sequences of nominal data as similar to or different from previously observed sequence data under the constraint that we wish to do so 'quickly'. This problem arises from the computer security task of *anomaly detection* (Kumar, 1995). The task in this domain is to characterize the behaviors of a computer user (the 'valid', or 'normal' user) with a profile so that unusual occurrences can be detected by comparing a current input event stream to the profile. To reduce the potential for hostile activities, we wish to make this classification by using as few events as possible.

The goal of the anomaly detection domain is to produce an agent which can detect, through observations of system state, audit logs, or user generated events, when a user or system deviates from 'normal' behavior. The presumption is that malicious behavior, especially on the part of an intruder who has penetrated a system account, will appear different from normal behavior in terms of some function of the present and historical observations of system state (Anderson, 1980; Denning, 1987). In this paper we refer to the individual observations as *events*. Taken over time, the events form an unbroken stream of temporally distributed nominal data. Our work focuses on an anomaly detection agent as a personal assistant that aids a single user in protecting his or her account from abuse. The alternative approach, of characterizing the *system's* state as normal or anomalous, entails a somewhat different set of problems and is examined in, for example, (Lunt, 1990; Forrest et al., 1996; Lee et al., 1997). The learning task for our domain is to form a *profile* describing the valid user's normal patterns of behavior, and to use that profile to classify incoming events as belonging to or differing from the valid user. We also wish to make such detections in the shortest possible time, to minimize the potential damage from a hostile user. We envision the techniques presented here as working in conjunction with other methods such as biometric measurements and attack signature detection to create an overall accurate and robust security assistant.

Because the space of possible malicious behaviors and intruder actions is potentially infinite, it is impractical to characterize normal behavior as a contrast to known abnormal behaviors (Spafford, 1998). It is also desirable, for privacy reasons, that an anomaly detection agent only employ data that originates with the profiled user or is publicly available — an important criterion to much of the computer security community. This requirement leads to a learning situation in which only instances of a single class ('valid user') are available.

In this environment, the anomaly detection agent sees only an unbroken and undifferentiated stream of incoming events and must classify each event as anomalous or normal. The associated learning task (training the agent to recognize a particular user) possesses a number of difficulties not faced by traditional, static learning tasks, including learning in the presence of concept drift, online learning, single class learning, and temporal sequence learning. These issues are further complicated by the high degree of noise introduced by normal variations in human activity patterns.

In other work, (Lane and Brodley, 1998b; Lane and Brodley, 1998a), we have explored some of the data representation, single class learning, online learning, and concept drift issues associated with the anomaly

detection domain. The purpose of this paper is to explore the issues of noise suppression and *time to detection*. In particular, we explore the behaviors to two noise suppression filters in terms of both classification accuracy and time to detection.

The Anomaly Detection System

In this section we describe the architecture of the anomaly detection system classification component and describe the sources of time lag within that component. We present the filtering techniques that are used for noise suppression in the classifier.

System Architecture

We have developed a learning classification system for the anomaly detection domain. The architecture of the classification system is shown in Figure 1. Input event tokens (appearing at the far left) are compared to the current user profile via a similarity function, Sim(), to yield a temporal stream of similarity values. Because this stream is highly noisy, and classification decisions on the raw stream are difficult to make accurately, we smooth it with a window-based filtering function, $F()$. The smoothed similarity stream is classified according to an induced learning model, yielding a stream of binary classifications. At each time step, the output stream classifies the current user as normal (1) or anomalous (0).

In other work we have investigated properties of the similarity function (Lane and Brodley, 1997b), data representation (Lane and Brodley, 1998b) and online pruning policies for the user profile (Lane and Brodley, 1998a), and model parameter estimation and updating issues (Lane and Brodley, 1998a). In this paper, we focus on the noise-suppression filtering function, $F()$, and its relation to classification accuracy and time to detection. To highlight the role of this fragment, we employ only the simplest, static version of the classifier. In this formulation, the profile is a dictionary of previously seen instances, the similarity function is the 1-nearest-neighbor rule with an appropriate distance measure between instances, and the model parameters are a pair of thresholds, $t_{\max}$ and $t_{\min}$. The final classification rule is that a sequence of events is considered normal if and only if its similarity to the profile is between the classification thresholds.

Time to detection

To minimize the potential for damage by a hostile user, we wish to make accurate classifications of 'abnormality' as quickly as possible, both in wall clock time and in number of input tokens required for classification. While a great deal of damage can be done in a very short time (a sort of 'hit-and-run' attack pattern), attack signature matching systems employing databases of known attacks can be employed to detect large classes of short term attacks (Kumar, 1995). We acknowledge that a learning system will have difficulty matching the temporal performance of such known-pattern detectors, so we focus our attention on longer term attacks in which an intruder penetrates a system for the purposes of exploiting its resources for a long period. (For an example of such an attack, see (Stoll, 1989).)

The anomaly detection system outlined in Figure 1 is sufficiently fast in terms of wall time[1], but requires a token lag between input and classification. That is, the classifier must accumulate t additional tokens before making a classification of the first input token. We refer to the lag period t as the *minimum detection length*, as it represents the minimum time in which an anomaly can be detected.

The lag factor is introduced by two components of the classifier: the similarity function, Sim(), and the filtering function $F()$. The similarity function compares subsequences of l input tokens to the user profile, yielding a similarity measure for the entire group. In (Lane and Brodley, 1997a), we examined the effect of the choice of l on classification accuracy, finding that the optimal value varied from user to user but that $l = 10$ was an acceptable compromise across users. The token subsequences are allowed to overlap, so no additional lag is required to align the subsequences. The filter function is a trailing window filter that operates on w similarity values to produce a single smoothed value. Thus, the entire system introduces a minimum detection length of $t = l + w$ tokens.

Filtering methods

We have examined two classes of filtering methods for this domain. The first, trailing window mean filtering, is defined by:

$$v_{\mathbf{D}}(i) = \frac{1}{w} \sum_{i-w+1}^{i} \mathrm{Sim}_{\mathbf{D}}(i)$$

where $\mathrm{Sim}_{\mathbf{D}}(i)$ is the similarity of the token sequence starting at time-step i to the user profile $\mathbf{D}$, W is the window length, and $v_{\mathbf{D}}(i)$ is the final value of sequence i with respect to $\mathbf{D}$. This can be viewed as the normalized convolution of a rectangular window with the raw similarity stream.[2] This filter is known to be good at removing high frequency effects such as noisy edges or ringing at the expense of small (relative to w) features and sharp gradients (Oppenheim and Schafer, 1989).

The second filtering method we investigate is trailing window median filtering, defined by:

$$v_{\mathbf{D}}(i) = \mathrm{med}\{\mathrm{Sim}_{\mathbf{D}}(i-w+1), \ldots, \mathrm{Sim}_{\mathbf{D}}(i)\}$$

Where the med{} operation denotes selection of the median element of a set and the other terms are as

[1] Our prototype anomaly detector can classify approximately six months of history data for a single user in four minutes, running on an Sparc Ultra 1.

[2] Other window shapes such as Hamming or Kaiser are possible, although we do not investigate these here.

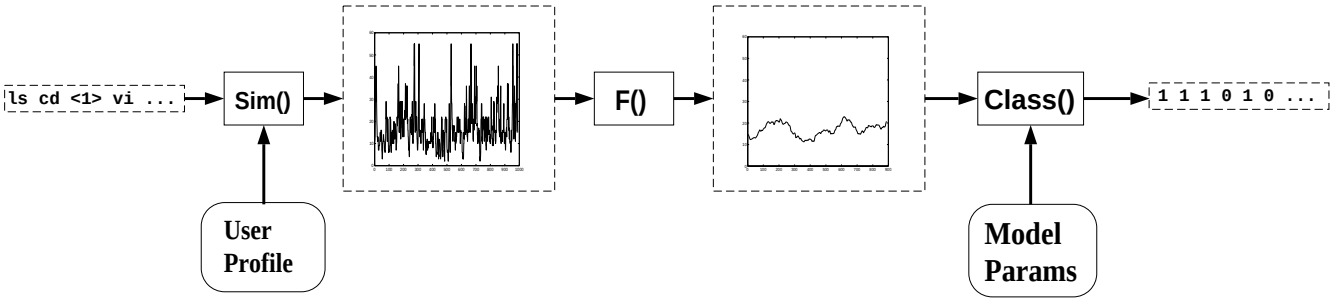

Figure 1: Architecture of the classification component of the anomaly detection system

given above for the mean filter. This filter is non-linear and does not possess as convenient an interpretation as the mean filter. The median filter is known to be generally strong at removing impulsive or shot noise, while preserving edges and features of intermediate size (Jain, 1989).

Empirical Evaluation

In evaluating the performance of alternative filtering functions for this domain, we must examine both the relative accuracy and detection length of each method. In the anomaly detection domain, there are two classes of errors which must be accounted for. We shall denote the rate of incorrectly flagging normal behaviors as the *false alarm* rate and the rate of failing to identify abnormal or malicious behaviors as the *false acceptance* rate. The converse measures are the *true acceptance* and *true detection* rates, respectively. The detection length is the mean time (in events) between a hostile user's first access to an account and the recognition of an anomaly. When the valid user is falsely accused, the detection length is the mean time between false alarms.

In this section, we present an empirical evaluation of the impacts of the choice of filter function, $F()$, and filter window length, w, on the accuracy and detection length of the anomaly detection classifier.

Data Sources and Structure

Of the thousands of possible data sources and features that might characterize a system or user, we chose to examine UNIX shell command data. We did so for two primary reasons: first, our interest is mainly in methods of characterizing human behavioral patterns and command traces reflect this more directly than do, say, CPU load averages and, second, shell data is simple and convenient to collect.[3] Lacking shell traces of actual intrusive or misuse behaviors, we demonstrate the behavior of the system on traces of normal system usage by different users. In this framework, an anomalous situation is simulated by testing one user's command data against another user's profile. This represents only a subset of the possible misuse scenarios — that of a naive intruder gaining access to an unauthorized account — but it allows us to evaluate the approach.

We have acquired shell command data from eight different users over the course of more than a year. The data events were tokenized into an internal format usable by the anomaly detector. In this phase, command names and behavioral switches were preserved, but file names were omitted under the assumption that behavioral patterns are at least approximately invariant across file names. The pattern '`vi <file> gcc <file> a.out`', for example, represents the same class of action regardless of whether `file` is `homework1.c` or `filter.c`.

From each user's data, seven thousand tokens were extracted and divided into train (5,000), parameter selection (1,000), and test (1,000) sets. For each user's data, a profile was constructed from the train data and classification model parameters were chosen by analysis of the parameter selection set. (The details of profile construction and parameter selection are given in (Lane and Brodley, 1998b).) Each user model was then used to classify the eight available test sets. Testing a model against the same user's test set yields the true accept rate, while testing against other users' data yields true detect rates. We use the static test model here to emphasize the effects of the filter component independent of the online learning components. The online learning version of this system is described in (Lane and Brodley, 1998a).

To examine the behaviors of each filter for different window lengths, w, we ran the entire test set for each filter with $w \in \{11, 21, 41, 81, 161\}$. Odd window sizes were chosen so that the median function yielded a single input point, rather than an interpolation.

Accuracy

Examples of the classification accuracies of the two filtering methods are given in Table 1. Because the per-

[3] The techniques discussed here could, of course, be extended to cover any discrete stream of nominal values such as system call logs, keystrokes, or GUI events. Furthermore, this classifier could likely be combined with classifiers based on other measures to yield a system with higher overall performance.

Profiled User	Tested User	Accuracy				
		Window length				
		11	21	41	81	161
		Mean filter				
USER2	USER0	58.1	65.9	90.6	100.0	100.0
	USER1	37.0	44.0	83.7	100.0	100.0
	USER7	87.1	93.4	100.0	100.0	100.0
	SELF	95.1	95.5	87.8	84.3	77.5
USER7	USER1	20.8	36.3	48.7	85.1	99.8
	USER5	21.9	38.9	60.0	94.3	100.0
	USER6	09.7	15.3	19.6	75.5	100.0
	SELF	99.0	99.6	100.0	98.8	98.0
		Median filter				
USER2	USER0	44.1	78.0	93.3	99.5	100.0
	USER1	20.9	63.4	83.1	94.5	100.0
	USER7	76.6	99.6	100.0	100.0	100.0
	SELF	96.4	91.2	91.7	88.1	100.0
USER7	USER1	17.6	34.6	62.8	72.4	91.1
	USER5	16.8	35.2	80.4	92.0	100.0
	USER6	07.8	14.7	35.8	46.3	100.0
	SELF	100.0	98.8	99.2	96.8	81.8

Table 1: True accept (SELF) and true detect (other UESRs) rates for the filtering methods.

formances of the techniques are quite disparate across users, variances for averaged accuracies are large making direct comparison difficult. Therefore, we present extreme cases for the techniques to illustrate strengths and weaknesses of each.

The first case appears in data tested against USER2's profile. Here we find that the median filter model has worse true detection rate than the mean filter at the shortest window lengths ($w = 11$ sequences), but wins substantially at $w = 21$. At longer window lengths, the median filter either wins or has comparable performance to the mean filter. In true acceptance accuracy, the median filter is superior in all cases except $w = 21$. The direct tradeoff between true detection and true acceptance accuracy is typical of this domain, and results from a high degree of overlap between the similarity value distributions of the valid and abnormal users.

An alternative case is displayed in data tested against USER7's profile. Here, the mean value filter has superior accuracies for both true detect and true accept rates in most cases. The exception is the value $w = 41$ sequences. We observe that, while there is a generally increasing trend in accuracy with increasing w, the trend is not uniform. A dramatic jump occurs between $w = 40$ and $w = 80$ for the mean filter, while a similar jump occurs between $w = 20$ and $w = 40$ for the median on USER7's profile. This jump represents the point at which the filter gains the majority of its noise suppression ability.

It seems, then, that while the median filter does not have the highest overall accuracies, it does achieve strong accuracies for smaller values of w than does the mean filter. Although we do not have space to display them here, we have observed cases in which each of the filters has generally decreasing performance with larger values of w. We find that, overall, the median filter has higher true detection accuracies than the mean filter in approximately 65% of the test cases, but outperforms mean on true accept rate only 52% of the time.

Detection length

Possibly a more practically useful measure than accuracy is *total detection length*. This is both a measure of how frequently the legitimate user will be bothered by false alarms and a measure of how quickly a hostile user can be detected. In Table 2, we give detection lengths (in sequences examined) for both filtering methods on the same cases examined above. We omit the minimum detection length (Section) as it is a fixed overhead for a given w. In this table, we wish the SELF detection length to be high, indicating infrequent false alarms, and the detection length for 'hostile' users to be low, indicating rapid detection.

In general, we observe the same trends that we found in accuracy. The important point to note is that detection length is not directly proportional to detection frequency. Specifically, a 10% true detection rate is equivalent to a 10% false alarm rate in *average detections per unit time*, yet the detection lengths for the two cases are quite different. Such a case can be observed in tests against USER7's profile for the median filter, in which the worst false alarm rate (18.2%) corresponds to a detection length of 187.3 sequences, while the much lower true detection rate of 7.8% corresponds to a detection length of only 82.7 sequences.

Profiled User	Tested User	Detection length				
		Window length				
		11	21	41	81	161
		Mean filter				
USER2	USER0	7.4	6.1	1.9	0.0	0.0
	USER1	16.3	15.3	3.1	0.0	0.0
	USER7	0.8	0.6	0.0	0.0	0.0
	SELF	256.8	276.6	167.7	94.3	128.8
USER7	USER1	45.6	31.1	33.3	7.8	0.0
	USER5	48.3	31.9	10.8	1.0	0.0
	USER6	64.4	61.6	60.6	8.0	0.0
	SELF	227.2	421.6	476.0	259.5	257.7
		Median filter				
USER2	USER0	10.1	4.0	1.4	0.0	0.0
	USER1	45.7	10.6	3.0	0.8	0.0
	USER7	1.7	0.0	0.0	0.0	0.0
	SELF	277.2	251.3	255.6	254.6	416.0
USER7	USER1	49.8	38.0	17.3	17.5	3.3
	USER5	59.1	43.6	4.1	2.5	0.0
	USER6	82.7	72.9	29.7	26.8	0.0
	SELF	491.0	266.2	268.5	247.9	187.3

Table 2: Detection lengths for the filtering methods.

This implies that the detector is performing as we desire — flagging hostile users often and quickly, while generating false alarms only rarely.

Finally, we note that because the minimum detection length factor is omitted, the actual values shown are not fully reflective of total time to detection. A tabulated detection length of 10.6 sequences at $w = 21$ corresponds to an actual detection length of 31.6 sequences or a total of 41.6 tokens (for sequence length $l = 10$), and is to be preferred to a tabulated value of 0.0 sequences for $w = 161$. Overall, we have found that the median filter has superior detection length performance in approximately 62% of the true detection cases and in 60% of the true acceptance cases.

Conclusions

We have examined a pair of filter functions for noise suppression in the learning and classification component of an anomaly detection system. Time to detection is a critical issue in this domain. We investigated the the impact of trailing window mean value and trailing window median value filtering on the time to detection for a variety of simulated attack data sets. We found that, though the median filter does not uniformly have the highest accuracies or best time to detection, it does achieve better results with shorter window lengths than does the mean filter. This is caused by an abrupt jump in accuracy which occurs at shorter window lengths for the median filter than it does for the mean filter.

We are currently investigating the nature of this jump, in terms of the type and degree of noise encountered in the similarity function signal. We are also examining the behavior of non-rectangular windows for the filter functions. Full characterization of these filtering techniques for this domain will need to include examination of interactions with the profile formation and classification model stages of the complete learning system. Finally, we intend to investigate methods for characterizing which filtering method is likely to be most successful for a given profile in terms of observed properties of the unfiltered similarity signal.

While the median filter evidences superior performance in between 60% and 65% of the test cases, this margin is not enough to declare it to be universally the superior method for this domain. Nevertheless, it does display strong performance, and if a single method is to be selected, the median filter is the preferred candidate.

References

Anderson, J. P. (1980). Computer security threat monitoring and surveillance. Technical Report Technical Report, Washington, PA.

Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2):222–232.

Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A. (1996). A sense of self for Unix processes. In *Proceedings of 1996 IEEE Symposium on Computer Security and Privacy.*

Jain, A. K. (1989). *Fundamentals of Digital Image Processing.* Prentice-Hall, Englewood Cliffs, NJ.

Kumar, S. (1995). *Classification and detection of computer intrusions*. PhD thesis, Purdue University, W. Lafayette, IN.

Lane, T. and Brodley, C. E. (1997a). Detecting the abnormal: Machine learning in computer security. Technical Report TR-ECE 97-1, Purdue University, School of Electrical and Computer Engineering, West Lafayette, IN.

Lane, T. and Brodley, C. E. (1997b). Sequence matching and learning in anomaly detection for computer security. In *Proceedings of AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management*.

Lane, T. and Brodley, C. E. (1998a). Approaches to online learning and concept drift for user identification in computer security. In *Fourth International Conference on Knowledge Discovery and Data Mining (to appear)*.

Lane, T. and Brodley, C. E. (1998b). Machine learning for computer security: Learning to classify anomalies in temporal sequence data. Technical report, Purdue University, School of Electrical and Computer Engineering, West Lafayette, IN.

Lee, W., Stolfo, S., and Chan, P. (1997). Learning patterns from UNIX process execution traces for intrusion detection. In *Proceedings of AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management*.

Lunt, T. F. (1990). IDES: An intelligent system for detecting intruders. In *Proceedings of the Symposium: Computer Security, Threat and Countermeasures*, Rome, Italy.

Oppenheim, A. and Schafer, R. (1989). *Discrete-Time Signal Processing*. Signal Processing. Prentice Hall, Englewood Cliffs, New Jersey.

Spafford, E. H. (1998). Personal communication.

Stoll, C. (1989). *The Cuckoo's Egg*. Pocket Books.