1st International Conference on Dependable Systems & Networks: New York, New York, USA. 25-28 June 2000.

Benchmarking Anomaly-Based Detection Systems

Roy A. Maxion

Department of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213 USA
Tel: 1-412-268-7556
Email: maxion@cs.cmu.edu

Kymie M.C. Tan

Department of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213 USA
Tel: 1-412-268-3266
Email: kmct@cs.cmu.edu

Abstract

Anomaly detection is a key element of intrusion-detection and other detection systems in which perturbations of normal behavior suggest the presence of intentionally or unintentionally induced attacks, faults, defects, etc. Because most anomaly detectors are based on probabilistic algorithms that exploit the intrinsic structure, or regularity, embedded in data logs, a fundamental question is whether or not such structure influences detection performance. If detector performance is indeed a function of environmental regularity, it would be critical to match detectors to environmental characteristics. In intrusion-detection settings, however, this is not done, possibly because such characteristics are not easily ascertained. This paper introduces a metric for characterizing structure in data environments, and tests the hypothesis that intrinsic structure influences probabilistic detection. In a series of experiments, an anomaly-detection algorithm was applied to a benchmark suite of 165 carefully calibrated, anomaly-injected datasets of varying structure. Results showed performance differences of as much as an order of magnitude, indicating that current approaches to anomaly detection may not be universally dependable.

Keywords: Anomaly detection, benchmarking, computer security, empirical methods, intrusion detection.

1. Introduction

Detection of anomalies in data is a core technology with broad applications: detection of clandestine nuclear blasts using seismic data [17], detection of cardiac arrhythmias in ECG data [1], detection of bridge failures (e.g., cracks, structural deteriorations, etc.) from vibration data, discovering semiconductor defects from plasma-etch data, detecting network faults from traffic data [11], ascertaining user-interface design defects from user data [10], etc. Over the last dozen years, another application has emerged: detection of unauthorized or malicious users on computer hosts and networks, often called intrusion detection. Intrusion-detection systems have become available commercially over the past few years [2, 4, 14]). Although their deployment in the marketplace suggests that these systems benefit their users, there is almost no data measuring their effectiveness. The same paucity of evaluation results plagues the research arena [7, 8, 13, 20].

Evaluating detection systems is a difficult undertaking, complicated by several common practices. For example, most evaluations are done according to a black-box testing regime (e.g., [7]). While black-box testing can demonstrate the overall performance capabilities of a detection *system*, it reveals almost nothing about the performance of components inside the black box, such as how phenomena affecting the components (e.g., a feature extractor or an anomaly detector) or the interactions among them will influence detection performance. If the performance aspects of components like anomaly detectors are not fully understood, then the performance aspects of any system composed of such elements cannot be understood either.

This paper proposes benchmarking as an approach toward understanding the performance characteristics of anomaly-detection algorithms applied to categorical data. Because operational environments are unlikely to be identical in their characteristic signal and noise structures across different application enterprises, the benchmarking scheme incorporates datasets that vary in these characteristics, as measured by entropy, across a continuum from zero to one. The paper shows how a series of benchmark datasets was constructed, both for training and test data, and illustrates how the datasets were varied across a continuum of intrinsic structure to reveal the influence of that structure on the performance of an anomaly detector.

2. Problem and approach

The principal problem addressed here is the expectation that an anomaly detector will perform uniformly, irrespective of the environment in which it is deployed. Benchmarking seeks to address the problem of evaluating detectors across different environments.

Environment variation. All anomaly-detection algorithms operate on data drawn from some kind of computing domain or environment, e.g., a process-control environment, an e-commerce environment, an air force base environment, etc. Embedded in each type of environment is a particular structuring of the data that is a function of the environment itself. Because most, if not all, anomaly-detection algorithms depend on the intrinsic structure embedded in the data upon which they operate, it seems reasonable to assume that differences in such structure would influence detector performance. Structure can be conceptualized as regularity, or the extent to which a sequence is either highly redundant or highly random. High-regularity data contain redundancies that facilitate predicting future events on the basis of past events; low-regularity data impede prediction.

Approach. Benchmarking is proposed as a methodology that can provide quantitative results of running an anomaly detector on various datasets containing different structure. Sets of benchmark data, each set with its own regularity, measured in ten equal steps from 0 to 1, were constructed and used to test anomaly-detection capabilities. Calibrated anomalies were injected into test sets at specified intervals.

Hypotheses. The work described in this paper tests the hypotheses that (1) differences in data regularities, *do* influence detector performances, and (2) such differences will be found in natural environments. If the working hypotheses are true, there are certain implications for anomaly detection, particularly in an information-assurance regime such as intrusion detection. First, the common practice of deploying a particular anomaly detector across several environments may have unintended consequences on detection outcomes, such as increasing false-alarm rates; second, detecting masqueraders by training on a user's behavior in one time period, and testing against that same user's behavior in another time period, may encounter difficulties if the environmental characteristics differ significantly among the several time periods. The extent to which such differences influence detector outcomes may be a function of the environment itself.

Anticipated results. If the first hypothesis is true (differences in data regularities do influence detector performances), then testing an anomaly detector across a range of environmental or behavioral regularities should produce a range of relative operating characteristic (ROC) curves; otherwise, the ROC curves should all be superimposed upon one another. If the second hypothesis is true (characteristic differences will be found in natural environments), then there is convincing evidence that a given anomaly detector cannot be moved arbitrarily from one environment to another without accommodating the differences in regularity (provided that the first hypothesis is true).

3. Structure in categorical data

The idea of intrinsic structure, or regularity, in a categorical data sequence is intuitive. A structured sequence has a definite organizational pattern. That a sequence's structure is intrinsic means that it is the essential nature of the sequence to contain such structure, on the basis of regularities in the process that generated the sequence. Such structure spans a continuum from highly irregular (e.g., random, no apparent structure) to highly regular (e.g., perfectly systematic, readily apparent structure). Examples of sequences with systematic, or regular, structure are: A A A A A ... or A B A B A B ... . The most interesting examples lie somewhere between the extremes of perfect regularity and perfect randomness.

The usual measure of randomness (uncertainty) in a sequence, X, of categorical data, is entropy, sometimes called the Shannon-Wiener Index [18]. Because entropy has no upper bound (in terms of bits), and because it is desirable to measure the randomness of a sequence on a 0-1 scale, relative entropy is used. Relative entropy is simply the entropy of the sequence divided by the maximum entropy for that sequence.

Some events in data sequences necessarily precede others, introducing a notion of sequential dependence amongst the elements of the sequence. Conditional relative entropy reflects this sequential dependency by accounting not only for the probability of an event, but also for the probability of its predecessor. These concepts are covered in most texts on information theory, such as [3]. Conditional relative entropy is used to measure the structure present in the benchmark datasets presented in this paper.

Many anomaly-detection systems depend on the presence of regularities in data (e.g., [5, 6, 15, 20]). In the remainder of this paper, the terms *regularity* or *regularity index* refer to the sequential dependencies of sequences as measured by entropy. A regularity index of 0 indicates perfect regularity (or redundancy); an index of 1 indicates no regularity, i.e., random.

4. Constructing the benchmark datasets

This section provides details of the benchmarking process, followed by an experiment validating the hypothesis that detector performance is influenced by intrinsic structure in background data.

In general, three kinds of data need to be generated: training data (normal background), testing data (background plus anomalous events), and the anomalies themselves. In benchmarking parlance, the training and testing data constitute the anomaly-detector workload. Several factors influence the composition of a sequence: alphabet size, alphabet symbols, regularity or randomness, sequential dependencies among symbols or subsequences, and length. This section describes how the datasets were generated and subsequently combined with anomalies to form the benchmark datasets.

4.1. Defining the sequences

The benchmark training datasets are made up of five suites of eleven files each, totaling 55 files. Each suite used a predetermined alphabet size whose constituency was held constant within the suite. Each suite contained eleven unidimensional datasets with regularities (conditional relative entropies) equally spaced at 0.1 intervals, from 0 to 1 inclusive.

Alphabet size. Alphabet sizes were 2, 4, 6, 8, and 10, yielding the five aforementioned suites. Low-order alphabets have the advantage that almost all operations on them are computationally easy; high-order alphabets, e.g., 10 or more, require significantly more computation, and hence the experiment was limited to a maximum alphabet size of 10.

Alphabet symbols. Symbols for categorical data can be almost anything. In this experiment, symbols were drawn from the standard English alphabet (e.g., A, B, C, D, E, F for alphabet size 6) for simplicity's sake. The symbols could have been system kernel-call names, but such names are longer than a single element, and hence slightly clumsier to handle.

Regularity. Regularity (conditional relative entropy) for a sequence is determined by the transition matrix from which the sequence is generated. There were eleven such matrices, one for each regularity level from 0 to 1 in steps of 0.1.

Sequence length. Because the detector (described later) used in this experiment is based on a moving window that encompasses n sequence elements at once, the datasets needed to be long enough to contain all possible n-grams for a given alphabet to guarantee equiprobable occurrence of n-grams. When the sequence does not contain all possible n-grams, empirical regularity (calculated from the data) will reflect this, and the equiprobable case will be impossible to obtain. For this reason, all datasets contained 500,000 characters (or events).

4.2. Defining the anomalies

An anomalous event is a surprising event. An event is a subsequence of one or more symbols. The extent to which an anomaly is considered surprising is determined by the anomaly detector itself, often on the basis of the expected probability of encountering the event. The anomalous events defined here are considered to be juxtapositional anomalies -- events juxtaposed in unexpected ways. Another type of anomaly, not considered here, is the temporal anomaly, which manifests as unexpected periodicities. Benchmark datasets could be built for either type of anomaly. Several types of juxtapositional anomalies are defined for this study:

Foreign-symbol anomalies. A foreign-symbol anomaly contains symbols not included in the training-set alphabet. For example, any symbol, such as a Q, not in the training-set alphabet comprised of A B C D E F would be considered a foreign symbol. Detection of such anomalies should be relatively simple.

Foreign n-gram anomalies. An n-gram that contains a sequence not found in any n-gram in the training dataset (but not containing a foreign symbol) is considered a foreign n-gram, because it is foreign to the training dataset. A foreign n-gram anomaly contains n-grams not present in the training data. For example, given an alphabet of A B C D E F, the set of all bigrams would contain AA AB AC ... FF, for a total of $6^2=36$ (in general, for an alphabet of α symbols and an n-gram of size n, total possible n-grams = α^n). If the training data contained all bigrams except CC, then CC would be a foreign n-gram. Note that if a foreign symbol appears in an n-gram, that would be a foreign-symbol anomaly, not a foreign n-gram anomaly. In real-world data it is quite common that not all possible n-grams are contained in the data, partly due to the relatively high regularity with which computers operate, and partly due to the large alphabets in, for example, kernel-call streams.

Rare n-gram anomalies. A rare n-gram anomaly contains n-grams that are infrequent in the training data. In the example above, if the n-gram AA constituted 96% of the events in the sequence, and the n-grams BB and CC constituted 2% each, then BB and CC would be rare n-gram anomalies. An n-gram whose exact duplicate is found only rarely in the training dataset is called a rare n-gram. The concept of *rare* is determined by a user-specified threshold. A typical threshold might be .05, which was the threshold used to generate the data sets for the present work. A threshold of .05 means that a rare n-gram would have a frequency of occurrence in the training data of not more than 5%. The selection of this threshold is arbitrary, but should be low enough for "rare" to be meaningful.

4.3. Generating the training and test data

The training data are generated from 11 transition matrices that produce the desired regularities for the sequences such that the regularity indices of the sequences run, in increments of .1, from 0 to 1 inclusive. The transition matrix is entered at a random point, determined by a random number generator. Once inside the transition matrix, each transition is determined by a random number between 0 and 1. To permit using any random number generator, particularly one that has been certified to be as random as technically possible [9], the 500,000 random numbers are computed first, stored in a table, and then used in determining the matrix transitions. In this way, the random-number sequence can be retained, if need be, to enable perfect repetition of the experimental sequence. The seed used for generation is different from the one used to enter the table, simply to go as far as possible to eliminate dependencies in the generation scheme. A single seed is used to generate data for all regularities. The seed is a 4-digit random integer produced from the Perl rand() function. Test data were generated in the same way in which the training data were generated, except that different random-number-generator seeds were used for generating the test data. Using new seeds for the test data guarantees that the generated sequences will retain the same intrinsic structure as seen in the training data, while ensuring that the specific order of symbols is not identical to that in the training data.

4.4. Generating the anomalies

A pool of anomalies is generated independent of generating the test data. A separate pool is generated for each anomaly type. After the pool is prepared, anomalies are drawn from the appropriate pool, and injected into the test data in accordance with the plan detailed in Section 4.5. Each of the anomaly types is generated in a different way, although the size of each anomaly is selected according to common criteria that reflect experimental goals. In the present case, the anomaly size was chosen to be 4, because the window parameter of the detector was set to 4. The paragraph below details the injection process for the various anomaly types.

4.5. Injecting anomalies into test data

The test data were generated without anomalies, as described in Section 4.3, and anomalies were injected into the test data later. The system determines the maximum number of anomalies to inject. This number is arbitrary, but is kept low enough so that the injected anomalies do not change the regularity of the test data. In practice, a heuristic is used that limits the number of injection replacements to not more than .24% of the uninjected test data. As a simple example, if the test data contains 500,000 elements, then .24% of these would be 1200 events which could be replaced by anomalies. If the anomaly size is 4, then only 300 4-gram anomalies could be injected into the test data. An injection interval is selected that determines the minimum and maximum spacing between anomalies. An anomaly is selected at random, with replacement, from the anomaly pool (foreign symbol, foreign n-gram or rare n-gram) and injected according to the constraints imposed by the injection interval. Each injected n-gram anomaly replaces n elements of the test sequence. After the injections have been done, the regularity of the test sequence is recalculated. If the recalculated regularity differs from the target regularity by more than .05, the injections are backed out, and the process is repeated using a broader spacing interval. Of course it is expected that the change in regularity will be larger when injecting foreign symbols.

5. Experiment one: synthetic benchmarks

This experiment tests the hypothesis that the performance of an anomaly detector is influenced by intrinsic structure (i.e., regularity as measured by conditional relative entropy) in data sequences; that is, the hypothesis that the nature of "normal" background noise affects signal detection. If the hypothesis is correct, then the same anomalies, injected into datasets that differ only in regularity, would be expected to generate different hit, miss and false-alarm rates; i.e., the ROC curve for a given regularity should not be superimposed on the ROC curve for any other regularity. If the hypothesis is not true, then all ROCs should be superimposed on one another. A noteworthy implication of this hypothesis, if true, is that if intrinsic structure does influence anomaly-detection capability, the performance of a given anomaly detector will vary across datasets of differing regularity, and as such it cannot be expected that the observed performance on one dataset will extend to datasets of other regularities, even if the detector is retrained. This means that one cannot use the same detector in different computational environments (e.g., research enterprises, educational enterprises, commercial enterprises, military bases, etc.) where different dataset regularities prevail, and expect to obtain results of the same level of accuracy for each environment. Simple retraining will not suffice. This is in absolute contrast to current practice.

5.1. Data sets

The data used were the calibrated benchmark datasets described in Section 4. The rare-4-gram anomalies had less than 5% occurrence in training datasets. For each alphabet size, all variables were held constant except for dataset regularity. There were 275 benchmark datasets total, 165 of which were anomaly-injected.

5.2. Description of anomaly detector

The anomaly detector used in this experiment was designed to detect anomalous subsequences embedded in longer sequences of categorical data. The choice of anomaly detector was arbitrary among the class of detectors that are probabilistically based. It works in two phases: a learning phase and a detection phase. Simply described, in the learning phase it constructs an internal table containing the probability of occurrence of every unique n-gram in the training sequence (e.g., normal background data), where n is user-specified. In the detection phase, it is given a test sequence consisting of normal background data (noise) mixed with injected anomalous-event subsequences (signal.) As the detector scans the test sequence, it raises an alarm each time it encounters an unusual event. The extent to which an event is unusual is the extent to which the probability of an event (i.e., subsequence) exceeds the probability of that event as stored in the table constructed during the training phase. A user-specified threshold between 0 and 1 determines the boundary above which an event is considered anomalous. The tunable parameters of the anomaly detector are window size (set to 4) and anomaly threshold (varied through the range 0-1). A window size of 4 means that the detector determines the level of surprise at a particular event, given the three events that preceded it; i.e., the detector is sensitive to sequential dependency in the data. A similar anomaly detector was reported by Nassehi [15]. Note that this kind of detector is designed to detect juxtapositional anomalies; it may be blind to certain kinds of temporal anomalies, unless a different encoding of input data stream is used. The same anomaly detector was used in all experiments.

5.3. Training the detector

Training, for any probabilistically-based detector, consists of establishing a representation of normal behavior in the detector's tables or data structures. Training the detector was done by running the training portion of the detection program on each of the 11 training datasets in each alphabet size; a total of 55 training sessions were conducted. Resultant data structures for each of the training sessions were stored for later use in detection.

5.4. Testing the detector

Testing the detector on test data consists of running the trained detector, with its data structures fully populated from the normal (training) data, on the test datasets. The detector is expected to indicate the presence of the injected anomalies, without falsely indicating subsequences that are not anomalous, and without missing any of the injected anomalies. Training and testing were done on data sets of the same regularity. Detection thresholds were swept through the range of 0.0 to 1.0 to yield different outcomes, depending on the detector's sensitivity at each threshold. For each of the 5 alphabet sizes, the detector was run on 33 test datasets, 11 for each anomaly type. Outcomes are presented in Section 5.6.

5.5. Scoring the detection outcomes

Several aspects of scoring are considered: exact determination of event outcomes, ground truth, detection threshold, anomaly scope, and presentation of results.

Event outcomes. The primary scoring task is to determine whether or not each event in the input datastream is correctly identified by the detector output in terms of hits, misses and false alarms. This is done by matching detector outcomes against a ground-truth oracle, or key.

Ground truth. Ground truth is ascertained automatically by the injection program, which produces a key to be compared against detector outcomes. The key identifies each event in the test dataset, showing the positions of injected events, their types (foreign-symbol, foreign n-gram, etc.), and their scope, as discussed above.

Threshold. The anomaly threshold determines the magnitude above which the anomaly is taken seriously (i.e., an alarm is raised), and below which the anomaly is disregarded. The magnitude, or surprise level, of an anomaly can vary from 0 to 1, where 0 is completely unsurprising and 1 is astonishing. In practice, one may decide that somewhere in between is the best set point for a particular environment coupled with a particular anomaly detector. For the detector used in the present study, the anomaly threshold was varied through its 0-1 range.

Scope. The scope of an anomaly refers to the locations in the test data where an anomaly would be detected. For example, if a 4-element anomaly had been injected at test-data location 101, then the basic scope of the anomaly would cover the four locations actually covered by the injected anomaly (101, 102, 103, and 104). For detectors that are sensitive to sequential dependencies, the scope of the anomaly may extend beyond the basic scope to cover residual anomalies. For example, the 4-gram AAAA would not only be anomalous by itself, but also in juxtaposition with whatever followed it in the data sequence. The residual anomalies would cover the distance that the detector window extends past the actual injected 4-gram; if the detection window was set to 4, then the scope would extend 3 elements beyond the actual anomaly, for a total scope of 7. Any detection within the basic scope or extended scope is considered a correct detection.

A correct detection, or hit, occurs when any point in the basic or extended scope of an injected anomaly is identified as anomalous. A false alarm is any detection that falls outside the total scope of the injected anomaly. A stricter scoring would count only detections in the basic scope, in which case detections in the extended scope would be regarded as false-alarm errors. A miss is the absence of a detection within the basic (not extended) scope of the injected anomaly.

Presentation of results. Experimental outcomes were analyzed graphically, using a technique from signal detection theory called ROC analysis -- the preferred method for measuring the accuracy of diagnostic systems [19]. Note that diagnosis is a classification process that assigns outcomes to predetermined classes; in the current case there are exactly two alternatives: each event in the test sequence is either anomalous or it is not. ROC analysis compares two aspects of detection systems: percent correct detections and percent false detections (sometimes called hits and false alarms, respectively, or true positives and false positives). The methodology, based on statistical decision theory, was originally developed in the context of electronic signal detection (e.g., radar) [16]. A detector operates through a range of sensitivities; the higher the sensitivity, the more likely the possibility of confusing signal and noise (e.g., anomaly and background), resulting in decisions that either identify noise as signal (false alarm) or, at the other extreme, fail to identify signal for what it is (miss). One wishes to find the point, between these extremes, at which to set the sensitivity; this is achieved by incrementing through a series of operating sensitivities, thereby sweeping out a relative operating characteristic (ROC) curve, with false alarms on the X-axis and hits on the Y-axis. As can be seen in Figure 5-1, the best operating point lies at the upper left (100% detection, 0% false alarms), and the worst at the lower right (all events falsely identified). The diagonal from [0,0] to [100,100] is the line indicating random guessing. Selecting the best operating point on the ROC curve is a matter of determining the relative cost of the two kinds of errors -- false alarms vs. missed detections, but this is outside the scope of the present work.

5.6. Results

A total of 165 separate tests was conducted, using 5 alphabets, 3 anomaly types, and 11 regularities. Results were graphically very similar across all alphabets and all regularities. Consequently, only one result, typical of the 165, will be presented. Figure 5-1 shows the ROC curve family for alphabet 6. Notice in the figure that the ROC curve for regularity index 0 (completely regular) would be a point at coordinate [0,100], indicating perfect performance, if there had been any rare-n-gram anomalies at regularity 0; by definition there could not have been. Also, the ROC curve for regularity-index 1 (completely random) is the 45-degree diagonal that, in signal-detection theory, represents pure guessing.

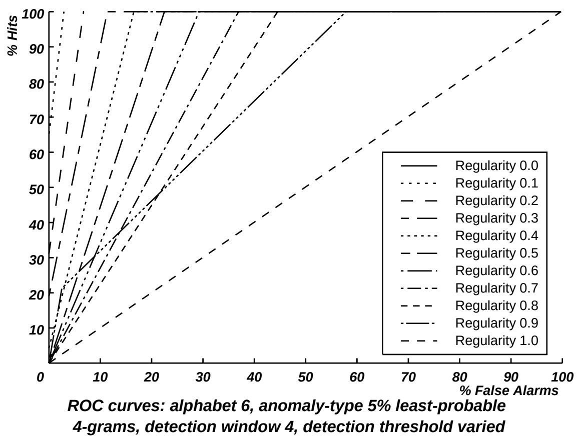

Figure 5-1: Synthetic benchmark datasets: ROC curve family; anomalies drawn from 5% least-probable 4-grams; alphabet 6, detector window 4, detector thresholds swept 0.0 to 1.0, data regularity (measured as entropy) indices 0.0 to 1.0. Different line types (dotted, dashed, etc.) are for visual aid only. Line across top appears solid only because many lines overlap.

It is noteworthy that none of the curves overlap until they reach the 100% hit rate, demonstrating that regularity does influence detector performance. If regularity had no effect, the fan effect clearly evident in the figure would not be present. What appears to be a solid line across the top of the figure is not actually solid; it's the superimposition of several curves at the 100% detection level. The hypothesis that regularity influences detector performance is confirmed.

Figure 5-2 shows the same results from another

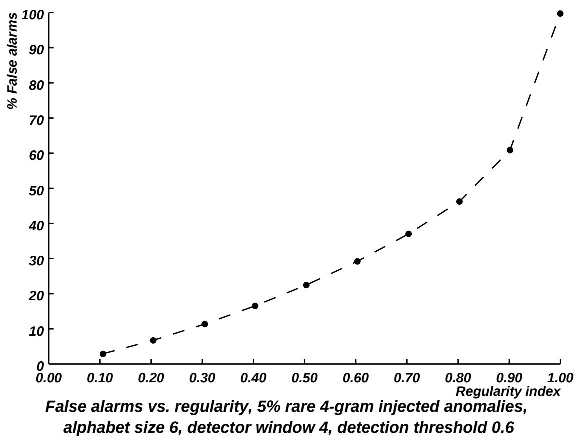

Figure 5-2: Synthetic benchmark data; false alarms vs. regularity index; detection threshold held constant at 0.6, 100% hit rate, no misses, rare-4-grams, detection window 4.

perspective: the false-alarm rate rises as the regularity index grows (i.e., the data become more and more random). Each point on the graph indicates the false alarm rate for each of the 11 datasets of increasing regularity index (increasing randomness), with detection threshold held constant, using the lowest value at which 100% hits could be achieved. As regularity degrades from highly regular (at the left) to highly irregular (at the right), the false alarms rise sharply, even though the detector did not miss any anomalous events. The deterioration in detector performance manifests as an increase in the false-alarm rate. It is important to note that the deterioration in detector performance is attributed solely to changes in regularity; nothing else in these experiments was manipulated.

These results demonstrate, starkly, that training and testing on datasets of the same regularity will facilitate a particular hit vs. false-alarm ratio; and that that ratio may not be similarly achieved when the same detector (using the same parameters) is trained and tested on datasets of different regularities. The results achieved on one set of data are very different from those achieved on another, and this difference grows substantially as the regularity index increases. The same detector cannot be expected to achieve the same results when used on datasets of differing regularities. Note that if regularity in a single dataset is nonstationary, detector performance will vary even within the dataset. This suggests that regularity needs to be tracked in real time, possibly changing either the nature of the detector in response to changes in regularity, or shifting confidence in the detector's results. Although these particular results are for the probabilistically-based detector used in this work, other detector architectures may produce similar differences, possibly manifested in different ways.

6. Experiment two: using real-world data

Experiment one demonstrated that intrinsic structure in data, as indicated by the regularity index, influences anomaly-detection performance. The results, however interesting, are of little use unless they can be grounded in naturally-occurring data; i.e., if natural data show differences in regularities, then perhaps regularity can be used as one predictor of detection performance. The results shown in this section demonstrate clearly, based on data obtained from a natural domain, that regularity is a characteristic of natural data, and that regularities can be different even within a single environment.

6.1. Natural dataset: undergraduate machine.

BSM[1] audit data were taken from an undergraduate student computer running the Solaris operating system. System-call events were extracted for each user session in the 24-hour monitoring period. Most users produced only a single session, but in cases for which multiple sessions existed for the same user, these sessions were concatenated into one session for the purpose of calculating regularities over a 24-hour period. These data were examined with the objective of determining whether or not data in different user enterprises exhibit different regularities.

Results. The regularities of the 58 user sessions active on one day are illustrated in Figure 6-1. The regularities differ considerably with respect to one another, illustrating that different users have different behaviors, at least with respect to the regularities of their system-call streams. Although there has long been an intuitive understanding that user behaviors differ, measures of regularity show these differences quantitatively. Note that the range of differences among user-session regularities equates to a difference in detector performance in the synthetic data in terms of a false-alarm range of about 10%, suggesting that using the same detector in different conditions may not yield the expected results.

7. Conclusion

The principle objective of this paper has been to show that intrinsic structure in data, as measured by conditional relative entropy, will affect the performance of anomaly detectors. This has been demonstrated experimentally through the use of synthetic benchmark datasets, generated so that each dataset had a different, carefully measured regularity. Regularity was manipulated in these datasets so that anomaly detectors could be evaluated at the component level, not at the system level. In the experiments conducted here, all variables were held constant except regularity, and it was established that a strong relationship exists between detector accuracy and regularity.

[1]BSM is the Sun SHIELD Basic Security Module; it provides a security-auditing subsystem for Solaris-based computers.

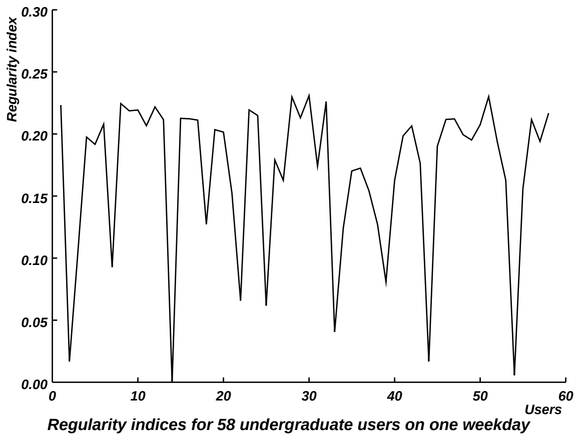

Figure 6-1: Natural dataset: Solaris BSM data; undergraduate machine; regularity indices for 58 undergraduate users.

Both of the main hypotheses were affirmed: regularity does influence the performance of a probabilistic detector in that false alarms rise sharply as the regularity index rises; and regularity differences were found to occur in natural data. There are important implications of this work. First, an anomaly detector cannot be evaluated on the basis of its performance on a dataset of one regularity, and be expected to perform similarly on datasets of different regularities, in contrast to current practice. Second, differing regularities do not necessarily occur only between different users or different environments; they also occur within user sessions and among users, indicating that shifting regularities may prevent effective anomaly-detection performance even within one dataset. Overcoming this obstacle may require a mechanism to swap anomaly detectors or change the parameters of the current anomaly detector whenever regularity changes.

Although it is beyond the scope of this paper, it can be shown [12] that other types of anomaly detectors (e.g., decision trees, neural networks, etc.) are similarly affected by shifts of intrinsic structure in data. In general, at least one of the quadrants of signal detection theory (hits, misses, false alarms or correct rejections) is affected by changes in regularity.

8. Acknowledgements

It is a pleasure to acknowledge the contributions of David Banks (U.S. Bureau of Transportation Statistics) and Robert Olszewski (CMU), both of whom participated in a number of fruitful discussions. We are especially grateful to Robert Olszewski, who implemented the detector used in the demonstration experiment. The authors thank the Defense Advanced Research Projects Agency for their support of this research through contracts F30602-96-1-0349 and F30602-99-2-0537.

References

[1] Akhavan, S. and Calva, G., ‘‘Automatic Anomaly Detection in ECG Signal by Fuzzy Decision Making’’, In *Proceedings of 6th International Conference on Fuzzy Theory and Technology*: Association for Intelligent Machinery, 23-28 October 1998, pp. 96-98, Research Triangle Park, North Carolina.

[2] Amoroso, Edward, *Intrusion Detection*, Intrusion.Net Books, Sparta, New Jersey, 1999.

[3] Cover, Thomas M. and Thomas, Joy A., *Elements of Information Theory*, Wiley, New York, 1991.

[4] Debar, Herve; Dacier, Marc and Wespi, Andreas, ‘‘Towards a Taxonomy of Intrusion-Detection Systems’’, *Computer Networks*, Vol. 31, No. 8, 23 April 1999, pp. 805-822.

[5] Forrest, Stephanie; Hofmeyr, Steven A. and Somayaji, Anil, ‘‘Computer Immunology’’, *Communications of the ACM*, Vol. 40, No. 10, October 1997, pp. 88-96.

[6] Lane, Terran and Brodley, Carla E., ‘‘Temporal Sequence Learning and Data Reduction for Anomaly Detection’’, In *5th ACM Conference on Computer and Communications Security*. New York: Association for Computing Machinery, 3-5 November 1998, pp. 150-158, San Francisco, California.

[7] Lippmann, Richard P.; Fried, David J.; Graf, Isaac; Haines, Joshua W.; Kendall, Kristopher R.; McClung, David; Weber, Dan; Webster, Seth E.; Wyschogrod, Day; Cunningham, Robert K. and Zissman, Marc A., ‘‘Evaluating Intrusion Detection Systems: The 1998 DARPA Off-Line Intrusion Detection Evaluation’’, In *Proceedings of the DARPA Information Survivability Conference and Exposition: DISCEX-2000, Volume 2*. Los Alamitos, California: IEEE Computer Society, 25-27 January 2000, pp. 12-26, Hilton Head Island, South Carolina.

[8] Lunt, Teresa F.; Tamaru, Ann; Gilham, Fred; Jagannathan, R.; Neumann, Peter G. and Jalali, Caveh, ‘‘IDES: A Progress Report’’, In *Annual Computer Security Applications Conference*. Tuscon, Arizona: IEEE Computer Society Press, 3-7 December 1990, pp. 273-285.

[9] Marsaglia, George, ‘‘A Current View of Random Number Generators’’, In *Computer Science and Statistics: Proceedings of the Sixteenth Symposium on the Interface*, L. Billard (Ed.): Elsevier Science Publishers, 1984, pp. 3-10.

[10] Maxion, Roy A. and deChambeau, Aimee L., ‘‘Dependability at the User Interface’’, In *25th International Symposium on Fault-Tolerant Computing*. Los Alamitos, California: IEEE Computer Society, 27-30 June 1995, pp. 528-535, Pasadena, California.

[11] Maxion, Roy A. and Feather, Frank E., ‘‘A Case Study of Ethernet Anomalies in a Distributed Computing Environment’’, *IEEE Transactions on Reliability*, Vol. 39, No. 4, October 1990, pp. 433-443.

[12] Maxion, Roy A. and Tan, Kymie M.C., ‘‘Comparing Anomaly-Detection Algorithms’’. Forthcoming.

[13] Me, Ludovic, ‘‘Security Audit Trail Analysis Using Genetic Algorithms’’, In *International Conference on Computer Safety, Reliability and Security*, Janusz Gorski (Ed.). Poznan-Kiekrz, Poland: Springer-Verlag, 27-29 October 1993, pp. 329-340.

[14] Mukherjee, Biswanath; Heberlein, L. Todd and Levitt, Karl N., ‘‘Network Intrusion Detection’’, *IEEE Network*, Vol. 8, No. 3, May/June 1994, pp. 26-41.

[15] Nassehi, Mehdi, ‘‘Anomaly Detection for Markov Models’’, Tech. report RZ 3011 (#93057), IBM Research Division, Zurich Research Laboratory, 30 March 1998.

[16] Peterson, W.W.; Birdsall, T.G. and Fox, W.C., ‘‘The Theory of Signal Detectability’’, *Transactions of the IRE Professional Group on Information Theory*, Vol. PGIT-4, 1954, pp. 171-212.

[17] Ringdal, Frode, ‘‘Teleseismic Event Detection Using the NORESS Array, with Special Reference to Low-Yield Semipalatinsk Explosions’’, *Bulletin of the Seismological Society of America*, Vol. 80, No. 6, December 1990, pp. 2127-2142.

[18] Shannon, Claude E. and Weaver, Warren, *The Mathematical Theory of Communication*, University of Illinois Press, Urbana, Illinois, 1949.

[19] Swets, John A., ‘‘Measuring the Accuracy of Diagnostic Systems’’, *Science*, Vol. 240, No. 4857, 03 June 1988, pp. 1285-1293.

[20] Teng, Henry S.; Chen, Kaihu and Lu, Stephen C-Y., ‘‘Adaptive Real-Time Anomaly Detection Using Inductively Generated Sequential Patterns’’, In *IEEE Computer Society Symposium on Research in Security and Privacy*. Oakland, California: IEEE Computer Society Press, 7-9 May 1990, pp. 278-284.