Cluster Ensembles for Network Anomaly Detection

Art Munson MMUNSON@CS.CORNELL.EDU
Rich Caruana CARUANA@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Abstract

Cluster ensembles aim to find better, more natural clusterings by combining multiple clusterings. We apply ensemble clustering to anomaly detection, hypothesizing that multiple views of the data will improve the detection of attacks. Each clustering rates how anomalous a point is; ratings are combined by averaging or taking either the minimum, the maximum, or median score. The evaluation shows that taking the median prediction from the cluster ensemble results in better performance than single clusterings. Surprisingly, averaging the individual predictions a) leads to worse performance than that of individual clusterings, and b) performs identically to taking the minimum prediction from the ensemble. This counter-intuitive result stems from asymmetric prediction distributions.

1. Introduction

Many data mining tasks involve looking for unusual, novel, or atypical data patterns. In the literature various names are given to the problem of finding "weird" data: outlier detection, novelty detection, fraud detection, anomaly detection, *etc.* Regardless of the domain, the interesting data for these tasks is typically both rare and defined in contrast to normal data (*i.e.* the interesting data is anything that is not normal).

We examine novelty detection in the context of **anomaly detection**, a subtask for intrusion detection. Compared with **misuse detection**, where known attacks are detected using hand crafted rules and/or signatures, anomaly detection focuses on detecting *novel* attacks. While current misuse detection systems achieve high detection rates with few or no false alarms, they are generally unable to detect novel attacks. In contrast, anomaly detection relies on models of 'normal' behavior and treats deviations from the model as anomalies that are potential attacks. While anomaly detection systems are able to detect new, previously unseen attacks, they suffer from high false alarm rates and the inability to identify the specific type of attack occurring (Ghosh & Schwartzbard, 1999). Typically misuse and anomaly detection are used together.

For intrusion detection false alarm rate is as important as detection rate. In practice, intrusion detection systems (IDSs) emit alarms that are checked by human operators who determine if an alarm is valid and then act accordingly. A high false alarm rate can overwhelm operators and make the IDS unusable.

This paper investigates using an ensemble of clusterings for anomaly detection. Recent research demonstrates that **ensemble clustering** can combine multiple and varied clusterings into a superior consensus clustering (Ayad & Kamel, 2003; Dudoit & Fridlyand, 2003; Fred & Jain, 2002; Hadjitodorov et al., 2005; Leisch, 1999; Strehl & Ghosh, 2003; Topchy et al., 2004). The concrete problem we consider is learning a model from historical network traffic (that is assumed to be normal), and using the model to identify anomalous patterns in new traffic. Our work is specifically interested in whether cluster ensembles improve anomaly detection performance (compared to single clusterings). That is, do different viewpoints and concepts of normalcy highlight the truly abnormal data while reducing noise (false alarms)?

Our approach is to create hundreds of different clusterings of the known normal data using k-means clustering (model building or training). When looking for anomalies in new data (classification), each clustering assigns a number in the range [0,1] to a data point that represents how much the point does not fit the clustering. The overall anomaly score is computed from the individual scores assigned by each clustering.

Presented at 1^{st} North East Student Colloquium on Artificial Intelligence (NESCAI), Ithaca, NY, 2006.

We evaluate the approach using the 1999 DARPA / Lincoln Laboratory Intrusion Detection Evaluation (IDEVAL-1999) data, an established high quality synthetic data set. The results show that combining an ensemble of clusters can improve performance, depending on how the individual scores are combined. Surprisingly, averaging the individual scores (taking the mean) performs worse than the expected performance of an individual clustering for low false alarm thresholds (the setting of most interest), while taking the median score convincingly outperforms the expected performance of a single clustering.

The rest of the paper is organized as follows. We summarize related work in Section 2. Section 3 describes our approach in detail. Experimental results and discussion follow in Sections 4 and 5; our conclusions are in Section 6. A description of the IDEVAL-1999 data set is in Appendix A.

2. Related Work

The defining characteristic of the anomaly detection problem is the assumption that labeled attack data is not available. Logically, it is impossible to anticipate what new attacks will be developed. Multiple approaches have been taken to address the lack of labeled attacks, including:

- Generating artificial attack data to train a supervised classifier (Ghosh & Schwartzbard, 1999; Lee et al., 2001).
- Learning a model that fits available normal data (or that could have generated the data). New data points are assigned a statistical probability of being generated by the model (Barbará et al., 2001; Mahoney, 2003; Sekar et al., 2001).
- Clustering the data and looking for outliers or atypical density patterns (*e.g.* an extremely dense and isolated cluster may indicate a denial of service attack) (Chan et al., 2003; Eskin et al., 2002; Sequeira & Zaki, 2002). These methods do not require "normal" data. Indeed, they are sometimes used as a way to remove anomalies from historical data to create a clean data set that is used in learning a model of normal data patterns.

Fitting a model to the normal data works well if a good model type can be found; the best-performing network anomaly detectors fall into this paradigm. Often, however, the choice of model restricts the kind of attacks that can be detected. (*E.g.* a finite state model of the HTTP protocol cannot detect port scanning.)

Anomaly detection as a field has received increased research attention following the two DARPA funded intrusion detection evaluation tests run by Lincoln Laboratory in 1998 and 1999. Most of the undetected intrusion attacks in these evaluations were novel denial of service attacks (DoS) and novel remote to local attacks (basically privilege escalation). The seriousness of these hard to detect attacks and the long analysis time prior to adding signatures to misuse detection systems has led researchers to investigate several models and techniques for anomaly detection, including: neural networks (Ghosh & Schwartzbard, 1999), rule learners such as RIPPER (Lee et al., 2001) and LERAD (Chan et al., 2003; Mahoney & Chan, 2003), finite state automata (Sekar et al., 2001; Sekar et al., 2002), and immune-system inspired feature space categorization (Dasgupta & González, 2002). Statistical network anomaly detectors include Barbará *et al.*'s Naïve Bayes detector (2001) and Mahoney and Chan's PHAD (2001) and NETAD (2003) systems. Like this work, PHAD and NETAD are trained using packet header bytes. Both systems also incorporate information about how much time passes between packets. This information is an important factor in NETAD achieving the best known performance on IDEVAL-1999 data (132/185 attacks with 100 false alarms).

Our work is perhaps most comparable to CLAD, a clustering network anomaly detector that builds fixed-width, soft clusters in two passes. CLAD looks for two kinds of anomalies: sparse clusters with few members, and dense clusters that are far away from other clusters (likely caused by flood attacks). At 100 false alarms, CLAD detects 76/185 attacks in the IDEVAL-1999 data (Chan et al., 2003).

Inspired by the successful application of ensemble methods to supervised learning (esp. bagging (Breiman, 1996)), researchers are now investigating ensemble clustering as a way to produce superior clusterings (Leisch, 1999; Dudoit & Fridlyand, 2003; Ayad & Kamel, 2003), reuse knowledge contained in existing clusterings (Strehl & Ghosh, 2003), and estimate the confidence of cluster assignments (Dudoit & Fridlyand, 2003). Leisch's work on bagged clustering (1999) provides a prototypical example: multiple clusterings are produced using k-means to cluster bootstrap samples of the data set, which are then combined into a consensus partitioning, or clustering, by using hierarchical clustering to group the produced centroids. Many different techniques have been proposed for combining clusterings; most of them are quadratic in the number of data points *(making them infeasible for this problem domain).*

Despite the surge in interest, cluster ensembles are not yet well understood. Hadjitodorov *et al.* (2005)

present the counter-intuitive result that maximizing ensemble diversity can reduce clustering accuracy. Their results also hint that larger ensembles may also hurt performance. This contrasts with Topchy *et al.*'s proof that the consensus partition converges to a true underlying partition as the number of clusterings in the ensemble increases (Topchy et al., 2004).

Our work differs from previous ensemble clustering work in that we do not combine clusterings to produce a consensus partitioning; instead we derive predictions from each clustering and combine them to improve performance.

3. Approach

For each test point—a packet in our case—multiple models predict if the point is normal or anomalous. Within this general framework the following decisions need to be made:

1. What do the individual models look like, and how are they created?
2. What does a single prediction look like?
3. How are predictions combined?

We use clusterings as our models to avoid requiring labeled attacks for training. It is worth noting that although we build the models from *clean* data—data that is free of attacks—the system does not require this. An interesting future question is how well this system performs when trained on dirty data.

The following subsections describe how our system answers the above questions.

3.1. K-Means Clustering

K-means clustering is a widely used iterative clustering algorithm. Given k, the number of clusters to form,

```
Choose k initial cluster centers
  (randomly in our case)
REPEAT
  0. Remove all members from the clusters.
  1. Assign each data point to closest cluster,
     based on distance to its center.
  2. Set each cluster center to the center of
     mass of its members (i.e., the mean of the
     members).
UNTIL clustering converges OR reach MAX_ITER
```

With each iteration the cluster centers move closer to a natural clustering of the data. (See Pelleg and Moore (1999) for a more efficient algorithm using kd-trees that can be applied to low-dimensional data.) For simplicity we use Euclidean distance as our distance measure and set `MAX_ITER` to 100. In practice `MAX_ITER` rarely terminates the loop; most training runs converge within 50 iterations.

To determine convergence, the fitness of the current clustering is compared to the previous clustering's fitness. When the current clustering's fitness is equal to or worse than the previous, iterative refinement is stopped. Fitness is measured using **distortion**, for which k-means is known to converge to a local minimum. Formally, the distortion$_\phi$ of a clustering ϕ is

$$\text{distortion}_\phi = \frac{1}{N}\sum_x [d(x, \phi(x))]^2$$

where N is the total number of points, x ranges over all data points, and $d(x, \phi(x))$ is the distance from x to its assigned cluster centroid $\phi(x)$. Intuitively, optimizing distortion minimizes the sum of squares for all x to their centers, thereby fitting a clustering to the data.

Despite k-means' simplicity, it works reasonably well. Importantly, it trains in $O(kN)$ time (compared with other clustering algorithms with $O(N^2)$ training time). We expect that most anomaly detection domains will require a large data set to establish the 'normal' areas, making quadratic clustering algorithms infeasible.

K-means does have a drawback. Specifically, it requires choosing the value of k *a priori*. Typically, a short parameter tuning experiment is run to find a good value for k that results in reasonably good clusterings. Luckily the ensemble framework obviates the need to perform this supervised optimization by letting us combine votes from different k clusterings. This not only means we do not have to run expensive experiments to find k, but also allows the ensemble to consider different granularity views of the data.[1]

3.2. Measuring Abnormality

Given a clustering the next question is how to classify new points as normal or anomalous. Intuitively, abnormality is a continuous scale; point x_1 can be more or less anomalous. For anomaly detection it is desirable to assign every new point a score that rates how anomalous the point is; a human expert can then focus investigations on the high scoring alarms that are most likely to be attacks.

We define **fringe factor** as a measure of a point's abnormality w.r.t. a clustering ϕ. Let $c \in \phi$ be the cluster closest to a new point x (*i.e.* c's centroid is the closest centroid to x). Then $\mathsf{ff}(x)$, x's fringe factor, is the percentage of c's members that are closer to the centroid. For example, if a cluster has 1,000 members and 337 of

[1] Indeed, our experiments find that the best k changes with how many false alarms are allowed.

them are closer to the centroid than x, $\mathsf{ff}(x) = 0.337$. Another point that lies wholly outside the cluster will get a 1 since 100% of the cluster's members are closer to the centroid.[2]

3.3. Combining Predictions

Predictions from multiple clusterings are combined to get an overall **anomaly score** that represents the constituents' combined point of view. We evaluate the system using four combination methods: computing the mean, taking the median prediction, taking the minimum prediction, and taking the maximum prediction. While averaging is the typical way to combine predictions, the mean is less suited to asymmetric distributions. Figure 1 shows the distributions of fringe factor scores for some sample points from the IDEVAL-1999 dataset. Most of the points, both anomalous and normal, have skewed fringe factor distributions. Note that the distributions are far more asymmetric than they might first appear since the y-axes use a log scale to make the graphs legible. Using the median as the ensemble score is potentially more robust in this situation. The minimum score is the most conservative prediction—all predictors agree that the point is at least as anomalous as the minimum. Conversely, using the maximum score aggressively flags a point as anomalous if any predictor thinks it is abnormal.

4. Evaluation

We evaluate cluster ensembles for anomaly detection using the mean, the median, the min, and the max as ways to combine fringe factor scores from individual clusterings. All clusterings are built using the filtered data from week 3. For each k in the range from 10 to 100 (increments of 10), 50 clusterings are built for a total of 500 clusterings. The range of k was chosen arbitrarily before looking at any data. Week 2 data was used for testing during development. All results are for testing on weeks 4 and 5.

4.1. Improvement from Cluster Ensemble

To determine if using an ensemble of clusters helps, we compare the ensemble performances to the detection performance of a single clustering. Unsurprisingly, k-means' tendency to find local minima results in clusterings with vastly different performance. Consequently, we define our **baseline** to be the expected performance of a single clustering *with the best k value*. In other words, we average the performance of the single clusterings with the same k value (yielding 10 potential baselines) and pick the best average. This approximates the performance one would get using a single clustering when k has been selected using parameter tuning data.

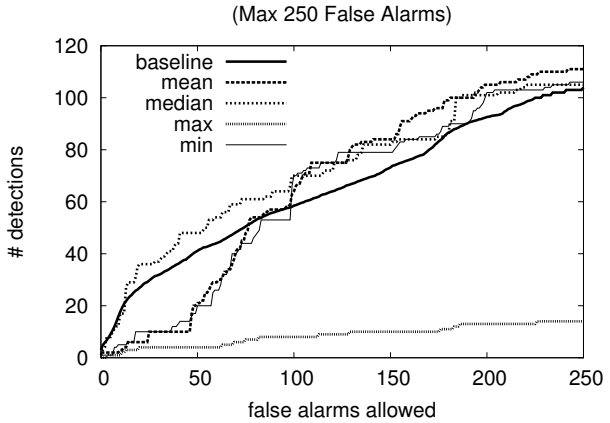

Figure 2. Detections as a function of false alarm threshold.

Figure 2 shows the trade-off between false alarms and detections. The x-axis is the false alarm threshold; moving to the right increases the maximum number of false alarms that can be tolerated. The y-axis is the number of detections made at a given false alarm threshold. Note that as the false alarm threshold changes, the baseline picks averages from different k.[3]

For 100 false alarms (the traditional threshold for this data set), MAX, MEAN, MIN, and MEDIAN detect 5.1% (8/157), 40.8% (64/157), 44.6% (70/157), and 44.6% (70/157) of the attacks, respectively. The ensemble performs better than the expected performance of a single clustering for all combination methods except MAX; the baseline detects 37.3% (58.5/157) of the attacks. As Figure 2 shows, however, the difference between the combination methods is more striking than these numbers suggest.

Two surprising observations can be made from Figure 2. First, MEAN performs much worse than the baseline for low false alarm thresholds. This is markedly different from how supervised ensemble learning methods, such as bagging (Breiman, 1996), usually improve performance—and almost never hurt it. Interestingly, as the threshold increases MEAN does steadily better, until it reliably performs better than the baseline. Second, MEAN and MIN perform almost identically at all false alarm thresholds. This is surprising because it implies that MEAN is making conservative predictions, and that its performance is strongly

[2] If the closest cluster has only a few points the ff scores are very coarse. In our experiments the smallest clusters have at least 200 points, making this a non-concern.

[3] The optimal k tends to increase as the false alarm threshold increases. For thresholds between 0 and 250, the best k is either 20, 30, or 40.

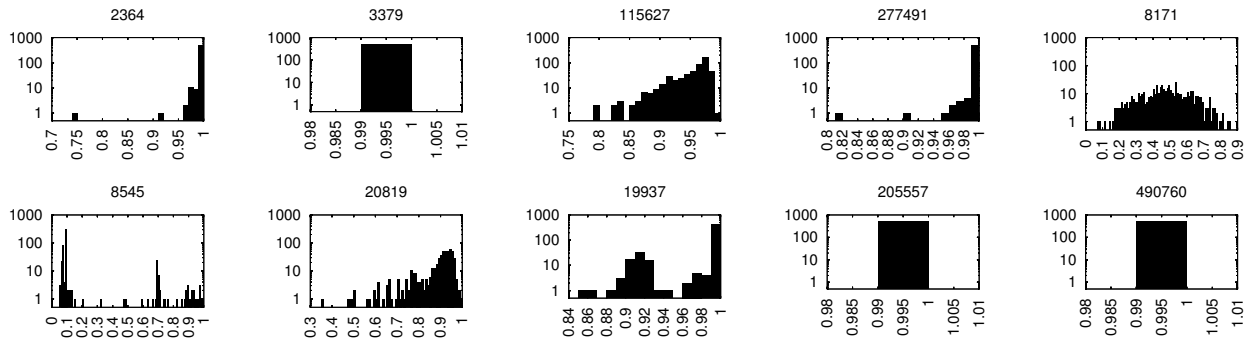

Figure 1. Fringe factor distribution from 500 clusterings for various packets. In each graph the frequency (y-axis) for each fringe factor score (x-axis) is depicted. Note that the range varies for the x-axis, and that the y-axis uses a log scale. The first 4 graphs correspond to anomalous packets; the remaining 6 are for non-attack packets. 2364: detected by MEDIAN. 3379: detected by MEAN and MEDIAN. 115627/277491: detected by neither MEAN nor MEDIAN. 8171, 8545, 20819: low ranked normal packets. 19937: false alarm for MEDIAN. 205557: false alarm for MEAN and MEDIAN. 490760: false alarm for MEAN.

influenced by the lowest fringe factor scores.

MEDIAN, on the other hand, outperforms the baseline at all but the lowest false alarm thresholds. Given that low outlier scores appear to strongly impact MEAN, it is not surprising that MEDIAN—which should be robust to asymmetric outliers—performs well.

Finally, MAX performs poorly. This suggests that there is a lot of noise in the highest score for the data points. In other words, most any point can look anomalous to at least one of the clusterings. This underlines the risk associated with using a single clustering. Combining multiple clusterings is a natural way to overcome the large variation in individual performances.

Beyond 500 false alarms performance plateaus with all methods (except MAX) detecting around 120 attacks, and attack detections become increasingly lost among the false alarms.

In summary, our results indicate that combining clusterings improves performance if the fringe factors are combined using the median. Averaging the individual predictions hurts performance unless a high false alarm threshold is used.

5. Discussion

In this section we investigate more closely why averaging individual predictions leads to below average performance for false alarm thresholds below 100. For MEAN to perform worse than the baseline, averaging must be promoting high ranking false alarms over true attacks. Consider the following scenario. Each individual clustering ranks a true attack first, followed by 3 false alarms. If the top attacks are all different and the false alarms are the same (across clusterings), then averaging the fringe factors will move the false alarms ahead of all the detections.

The above scenario is particularly likely if there are outliers in the distribution of the fringe factors. A handful of outlying scores that occur at the tail-end of an asymmetric disribution can strongly influence the distribution's mean. The fact that the median, a combiner that is more resistant to outliers and skewed distributions, outperforms the mean and the baseline is evidence that outliers are detrimentally affecting the mean.

Figure 1 shows sample score distributions for attack and normal packets. It is clear that most packets, particularly the attacks and false alarms, have skewed score distributions that make the mean unreliable. It is also interesting that the profile for the normal packets varies a great deal.

We also note that the graphs support the observation that anomaly detection is difficult: there is virtually no difference between some of the attacks and some of the false alarms. This suggests that the features we use do not sufficiently capture the difference between anomalous and normal data.

If the problem with averaging the scores is outliers, as it appears to be, then using the n^{th} percentile score is a way to ignore different percentages of outliers. Figure 3 plots the detection rate as a function of which percentile score is chosen to represent the cluster ensemble. For example, taking percentiles 0.0, 0.5, and 1.0 correspond to using the minimum score, the median score, and the maximum score, respectively. Performance is shown for different false alarm thresholds.

For low false alarm thresholds, the best percentile is either 0.4 or 0.5. At higher thresholds the percentile used matters less, with the caveat that 1.0 always per-

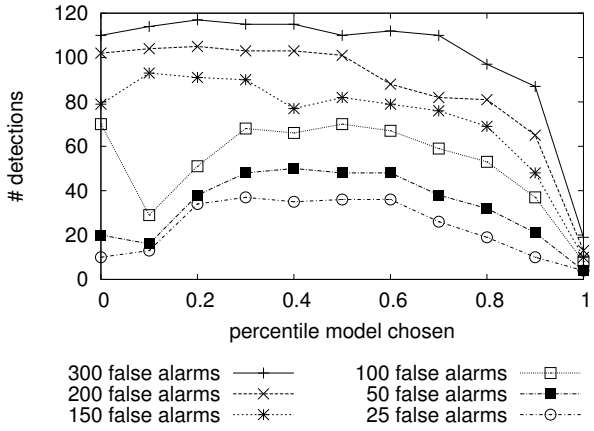

Figure 3. Attack detections as a function of percentile model chosen.

forms poorly. The graph supports our hypothesis that extreme scores, especially at the high end, are unreliable and noisy.

More generally, these results raise the question of whether supervised ensemble learning methods, such as bagging (Breiman, 1996), can yield better results by combining model predictions using the median instead of the mean. While it could be difficult to change merging techniques from one test point to another, always using the median could be beneficial for certain data sets. This seems particularly likely if the data set exhibits the same asymmetric prediction distributions as our work has found for network anomaly detection.

6. Conclusions

We investigate using cluster ensembles for anomaly detection, reasoning that multiple views of the data can better highlight atypical points. To measure abnormality, we define fringe factor as a score of how much a point is on the fringe of its assigned cluster. Our experiments test merging the fringe factor scores from each clustering using the mean, the median score, the minimum score, and the maximum.

We find that the interesting data—attacks and false alarms—have a skewed distribution of scores. As a result, the cluster ensemble that merges predictions using the mean performs worse than the expected single clustering performance—unless many false alarms are allowed. Taking the median, however, reliably improves upon single clustering performance. Using the minimum performs almost identically to the mean, while using the maximum yields very bad performance.

In the future we hope to apply ensemble clustering to other novelty detection problems. Another interesting question is whether supervised ensemble learning methods can benefit from using the median to merge predictions instead of the mean, and under what circumstances.

Acknowledgments

We thank Alexandru Niculescu-Mizil and Mohamed Elhawary for helpful discussions and the anonymous NESCAI reviewers for insightful comments that will influence our future work in this area. This work was supported by AFOSR/AFRL Grant through the AFRL/Cornell Information Assurance Institute, by NSF CAREER grant 0347318, and by a Cornell University Fellowship.

References

Ayad, H., & Kamel, M. (2003). Finding natural clusters using multi-clusterer combiner based on shared nearest neighbors. *Multiple Classifier Systems* (pp. 166–175).

Barbará, D., Wu, N., & Jajodia, S. (2001). Detecting novel network intrusions using bayes estimators. *Proc. of SIAM Intl. Conf. Data Mining*.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Chan, P., Mahoney, M., & Arshad, M. (2003). *A machine learning approach to anomaly detection* (Technical Report). Florida Institute of Technology.

Dasgupta, D., & González, F. (2002). An immunity-based technique to characterize intrusions in computer networks. *IEEE Trans. Evol. Comput.*, *6*, 1081–1088.

Dudoit, S., & Fridlyand, J. (2003). Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, *19*, 1090–1099.

Eskin, E., Arnold, A., Prerau, M., Portnoy, L., & Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *Data Mining for Security Applications*. Kluwer.

Fred, A. L., & Jain, A. K. (2002). Data clustering using evidence accumulation. *Proc. of the 16th Intl. Conf. on Pattern Recognition (ICPR '02)*. IEEE Computer Society.

Ghosh, A., & Schwartzbard, A. (1999). A study in using neural networks for anomaly and misuse detection. *Proc. of the 8th USENIX Security Symposium*.

Hadjitodorov, S. T., Kuncheva, L. I., & Todorova, L. P. (2005). Moderate diversity for better cluster ensembles. *Information Fusion*.

Lee, W., Stolfo, S. J., Chan, P. K., Eskin, E., Fan, W., Miller, M., Hershkop, S., & Zhang, J. (2001). Real time data mining-based intrusion detection. *Proc. Second DARPA Information Survivability Conference and Exposition* (pp. 85–100).

Leisch, F. (1999). *Bagged clustering* (Technical Report 51). SFB "Adaptive Information Systems and Modeling in Economics and Management Science".

Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., & Das, K. (2000). The 1999 DARPA off-line intrusion detection evaluation. *Comput. Networks*, *34*, 579–595.

Mahoney, M., & Chan, P. (2001). *PHAD: Packet header anomaly detection for identifying hostile network traffic* (Technical Report). Florida Institute of Technology.

Mahoney, M., & Chan, P. (2003). *Learning rules for anomaly detection of hostile network traffic* (Technical Report). Florida Institute of Technology.

Mahoney, M. V. (2003). Network traffic anomaly detection based on packet bytes. *Proc. of the 2003 ACM Symposium on Applied Computing* (pp. 346–350). Melbourne, Florida: ACM Press.

Pelleg, D., & Moore, A. (1999). Accelerating exact k-means algorithms with geometric reasoning. *Proc. of the 5th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining* (pp. 277–281). San Diego, California, United States: ACM Press.

Sekar, R., Bendre, M., Dhurjati, D., & Bollineni, P. (2001). A fast automaton-based method for detecting anomalous program behaviors. *Proc. IEEE Symposium Security and Privacy* (pp. 144–155).

Sekar, R., Gupta, A., Frullo, J., Shanbhag, T., Tiwari, A., Yang, H., & Zhou, S. (2002). Specification-based anomaly detection: A new approach for detecting network intrusions. *ACM Conf. on Computer and Communications Security* (pp. 265–274).

Sequeira, K., & Zaki, M. (2002). ADMIT: Anomaly-based data mining for intrusions. *Proc. of the 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining* (pp. 386–395). Edmonton, Alberta, Canada: ACM Press.

Strehl, A., & Ghosh, J. (2003). Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, *3*, 583–617.

Topchy, A. P., Law, M. H. C., Jain, A. K., & Fred, A. L. N. (2004). Analysis of consensus partition in cluster ensemble. *ICDM* (pp. 225–232).

A. Anomaly Dataset

We evaluate our work using the 1999 DARPA/Lincoln Laboratory Intrusion Detection Evaluation (IDEVAL-1999) data, specifically the `tcpdump` data from a packet sniffer inside the router (the *inside* packet trace). The dataset contains three weeks of training data and two weeks of test data. Weeks two, four, and five contain **denial of service** (DoS) attacks, **probes** (*e.g.* port scans), **remote to local** (R2L) attacks where remote access is elevated to local user privileges, **user to root** (U2R) attacks where local user privileges are elevated to root privileges, and **data** attacks where proprietary data is released accidentally or maliciously by an employee. Since the data sets were created by simulation, the test data does not contain unknown, unlabeled attacks. Scoring during the original evaluation was based on the number of attacks detected while constrained to 10 false alarms per day of training data; a detection needed to be supported by identifying the IP address of the attack target.

A stated goal of the evaluation was to test the ability of systems to detect novel attacks. To this end, several attacks in the test data were not present in the training data. Out of 201 total attacks, 35 were stealthy versions of attacks used in the 1998 Intrusion Detection Evaluation (IDEVAL-1998) and 62 were new attacks developed for the evaluation. The systems participating in the evaluation detected, on average, 72% of the old attacks but only 19% of the new/stealthy attacks. Lippmann *et al.*'s report of the evaluation (2000) lists 21 attack types (out of a 56 total attack types) that were poorly detected by all systems *regardless of the false alarm rate*.

While there are 201 attacks present in IDEVAL-1999's test data, not all of these attacks are detectable from the inside packet trace dataset. As a result, much of the research using these data sets calculates the percentage detected out of those attacks that could conceivably be detected. For example, the inside network trace only contains evidence for about 177 attack events. Since each work tends to define the detectable attacks slightly differently (*e.g.* based on whether or not network traffic outside the router was examined some attacks were or were not detectable; some systems only targeted certain attack types; *etc.*) comparisons based on percentages of attacks detected are

not straightforward. Often the number of attack detections, when 10 false alarms per day are allowed, are reported. This works out to 100 false alarms for the two week test period. (Data was not provided for Saturdays or Sundays.)

The IDEVAL-1999 data is not labeled in the traditional sense. Based on experience with the previous evaluation (IDEVAL-1998), the organizers felt that labeling every single packet as attack / non-attack was too time consuming, difficult, and error prone. Instead, IDEVAL-1999 comes with a list of attacks, the times at which they occur, and the IP addresses involved (source and destination). To allow for small inconsistencies in host clocks, any alarm with the correct destination IP address and an attack time that falls within plus or minus 60 seconds of the attack time is considered a positive detection. An important consequence of this evaluation scheme is that *normal packet traffic occurring within an attack's window and headed to the attack's target is effectively labeled as anomalous.* Short of manually examining packets, there is no way to know how many normal packets fall within these windows. Nonetheless, we use IDEVAL-1999 since it is the best available data set.

Since many attacks are identifiable—rightly or wrongly—by multiple packets, multiple positive alarms within the window are treated as one positive detection to make the detection counts understandable. False alarms are not accorded the same privilege. As a result false alarms accumulate very quickly. Most research using this dataset compensates by limiting the rate of alarms. We follow Mahoney (2003) in filtering the alarms emitted by the system to keep only the highest scored (most anomalous) alarm for each minute. In our experience this substantially improves the system's detection rate.

A.1. Feature Representations

Following Mahoney (2003), we use the first 48 bytes of the packet, starting with the IP header, for the training features. Each feature corresponds to one byte. For TCP data the features include the IP header, the TCP header, and a few bytes of the data payload. For simplicity we have chosen to not interpret the header bytes. For example, no effort is made to treat bytes 12-15 (the source IP address) of TCP data as a single number or to combine the two bytes comprising the destination port into a single number. This saves us from actively interpreting different protocols when processing data (*e.g.* web traffic, email, DNS updates, *etc.*). Such higher level interpretation would be desirable in a more complete implementation. Mahoney achieves good performance using the same uninterpreted features and a feature that approximated the passage of time.[4]

Logically, these byte-based feature vectors seem like a poor fit with a Euclidean distance measure. To see this, note that packet fields are almost universally discrete. Considering only the destination port, port 79 is logically no closer to port 80 than port 23 or port 50,000 (ignoring reserved port designations). The Euclidean distance metric, however, treats these values as continuous and places undue emphasis on numeric separation.

To address this we transform the feature values into a frequency space using the technique Chan *et al.* employ for CLAD (2003), which we motivate below. To make the values continuous, a given value v_f for a feature f is replaced by the frequency with which that value occurs for that feature in the training data. Intuitively this can be seen as estimating the probability of v_f occurring based on the distribution of values observed during training. (The frequency values could be normalized by the number of training instances to get a real probability; we have not done so since we do not require the values to be in the range [0,1].)

Within the frequency space of the transformed values, every feature will necessarily have a power law distribution with larger values (those with high frequency before transformation) occurring much more frequently than other values. The distance between two instances could potentially be dominated by one feature with very different values. We mitigate this effect by using a log scale in the frequency space.

Even using the log scale, there is still the problem that between features the range of observed values is different. For example, feature one may range from 0 to 10, while feature two ranges from 0 to 5. As a result, feature one is implicitly given more weight than feature two by Euclidean distance since larger distances are possible in feature one than in feature two. We remove this implicit weighting by normalizing a feature's

[4] Mahoney and Chan (2001; 2003) discovered a simulation artifact in the IDEVAL-1999 data set where the time-to-live (TTL) field of packets during training was limited to only 8 values. In contrast, 80 distinct values were observed by the authors during a 9 hour period on their department web server. Consequently the values of 126 and 253 that only occur in the IDEVAL-1999 test data allow PHAD (Mahoney & Chan, 2001) to detect 33 attacks (19 different types) and cause 20 false alarms. This indicates that unless the TTL field is ignored by anomaly detection systems, performance may be artificially inflated on the IDEVAL-1999 data set. Following them we zero out the TTL field in the data.

values by the range of the feature.

Formally we can express the transformation T as,

$$T(v_f) = RangeNormalize(\ln(\lfloor v_f \rfloor + 1), f)$$

where

$$RangeNormalize(l, f) = (l - min_f)/(max_f - min_f)$$

A.2. Filtering

Following Mahoney (2003) we have filtered the packet stream to reduce the data size and remove noise. Our filtering is closely based on Mahoney's, but differs in that we do not rate limit new TCP connections to 16 per 60 seconds and we do not remove UDP packets to ports > 1023. In more detail,

1. Non-IP packets (address resolution protocol (ARP), hub test, *etc.*) are removed. This focuses our modeling on the most likely attack space. Ideally non-IP traffic would be modeled in a separate clustering. For evaluation purposes this is not possible since identifying attacks requires giving the IP address of the attack target, and this information is not available from non-IP packets alone. The filter actually removes all IP packets that are not TCP or user datagram protocol (UDP) traffic as well. In practice though the only IP packets in the data are TCP and UDP packets.
2. Outgoing traffic is removed. This choice was made simply to limit the size of the data. Most of the attacks are not in the outgoing traffic at any rate.
3. TCP packets that do not open new connections are removed. This includes acknowledgment packets (ACK's), goodbye packets (FIN's), and packets containing data after the first 100 bytes (determined using the sequence number). Connections are hashed to a 4K hash table using their addresses and ports (both source and destination). The rationale is to try and model TCP data as a series of connections and not as discrete packets. While tracking the normalcy of the data payloads can be very powerful, it is very hard to generalize to a small model. Furthermore, a significant portion of today's traffic is encrypted, making the data segment of packets useless.
4. All remaining packets are truncated to 48 bytes, starting at the beginning of the IP packet (*i.e.* the Ethernet header is discarded).

Filtering keeps 241,173 of 10,814,374 packets for week 3 and reduces the size of the data files from 2.06 GB to 11 MB. Reductions for the other weeks are similar.

The obvious concern with filtering is that interesting data—attacks—might be removed. To determine the maximum number of detectable attacks remaining in the data, we output an alarm for every packet in the test data and find that 157/177 attacks are detected. Thus, filtering only removes evidence for 20 attacks.