Feedback-Guided Anomaly Discovery via Online Optimization

Md Amran Siddiqui
Oregon State University
Corvallis, OR 97331, USA
siddiqmd@eecs.oregonstate.edu

Alan Fern
Oregon State University
Corvallis, OR 97331, USA
afern@eecs.oregonstate.edu

Thomas G. Dietterich
Oregon State University
Corvallis, OR 97331, USA
tgd@eecs.oregonstate.edu

Ryan Wright
Galois, Inc.
Portland, OR 97204, USA
ryan@galois.com

Alec Theriault
Galois, Inc.
Portland, OR 97204, USA
atheriault@galois.com

David W. Archer
Galois, Inc.
Portland, OR 97204, USA
dwa@galois.com

ABSTRACT

Anomaly detectors are often used to produce a ranked list of statistical anomalies, which are examined by human analysts in order to extract the actual anomalies of interest. This can be exceedingly difficult and time consuming when most high-ranking anomalies are false positives and not interesting from an application perspective. In this paper, we study how to reduce the analyst's effort by incorporating their feedback about whether the anomalies they investigate are of interest or not. In particular, the feedback will be used to adjust the anomaly ranking after every analyst interaction, ideally moving anomalies of interest closer to the top. Our main contribution is to formulate this problem within the framework of online convex optimization, which yields an efficient and extremely simple approach to incorporating feedback compared to the prior state-of-the-art. We instantiate this approach for the powerful class of tree-based anomaly detectors and conduct experiments on a range of benchmark datasets. The results demonstrate the utility of incorporating feedback and advantages of our approach over the state-of-the-art. In addition, we present results on a significant cybersecurity application where the goal is to detect red-team attacks in real system audit data. We show that our approach for incorporating feedback is able to significantly reduce the time required to identify malicious system entities across multiple attacks on multiple operating systems.

CCS CONCEPTS

• **Computing methodologies → Anomaly detection**; **Online learning settings**;

KEYWORDS

Anomaly detection; anomaly detection feedback; anomaly detection on security; online convex optimization; feedback in linear model.

ACM Reference Format:
Md Amran Siddiqui, Alan Fern, Thomas G. Dietterich, Ryan Wright, Alec Theriault, and David W. Archer. 2018. Feedback-Guided Anomaly Discovery via Online Optimization. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220083

1 INTRODUCTION

General-purpose anomaly detectors typically operate by identifying and ranking statistical outliers in the data. Different detectors make different choices for the statistics to monitor and combine, which can result in very different anomaly rankings for the same data set. Thus, the utility of a particular detector for an application depends on how well its ranking aligns with what is interesting from an application perspective. For example, in a computer security application, the password file may appear anomalous compared to other system files based on access-pattern statistics. However, an analyst would understand the reason for this statistical deviation and would not be interested in that particular anomaly by itself. In general, the gap between statistical outliers and the semantics of an application can result in high false-positive rates and easily render an anomaly detector unusable.

One way to reduce false-positive rate is to build domain knowledge into a detector. For example, a designer might apply domain expertise to select statistics that are more likely to produce interesting anomalies and/or filter anomalies based on semantically defined white lists. Unfortunately, this requires significant expertise about both the application and anomaly detection. There is also a risk that important anomalies will be missed due to bias introduced by the designer.

In this paper, we consider an alternative approach to reduce false positives based on incorporating feedback from an end-user analyst. In traditional anomaly detection (without feedback), an analyst is presented with a ranked list of anomalies and then investigates anomalies in that order until time is up. If anomalies of interest are low in this list, then significant effort will be required to find them. Rather, in our setting of feedback-guided anomaly discovery, after investigating the currently top-ranked instance, the analyst provides feedback about whether the instance was of interest or not. This feedback is used by the detector to adjust the anomaly scores with the goal of moving anomalies of interest higher in the ordering. This approach has the advantage of being customizable by the end-user analyst with little overhead on the analyst's time.

The primary question is whether learning can be made efficient and accurate enough to demonstrate a benefit in practice. In this paper, we demonstrate that significant benefits are possible on benchmark data and on a large-scale computer security application.

The main contribution of this paper is to formulate feedback-guided anomaly detection within the framework of online convex optimization and derive simple, efficient, and effective algorithms. This is done by associating a convex loss function to each feedback response, which rewards the anomaly score for aligning with the feedback. We consider two different methods, which differ in their choice of loss function. By leveraging the well-studied area of online convex optimization, our approaches inherit the simplicity and efficiency of the online algorithms as well as the solid theoretical grounding, where convergence is understood. Prior state-of-the-art approaches, such as AAD [1], are significantly more complex to implement, incur much higher computational complexity, and have less clear theoretical underpinnings. In addition, prior approaches incorporate a significant number of parameters, whereas, our approaches have a single learning rate parameter, for which we show that a single value works well across all of our experiments.

While our feedback approach can be applied quite generally, in this paper we focus on the popular class of tree-based anomaly detectors, which includes the state-of-the-art Isolation Forest detector [13], among others. At a high-level, detectors in this class are distinguished based on how they "assign weights" to edges in their trees. Our feedback approach is able to automatically span this space, covering existing and new tree-based detectors, by tuning the weights in response to feedback. We conduct experiments on individual benchmark problems from prior work and on six large anomaly detection benchmark sets. The results show that our online approaches are able to significantly accelerate the rate that an analyst can find anomalies compared to not using feedback. We also show significant improvements over the prior state-of-the-art.

We also demonstrate our approach on real cybersecurity attack data collected from a recent DARPA red team exercise, which contains multiple attacks on multiple operating systems, over multiple days. We focus on the problem of detecting anomalous system entities, with the goal of quickly identifying the malicious entities that are part of an attack. The malicious entities are extremely rare in the data, which yields a very difficult anomaly detection problem. Our results show that our feedback-based approach allows for malicious system entities to be discovered significantly earlier than when no feedback is used by the system.

2 ANOMALY DETECTION PROBLEM SETUP

We consider anomaly detection problems defined over a set of m data instances $D = \{x_1, \ldots, x_m\}$. Often the data instances are described as real-valued vectors, but our approach is agnostic to the instance representation. We will assume, however, that the associated anomaly detector for the instances has a particular generalized linear form (see Section 4). There is an unknown partition of D into a set of *nominal instances* N and a set of *alien instances* A, so that $D = N \cup A$. Generally the nominal instances account for an overwhelming fraction of the data, that is, $|A| \ll |N|$, which is one of the key challenges in anomaly detection. We refer to instances in A as aliens, rather than just anomalies, to clarify that those are the semantically interesting instances that we wish to discover in the data. In most applications, the aliens are generated by a process distinct from the process generating the nominals, in particular, a process that is important to detect for the application. For example, in our computer security application, the data instances describe the behavior of computer system entities, such as processes, files, and netflows, and the aliens correspond to the malicious entities that are part of an attack.

Since D is typically large, manual search for aliens through all instances is not practical. Anomaly detectors help address this problem by assigning anomaly scores to instances in D to yield an overall anomaly raking, with the goal of having the aliens be higher ranked than the nominals. Given such a ranking, an analyst can inspect instances in decreasing order of anomaly rank to better direct their search for aliens. Of course, anomaly detectors are rarely perfect, and nominals are often ranked above aliens. Thus, the practical utility of an anomaly detector, in this setting, is related to the amount of analyst effort required to uncover alien instances. In our experiments, we will measure how many aliens are discovered per instance investigated and how many instances must be investigated until the first alien is discovered.

The above setting uses a static anomaly ranking throughout the investigation. This fails to take into account information about the result of each analyst investigation. The setting of *feedback-guided anomaly discovery* aims to use that information to improve the anomaly detector during its operation. In particular, on each round, the analyst investigates the currently top-ranked instance in D and labels it as "nominal" or "alien", depending on whether it is of interest or not. The labeled instance is then given to the anomaly detector as feedback, which is used to possibly adjust the anomaly ranking. The process of investigating the top ranked instance, giving feedback to the detector, and adjusting the ranking continues until the analyst resources are exhausted.

This setting is related to active learning in the sense that the algorithm is providing the analyst with instances to be labeled. Whereas most traditional active learning work focuses on deciding which instance to query next, we fix that choice in this work (query the top ranked instance) and instead focus on how to best update the anomaly detector based on the feedback. This strategy of querying the top ranked instance makes sense, given our goal of quickly identifying aliens. There is work, however, on related problems that suggests non-myopic query strategies may be beneficial (e.g. [11]).

3 RELATED WORK

The general problems of active learning [18] is closely related to our work. Active learning mainly focuses on finding a query strategy to select instances that provide the most useful information for learning. In this work, rather, we use a simple greedy instance selection mechanism and place our emphasis on how to best adapt the anomaly detector in response to feedback. It is possible that more complex active learning query strategies may also benefit our setting in future work.

Rare-category detection [14] is another closely related problem. The goal of rare-category detection is to uncover at least one instance from each of some set of unknown categories as quickly as possible. While related to our problem of feedback-guided anomaly discovery, our focus is to uncover as many instances from the alien class as possible using the minimal number of queries to the analyst.

There are several other efforts that have considered incorporating feedback into a sequential setting for anomaly detection [16, 23], where data arrives in an online fashion and must be queried when it arrives or be ignored. Rather, in our work, we focus on the batch anomaly detection setting, where we have a large set of data that is being analyzed in order to find the aliens as efficiently as possible.

The most closely related prior work is the Active Anomaly Discovery (AAD) algorithm [1, 2], where the same setting for incorporating feedback into anomaly detection is studied. In [1] AAD was introduced as a way to incorporate feedback into a the LODA anomaly detector [15]. LODA forms an ensemble of component anomaly detectors based on random linear projections, which are combined via uniform weights. At each step, the AAD approach [1] defines and solves an optimization problem based on all prior analyst feedback, which results in new weights for the components. This optimization approach was later applied to adjust the weights of the Isolation Forest anomaly detector [2]. One disadvantage of this approach is that the optimization problem involves a number of components that each have hyper-parameters associated with them. This makes it difficult to develop a formal characterization of the overall objective being optimized and requires careful selection of the hyperparameters. In addition, the optimization problem is non-convex and requires a relatively complex "alternating" convex optimization approach with its own parameters. Rather, one of our main motivations is to do as well or better than AAD using a much simpler and efficient approach with a minimal number of parameters. In addition, our approach is built on top of the well-established area of online convex optimization, which allows it to inherit the theoretical characterizations developed by that community.

Some additional efforts on incorporating feedback into anomaly detection include: SSAD (Semi-Supervised Anomaly Detection) [8] and AI2 [24]. SSAD incorporates knowledge of labeled instances within a semi-supervised learning setting and AI2 is an ensemble of unsupervised anomaly detectors with a supervised learner. Another supervised method called ATGP [9] uses some labeled data to estimate a convex combination of a set of anomaly detectors. In [1], it was shown that AAD is able to significantly outperform these methods, establishing AAD as the state-of-the-art, which we compare to in this paper.

Since we employ our feedback approach on cybersecurity data, we also mention closely related work in security. Prior work has studied anomaly detection for host-based intrusion detection [3, 6, 7, 17, 20]. The major focus of existing work is on learning normal behavior from sequences of system calls or execution control flow. These approaches show promise but are prone to high false positive rate, which hinders their use. Our approach is motivated by recognizing the difficulty of achieving low false positive rates using fixed anomaly detection schemes across a wide range of host systems. By adapting the detection system based on analyst feedback, we aim to provide a more robust system that can quickly be tuned to be effective on any given host system.

4 INCORPORATING FEEDBACK VIA ONLINE CONVEX OPTIMIZATION

In this section, we first overview the online convex optimization (OCO) framework and our corresponding formulations of feedback-guided anomaly discovery. Next, we describe the OCO algorithms used in this work, their properties, and some implementation choices.

4.1 Online Convex Optimization

Many classic and new online learning algorithms can be described and analyzed within the OCO framework. A good introduction to OCO can be found in [19]. Here we describe only the elements essential for our work.

OCO is formulated as an iterative game against a potentially adversarial environment where our moves are vectors from a convex set $\mathcal{S}$. At discrete time steps t the game proceeds as follows:

(1) We select a vector $w_t \in \mathcal{S}$.
(2) The environment selects a convex function $f_t : \mathcal{S} \to \mathcal{R}$.
(3) We suffer a loss $f_t(w_t)$.

Roughly speaking, the goal is to select a sequence of vectors with small accumulated loss over time. Because there are no restrictions on the sequence of convex functions and because we must select w_t before seeing f_t, the total loss over T steps can be arbitrarily large. Thus, the objective of OCO is typically stated in terms of the regret with respect to the best vector w^* in hindsight. More formally, given a T-step game episode where we play $(w_1, w_2, \ldots, w_T)$ against $(f_1, f_2, \ldots, f_T)$ the total T *step regret* is equal to

$$\text{Regret}_T = \sum_{t=1}^{T} f_t(w_t) - \min_{w^* \in \mathcal{S}} \sum_{t=1}^{T} f_t(w^*), \tag{1}$$

which is our accumulated loss minus the minimum loss possible on the sequence of functions using any fixed vector in $\mathcal{S}$. A fundamental goal of OCO algorithms is to achieve worst-case regret that grows sub-linearly in T, since the time-averaged regret then goes to zero. This "no regret" property is powerful, since it indicates that an algorithm is competitive with the best possible solution in hindsight (for large enough T) even without the advantage of hindsight. As an example, the family of Follow-the-Regularized-Leader algorithms achieves a regret that grows as $O\left(\sqrt{T}\right)$ under fairly general conditions.

4.2 Loss Functions for Query-Guided Anomaly Discovery

Query-guided anomaly discovery can also be viewed as a game where on each round we output an anomaly ranking over the data instances and we get feedback on the top-ranked instance. We wish to minimize the number of times we receive "nominal" as the feedback response. To put this problem in the OCO framework, we first need to place some reasonable restrictions on the form of the anomaly detectors that we will consider. We study the family of *generalized linear anomaly detectors (GLADs)*, which are defined by a feature function $\phi : D \mapsto \mathcal{R}^n$, which maps data instances to n-dimensional vectors and an n-dimensional weight vector w. In this work, the *anomaly score* assigned to an instance x is defined to be $\text{SCORE}(x; w) = -w \cdot \phi(x)$, with larger scores corresponding to more anomalous instances.[1] Our algorithm will adapt an anomaly detector by adjusting the weight vector w in response to feedback.

We will see in Section 5 that tree-based anomaly detectors are easily modeled as GLADs. Many other types of detectors can also be parameterized in various ways as GLADs to support adaptation. For example, prior work on query-guided anomaly discovery [1] parameterized the LODA anomaly detector [15] as a GLAD by assigning a weight to each of its local models. As another example, given any collection of anomaly detectors, we can form an ensemble by using their scores as the components of ϕ and weighting their relative influences via w. Depending on the type of detector, it will sometimes be natural to place constraints on w to capture prior expectations. For example, in our tree-based models, it will be natural to place non-negativity constraints on the weights. As long as the constraint sets are convex (such as non-negativity) and there is an associated projection operator (see below), our algorithm can easily incorporate such constraints.

Given a GLAD parameterization of an anomaly detector, we can now connect query-guided anomaly discovery to OCO. On each feedback round we select a vector w_t for the detector, which specifies an anomaly ranking over instances. Rather than receiving a convex function f_t from the analyst, we receive feedback y_t on the top-ranked instance, where $y_t = +1$ if the instance is alien and $y_t = -1$ if it is nominal. The question is how to translate that feedback y_t into a convex function for the OCO framework that will result in effective use of the feedback. Intuitively, if $y_t = +1$ then our current cost vector should suffer a smaller loss than if $y_t = -1$, since the ranking based on w_t correctly put an alien at the top. Further, if $y_t = -1$ then it may also be sensible to have the loss suffered increase with the "confidence" the detector has in the top ranked instance being anomalous. Below we give two simple loss functions based on these principles.

Linear Loss. Let x_t be the top-ranked instance in D under the ranking given by w_t. The linear loss is given by

$$f_t(w_t) = -y_t \text{SCORE}(x_t; w_t) = y_t w_t \cdot \phi(x_t), \quad (2)$$

which is a linear (and hence convex) function of w_t. This function is similar to margin-based losses used for classification problems. When we receive alien (nominal) feedback, this loss decreases with increasing (decreasing) anomaly score as desired.

Log-Likelihood Loss. In linear classification it is common to define loss functions in terms of a probabilistic semantics. Our next cost function follows this approach by using the anomaly score SCORE$(x; w)$ to define a discrete probability distribution P_a over D. We will interpret $P_a(x; w)$ as the *alien distribution* from which alien instances are drawn when aliens occur. We use the following log-linear form for P_a:

$$P_a(x; w) = \exp\left(\text{SCORE}(x; w)\right)/Z = \exp\left(-w \cdot \phi(x)\right)/Z, \quad (3)$$

where Z is the normalizing constant $Z = \sum_{x' \in D} \exp\left(-w \cdot \phi(x')\right)$. Thus, P_a assigns higher probabilities to instances with higher anomaly scores. The corresponding loss function in the OCO framework is the signed log-likelihood of the top-ranked instance x_t:

$$f_t(w_t) = -y_t \log\left(P_a(x_t; w_t)\right). \quad (4)$$

[1]We could also define the score without the negative sign. However, this definition is more natural for the class of tree-based anomaly detectors described in Section 5.

The function f_t is a convex function of w_t over any convex vector set. For alien feedback ($y_t = +1$), the loss becomes positive and becomes smaller when $P_a(x_t)$ is larger, as desired. The situation is reversed for nominal feedback. An algorithm that achieves low regret with respect to this choice of loss function will tend to select weight vectors that assign lower probabilities to nominals and higher probabilities to anomalies. Compared to the linear loss, this loss function incorporates information from all of D via the normalization term Z. While our experiments show that this appears to sometimes improve the anomaly detection performance, it also incurs computational overhead. If computing the full normalization term is too expensive for an application, one can use standard techniques such as sampling to approximate Z [12].

Logistic Loss. We have also considered using logistic loss commonly employed for binary logistic regression classifiers. This loss is qualitatively similar to the linear loss, and in our experiments it never outperformed linear loss. Thus, we do not define it here or include it in our experiments.

We note that the above loss functions are proxies for the true objective of discovering alien instances as quickly as possible. Any OCO theoretical guarantees concerning regret are with respect to the proxies and not necessarily the true objective. If there is a weight vector w^* that tends to rank aliens above nominals, then achieving a low regret with respect to the proxies will tend to assign higher anomaly scores to aliens and lower scores to nominals as desired. It is reasonable to expect that this will translate to an improved rate of alien discovery. This is not guaranteed, however, and it is possible that within a practical number of rounds, low losses are being achieved but aliens are never ranked at the top of the ordering. Note, however, that our strategy for selecting the instance x_t for the analyst is designed to result in maximum loss when nominal feedback is received. That is, x_t is selected to have the highest anomaly score and hence is the instance that w_t is currently most confident about being an alien. In practice, we have found that applying OCO to both objectives using this strategy for selecting x_t is quite effective.

4.3 The Mirror Descent Learning Algorithm

A general algorithm for OCO is *Follow-the-Regularized-Leader (FoReL)*. Given a regularization function $R : \mathcal{S} \mapsto \mathcal{R}^+$ over vectors, at step t, the algorithm plays the vector

$$w_t = \arg\min_{w \in \mathcal{S}} \sum_{j=1}^{t-1} f_j(w) + \frac{1}{\eta} R(w), \quad (5)$$

which is the vector that would have minimized the accumulated loss over the previous steps while also considering the "regularization loss". The regularizer R is often chosen to be the L_2 norm, and η determines the influence of regularization. Given an appropriate choice for η (see below) and under fairly general conditions, this algorithm with L_2 regularization yields a sublinear regret of $O\left(\sqrt{T}\right)$. However, the algorithm can be computationally expensive, since each step solves an optimization problem that grows over time.

Algorithm 1 Online Mirror Descent for Query-Guided Anomaly Discovery

```
Require: Data Instances $D$; Generalized-Linear Anomaly Detector
    with features $\phi$ and weight parameters $w \in \mathcal{S}$; regularization
    parameter $\eta > 0$; weight prior $w_0$
  $\theta_1 = w_0$
  for $t = 1, 2, 3, \ldots$ do
    $w_t = \arg\min_{w \in \mathcal{S}} \|w - \theta_t\|_2$
    $x_t = \arg\max_{x \in D} -w_t \cdot \phi(x)$ ;; max anomaly score
    Get feedback $y_t \in \{-1, 1\}$ from analyst on $x_t$
    $D = D - \{x_t\}$
    Use $y_t$ to define loss function $f_t$ based on either Eq. 2 or Eq. 4
    $\theta_{t+1} = \theta_t - \eta \partial f_t(w_t)$ ;; subtract $\eta$-weighted sub-gradient
  end for
```

Fortunately, there are many choices of R for which a completely online version of FoReL can be derived, which achieves the same theoretical guarantees.[2] In particular, we use a variant of *Online Mirror Descent (OMD)* [19], which corresponds to FoReL using the regularizer $R(w) = \|w - w_0\|_2^2$, where w_0 is a reference or prior vector. Intuitively, if we have prior knowledge that w_0 is a reasonable starting point for a solution, then using this regularizer will encourage solutions to stay close to w_0 unless the data suggests otherwise.

Algorithm 1 gives pseudo-code for our OMD algorithm specialized for query-guided anomaly discovery. Each iteration starts by selecting a weight vector w_t for the anomaly detector (more on this below), which is used to select the top-ranked anomalous instance x_t remaining in D. The feedback y_t is then employed to define the loss function f_t, which is applied to update the weight vector. In order to compute the sequence of weight vectors, the algorithm uses θ_t to accumulate $-\partial f_j(w_j)$ across iterations, which is a negative sub-gradient of the observed loss function f_j at the selected vector w_j. Our functions f_j are continuous, so this is just the accumulation of gradients. The next weight vector w_t for the next round is selected as the closest vector in $\mathcal{S}$ to θ_t with respect to the L_2 norm. In our instantiation for tree-based anomaly detectors (Section 5), the set $\mathcal{S}$ includes only non-negative weights. In this case, the projection operation simply returns θ_t, except that negative vector components are set to zero. A special case of this algorithm is when $\mathcal{S} = \mathcal{R}^n$, which yields the standard unconstrained online gradient descent algorithm with learning rate η.

The regularization parameter η plays the role of a learning rate. In theory, if we know the number of rounds T to be played, then a choice of $\eta = \Theta(1/\sqrt{T})$ yields the regret bound $O\left(\sqrt{T}\right)$. If T is unknown, then a similar result can be obtained using the "doubling trick" [19]. Note that the constant factors implicit in the definition of η depend on theoretical quantities of the OCO problem, which are generally not known. In practice, there is a large literature on learning rate selection, including adaptive learning rates, which we could draw on here. However, we found that simply setting $\eta = 1$ worked well throughout our range of experiments. We experimented with various adaptive choices from the literature, for example, $\eta_t = 1/\sqrt{t}$, but did not see significant differences. Our experiments do include an investigation into the sensitivity to learning rate.

[2]The online algorithms are derived by considering a "linearized" version of the original OCO problem and developing online updates for FoReL on that problem. The regret guarantees of FoReL hold for the original problem, since it can be shown that the regret of the linearized problem upper bounds the regret on the original problem.

5 APPLICATION TO TREE-BASED ANOMALY DETECTION

There are a number of state-of-the-art anomaly detectors based on constructing randomized decision trees, where each tree assigns a score to an instance and the scores are combined, usually via averaging [10, 13, 21, 22, 25]. The different algorithms vary in the exact way trees are constructed and the scores they define. However, all of the detectors can be exactly or closely simulated using a single tree-based GLAD model, where different weight settings result in different algorithms. Thus, learning from feedback using this GLAD model can be viewed as attempting to select among the space of tree based algorithms (both known and unknown) based on feedback.

We now describe the *Isolation Forest (IF)* algorithm [13], which is a state-of-the-art algorithm that helps motivate our model. Then we describe the generic tree-based GLAD model.

Isolation Forest. Given a data set D of vector-valued instances, IF analyzes D to construct a forest of randomized decision trees. Trees are constructed recursively. Starting at the root node with the full data set D, a random test is selected by first selecting a feature and then selecting a random threshold for that feature that will separate at least one data instance from the rest. The test is then used to split the data into left and right children and the recursion continues on each child. The tree is built until each leaf node contains a single instance. We can view each node of a tree as defining axis-aligned hyper-rectangular regions determined by the tests from the root to that node. Leaf nodes define regions that contain single data points.

The isolation depth of an instance in a tree is the depth of the leaf that it belongs to. Intuitively, if a data instance is very different from other instances, there is a higher chance that a randomly-selected test will isolate that instance. This suggests that instances that are more anomalous may be expected to have lower isolation depths on average. The IF algorithm assigns an anomaly score to an instance x based on its average isolation depth across the randomized forest. In particular, the score is (a normalized version of) the negative of this average depth.

Tree-Based GLAD Model. After constructing a forest of trees, it is straightforward to define a GLAD model that replicates IF. For each edge e in one of the trees, let $\phi_e(x)$ be a binary feature that is 1 if instance x goes through the edge and 0 otherwise. Note that this is a very sparse set of features and most will be zero for any particular instance. Similarly, associate a weight w_e to each edge. Now let ϕ and w be vectors that concatenate all of the features and weights across the forest in a consistent order, and let $w_e = 1$ for all e. Then the GLAD scoring function $\text{SCORE}(x; w) = -w \cdot \phi(x)$ corresponds exactly to the (unnormalized) IF anomaly score.

If we think of w_e as the cost of traversing edge e, then IF is using the average "cost-of-isolation" across the forest as the anomaly score. A low cost-of-isolation indicates a more anomalous instance. Given this view, it makes sense to consider assigning alternative weights to edges. For example, if we happen to know that the hyper-rectangular region of a node n was not interesting from an application perspective, then we could assign high weights on edges leading to that node. The isolation cost of instances going through that node would then increase and hence lower the anomaly score compared to IF. In practice, we generally will not have such information. However, through the use of analyst feedback we can hope to adjust the weights in ways that match this intuition.

By adjusting the weights it is possible to generate scores that exactly or closely match the scores that would be assigned by many other tree-based algorithms. For example, the average version of RPAD for random forests [21] assigns the normalized frequency as the weight to each edge and hence can be directly captured by our GLAD model. As another example, in the RS-Forest algorithm [25], which is based on randomized trees, only the score (estimated density) from the leaf node are considered from each tree. Hence the corresponding edge weight of the leaf can be set to estimate the log of the density value and the remaining weights set to zero. A similar weight adjustment can be used to arrive at other tree-based algorithms including [22] and [10]. Some algorithms, including some parameter settings of IF, do not grow the trees to full isolation as we do in our experiments, but rather impose a depth limit on trees. In this case, we can replace the cost-of-isolation with the cost of the path from the root to a leaf for a given instance.

Mirror Descent for Tree-Based Models. OMD can be implemented very efficiently for our tree-based models. The trees can store the weights on the edges, so the cost-of-isolation of data point x can be computed using a simple tree traversal by summing the weights as x traverses the edges. The main computation performed by OMD at each step is the gradient of the loss, which is needed to update θ_t. Importantly, the only non-zero components of the gradient vector for our two loss functions will be those corresponding to non-zero components of $\phi(x_t)$, which are those components on edges traversed by x_t. Thus, by storing components of ϕ_t in the appropriate tree edges, we are assured that we only need to update those that are traversed by x_t.

We build our forests exactly as IF with no depth limit, so that all instances are completely isolated in all trees. Since IF is known to be an effective anomaly detector, we select the prior weight vector of OMD to be $w_0 = 1$, which will exactly simulate IF on the first round. We also constrain the weights to be non-negative in this work, which matches the interpretation of the weights as edge costs. We found that allowing for negative weights did not significantly hurt performance, but the non-negativity constraint did show minor improvements in some cases.

6 EMPIRICAL RESULTS

We performed experiments to answer five questions.

- *Question Q1:* how do the OMD algorithms for incorporating feedback compare to the baseline with no feedback on benchmark data?
- *Question Q2:* how do the OMD algorithms compare to the previous state-of-the-art AAD algorithm on benchmark data?
- *Question Q3:* what is the impact of the learning rate parameter on the OMD algorithm?
- *Question Q4:* what is the relative value of feedback on nominals versus feedback on anomalies?
- *Question Q5:* do the OMD algorithms show utility for using feedback on real-world cybersecurity data?

For all of the above questions, our experimental focus is on evaluating the effectiveness of feedback for tree-based anomaly detectors. Due to this focus and space limitations, we do not include experiments with other types of anomaly detectors in this paper. We note, however, that the baseline version of our system, which does not use feedback, corresponds to the Isolation Forest algorithm [13], which has been shown to be among the top performers on a large-scale evaluation [5]. In addition, results in prior work on AAD [2] have shown that the AAD approach, which we compare to here, significantly outperforms the state-of-the-art LODA [15] anomaly detector. For our comparison to AAD, we use the currently available implementation for tree-based detectors [2] with the parameter settings recommended in that work. AAD has been shown to be a state-of-the-art feedback mechanism in prior comparisons using both tree-based methods [2] and the LODA anomaly detector [1].

We used the same randomized forest construction strategy for both our approach and AAD, which is identical to that used by Isolation Forest. All of our experiments used 100 trees and a subsample size of 256 to grow each tree (an Isolation Forest default). All trees were grown until all data instances were isolated. Because the tree construction is randomized, we repeat all of the experiments 10 times and report the average results with confidence intervals. Our primary evaluation metric is to count the average number of alien instances found after each number of feedback rounds.

In order to run the algorithms that incorporate feedback we use simulated analysts to label the top ranked instance at each round as either "nominal" or "alien". In particular, each of the benchmark data sets and the computer security data sets have ground truth labels that are used to provide feedback and for evaluation of the algorithms. The labels are hidden from the anomaly detectors, except those that are revealed as feedback by the simulated analyst.

6.1 Experiments on Benchmark Data

Single Benchmark Results. To address Q1 and Q2, we first show results on six anomaly detection data sets used previously to evaluate AAD Das et al. [2]. Figure 1, shows results of our two OMD[3] methods with Linear and Log-Likelihood loss functions, the state-of-the-art AAD method, and the baseline IF algorithm that does not incorporate feedback. Each graph shows 100 feedback iterations where the top-ranked entity is presented to the simulated analyst and the resulting label is used to update the anomaly detector (or ignored by the pure IF). Each graph illustrates the number of aliens found versus the number of feedback iterations. An ideal result would be a line with slope 1, which indicates that the anomaly detector never shows the analyst a false positive.

For Q1 we observe that with very few exceptions, the algorithms that incorporate feedback are able to improve upon the baseline, sometimes by a substantial factor. In some cases, the rate of anomaly discovery increases by more than 2x. For Q2 both of our OMD approaches improve over AAD on AAN_Thyroid, Mammography, and Yeast. They are about the same as AAD for Covtype and Cardiotocography, and slightly worse on Abalone. This gives evidence that the OMD approaches are highly competitive with the prior state-of-the-art, despite being dramatically simpler to implement,

[3]Codes available: https://github.com/siddiqmd/FeedbackIsolationForest

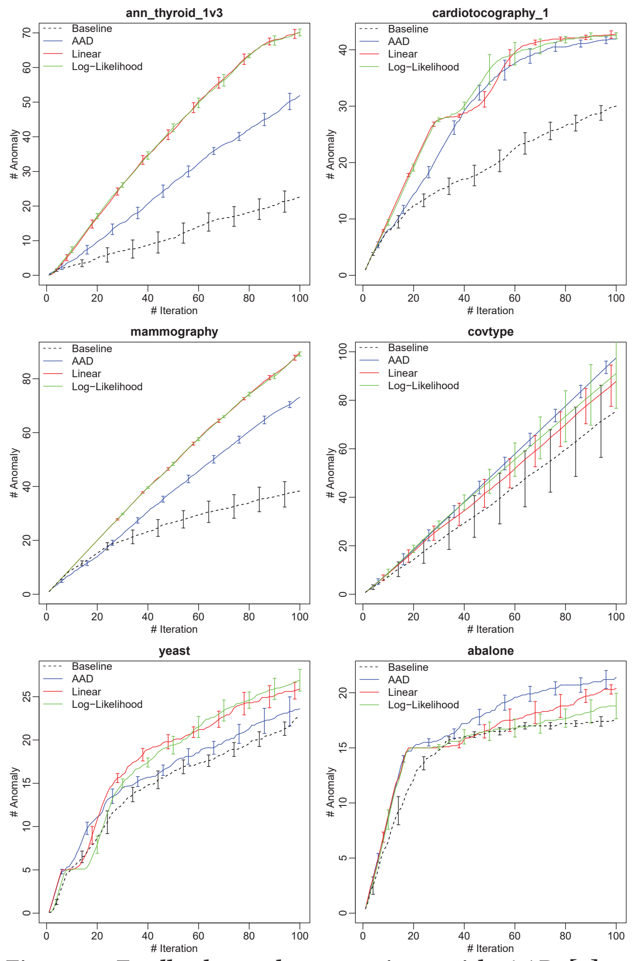

Figure 1: Feedback result comparison with AAD [2] on some standard anomaly detection datasets [2]. 95% confidence interval is also shown for some feedback iterations, others are similar, hence not shown to make the plot clear.

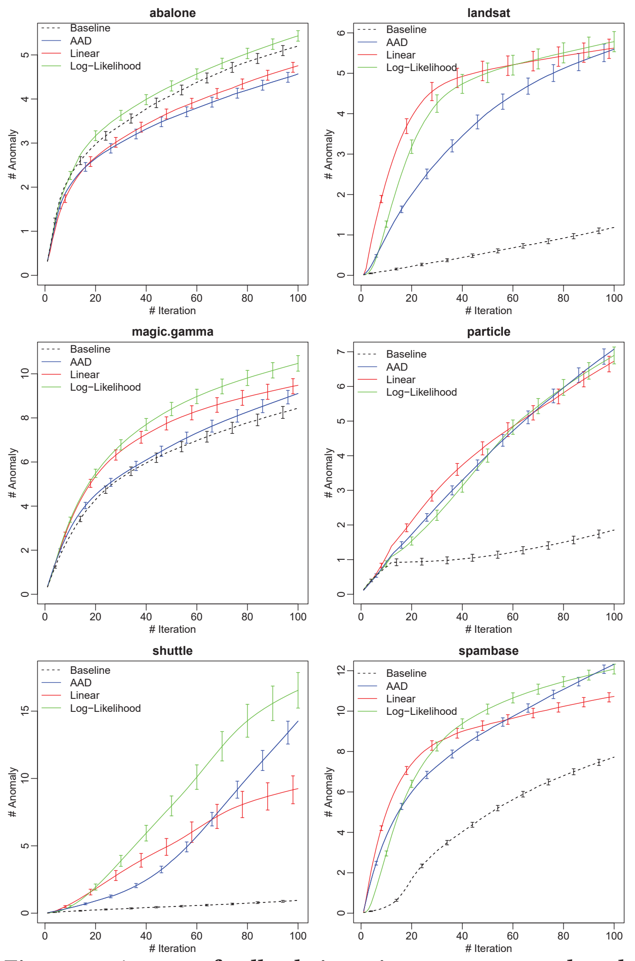

Figure 2: Average feedback iteration curves on 6 benchmark datasets from [4]. Each curve is averaged over 120-300 benchmark datasets.

much more computationally efficient, and having just a single parameter that is held fixed at 1.

Mother Set Benchmark Results. To provide more evidence for Q1 and Q2, we ran experiments on a much larger number of benchmarks from a recent large-scale anomaly detection evaluation Emmott et al. [5]. That study used UCI data sets as "mother sets" for creating large sets of anomaly detection benchmarks with varying characteristics. We selected 6 mother sets and used all benchmarks from each set with an alien rate of 0.01 (120 to 300 benchmarks per mother set). These benchmarks tend to be more challenging than the data sets used above. We ran the same experiments as above for each benchmark and for each mother set report average results across its benchmarks in Figure 2. Runs for each benchmark are repeated 10 times.

For Q1 we see that the feedback methods are able to outperform the baseline on these benchmarks, except for Abolone, where AAD and Linear does slightly worse (decrease in less than 1 discovered anomaly on average). For Q2 overall we see that both OMD methods tend to be competitive with AAD. The Log-Likelihood loss function is as good or better than AAD across all mother sets and better on at least 4 of the 6. The Linear loss function is better on at least 2 of the 6 and the same or incomparable (sometimes better sometimes worse) for the remaining mother sets. These results again suggest that the OMD approaches can significantly improve over the baseline and are competitive with the prior state-of-the-art despite the simplicity.

Sensitivity to Learning Rate. For Q3 we ran experiments with the OMD approaches using a range of fixed learning rates, noting that all previous experiments used a fixed learning rate of 1. Figure 3 shows results for the Log-Likelihood loss for three benchmarks. These results are representative of the learning rate behavior on other benchmarks and for the Linear loss. We varied the learning rate from 0.05 to 10 and also included a commonly used variable learning rate of $\eta_t = 1/sqrt(t)$, where t is the feedback iteration number (labeled as 0 in the plot). We see that all of these learning rates are still able to outperform the baseline IF method. However, the performance drops for learning rates that are on the small or large range. Overall it appears that a learning rate of 1 is a robust and recommended setting.

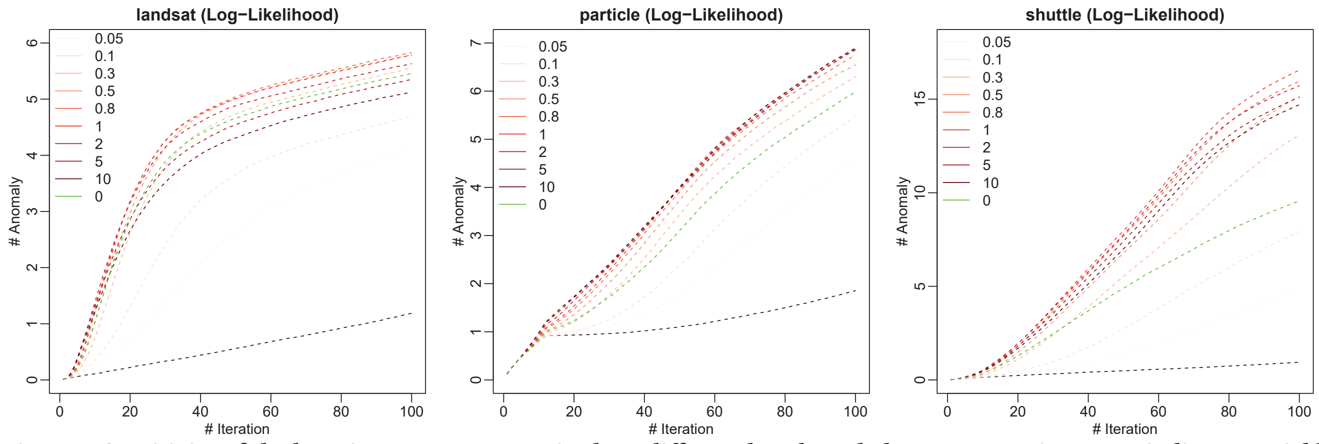

Figure 3: Sensitivity of the learning rate parameter in three different benchmark datasets. Learning rate 0 indicates variable learning rate. The black dotted line is the baseline performance without feedback.

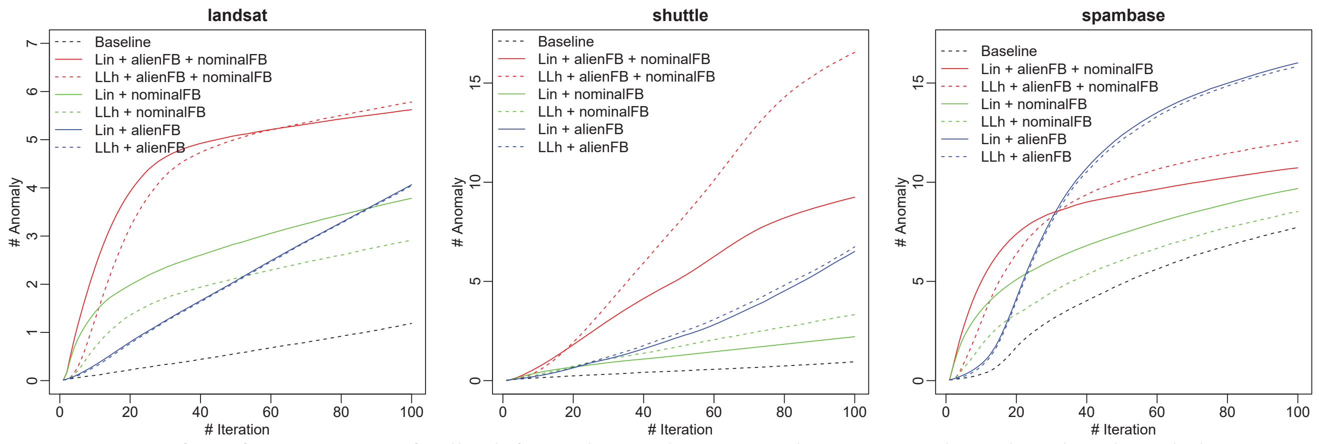

Figure 4: Effect of incorporating feedback from alien and/or nominal instances only in three benchmark datasets.

Influence of Nominal and Alien Feedback. For Q4 we ran experiments where we restricted the types of feedback used for learning. Figure 4 gives results for the Linear and Log-Likelihood loss functions for three cases: (+ nominalFB) where only feedback on nominal instances is given to OMD, (+ alienFB) where only feedback on aliens is given to OMD, (+ alienFB + nominalFB) where all feedback is given to OMD to update the weights. This last configuration is the normal mode of operation. Results are provided for three representative benchmarks.

The aim of this experiment is to help provide some understanding about the relative importance of alien versus nominal feedback and whether this changes across problems. This can give insight into the structure of the anomaly detection problems that is being exploited through the use of feedback. There are several main observations. First, the performance when using just nominal feedback (+ nominalFB) is generally better than using no feedback (baseline) and sometimes significantly worse than using all feedback. Second, using just alien feedback (+ alienFB) can sometimes start out worse than (+ nominalFB) for small numbers of iterations, but eventually catches up and overtake (+ nomialFB). In fact for Spambase using only alien feedback was actually better than using all of the feedback. This was also observed for Abolone (not shown), though not as dramatically. In these cases the alien feedback provides such as useful signal for learning that including nominalFB apparently negatively interferes with that signal.

Overall it is clear that both types of feedback provide significant value and in most cases the combination of the two is better than either alone. The relative value of alien versus nominal feedback, however, depends on the problem and on how much feedback has been acquired. The fact that alien feedback is quite powerful by itself suggests that our methods can effectively exploit cluster structure among aliens. That is, when some aliens are discovered the feedback on those aliens allows for "similar" aliens to be discovered.

Finally, we can also observe that the Linear and Log-Likelihood loss functions lead to very similar performance when given only alien feedback (+ alienFB). In contrast, there is often a more significant performance difference between them for only nominal feedback (+ nominalFB). This indicates that the methods may exploit alien feedback in a similar way, but exploit nominal feedback quite differently. This may be due to the global normalization of the Log-Likelihood model, since if the probability of one instance is decreased due to nominal feedback, then it must increase for some other instance(s). However, we do not see a clear winner between Log-Likelihood and Linear for nominal-only feedback, meaning that the normalization does not necessarily have a positive impact.

6.2 Experiments on Cybersecurity Data

Attack Datasets. To address Q5 we use datasets based on audit logs that were generated during attack campaigns carried out by a red team as part of the DARPA Transparent Computing program. We collected data from three different hosts (Host1, Host2, Host3): two running the FreeBSD server operating system and one running the Ubuntu desktop operating system. Data for two of the hosts (one FreeBSD and one Ubuntu) were collected over a three-day period and from the other over a five-day period. During the data collection period, all hosts were running benign activities resembling normal workloads. On each host, the red team executed an unknown attack campaign, which started at a time that was unknown to us. Our goal is to use anomaly detection to detect "alien" system entities that are involved in the attacks. The data includes events involving three types of entities: processes, files, and netflows.

After the attacks were completed and our results reported to the red team, the red team released a description of each attack scenario, which outlined the key entities and events. We used the descriptions to produce a ground truth encoding of the attacks, where each entity and event was labeled as being part of the attack or not. We use this ground truth to evaluate our approach and also to simulate feedback provided by a system analyst. The total number of system events in the logs for each host was 2.3M (Host1), 9M (Host2), and 6M (Host3) and the number of events associated with attacks was 316 (Host1), 473 (Host2), and 1139 (Host3). The number of total system entities in the data was 137K (Host1), 372K (Host2), and 78K (Host3) with the number of alien entities associated with the attacks being 42 (Host1), 51 (Host2), and 51 (Host3). Note that the fraction of alien entities here is exceedingly small.

Anomaly Detector Configuration. Each instance for our anomaly detector corresponds to a system entity and we described these system entities via a set of 20 "views", where each view computed a set of descriptor features for the particular entity type. There were approximately an equal number of views per type. For example, there is a view that computes features about the statistics of process file access patterns. The views were constructed using domain knowledge about what subsets of information would potentially be useful to consider jointly when searching for anomalies. It was not clear to the domain experts, however, which of the views would actually be useful, if any.

We created a single tree based detector that incorporated all views. To do this, we built a randomized IF for each of the views individually, containing 100 trees each. Then we put those trees together into a single forests of 2000 trees. Given a system entity its score can be computed using all of the trees, except that if a tree does not apply to the type of entity being considered it is ignored. Given such a forest we run our OMD feedback algorithms using a simulated analyst along with the IF baseline. Interestingly this overall approach can be viewed as a principled way of combining multiple views via feedback, since the feedback will effectively help up-weight and down-weight information associated with different views according to the inferred utility. For these experiment, we were unable to run AAD due to the currently available code not supporting the composition of trees across views as described above.

Table 1: Time to detect first malicious entity

Dataset	# of iterations to 1^{st} malicious entity		
	Baseline	Linear	Log-Likelihood
Host1	42.7 ± 24.3	**2.7 ± 0.7**	12.9 ± 6.3
Host2	61.8 ± 26.9	**8.1 ± 1.1**	32.2 ± 22.7
Host3	41.4 ± 1.1	**21 ± 3.5**	**20.6 ± 3.5**

Results. Figure 5 shows the number of malicious entities detected as a function of the number of feedback iterations for each of the three hosts. We see that both feedback iterations are able to improve the rate of discovered aliens compared to the IF baseline. This is particularly significant on Host2. Note that not all attack entities are detected. This is expected, since many of the entities involved in attack behave quite normally. The other entities exhibit unusual behavior that can be detected, for example, entities involved in initial exploits or exfiltration. The key utility in such a system is to provide analysts with starting points for uncovering the full attack via more detailed post-attack analysis.

Finally, to provide another utility of our approach, Table 1 records the average number of iterations until the first attack entity is discovered with our feedback approaches and the IF baseline. This is a particularly relevant metric, since it roughly quantifies the amount of investigative work required to identify the first entry point into the attack to seed further investigation. We see that both feedback methods Linear and Log-Likelihood significantly reduce the number of iterations required compared to the IF baseline. The linear loss function is particularly effective here and for Host1 and Host2 is significantly better than the Log-Likelihood loss.

7 SUMMARY

In this paper, we developed a human-in-the-loop anomaly detection system, where an analyst can provide direct feedback to the unsupervised anomaly detector. The goal is for the anomaly detector to then align its anomaly scores with the application-specific notion of interestingness. We formulated this problem within the framework of online convex optimization and defined two loss functions, which lead to two simple and efficient algorithms with a single parameter that was held constant throughout our experiments. We instantiated the algorithm for the wide class of tree-based anomaly detectors. Our results on a large number of benchmark data sets demonstrates that our approach is able to significantly improve upon a strong baseline that does not use feedback and improve over the prior state-of-the-art algorithm AAD, which incorporates feedback. Overall both of our loss functions were effective, with the Log-Likelihood loss function showing slighly more robust performance. We also showed the effectiveness of our approach when applied to large cybersecurity datasets generated by a red team for a recent DARPA evaluation exercise. Our method also improved significantly the rate that attack entities can be found and reduce the effort needed to find the first attack entity. In future work we would like to extend our current approach to incorporate feedback for any arbitrary anomaly detection method. Another critical component moving forward is to study interfaces that best support a system analyst in investigating proposed anomalies, which will

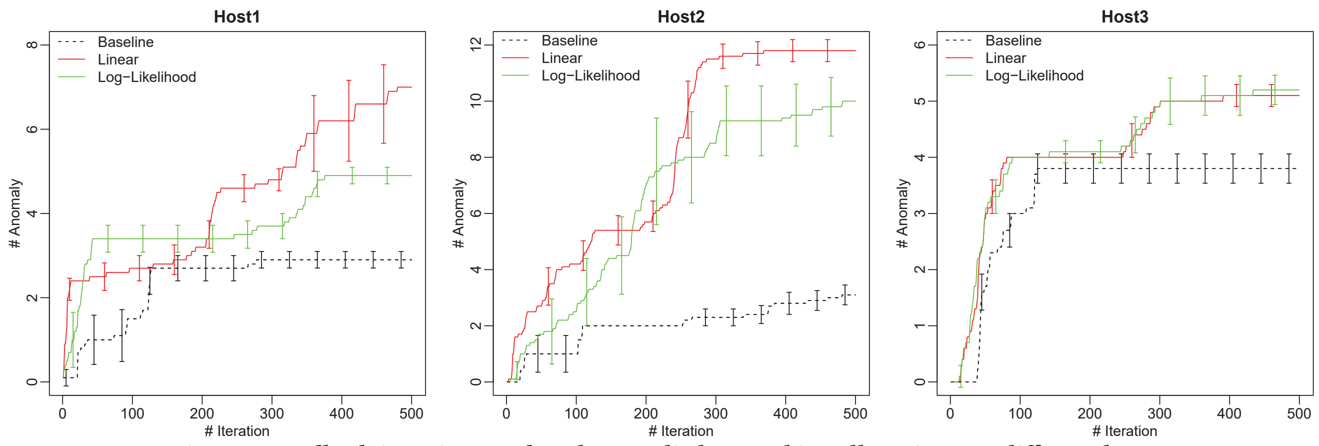

Figure 5: Feedback iteration results when applied to combine all 20 views on different hosts.

involve providing the analyst with clear explanations about why the system believes an entity is potentially anomalous. To be able to incorporate feedback on such explanations would be another interesting future direction.

Acknowledgements

This work is supported by DARPA under contract number: FA8650-15-C-7557. Any opinions, findings and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the DARPA.

REFERENCES

[1] Shubhomoy Das, Weng-Keen Wong, Thomas G. Dietterich, Alan Fern, and Andrew Emmott. 2016. Incorporating Expert Feedback into Active Anomaly Discovery. In *Proceedings of the IEEE ICDM*. 853–858.

[2] Shubhomoy Das, Weng-Keen Wong, Alan Fern, Thomas G Dietterich, and Md Amran Siddiqui. 2017. Incorporating Feedback into Tree-based Anomaly Detection. *arXiv preprint arXiv:1708.09441* (2017).

[3] Boxiang Dong, Zhengzhang Chen, Hui Wendy Wang, Lu-An Tang, Kai Zhang, Ying Lin, Zhichun Li, and Haifeng Chen. 2017. Efficient Discovery of Abnormal Event Sequences in Enterprise Security Systems. In *The ACM International Conference on Information and Knowledge Management (CIKM). Pan Pacific, Singapore.*

[4] Andrew Emmott, Shubhomoy Das, Thomas G. Dietterich, Alan Fern, and Weng-Keen Wong. 2015. Systematic Construction of Anomaly Detection Benchmarks from Real Data. *CoRR* abs/1503.01158 (2015). http://arxiv.org/abs/1503.01158

[5] Andrew F Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. 2013. Systematic construction of anomaly detection benchmarks from real data. In *Proceedings of the ACM SIGKDD workshop on outlier detection and description*. ACM, 16–21.

[6] Stephanie Forrest, Steven A Hofmeyr, Anil Somayaji, and Thomas A Longstaff. 1996. A sense of self for unix processes. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*. IEEE, 120–128.

[7] Debin Gao, Michael K Reiter, and Dawn Song. 2004. Gray-box extraction of execution graphs for anomaly detection. In *Proc. of the 11th ACM conference on Computer and communications security*. 318–329.

[8] Nico Görnitz, Marius Micha Kloft, Konrad Rieck, and Ulf Brefeld. 2013. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research* (2013).

[9] Martin Grill and Tomáš Pevnỳ. 2016. Learning combination of anomaly detectors for security domain. *Computer Networks* 107 (2016), 55–63.

[10] Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. 2016. Robust Random Cut Forest Based Anomaly Detection On Streams. In *Proceedings of The 33rd International Conference on Machine Learning*, Vol. 48.

[11] Shali Jiang, Gustavo Malkomes, Geoff Converse, Alyssa Shofner, Benjamin Moseley, and Roman Garnett. 2017. Efficient Nonmyopic Active Search. In *International Conference on Machine Learning*. 1714–1723.

[12] F Korč and W Förstner. 2008. Approximate parameter learning in conditional random fields: An empirical investigation. In *Joint Pattern Recog. Symp.* Springer.

[13] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 413–422.

[14] Dan Pelleg and Andrew W Moore. 2005. Active learning for anomaly and rare-category detection. In *Advances in neural information processing systems*.

[15] Tomáš Pevný. 2016. Loda: Lightweight On-line Detector of Anomalies. *Mach. Learn.* 102, 2 (Feb. 2016), 275–304.

[16] Maxim Raginsky, Rebecca M Willett, Corinne Horn, Jorge Silva, and Roummel F Marcia. 2012. Sequential anomaly detection in the presence of noise and limited feedback. *IEEE Transactions on Information Theory* 58, 8 (2012), 5544–5562.

[17] R Sekar, Mugdha Bendre, Dinakar Dhurjati, and Pradeep Bollineni. 2001. A fast automaton-based method for detecting anomalous program behaviors. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*. IEEE, 144–155.

[18] Burr Settles. 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1 (2012), 1–114.

[19] Shai Shalev-Shwartz et al. 2012. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning* 4, 2 (2012), 107–194.

[20] Xiaokui Shu, Danfeng Yao, and Naren Ramakrishnan. 2015. Unearthing stealthy program attacks buried in extremely long execution paths. In *Proc. of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 401–413.

[21] Md Amran Siddiqui, Alan Fern, Thomas Dietterich, and Shubhomoy Das. 2016. Finite Sample Complexity of Rare Pattern Anomaly Detection. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.

[22] Swee Chuan Tan, Kai Ming Ting, and Tony Fei Liu. 2011. Fast Anomaly Detection for Streaming Data. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence - Volume Two*. 1511–1516.

[23] Jagannadan Varadarajan, Ramanathan Subramanian, Narendra Ahuja, Pierre Moulin, and Jean-Marc Odobez. 2017. Active Online Anomaly Detection using Dirichlet Process Mixture Model and Gaussian Process Classification. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. 615–623.

[24] Kalyan Veeramachaneni, Ignacio Arnaldo, Vamsi Korrapati, Constantinos Bassias, and Ke Li. 2016. AIˆ 2: training a big data machine to defend. In *Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2016 IEEE 2nd International Conference on*. IEEE, 49–54.

[25] Ke Wu, Kun Zhang, Wei Fan, Andrea Edwards, and S Yu Philip. 2014. Rs-forest: A Rapid Density Estimator for Streaming Anomaly Detection. In *ICDM, 2014 IEEE International Conference on*. 600–609.