AN ABSTRACT OF THE DISSERTATION OF

Md Amran Siddiqui for the degree of Doctor of Philosophy in Computer Science presented on June 7, 2019.

Title: Anomaly Detection: Theory, Explanation and User Feedback

Abstract approved: _____

Alan P. Fern

Anomaly detection has been used in variety of applications in practice, including cyber-security, fraud detection and detecting faults in safety critical systems, etc. Anomaly detectors produce a ranked list of statistical anomalies, which are typically examined by human analysts in order to extract the actual anomalies of interest. Unfortunately, most anomaly detectors provide no explanations about why an instance was considered anomalous. To address this issue, we propose a feature based explanation approach called sequential feature explanation (SFE) to help the analyst in their investigation. A second problem with the anomaly detection systems is that they usually produce a large number of false positives due to a mismatch between statistical and semantic anomalies. We address this issue by incorporating human feedback, that is, we develop a human-in-the-loop anomaly detection system which can improve its detection rate with a simple form of true/false positive feedback from the analyst. We show empirically the efficacy and the superior performance of both of our explanation and feedback approaches on significant cyber security applications including red team attack data and real corporate network data along with a large number of benchmark datasets. We also delve into a set of state-of-the-art anomaly detection techniques to understand why they perform so well with a small number of training examples. We unify their working principle into a common framework underlying different pattern spaces and compute their sample complexity for achieving performance guarantees. In addition, we empirically investigate learning curves for anomaly detection in this framework.

Anomaly Detection: Theory, Explanation and User Feedback

by

Md Amran Siddiqui

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented June 7, 2019
Commencement June 2019

Doctor of Philosophy dissertation of Md Amran Siddiqui presented on June 7, 2019.

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Md Amran Siddiqui, Author

ACKNOWLEDGEMENTS

I am immensely grateful to my advisor Dr. Alan Fern for giving me the opportunity to pursue my PhD under his advising. I learned a lot of skills like how to conduct research, how to brainstorming ideas, writing papers and presentation from his close supervision and feedback.

I am very grateful to Dr. Thomas Dietterich for providing feedback and discussing interesting research directions during our anomaly detection meetings.

I am also grateful to my committee members Dr. Prasad Tadepalli, Dr. Raviv Raich and my GCR Dr. Debashis Mondal for their valuable time and feedback during the exams.

I am grateful to my industry collaborators from Galois, Inc. Ryan Wright, Alec Theriault, David W. Archer and Nichole Schimanski. I learned a lot from them about systems, software engineering and computer security for which I didn't have much background before.

I would like to thank my fellow talented graduate students and faculty members with whom I spent a lot of time discussing and sharing ideas and so on: Tadesse Zemicheal, Shubhomoy Das, Andrew Emmott, Jed Irvine, Michael Slater, Liu Si and Risheek Garrepalli.

Finally, I would like to thank my friends, family members and communities in Corvallis for helping me with other than academic aspects.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1: Introduction

This dissertation focuses on the anomaly detection problem in general with a special focus on computer security as an application. The frequency of cyberattacks is rapidly increasing in all sectors of personal, enterprise, government, and medical computer systems. While there are many effective tools for detecting and mitigating known attacks, tools for detecting novel attacks are still unreliable. While fully automated detection of cyberattacks is a clear goal for the future, a more realistic and important near-term goal is to develop security systems that interact with a security analyst in order to detect the attacks as quickly as possible. We address these issues by employing anomaly detection techniques and develop methodologies and extensions around anomaly detection. We try to understand why modern anomaly detection often perform well and what are the practical problems people face when an anomaly detection system is deployed. We discuss those problems in detail and developed methodologies to address those problems scientifically.

1.1 Anomaly Detection

The problem of (unsupervised) anomaly detection is to identify anomalies in (unlabeled) data, where an anomaly is a data point that is generated by a process that is distinct from the process that generates "normal" points. This problem arises in a large number of applications, from security to data cleaning, and there have been many approaches proposed in the literature [Chandola et al., 2009, Hodge and Austin, 2004]. Figure 1.1 shows a simple one dimensional anomaly detector based on density computation, that is, lower density points are declared as anomalous.

While most applications seek to identify semantically-interesting anomalies, it is typically not possible to predefine a functional notion of semantic interestingness. Instead, the vast majority of anomaly detectors use a surrogate measure of interestingness. For example, a point may be interesting if it is a statistical outlier or if it is far away from other data points. The performance of a given detector in a domain depends on how well it can optimize the statistical measure and on how

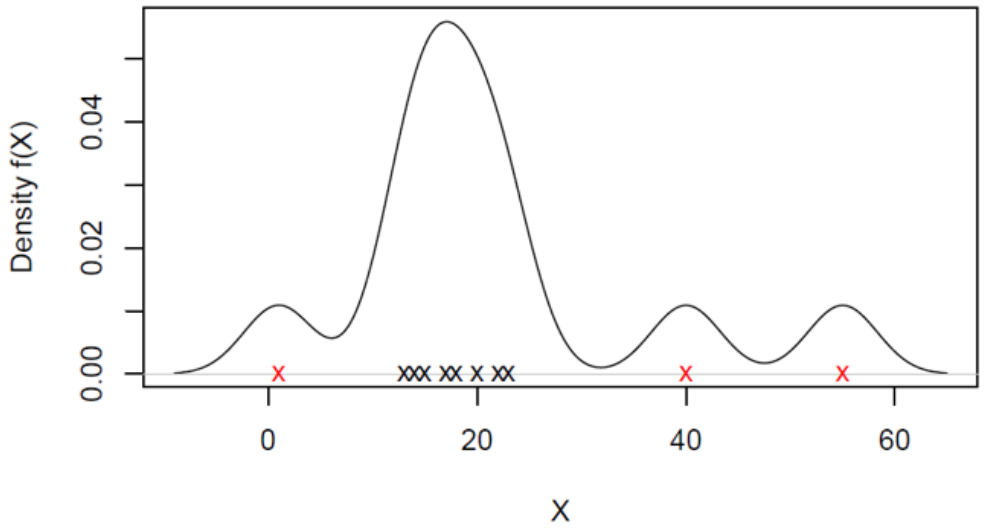

Figure 1.1: Density based anomaly detection. The red points are outliers due to lying in a low density region.

well that measure aligns with the behavior of anomalies in the application domain. One way to align the statistical measure to the target behavior is via some supervision which can be collected from some end user domain expert. The domain expert also needs to understand why the anomaly detector surfaces certain points as anomalous and others as nominal. So, we need to provide some explanations about why the anomaly detector actually choosing certain points as anomalous. We address both of these issues in this dissertation by developing methodologies to produce explanations and how to incorporate analysts feedback into anomaly detection.

1.2 Outline

Chapter 2 addresses the question of why modern anomaly detection performs well with a surprisingly small number of examples. Specifically, we develop some theory around the sample complexity of different anomaly detection techniques underlying different pattern spaces and provide PAC guarantees. This work resulted in a conference paper at UAI 2016 [Siddiqui et al., 2016].

Chapter 3 addresses the problem that most anomaly detectors do not provide

explanations about why they detected certain points as anomalous. Specifically, we proposes an explanation idea based on the input features and develop some efficient practical algorithms to produce such explanations. This work resulted an ODDx3 workshop paper at KDD 2015 [Siddiqui et al., 2015] and a journal paper at ACM TKDD 2019 [Siddiqui et al., 2019a].

Chapter 4 focuses on how the anomaly detection performance can be improved from the feedback of an end-user analyst to reduce number of false positives. We show how such approaches can be applied in a significant cyber security application. This work resulted an AICS workshop paper at AAAI 2018 [Siddiqui et al., 2018b] and a conference paper at KDD 2018 [Siddiqui et al., 2018a].

Chapter 5 explores the class discovery problem which focuses on how quickly we can discover all of the interesting classes from a given dataset, for example, discovering all of the different types of attack from network data. Finally, Chapter 6 provides some conclusions along with some future directions.

Chapter 2: Rare Pattern Anomaly Detection (RPAD)

2.1 Introduction

For moderately high-dimensional data, all data points become far apart from each other, so they are all statistical outliers in a sense. This suggests that anomaly detection by identifying outliers or distant points should perform poorly and degrade to random selection as the dimensionality grows. Empirical results, however, have shown that state-of-the-art anomaly detectors often perform quite well [Emmott et al., 2013] even for high-dimensional data. Further, these detectors tend to reach their peak performance with a relatively small amount of training data compared to what might be expected based on the dimensionality. The primary goal of this chapter is to move toward an understanding of these empirical observations by analyzing the sample complexity of a certain class of anomaly detectors.

The sample complexity of supervised learning has been widely studied and is quite well understood via the framework of Probably Approximately Correct (PAC) learning. However, this is not the case for anomaly detection, where virtually all published work has focused on algorithms with good empirical performance (with additional attention to computational speed, especially on big data sets). A key step in the development of PAC learning theory was to formalize the notion of a hypothesis space and to quantify the relationship between the complexity of this space and the amount of training data required to identify a good hypothesis in the space. In this chapter, we follow a similar approach. Our framework is motivated by the observation that many state-of-the-art anomaly detectors can be viewed as monitoring the probabilities of certain "patterns" in the data, where a "pattern" is a subset (typically closed and compact) of the feature space. Outliers are then identified based on measures of those probabilities, where points are ranked as more anomalous if they satisfy lower-probability patterns. For example, the highly-competitive anomaly detection algorithm, Isolation Forest [Liu et al., 2008], finds outliers by monitoring probabilities in the pattern space of axis-aligned hyper-rectangles. Section 2.5 provides additional examples of

the pattern spaces underlying a number of other state-of-the-art detectors. In our analysis, a "pattern" will play the same role as a "hypothesis" in PAC learning, and the pattern space complexity will determine the number of training examples required for high accuracy.

A second key step in the development of PAC theory was to relax the goal of finding the best possible hypothesis. Similarly, we will introduce an error parameter ϵ that determines how accurately the algorithm must estimate the probabilities of the patterns in the pattern space. We will then show that the required sample size scales polynomially in $1/\epsilon$ (as well as in several other parameters).

We call our formulation Rare Pattern Anomaly Detection (RPAD), and an algorithm that provides PAC guarantees will be referred to as a PAC-RPAD algorithm. We prove sample complexity results for any algorithm within the RPAD framework. The framework captures the qualitative essence of many anomaly detection algorithms. Note that we focus exclusively on sample complexity. Experience with the supervised PAC analysis has shown that sample complexity results often give more insight than computational complexity results. Indeed, many computational problems in PAC learning are NP-Hard and yet practical approximate algorithms are known. Similarly, we expect that some of the computational PAC-RPAD problems will also be NP-Hard, but existing algorithms, such as Isolation Forest, already provide practical approximate solutions.

Prior work on one-class SVMs [Schölkopf et al., 2001] and learning minimum volume sets [Scott and Nowak, 2006] have also provided sample complexity analysis relevant to anomaly detection. These approaches, however, are fundamentally different than RPAD-style approaches. In particular, these approaches focus on finding a common region/pattern in the input space that capture the normal points. In contrast, our RPAD framework is based on finding rare regions/patterns for directly extracting anomaly characteristics. Anomaly detection benchmarking studies [Emmott et al., 2013] have shown that RPAD-style approaches tend to significantly outperform "common pattern" approaches such as one-class SVM. Our work is the first to analyze the sample complexity of the former approach.

The main contributions of this chapter are as follows. First, we present a formal framework for RPAD (Section 2.2), which leads to the definition of PAC-RPAD (Section 2.3). Second, we specify a simple generic algorithm, RarePatternDetect, based on finding rare patterns. We derive sample complexity results for

both finite pattern spaces and uncountable spaces of bounded complexity (Section 2.4). Third, we give a number of applications of the theory to pattern spaces that underly a number of state-of-the-art anomaly detection algorithms (Section 2.5). This, in part, helps explain why such algorithms consistently perform much better than random in high-dimensional data. Fourth, we measure learning curves on several benchmarks and for pattern spaces of varying complexity for RarePatternDetect and another state-of-the-art anomaly detector over the same spaces (Section 2.6). The results show that the RPAD-based algorithm can be competitive with the state-of-the-art and that the detectors' performances converge for surprisingly small training sets.

2.2 Rare Pattern Anomaly Detection

We consider anomaly detection over a space of possible data points $\mathcal{X} \subseteq \mathcal{R}^d$, which may be finite or infinite. Data from this space is generated according to an unknown probability density function $\mathcal{P}$ over $\mathcal{X}$. A common assumption in anomaly detection is that $\mathcal{P}$ is a mixture of a normal component, which generates normal data points, and an anomaly component, which generates anomalous data points. Further, it is typically assumed that there is a much higher probability of generating normal data than anomalies. This set of assumptions motivates one approach to anomaly detection, which we call *Anomaly Detection via Outlier Detection*. The idea is to estimate, for each query point x, the density $\mathcal{P}(x)$ based on an (unlabeled) training sample of the data and assign an anomaly score to x proportional to $-\log \mathcal{P}(x)$, the "surprise" of x.

There are many problems with this approach. First, the probability density may not be smooth in the neighborhood of x, so that $\mathcal{P}(x)$ could be very large and yet be surrounded by a region of low or zero density (or vice versa). Second, even under smoothness assumptions, density estimation is very difficult. For example, the integrated squared error of kernel density estimation in d-dimensional space (for a second-order kernel, such as a Gaussian kernel) converges to zero at a rate of $O(N^{-4/(4+d)})$ [Shalizi, 2016]. It follows that the sample size N required to achieve a target accuracy grows exponentially in the dimension d.

In this chapter, we consider an alternative anomaly detection framework, which we will refer to as *Rare Pattern Anomaly Detection (RPAD)*. Informally, the main

idea is to judge a point as anomalous if it exhibits a property, or pattern, that is rarely exhibited in data generated by $\mathcal{P}$. For example, in a computer security application, a detector may monitor various behavior patterns associated with processes accessing files. A process that exhibits an access behavior pattern that has been rarely seen would be considered anomalous. One attractive feature of the RPAD framework is that the notion of anomaly is grounded in the estimation of pattern probabilities, rather than point densities. Pattern probability estimation is quite well understood compared to density estimation. A second attractive feature of the RPAD framework is that each detected anomaly comes with an explanation of why it was considered anomalous. Specifically, the explanation can report the rare patterns that the anomaly satisfies. Explanation methods have been developed for density-estimation approaches (e.g. [Siddiqui et al., 2015]), but they are less directly tied to the anomaly detection criterion.

Formally, a *pattern* h is a binary function over $\mathcal{X}$. A *pattern space* $\mathcal{H}$ is a set of patterns, which may be finite or infinite. As an example, if $\mathcal{X}$ is a finite space of n-dimensional bit vectors, a corresponding finite pattern space could be all conjunctions of length up to k. As another example, let $\mathcal{X} = [0,1]^n$, the n-dimensional unit cube, and consider the uncountable pattern space of all axis-aligned k-dimensional hyper-rectangles in this cube. In this case, each pattern h is a hyper-rectangle, such that $h(x)$ is true if x falls inside it. The choice of pattern space is an important consideration in the RPAD framework, since, in large part, this choice controls the semantic types of anomalies that will be detected.

Each pattern $h \in \mathcal{H}$ has a probability $P(h) = \Pr(\{x : h(x) = 1\})$ of being satisfied by data points generated according to $\mathcal{P}$. It will be useful to specify the set of all patterns in $\mathcal{H}$ that a point x satisfies, which we will denote by $\mathcal{H}[x] = \{h \in \mathcal{H} : h(x) = 1\}$. One approach to RPAD is to declare x to be anomalous if there is a pattern $h \in \mathcal{H}[x]$, such that $P(h) \leq \tau$. This approach is sensible when all patterns are approximately of the same "complexity". However, when $\mathcal{H}$ contains a mix of simple and complex patterns, this approach can be problematic. In particular, more complex patterns are inherently less likely to be satisfied by data points than simpler patterns, which makes choosing a single threshold τ difficult. For this reason, we introduced the *normalized pattern probability* $f(h) = P(h)/U(h)$, where $U(h)$ is the probability of h being satisfied according to a *reference density* U over $\mathcal{X}$. When $\mathcal{X}$ is bounded, we typically take

U to be a uniform density function. Thus, a small value of $f(h)$ indicates that under $\mathcal{P}$, h is significantly more rare than it would be by chance, which provides a better-calibrated notion of rareness compared to only considering $\mathcal{P}(h)$. If $\mathcal{X}$ is unbounded, then an appropriate maximum entropy distribution (e.g., Gaussian) can be chosen.

We can now define the notion of anomaly under the RPAD framework. We say that a pattern h is τ-*rare* if $f(h) \leq \tau$, where τ is a detection threshold specified by the user. A data point x is a τ-*outlier* if there exists a τ-rare h in $\mathcal{H}[x]$ and otherwise x is said to be τ-common. Given τ and a stream of data drawn from $\mathcal{P}$, an optimal detector within the RPAD framework should detect all τ-outlier points and reject all τ-common points. That is, we want to detect any point that satisfies a τ-rare pattern and otherwise reject. An anomaly detector will make its decisions based on some finite amount of sampled data, and we expect that the performance should improve as the amount of data grows. Further, in analogy to supervised learning, we would expect that the amount of data required to achieve a certain level of performance should increase as the complexity of the pattern space increases. We now introduce a formal framework for making these intuitions more precise.

2.3 Probably Approximately Correct Framework

To address the sample complexity of RPAD, we consider a learning protocol that makes the notion of training data explicit. The protocol first draws a training data set $\mathcal{D}$ of N i.i.d. data points from $\mathcal{P}$. An anomaly detector is provided with $\mathcal{D}$ along with a test instance x that may or may not be drawn from $\mathcal{P}$. The anomaly detector then outputs "detect" if the instance is considered to be an anomaly or "reject" otherwise. Note that the output of a detector is a random variable due to the randomness of $\mathcal{D}$ and any randomization in the algorithm itself. This protocol models a common use case in many applications of anomaly detection. For example, in a computer security application, data from normal system operation will typically be collected and provided to an anomaly detector before it is activated.

The *ideal* correctness criterion requires that the test instance x be detected if it is a τ-outlier and rejected otherwise. However, as we discussed above, this

notion of correctness is too strict for the purpose of sample complexity analysis. In particular, such a criterion requires distinguishing between pattern probabilities that fall arbitrarily close to each side of the detection threshold τ, which can require arbitrarily large training samples. For this reason, we relax the correctness criterion by introducing a *tolerance parameter* $\epsilon > 0$. The detector is said to be approximately correct at level ϵ if it detects all τ-rare points and rejects all $(\tau+\epsilon)$-common points. For test points that are neither τ-rare nor $(\tau+\epsilon)$-common, the detector output can be arbitrary. The value of ϵ controls the false positive rate of the detector, where smaller values of ϵ will result in fewer false positives relative to the detection threshold.

We now define the PAC learning objective for RPAD. A detection algorithm will be considered PAC-RPAD if with high-probability over draws of the training data it produces an approximately correct output for any $x \in \mathcal{X}$.

Definition 2.3.1. *(PAC-RPAD) Let $\mathcal{A}$ be a detection algorithm over pattern space $\mathcal{H}$ with input parameters $0 < \delta < 1$, $0 < \tau$, $0 < \epsilon$, and the ability to draw a training set $\mathcal{D}$ of any size N from $\mathcal{P}$. $\mathcal{A}$ is a PAC-RPAD algorithm if for any $\mathcal{P}$ and any τ, with probability at least $1-\delta$ (over draws of $\mathcal{D}$), $\mathcal{A}$ detects all τ-outliers and rejects all $(\tau+\epsilon)$-commons.*

The *sample complexity* of a PAC-RPAD algorithm for $\mathcal{H}$ is a function of the inputs $N(\delta, \epsilon)$ that specifies the number of training examples to draw. We expect that the sample complexity will increase as the complexity of $\mathcal{H}$ increase, as the dimensionality d of points increases, and as the failure probability δ decreases. Further, we expect that the sample complexity will increase for smaller values of the tolerance parameter ϵ, since this controls the difficultly of distinguishing between τ-rare and $(\tau+\epsilon)$-common data points. Accordingly we say that a PAC-RPAD algorithm is *sample efficient* if its sample complexity is polynomial in d, $\frac{1}{\delta}$, and $\frac{1}{\epsilon}$.

2.4 Finite Sample Complexity of RPAD

We now consider a very simple algorithm, called RarePatternDetect, which will be shown to be a sample efficient PAC-RPAD algorithm for bounded complexity pattern spaces. The algorithm is given in Table 2.1 and first draws a training set $\mathcal{D}$ of size $N(\delta, \epsilon)$. Here $N(\delta, \epsilon)$ will depend on the pattern space complexity

Table 2.1: RarePatternDetect Algorithm

Input: δ, τ, ϵ

1. Draw a training set $\mathcal{D}$ of $N(\delta, \epsilon)$ instances from $\mathcal{P}$.
2. **Decision Rule for any x:**
   If HasRarePattern$(x, \mathcal{D}, \mathcal{H}, \tau + \epsilon/2)$ then return "detect"
   Otherwise return "reject".

HasRarePattern$(x, \mathcal{D}, \mathcal{H}, \mu)$
$:= \{h \in \mathcal{H}[x] : \widehat{f}(h) \leq \mu\} \neq \emptyset$

and will be specified later in this section. The training set is used to estimate the normalized pattern probabilities given by

$$\widehat{f}(h) = \frac{1}{|D| \cdot U(h)} |\{x \in \mathcal{D} : h(x) = 1\}|.$$

Here, we assume that $U(h)$ can be computed analytically or at least closely approximated. For example, when U is uniform over a bounded space, $U(h)$ is proportional to the volume of h.

After drawing the training set, RarePatternDetect specifies a decision rule that detects any x as anomalous if and only if it satisfies a pattern with estimated frequency less than or equal $\tau + \epsilon/2$. This test is done using the subroutine HasRarePattern$(x, \mathcal{D}, \mathcal{H}, \mu)$, which returns true if there exists a pattern h in $\mathcal{H}[x]$ such that $\widehat{f}(h) \leq \mu$. For the purposes of sample complexity analysis, we will assume an oracle for HasRarePattern. For sufficiently complex pattern spaces, the problem addressed by HasRarePattern will be computational hard. Thus, in practice, a heuristic approximation will be needed, for example, based on techniques developed in the rare pattern mining literature. In practice, we are often interested in having an anomaly detector return an anomaly ranking over multiple test data points. In this case, the algorithm can rank a data point x based on a score equal to the minimum normalized frequency of any pattern that it satisfies, that is, score$(x) = \min\{\widehat{f}(h) : h \in \mathcal{H}[x]\}$. It remains to specify $N(\delta, \epsilon)$ in order to ensure that RarePatternDetect is PAC-RPAD. Below we do this for two

cases: 1) finite pattern spaces, and 2) pattern spaces with bounded VC-dimension. Later, in Section 2.5 we will instantiate these results for specific pattern spaces that underly several existing anomaly detection algorithms.

2.4.1 Sample Complexity for Finite $\mathcal{H}$

For finite pattern spaces, it is relatively straightforward to show that as long as $\log|\mathcal{H}|$ is polynomial in d then RarePatternDetect is a sample efficient PAC-RPAD algorithm.

Theorem 2.4.1. *For any finite pattern space $\mathcal{H}$,* RarePatternDetect *is PAC-RPAD with sample complexity $N(\delta,\epsilon) = O\left(\frac{1}{\epsilon^2}\left(\log|\mathcal{H}| + \log\frac{1}{\delta}\right)\right)$.*

Proof. Suppose, X is a Bernoulli random variable with parameter $P(h)$ for a pattern h, i.e., $E[X] = P(h)$. Let, $Y = \frac{X}{U(h)}$, hence, Y is a random variable with $E[Y] = \frac{E[X]}{U(h)} = \frac{P(h)}{U(h)} = f(h)$. We also observe that the maximum value of Y is $\frac{1}{U(h)}$. Given N samples $x_1, x_2, ..., x_N$, each $x_i \sim \mathcal{P}$, we estimate $\widehat{f}(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{\mathcal{I}[x_i \in h]}{U(h)}$. We seek a confidence interval $[L(h), R(h)]$ for $f(h)$ that is narrow enough that it does not simultaneously contain both τ and $\tau+\epsilon$. The reason is that if $L(h) > \tau$, then with probability $1-\delta$, $f(h) > \tau$, so h is not a τ-rare pattern. If $R(h) < \tau+\epsilon$, then with probability $1-\delta$, h is not a $\tau+\epsilon$-common pattern, so it is safe to treat it as τ-rare. Hence, the confidence interval should be $[\widehat{f}(h) - \epsilon/2, \widehat{f}(h) + \epsilon/2]$, and its "half width" is $\epsilon/2$. So, we want to bound (by δ) the probability that $\widehat{f}(h)$ is more than $\frac{\epsilon}{2}$ away from its true value $f(h)$. Now, using the Hoeffding bound we have

$$P\left(|E_{\mathcal{P}}[\widehat{f}(h)] - \widehat{f}(h)| > \frac{\epsilon}{2}\right) \iff P\left(\left|f(h) - \frac{1}{N}\sum_{i=1}^{N}\frac{\mathcal{I}[x_i \in h]}{U(h)}\right| > \frac{\epsilon}{2}\right)$$
$$\leq 2\exp\left(-\frac{\epsilon^2}{2}U(h)^2 N\right).$$

Since $\mathcal{H}$ is finite, we can bound the above probability for all $h \in \mathcal{H}$ using the union bound: $2|\mathcal{H}|\exp\left(-\frac{\epsilon^2}{2}U_{min}^2 N\right)$, where, $U_{min} = \min_{h\in\mathcal{H}} U(h)$. We want this

quantity to be less or equal to δ:

$$2|\mathcal{H}| \exp\left(-\frac{\epsilon^2}{2} U_{min}^2 N\right) \leq \delta \implies N \geq \frac{2}{\epsilon^2} \frac{1}{U_{min}^2} \log \frac{2|\mathcal{H}|}{\delta}.$$

Hence, the sample complexity is $O\left(\frac{1}{\epsilon^2}\left(\log|\mathcal{H}| + \log\frac{1}{\delta}\right)\right)$. □

2.4.2 Sample Complexity for Infinite $\mathcal{H}$

When the sample space $\mathcal{X}$ is continuous, it is typically the case that the corresponding pattern space will be infinite and hence not covered by the above result. As is standard in supervised learning, we will characterize the complexity of infinite pattern spaces via the VC-dimension [Vapnik and Vapnik, 1998], which we denote by $\mathcal{V}_{\mathcal{H}}$. The VC-dimension of $\mathcal{H}$ is equal to the maximum number of points that can be shattered by patterns in $\mathcal{H}$. Here a set of points D is shattered by $\mathcal{H}$ if for any subset D' of D there is an $h \in \mathcal{H}$ such that $h(x) = 1$ for all $x \in D'$ and $h(x) = 0$ for all $x \in D - D'$. That is, patterns in $\mathcal{H}$ can be used to define all possible bi-partitions of D. For many interesting pattern spaces, the VC-dimension scales polynomially with the data dimension d. The following result exploits this property by showing that if a space has VC-dimension that is polynomial in d, then the space is sample-efficient learnable in the PAC-RPAD model.

Theorem 2.4.2. *For any pattern space $\mathcal{H}$ with finite VC-dimension $\mathcal{V}_{\mathcal{H}}$,* RarePatternDetect *is PAC-RPAD with sample complexity* $N(\delta, \epsilon) = O\left(\frac{1}{\epsilon^2}\left(\mathcal{V}_{\mathcal{H}} \log\frac{1}{\epsilon^2} + \log\frac{1}{\delta}\right)\right)$.

Proof. When the pattern space $\mathcal{H}$ is infinite, we want to bound the probability $P\left(\sup_{h\in\mathcal{H}} |\widehat{f}(h) - f(h)| > \frac{\epsilon}{2}\right)$, which can be achieved by bounding

$$P\left(\sup_{h\in\mathcal{H}} |\widehat{P}(h) - P(h)| > \frac{\epsilon}{2} U_{min}\right),$$

where, $\widehat{P}(h)$ is an estimate of $P(h)$ based on sampled data. Let, $\epsilon_f = \frac{\epsilon}{2} U_{min}$. Using the VC uniform convergence bound on frequency estimates [Devroye et al., 2013,

Thm. 12.5] we have

$$P\left(\sup_{h\in\mathcal{H}}|\widehat{P}(h)-P(h)|>\epsilon_f\right)\leq 8\mathcal{S}_{\mathcal{H}}(N)e^{-N\epsilon_f^2/32}. \tag{2.1}$$

where, $\widehat{P}(h)$ is an estimate based on N i.i.d. samples from $\mathcal{P}$ and $\mathcal{S}_{\mathcal{H}}(N)$ is the Shatter Coefficient, which is the largest number of subsets that can be formed by intersecting some set of N points with patterns from $\mathcal{H}$.

Now, for any $N > 2\mathcal{V}_{\mathcal{H}}$, we can bound the Shatter Coefficient as: $\mathcal{S}_{\mathcal{H}}(N) < (\frac{eN}{\mathcal{V}_{\mathcal{H}}})^{\mathcal{V}_{\mathcal{H}}}$ [Devroye et al., 2013, Thm. 13.3]. Hence, from Equation 2.1 we have

$$P\left(\sup_{h\in\mathcal{H}}|\widehat{P}(h)-P(h)|>\epsilon_f\right)<8(\frac{eN}{\mathcal{V}_{\mathcal{H}}})^{\mathcal{V}_{\mathcal{H}}}e^{-N\epsilon_f^2/32}. \tag{2.2}$$

We want to bound this probability by δ, which yields

$$N\geq\frac{32}{\epsilon_f^2}\left(\mathcal{V}_{\mathcal{H}}\log(N)+\mathcal{V}_{\mathcal{H}}\log\frac{e}{\mathcal{V}_{\mathcal{H}}}+\log\frac{8}{\delta}\right). \tag{2.3}$$

Using the fact that $\log(N)\leq\alpha N-\log(\alpha)-1$, where, $N,\alpha>0$ and setting $\alpha=\frac{\epsilon_f^2}{64\mathcal{V}_{\mathcal{H}}}$, we get

$$\begin{aligned}\frac{32\mathcal{V}_{\mathcal{H}}}{\epsilon_f^2}\log(N)&\leq\frac{32\mathcal{V}_{\mathcal{H}}}{\epsilon_f^2}\left(\frac{\epsilon_f^2}{64\mathcal{V}_{\mathcal{H}}}N-\log\frac{\epsilon_f^2}{64\mathcal{V}_{\mathcal{H}}}-1\right)\\&=\frac{N}{2}+\frac{32\mathcal{V}_{\mathcal{H}}}{\epsilon_f^2}\log\frac{64\mathcal{V}_{\mathcal{H}}}{\epsilon_f^2e}.\end{aligned} \tag{2.4}$$

Applying results from Equation 2.4 into Equation 2.3 and substituting the original value of ϵ_f we prove the Theorem 2.4.2:

$$N\geq\frac{256}{\epsilon^2}\frac{1}{U_{min}^2}\left(\mathcal{V}_{\mathcal{H}}\log\left(\frac{256}{\epsilon^2}\frac{1}{U_{min}^2}\right)+\log\frac{8}{\delta}\right).$$

□

2.5 Application to Specific Pattern Spaces

Most state-of-the-art anomaly detectors assign an anomaly score to data points and then detect points based on a score threshold or present a ranked list to the user. Further, while not usually explained explicitly, the scores are often based on frequency estimates of patterns in some space $\mathcal{H}$ with rare patterns leading to higher anomaly scores. While RarePatternDetect was designed as a pure implementation of this principle, it is reasonable to expect that its sample complexity is related qualitatively to the sample complexity of other algorithms grounded in pattern frequency estimation.

In this section, we consider a number of pattern spaces underlying existing algorithms and show that the sample complexity of those spaces is small. The spaces are thus all learnable in the PAC-RPAD framework, which offers some insight into why existing algorithms often show strong performance even in high-dimensional spaces.

2.5.1 Conjunctions

Consider a space $\mathcal{X}$ over d Boolean attributes and a pattern space $\mathcal{H}$ corresponding to conjunctions of those attributes. This setup is common in the data mining literature, where each boolean attribute corresponds to an "item" and a conjunction corresponds to an "item set", which indicates which items are in the set. Rare pattern mining has been studied for such spaces and applied to anomaly detection [Adda et al., 2007, Szathmary et al., 2007]. In this case, the pattern space has a finite size 2^d and thus by Theorem 2.4.1 is efficiently PAC-RPAD learnable with sample complexity $O\left(\frac{1}{\epsilon^2}\left(d + \log\frac{1}{\delta}\right)\right)$. If we limit attention to conjunctions of at most k attributes, then the sample complexity drops to $O\left(\frac{1}{\epsilon^2}\left(k\log(d) + \log\frac{1}{\delta}\right)\right)$, which is sub-linear in d.

2.5.2 Halfspaces

Given a continuous space $\mathcal{X} \subseteq \Re^d$, a *half space pattern* is an oriented d-dimensional hyperplane. A data point satisfies a half space pattern if it is on the positive side of the hyperplane. The half-space mass algorithm [Chen et al., 2015] for anomaly detection operates in this pattern space. Roughly speaking, the algorithm assigns

a score to a point x based on the frequency estimates of random half spaces that contain x. The VC-dimension of d-dimensional half spaces is well known to be $d+1$ and hence this space is sample-efficient learnable in the PAC-RPAD model.

2.5.3 Axis Aligned Hyper Rectangles

For a continuous space $\mathcal{X} \subseteq \Re^d$, an axis-aligned hyper rectangle (bounded or unbounded) can be specified as a conjunction of threshold tests on a subset of the dimensions. The pattern space of axis-aligned rectangles are often implicit in decision tree algorithms, where each internal tree node specifies one threshold test.

The state-of-the-art anomaly detectors, Isolation Forest [Liu et al., 2008] and RS-Forest [Wu et al., 2014] (among others), are based on this space. The core idea of these algorithms is to build a forest of T random decision trees, where each node specifies a random threshold test. The trees are grown until either a maximum depth is reached or a leaf node contains only a single data point. Each leaf corresponds to an axis-aligned hyper rectangle. Given a point x, the algorithm can compute the leaf node it reaches in each tree, yielding a set of T hyper-rectangle patterns. The score for x is then based on the average score assigned to each pattern, which is related to the pattern frequency, dimension, and volume.

The VC-dimension of the space of axis parallel hyper rectangles in $\Re^d$ is $2d$ [Blumer et al., 1989]. Thus, the pattern space underlying Isolation Forest and RS-Forest is sample-efficient learnable in the PAC-RPAD model.

2.5.4 Stripes

A stripe pattern in $\Re^d$ can be thought of as an intersection of two parallel halfspaces with opposite orientations and can be defined by the inequalities: $a \leq w^\top x \leq a+\Delta$, where, $w \in \Re^d$, a and $\Delta \in \Re$ and Δ represents the width of the stripe. The stripe pattern space consists of the set of all such stripes.

The very simple, but effective, anomaly detector, LODA [Pevnỳ, 2016], is based on the stripes pattern space. The main idea of LODA is to form a set of T sparse random projections along some random directions in the subspaces of $\Re^d$ and then estimate a discretized 1D histogram based on the projected values. Each bin of each histogram can be viewed as corresponding to a stripe in the original $\Re^d$ space with

orientation defined by the direction of the random projection and location/width defined by the bin location/width.

To the best of our knowledge the VC-dimension of the stripes pattern space has not been previously derived. A loose bound can be found by noting that stripes are a special case of the more general pattern space of intersections of half spaces and then applying the general result for bounding the VC-dimension of intersections [Blumer et al., 1989], which gives an upper bound on the VC-dimension of stripes of $4\log(6)(d+1) = O(d)$. Hence, stripes are PAC learnable in the RPAD model.

2.5.5 Ellipsoids and Shells

Anomaly detectors based on estimating "local densities" often form estimates based on frequencies observed in ellipsoids around query points. In particular, an ellipsoid pattern in a d dimensional space can be represented by

$$(x-\mu)^\top A\ (x-\mu) \leq t,$$

where $t \in \Re$, $\mu \in \Re^d$ and A is a $d \times d$ positive definite symmetric matrix. An upper bound for the VC-dimension of ellipsoids in d-dimensional space $\Re^d$ is $(d^2+3d)/2$ [Auger and Doerr, 2011]. Hence the ellipsoid pattern space is sample-efficient learnable in the PAC-RPAD model. However, we see that the complexity is quadratic in d rather than linear as we saw above for spaces based on hyperplane separators. A related pattern space is the space of *ellipsoidal shells*, which is the analog of stripes for ellipsoidal patterns. An ellipsoidal shell in $\Re^d$ can be thought of as the subtraction of two ellipsoids with the same center and shape, but different volumes. That is, the shell is a region defined by $t \leq (x-\mu)^\top A(x-\mu) \leq t+\Delta$, where $\Delta \in \Re$ is the width. Shells naturally arise as the discretized level sets of multi-dimensional Gaussian density functions, which are perhaps the most widely-used densities in anomaly detection. We are unaware of previous results for the VC-dimensions of shells and show below that it is also $O(d^2)$.

Theorem 2.5.1. *The VC-dimension of the ellipsoidal shell pattern space in $\Re^d$ is upper bounded by $2\log(6)(d^2+3d+2)$.*

Proof. We can represent an ellipsoidal shell in $\Re^d$ as:

$$t \leq (x-\mu)^\top A\ (x-\mu)\ \leq t+\Delta. \tag{2.5}$$

Rewriting the equation of an ellipsoid in $\Re^d$:

$$\begin{aligned} & (x-\mu)^\top A\ (x-\mu)\ \leq t \\ \implies & x^\top A\ x - 2x^\top A\ \mu \leq t - \mu^\top A\ \mu \\ \implies & \sum_{i,j=1}^{d} A_{ij}x_i x_j - \sum_{i=1}^{d} 2(A\mu)_i x_i \leq t - \mu^\top A\mu \\ \implies & w^\top z \leq b. \end{aligned} \tag{2.6}$$

where, $w, z \in \Re^{d(d+1)/2+d}$. The vector w is a new parameter vector constructed from the original parameters. The matrix A gives $d(d+1)/2$ unique parameters, since A is symmetric, and the vector $A\mu$ gives another d parameters. The vector z is a new input constructed from the original input x, and $b = t - \mu^\top A\mu$.

Now, Applying result of Equation 2.6 to Equation 2.5 we get

$$\begin{aligned} & t - \mu^\top A\ \mu \leq w^\top z \leq t + \Delta - \mu^\top A\ \mu \\ \implies & t' \leq w^\top z \leq t' + \Delta. \end{aligned} \tag{2.7}$$

The Equation 2.7 is a representation of a stripe in $\Re^{d(d+1)/2+d}$. So, we apply the same approach from previous Section 2.5.4 i.e. the case of two halfspaces, which gives the upper bound of

$$4\log(6)(d(d+1)/2+d+1) = 2\log(6)(d^2+3d+2) = O(d^2)$$

. □

2.6 Experiments

The above results suggest one reason for why state-of-the-art anomaly detectors often perform significantly better than random, even on high-dimensional data. However, existing empirical work does not go much further in terms of providing an understanding of learning curves for anomaly detection. To the best of our knowledge, there has not been a significant study of how the empirical per-

formance of anomaly detectors varies as the amount of training/reference data increases. Typically empirical performance is reported for benchmark data sets without systematic variation of training set size, though other factors such as anomaly percentage are often varied. This is in contrast to empirical studies in supervised learning, where learning curves are regularly published, compared, and analyzed.

In this section, we present an initial investigation into empirical learning curves for anomaly detection. We are interested in the following experimental questions: 1) Will the empirical learning curves demonstrate the fast convergence predicted by the PAC-RPAD framework, and how is the convergence impacted by the data dimension and pattern space complexity? 2) How does the RPAD approach compare to a state-of-the-art detector based on the same pattern space on anomaly detection benchmarks? 3) In what ways do empirical learning curves for anomaly detection differ qualitatively from learning curves for supervised learning? While a complete empirical investigation is beyond the scope of this study, below we provide experiments that shed light on each of these questions.

2.6.1 Pattern Space and Anomaly Detector Selection

A recent large scale evaluation [Emmott et al., 2013] has shown that the Isolation Forest (IF) is among the top performing anomaly detectors across a wide range of benchmarks. This motivates us to focus our investigation on IF's pattern space of axis aligned hyper rectangles (see above). In order to allow for the complexity of this pattern space to be varied, we define *REC(k)* to be the space of all axis aligned hyper rectangles defined by at most k threshold tests on feature values.

The first step of IF is to construct a random forest of trees that are limited to a user-specified maximum leaf depth of k. Each tree leaf in the forest corresponds to a single pattern in REC(k). Thus, the first step of IF can be viewed as generating a random pattern space $\mathcal{H}_k \subseteq \text{REC}(k)$ that contains all leaf patterns in the forest. IF then operates by using training data to compute empirical frequencies $\widehat{P}(h)$ of patterns in $\mathcal{H}_k$ and then for any test point x computes an anomaly score based on those frequencies as follows (smaller is more anomalous):

$$\text{IF}(x) = \sum_{h \in \mathcal{H}_k[x]} d_h + c(\widehat{P}(h))$$

where, $d_h \leq k$ is the number of tests in pattern h and $c(h)$ is a function of the empirical frequency of h.[1]

In order to directly compare IF to our RPAD approach we will conduct multiple experiments, each one using a different randomly generated pattern space $\mathcal{H}_k$. We can then compute the scoring function corresponding to RarePatternDetect on those pattern spaces and compare to the performance of the IF scoring function. In particular, RarePatternDetect effectively assigns a score to x based on the minimum frequency pattern as follows:

$$\text{MIN}(x) = \min_{h \in \mathcal{H}_k[x]} \widehat{f}(h)$$

where the normalized frequency estimate $\widehat{f}(h)$ is normalized by a uniform density $U(h)$ over a region of bounded support defined by the training data. This normalizer is proportional to the volume of h and is easily computed. We see that compared to the IF scoring function with sums/averages over functions of each pattern in $\mathcal{H}_k[x]$, the MIN scoring function is only sensitive to the minimum frequency pattern. In order to provide a baseline in between these two, we also compare to the following alternative scoring function that averages normalized frequencies. This scoring function is given by

$$\text{AVE}(x) = \frac{1}{|\mathcal{H}_k[x]|} \sum_{h \in \mathcal{H}_k[x]} \widehat{f}(h).$$

AVE is included in order to observe whether averaging is a more robust approach to using normalized frequencies compared to MIN.

2.6.2 Learning Curve Generation

We generate learning curves using three existing anomaly detection benchmarks: **Covertype** [Wu et al., 2014]: $d = 10$ features, approximately 286k instances 0.9%

[1] In particular, $c(h)$ is an estimate of the expected number of random tests required to completely isolate training data points that satisfy h, which is a function of the number of training points that satisfy h [Liu et al., 2008]. This score is motivated by attempting to estimate the "isolation depth" of x, which is the expected length of a random path required to isolate a point. Intuitively smaller isolation depths indicate a more anomalous point since it is easier to separate from other points.

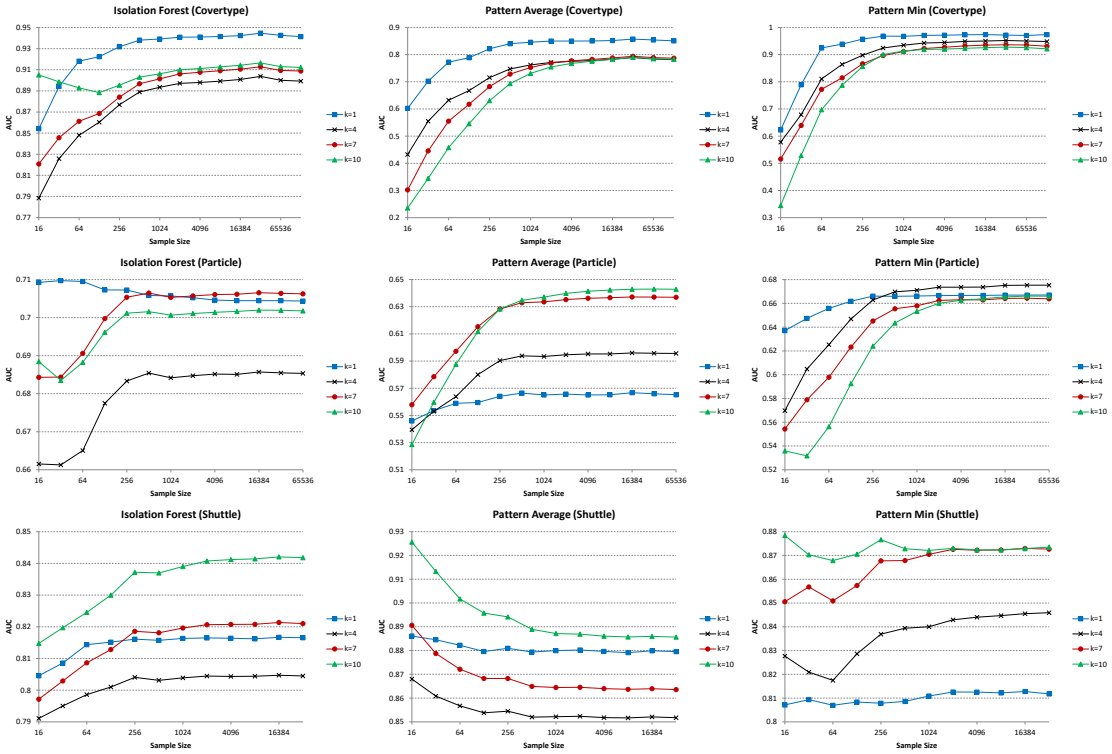

Figure 2.1: Learning Curves for the Three Scoring Methods (IF, AVE, MIN) with Varying Pattern Space Complexity k Over Three Benchmarks (Rows). MIN Represents the Main RPAD Approach Analyzed in this chapter.

anomalies.

Particle [Emmott et al., 2013]: $d = 50$ features, approximately 130k instances with 5% anomalies.

Shuttle [Emmott et al., 2013]: $d = 9$ features, approximately 58k instances with 5% anomalies.

These datasets were originally derived from UCI supervised learning benchmarks [Lichman, 2013] by treating one or more classes as the anomaly classes and subsampling at an appropriate rate to produce benchmarks with certain percentages of anomalies. We have conducted experiments on a number of additional benchmarks, which are not included for space reasons. These three data sets were selected as being representative of the qualitative learning curve types that we have observed. The data sets are divided into three sets of data: pattern generation data, training

data, and test data for which we ensure that the fraction of anomaly points in each data set is approximately the same as for the full benchmark.

Given a benchmark and a specified pattern complexity k, we generate learning curves for each algorithm as follows. First, we use IF to generate a random pattern space $\mathcal{H}_k$, based on the pattern generation data using a forest of 250 random trees. Next for each desired training set size, we sample a training set of the appropriate size and use that data to estimate frequencies over $\mathcal{H}_k$, which can be done efficiently by passing each data point through each tree. Next, the scores defined above for IF, MIN, and AVE are computed for each test instance based on the frequency estimates and the *Area Under the Curve (AUC)* of those scores is computed relative to the ground truth anomalies. This process is repeated 50 times for each training set size and the resulting AUCs are averaged to produce a final learning curve.

2.6.3 Results

Figure 2.1 gives learning curves for each benchmark (rows) and each of the anomaly detectors IF, MIN, and AVE (columns). Recall that MIN represents the main RPAD approach analyzed in this chapter. In each case, four learning curves are shown for pattern space complexities $k = 1, 4, 7, 10$ and training instances range from 16 to 2^{15}.

Convergence rate: Each learning curve for each algorithm and benchmark follows a trajectory starting at 16 training instances to a converged performance at 2^{15} points. We see that in all cases, convergence to the final performance occurs very quickly, in particular around 1024 samples. This convergence rate is not significantly impacted by the dimensionality of the data, which can be seen by comparing the results for Particle with $d = 50$ to the other data sets with $d = 9$ and $d = 10$. Rather we see that the convergence rate visibly depends on the pattern space complexity k, though to a relatively small degree. In particular, we see that the convergence for the simplest space $k = 1$ tends to be faster than for $k = 10$ across our experiments. These observations agree with our analysis. The VC-dimension of REC(k), which controls worst case convergence, is dominated by the limiting effect of k. These observations are consistent across additional benchmarks not shown here.

Relative Algorithm Performance: Here we focus on comparing the differ-

ent detectors, or scoring functions, in terms of their converged performance. For the Cover and Shuttle data sets we see that the converged performance of MIN is better or competitive than the converged performance of IF for all values of k. For the Particle data set, the IF scoring function outperforms MIN consistently by a small margin. This shows that despite its simplicity the RPAD approach followed by MIN appears to be competitive with a state-of-the-art detector based on the same pattern space. Experiments on other benchmarks, not shown, further confirm this observation.

The converged performance of AVE tends to be worse than both IF and MIN for Covertype and Particle and is slightly better on Shuttle. It appears that for these data sets (and others not shown) that averaging is not significantly more robust than minimizing and can even hurt performance. One reason for degraded performance is that AVE can be influenced by the cumulative effect of a large number of non-rare patterns, which may sometimes overwhelm the signal provided by rare patterns.

An interesting observation is that for each data set, the best performing pattern space (i.e. value of k) is usually the same across the different learning algorithms. For example, for Covertype, $k = 1$ yields the best converged performance for all scoring functions. This observation, which we also frequently observed in other data sets, suggests that the choice of pattern space can have a performance impact that is as large or larger than the impact of the specific scoring function used. To understand this, note that the performance of an anomaly detector depends on both the convergence of its scoring function and the match between the scoring function and the semantic notion of anomaly for the application. The pattern space choice has a large influence on the this match since it controls the fundamental distinctions that can be made among data points.

Qualitative Properties: The qualitative behavior of the learning curves exhibits a couple of nonintuitive properties compared to supervised learning curves. First, in supervised learning, we typically expect and observe that more complex hypothesis spaces converge to a performance that is at least as good as simpler spaces, though more complex spaces may underperform at small sample sizes due to variance. This does not appear to be the case for anomaly detection learning curves in general. For example, the performance of the simplest pattern space ($k = 1$) on Covertype converges to better performance than the more complex

spaces. This has also been observed in other data sets and does not appear to be due to premature termination of the learning curve. Rather, we hypothesize that this behavior is due to the mismatch between the anomaly detection scores and the semantic notion of anomaly in the benchmark. In particular, rare patterns in REC(1) are apparently a better indicator of the semantic notion of anomaly than some distracter rare patterns in REC(10). Indeed, it is straightforward to construct synthetic examples with such behavior.

Another nonintuitive aspect is that, in at least two cases, the learning curves consistently decrease in performance, while typically in supervised learning, ideal learning curves are non-decreasing. The most striking example is the performance of AVE on Shuttle, where all learning curves steadily decrease. After further analysis, this type of behavior again appears to be explained by the mismatch between the semantic notion of anomalies and the scoring function. In particular, the variance across different trials of the learning curve for large sample sizes is much smaller than for small sample sizes. It also turns out that the distribution of scoring functions generated for the small sample sizes is skewed toward solutions that better match the ground truth anomalies compared to the converged scoring function. Thus, the average performance for small sample sizes is better. We have observed this decreasing learning curve behavior in other data sets as well, though it is much more common for learning curves to increase.

2.7 Summary

This chapter is motivated by the observation that many statistical anomaly detection methods perform much better than random in high dimensions using relatively small amounts of training data. Our PAC-RPAD framework attempts to explain these observations by quantifying the sample complexity in terms of the complexity of the pattern spaces underlying such anomaly detectors. Our results mirror those in supervised learning by showing that the VC-dimension of the pattern spaces is the dominating factor that controls sample complexity. We showed that for several state-of-the-art detectors, the underlying pattern spaces had well-behaved VC-dimensions with respect to the data dimensionality, which offers a partial explanation for their good performance. On the empirical side, we investigated for the first time, to the best of our knowledge, learning curves for anomaly detection.

The experiments confirmed the fast convergence predicted by the theory. The results also suggest that our simple algorithm, which was shown to be PAC-RPAD, is competitive with the state-of-the-art algorithm Isolation Forest when using the same pattern space. Finally, the learning curves showed some interesting qualitative differences compared to supervised learning curves. In particular, the results highlight the importance of selecting a pattern space that is likely to be a good match to the semantic notion required for an application.

Chapter 3: Sequential Feature Explanations (SFE)

3.1 Introduction

As described in Chapter 1, anomaly detection is the problem of identifying anomalies in a data set, where anomalies are those points that are generated by a process that is distinct from the process generating "normal" points. Statistical anomaly detectors address this problem by seeking statistical outliers in the data. In most applications, however, statistical outliers will not always correspond to the semantically-meaningful anomalies. For example, in a computer security application, a user may be considered statistically anomalous due to an unusually high amount of copying and printing activity, which may have a benign explanation and hence is not a true and interesting anomaly. Because of this gap between statistics and semantics, an analyst typically investigates the statistical outliers in order to decide which ones are likely to be true anomalies and deserve further action.

Given an outlier point, an analyst faces the problem of examining the data associated with that point in order to make a judgment about whether it is an anomaly or not. Even when points are described by just tens of features this can be challenging, especially so when feature interactions are critical to the judgment. In practice, the situation is often much worse with points being described by thousands of features. In these cases, there is a significant risk that even when the detector passes a true anomaly to the analyst, the analyst will not recognize the key properties that make the point anomalous due to information overload. This means that, in effect, the missed anomaly rate of the overall system is a combination of the miss rates of both the anomaly detector and the analyst. Thus, one avenue for improving detection rates is to reduce the effort required by an analyst to correctly identify anomalies, with the intended side-effect of reducing the analyst miss rate. This anomaly detection pipeline is depicted in Figure 3.1.

In this chapter, we consider reducing the analyst's detection effort by providing them with explanations about why points were judged to be anomalous by the detector. Given such an explanation, the analyst can minimize his effort in the

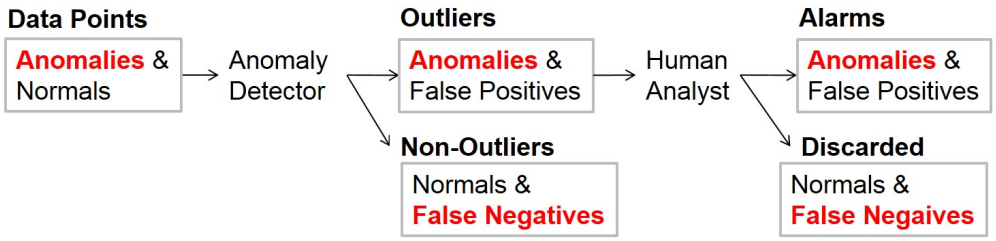

Figure 3.1: The anomaly detection pipeline addressed by this chapter (from left to right). The original data set contains both normal points and anomalies. Our goal is detect the true semantic anomalies. First an anomaly detector is applied to identify a set of statistical outliers, which are further analyzed by a human analyst in order to identify the true semantic anomalies. The false negatives (missed true anomalies) of the overall system are composed of two types of true anomalies: 1) anomalies that are considered to be statistically normal and are never presented to the human analyst, and 2) anomalies that are statistical outliers but are misidentified by the human analyst as normal. The first type of false negative can only be avoided by changing the anomaly detector and are not the focus of this chapter. The second type of false negative can be avoided by making it easier for analysts to detect true anomalies when presented with them. The focus of this chapter is to compute explanations of statistical outliers that reduce the effort required to detect such true anomalies.

investigation by focusing on information related to the explanation.

Our first contribution is to introduce an intuitive and simple form of explanation, which we refer to as *sequential feature explanations (SFEs)*. Given a point judged to be an outlier by a detector, an SFE for that point is an ordered sequence of features where the order indicates the importance with respect to causing a high outlier score. An SFE is presented to the analyst by incrementally revealing the features one at a time, in order, until the analyst has acquired enough information to make a decision about whether the point is an anomaly or not (e.g. in the security domain, a threat or non-threat). The investigative work of the analyst is roughly related to the number of features that must be revealed. Hence, the goal for computing SFEs is to minimize the number of features that must be revealed in order for the analyst to confidently identify true anomalies.

Our second contribution is to formalize the problem of optimizing SFEs and

to develop algorithms to solve that problem. We develop both greedy algorithms and an optimal algorithm based on branch-and-bound search. These algorithms can be applied to any density-based anomaly detector given that it is possible to (approximately) compute joint marginals of a detector's density function which is an operation that is supported by most commonly-used densities.

Our third contribution is to formulate a quantitative method for evaluating SFEs which allows for the comparison of different SFE algorithms. The key idea of the approach is to construct a simulated analyst for each anomaly detection benchmark using supervised learning and ground truth about which points are anomalies. The simulated analyst can then be used to evaluate the quality of SFEs with respect to the number of features that must be revealed to reach a specified confidence level. While in concept a human analyst could be used in the evaluation process, it is impractical to conduct a large number of such human-analyst experiments over a large number of benchmarks. Hence, in this chapter we focused on how to conduct large scale quantitative evaluations—necessarily using simplified analyst models. To the best of our knowledge this is the first methodology for quantitatively evaluating any type of anomaly explanation method.

Finally, our fourth contribution is to provide an empirical investigation of several methods for computing SFEs. Our primary evaluations use a recently constructed set of anomaly detection benchmarks derived from real-world supervised learning data. In addition we provide an evaluation on the standard KDD-Cup benchmark. The investigation leads to a recommended method and additional insights into the methods.

The remainder of the chapter is organized as follows. Section 3.2 reviews related work on explanations for both supervised learning and anomaly detection. Next, Section 3.3 presents the anomaly-detection framework used in this chapter. Section 3.4 then more formally presents the concept of SFEs and possible quality metrics. Section 3.5 formulates an optimal SFE computation method and then describes two greedy methods for computing SFEs. Section 3.7.2 shows a comparison between the optimal and greedy methods in achieving the optimal objective value. Section 3.6 introduces our quantitative evaluation framework for SFEs and finally Section 3.7 presents experiments evaluating the introduced methods within the framework.

3.2 Related Work

In both supervised learning tasks and unsupervised ones like anomaly detection, the problem of computing explanations has received relatively little attention. Related work in the area of supervised classification aims to provide explanations about why a classifier predicted a particular label for a particular instance. For example, a number of methods have been proposed to produce explanations in the form of relevance scores for each feature, which indicate the relative importance of a feature to the classification decision. Such scores have been computed by comparing the difference between a classifier's prediction score and the score when a feature is assumed to be unobserved [Robnik-Sikonja and Kononenko, 2008], or by considering the local gradient of the classifier's prediction score with respect to the features for a particular instance [Baehrens et al., 2010].

Other work has considered how to score features in a way that takes into account the joint influence of feature subsets on the classification score, which usually requires approximations due to the exponential number of such subsets [Strumbelj and Kononenko, 2010, Štrumbelj and Kononenko, 2013]. Since these methods are typically based on the availability of a class-conditional probability function, they are not directly generalizable to computing explanations for anomaly detectors. Our experiments, however, do evaluate a method, called Dropout, which is inspired by the approach of [Robnik-Sikonja and Kononenko, 2008].

The form of such feature-relevance explanations is similar in nature to our SFEs in that they provide an ordering on features. However, prior work has not explicitly considered the concept of sequentially revealing features to an analyst, which is a key part of the SFE proposal for reducing analyst effort.

Prior work on feature-based explanations for anomaly detection has focused primarily on computing explanations in the form of feature subsets. Such explanations are intended to specify the subset of features that are jointly responsible for an object receiving a high anomaly score. For example, Micenkova, et al. [Micenková et al., 2013] computed a subset of features such that the projection of the anomalous object onto the features shows the greatest deviation from normal instances. Some authors have referred to this explanation task as "outlying aspects mining" [Duan et al., 2014, Vinh et al., 2015]. For example, Duan et al. developed a method called OAMiner [Duan et al., 2014], which finds the most outlying subspace where the object of interest is ranked highest in terms of probability density

measure. Vinh et al. proposed a method called OARank [Vinh et al., 2015], which gives a feature ranking based on their potential contribution toward the outlyingness of a query point. One issue with these approaches is that the computation of an explanation is independent of the anomaly detector being employed, i.e., they don't consider the very anomaly detector that judged the instances as anomalous. Rather, they generate pseudo-training data and train a completely different model that has no information at all about the original anomaly detector. This is contrary to the goal of trying to explain why a particular anomaly detector judged a particular object to be anomalous. In contrast, the explanation approaches we consider in this chapter are sensitive to the particular anomaly detector.

Other work on computing feature-subset explanations [Dang et al., 2013] developed an anomaly detection system called LODI which includes a specialized explanation mechanism for the particular anomaly detector. A similar approach is considered by Dang, et al. [Dang et al., 2014], where the anomaly detection mechanism directly searches for discriminative subspaces that can be used for the purpose of explanation. In contrast, the explanation approaches we consider in this work can be instantiated for any anomaly detection scheme based on density estimation, which includes a large fraction of existing detectors.

Existing approaches for evaluating explanation methods in both supervised and unsupervised settings are typically quite limited in their scope. Often evaluations are limited to visualizations or illustrations of several example explanations [Baehrens et al., 2010, Dang et al., 2014] or to test whether a computed explanation collectively conforms to some known concepts in the corresponding data set [Baehrens et al., 2010], often for synthetically generated data. Prior work has not yet proposed a larger scale quantitative evaluation methodology for explanations, which is one of the main contributions of our work.

3.3 Anomaly Detection Formulation

We reiterate the anomaly detection problem which can be defined over a set of N data points $\{x_1, \ldots, x_N\}$, where each point x_i is an n dimensional real-valued vector. The set contains a mixture of *normal points* and *anomaly points*, where generally the normal points account for an overwhelming fraction of the data. In most applications of anomaly detection, the anomaly points are generated by

a distinct process from that of the normal points, in particular, a process that is important to detect for the particular application. The goal is to detect this process. For example, the data points may describe the usage behavior of all users of a corporate computer network and the anomalies may correspond to insider threats.

Since N is typically large, manual search for anomalies through all points is generally not practical. Statistical anomaly detectors address this issue by seeking to identify anomalies by finding statistical outliers. The problem, however, is that not all outliers correspond to anomalies, and in practice an analyst must examine the outliers to decide which ones are likely to be anomalies. We say that an analyst *detects* an anomaly when he or she is presented with a potential anomaly and is able to determine that there is enough evidence that the point is indeed an anomaly. The success of this approach depends on the anomaly detector's precision of identifying anomalies as outliers, and also on the analysts' ability to correctly detect anomalies. Without further assistance, an analyst may need to consider information related to all n features of an anomaly point during analysis. In many cases, considering this information thoroughly will be impossible and increases the chance of not detecting anomalies. Something which can be costly in many domains.

3.4 Sequential Feature Explanations

In order to reduce the analyst's effort for detecting anomalies, we propose to provide the analyst with *sequential feature explanations (SFEs)* that attempt to efficiently explain why a point was considered to be an outlier. A length k SFE for a point is an ordered list of feature indices $E = (e_1, \ldots, e_k)$, where $e_i \in \{1, \ldots, n\}$. The intention is that features that appear earlier in the order are considered to be more important to the high outlier score of a point (e.g. x_{e_1} is the most important). We will use the notation E_i to denote the set of the first i feature indices of E. Also, for any set of feature indices S and a data point x, we let x_S denote the projection of x onto the subspace specified by S.

Given an SFE E for a point x, the point is incrementally presented to the analyst by first presenting only feature x_{E_1}. If the analyst is able to make a judgement based on only that information then we are finished with the point.

Otherwise, the next feature is added to the information given to the analyst, that is, the analyst now sees x_{E_2}. The process of incrementally adding features to the set of presented information continues until the analyst is able to make a decision. The process may also terminate early because of time constraints; however, we don't study that case in this chapter.

For normal points, the incremental presentation of SFEs may not help the analyst more efficiently exonerate the points. In contrast, for anomalies, it is reasonable to expect that an analyst would be able to detect the anomalies more easily by considering a much smaller amount of information than they would have to without the SFE, which should reduce the chance of missed detections. To clarify further, we assume the analyst has the expertise to decide with certainty whether an instance is an anomaly from the entire set of features if given enough time. However, the effort required to determine the anomaly may be large if all the information is shown at the start. Further, it is assumed that if the minimal set of features responsible for the anomaly are shown, the effort may be reduced (they see the minimal number of feature interactions). Hence, we assume that the amount of analyst effort is a monotonically increasing function of the number of features considered. This motivates measuring the quality of an SFE for a target by the number of features that must be revealed to an analyst for correct detection. More formally, given an anomaly point x, an analyst a, and an SFE E for x, the *minimum feature prefix*, denoted $\mathrm{MFP}(x, a, E)$, is the minimum number of features that must be revealed to a, in the order specified by E, for a to detect x as an anomaly. The analyst may very well consult other information during an investigation. The hope is that simple and good explanations will allow the analyst to efficiently direct their attention to the key external information.

While MFP provides a quantitative measure of SFE quality, its definition requires access to an analyst. This complicates the comparison of SFE computation methods in terms of MFP. Section 3.6 addresses this issue and describes an approach for conducting wide evaluations in terms of MFP.

3.5 Explanation Methods

We now consider methods for computing SFEs for anomaly detectors. Prior work on computing explanations for anomaly detectors has either computed ex-

planations that do not depend on the particular anomaly detector used (e.g., [Micenková et al., 2013]) or used methods that were specific to a particular anomaly detector (e.g., [Dang et al., 2013]). We wish to avoid the former approach since intuitively an explanation should attempt to indicate why the particular detector being employed found a point to be an outlier. Considering the latter approach, we seek more general methods that can be applied more widely across different detectors. Thus, here we consider explanation methods for the widely-studied class of *density-based detectors*.[1]

Density-based detectors operate by estimating a probability density function $f(x)$ (e.g. a Gaussian mixture) over the entire set of N points and treating f as the density over normal points. This is reasonable under the usual assumption that anomalies are very rare compared to normal points. Points are then ranked according to ascending values of $f(x)$ so that the least normal objects according to f are highest in the order. Our methods do not assume knowledge of the form of f, but do require an interface to f that allows for joint marginal values to be computed. That is, for any subset of feature indices S and point x, we require that we can compute $f(x_S)$. For many choices of f, such as mixtures of Gaussians, these joint marginals have simple closed forms. If no closed form is available, then exact or approximate inference techniques (e.g., MCMC) may be employed.

It is worth noting that by considering SFE methods that depend on the anomaly detector being used, the performance in terms of MFP will depend on the quality of the anomaly detector as well as the SFE method. For example, consider a situation where the anomaly detector judges an anomaly point x to be an outlier for reasons that are not semantically relevant to why x is an anomaly. The SFE for x is not likely to help the analyst to more efficiently determine that x is an anomaly, since the semantically critical features may appear late in the ordering. While this is a possibility, it is out of the control of the SFE method. Thus, when designing SFE methods we will assume that outlier judgements made on f are semantically meaningful with respect to the application. This is a reasonable assumption since the SFE methods aim to explain the "reasoning" of the anomaly detector, regardless of whether or not the anomaly detector is bad (e.g. a mis-

[1] Our methods can actually be employed on the more general class of "score-based detectors" provided that scores can be computed given any subset of features. For simplicity, we focus on density-based detectors in this chapter, where the density function is used to compute scores.

match with what a human judges as important). Note that the SFE methods have no information with which to judge whether the anomaly detector is bad or not, so making the above assumption is the only reasonable assumption to make without further information. Addressing the mismatch between an anomaly detector and the semantically interesting anomalies is a fundamental problem on its own (presumably requiring some form of feedback from the analyst).

We now present the formulation of the SFE objective function and an exact method based on branch-and-bound search along with our two main classes of greedy SFE methods which we refer to as *marginal methods* and *dropout methods*.

3.5.1 SFE Objective Function

We model the SFE objective function from the perspective of an analyst. In particular, we view the analyst as a Bayesian classifier that assumes normal points are generated according to f and that anomalies have a uniform distribution u over the support of the feature space, which is a reasonable assumption in the absence of prior knowledge about the anomaly distribution. Given a point x, an SFE E, and a number of revealed features i, such an analyst would make the decision of whether x is an anomaly or not by comparing the likelihood ratio $\frac{f(x_{E_i})}{u(x_{E_i})}$ to some threshold. Since u is assumed to be uniform, this is equivalent to comparing the joint marginal $f(x_{E_i})$ to a threshold. Intuitively this means that if our goal is to cause the analyst to quickly decide that x is an anomaly, then we should choose an E that yields small values of $f(x_{E_i})$, particularly for small i.

In order to formalize the above intuition into an objective function, we first need to more precisely specify the thresholds used in the above analyst model. It is important to note that any choice of threshold on $f(x_{E_i})$ should depend on i, since larger values of i tend to lead to smaller density values. For this purpose, we introduce the *threshold function* $\tau_i(E, \alpha)$, where E is an SFE, i is an SFE index, and $0 < \alpha < 1$ is a percentile parameter. In particular, $\tau_i(E, \alpha)$ is the α-percentile density value in the set $\{f(x_{E_i}) : x \in D\}$, where D is the set of all data instances. That is, a fraction α of the data points will have marginal density $f(x_{E_i})$ less than $\tau_i(E, \alpha)$. If α is equal to the expected rate of anomalies, then this is a natural way to model the analyst detection threshold. In practice, we do not know the value of α and thus our approach below will assume a prior $p(\alpha)$, which in turn implies a

prior over thresholds given by $\tau_i(E, \alpha)$. In our implementation, we use a discrete distribution over α that assigns non-zero probability to reasonably small values.

Given the above threshold function, we can now formulate our SFE objective function. For this purpose, given an instance x, SFE E, and percentile α, we define the *Smallest Prefix (SP)* of E as the smallest number i of top features in E that would cause the analyst to detect x using threshold $\tau_i(E, \alpha)$, i.e.

$$\mathrm{SP}(x, E, \alpha) = min\{i : f(x_{E_i}) < \tau_i(E, \alpha)\} \tag{3.1}$$

Note that if no value of i satisfies the inequality, then we define $\mathrm{SP}(x, E, \alpha) = n$. Intuitively, $\mathrm{SP}(x, E, \alpha)$ is the amount of the modeled analyst's effort (i.e. the number of features) required to detect x as an outlier when using SFE E and assuming threshold corresponding to α. Since we do not know the true value of α, our final objective is the expected value of SP, denoted by ESP, with respect to $p(\alpha)$, that is,

$$\mathrm{ESP}(x, E) = \sum_{\alpha} \mathrm{SP}(x, E, \alpha) p(\alpha) \tag{3.2}$$

recalling that we are assuming $p(\alpha)$ is a discrete distribution. Given this objective function, we can now specify the SFE optimization problem for a given instance x, which is to compute the SFE with minimal ESP:

$$\operatorname*{argmin}_{E} \mathrm{ESP}(x, E) \tag{3.3}$$

To understand the computational complexity of this problem, consider the associated decision problem SFE-Decide.

SFE-Decide: *Does there exist an SFE E that satisfies* $\sum_{\alpha} \mathrm{SP}(x, E, \alpha) p(\alpha) \leq t$

This problem turns out to be computationally hard.

Theorem 3.5.1. SFE-Decide *is NP-hard.*

The proof is in Appendix A.1. This hardness result motivates us to consider two optimization approaches described below. First, we consider greedy algorithms that are guaranteed to be efficient, but without any guarantees with respect to

optimality. Second, we design an anytime branch and bound search, which is guaranteed to find optimal solutions if run long enough and also provides bounds on the sub-optimality of the solution returned at any time. Our experiments will compare both types of approaches.

3.5.2 Greedy Algorithms

In this section we first describe two greedy approaches and then an optimal algorithm based on branch and bound search to solve the optimization problem of Equation 3.3.

3.5.2.1 Marginal Methods

A natural greedy approach to optimizing the above objective is to greedily construct an E that attempts to minimize $f(x_{E_i})$ as a function of i as quickly as possible. This leads to our first SFE method, called *sequential marginal (SeqMarg)*. The SeqMarg method adds one feature to the k-length SFE $E = (e_1, \ldots, e_k)$ at a time, at each step adding the feature that minimizes the joint marginal density with the previously-selected features. This gives a nice way to stop if the analyst can decide whether x is anomaly just from the first k features, otherwise we need to keep adding a feature at each iteration. More formally, SeqMarg computes the following explanation:

$$\textbf{SeqMarg:} \qquad e_i = \arg\min_{j \in \overline{E}_{i-1}} f(x_{E_{i-1}}, x_j)$$

where $\overline{S}$ is the complement of set S. SeqMarg requires $O(kn)$ joint marginal computations in order to compute an explanation of length k. Note that due to the inherent greediness of SeqMarg, x_{E_i} may not necessarily be the optimal set of i features for minimizing f. Rather, if the goal were to optimize for a particular value of i, we would need to consider all $O(n^i)$ feature subsets of size i. However, our problem formulation does not provide us with a target value of i, and thus SeqMarg offers a more tractable approach that focuses on minimizing f as quickly as possible in a greedy manner.

It is worth noting that if our objective function were submodular, then SeqMarg

would provide an approximation guarantee [Krause and Golovin, 2014]. However, our objective function $f(x_E)$ is not a submodular function of the set E. Intuitively, a submodular function is one that exhibits diminishing returns, meaning that if $E' \subseteq E$ then adding an item to E' will improve the objective at least as much as adding the item to E. Unless we place strict restrictions on f this is not the case in general, due to interactions between features. For example, consider a density f for which the feature values of x_i and x_j are very rare when considered together, but common individually. If we let $E' = \{k\}$ and $E = \{k, i\}$, then it is easy to design f so that adding j to E results in a larger drop in the marginal f value than adding j to E'. Because, adding j to E would cause more rarity than adding j to E' even though E' is a subset of E. Indeed using this type of construction it is easy to construct examples where SeqMarg can be arbitrarily far from optimal. However, in practice we find that it tends to work very well across a wide range of problems.

In addition to SeqMarg we also consider a computationally cheaper alternative, called *independent marginal (IndMarg)*, which only requires the computation of individual marginals $f(x_i)$. This approach simply selects an explanation E for x by sorting the features in increasing order of $f(x_i)$. This only requires $O(n)$ marginal computations for computing an explanation of any length. IndMarg offers a computationally cheaper alternative to SeqMarg, but fails to capture joint feature interactions. For example, SeqMarg will select e_i in a way that optimizes the joint value when combined with previous features E_{i-1}. Instead, IndMarg ignores interactions with previously-selected features. Thus, IndMarg serves as a baseline for understanding the importance of accounting for joint feature interactions when computing explanations.

3.5.2.2 Dropout Methods

The next two methods are inspired by the work of Robnik-Sikonja and Kononenko [Robnik-Sikonja and Kononenko, 2008] on computing feature-relevance explanations for supervised classifiers. In their work, the relevance score for a feature is the difference between the classification score when the feature is provided to the classifier and the classification when the feature is omitted ("dropped out"). The analogous approach for anomaly detection is to score features according to the

change in the density value when the feature is included and when the feature is not included, or marginalized out. This yields the first dropout method, referred to as *independent dropout (IndDO)*: given a point x, each feature is assigned a score of $f(x - x_i) - f(x)$, where we abuse notation and denote the removal of x_i from x by $x - x_i$. Intuitively, features with larger scores are ones that make the point appear most normal when removed. The SFE E is then obtained by sorting features in decreasing order of score.

We can also define a sequential version of dropout, by following the same recipe we considered for IndMarg versus SeqMarg. Let the *sequential dropout (SeqDO)* be defined as follows:

$$\textbf{SeqDO: } e_i = \arg \max_{j \in \overline{E}_{1:i-1}} f(x_{\overline{E}_i} - x_j)$$

This approach requires the same number of marginal computations as SeqMarg. This algorithm can be viewed as a dual of SeqMarg in that it measures the contribution of feature sets according to how much more normal a point looks after their removal, whereas SeqMarg measures how abnormal a point looks with only those features included.

3.5.3 Branch and Bound Search

We now develop an optimal algorithm for optimizing ESP based on branch and bound search. This algorithm will search through the exponentially large space of SFEs, while attempting to soundly prune as much of the space from consideration as possible. Ideally, the pruning will result in finding the optimal SFE very quickly, though in the worst case the procedure may need to enumerate an exponentially large set of SFEs. This worst case behavior is unavoidable under standard complexity assumptions due to the NP-hardness of the optimization problem. Importantly, our branch and bound procedure can be run in an "anytime" mode, where it can be terminated at any point to return the best solution found so far.

The branch and bound search operates over a rooted tree, where a node at depth i is labeled by a length i feature sequence E_i, which represents a partial SFE. Since there will be a one-to-one correspondence between tree nodes and feature subsequences, we will abuse notation and refer to nodes by the corresponding

Algorithm 1 Branch and Bound Search

Input: $x \in \mathbb{R}^d$: input instance, d: dimension, MAX: maximum number of nodes to explore
Output: SFE for x

```
root := CreateEmptyNode(x) // Creates node with empty feature list.
Q.Insert(root)
BestNode := root
NodesExplored := 0
while Q.NotEmpty() AND NodesExplored < MAX do
    NodesExplored := NodesExplored + 1
    node := Q.GetSmallestUpperBoundNode() // Expand node with best upper
    bound.
    if node.LB < bestNode.UB then
        for f ∈ (1 : d) AND f ∉ node.RankedFeatureList do
            child := copy(node)
            child.AppendFeature(f)
            child.UpdateBounds()
            if BestNode.UB > child.UB then
                BestNode := child
            end
            Q.Insert(child)
        end
    end
end
return BestNode.RankedFeatureList
```

sequences. The root of the tree is the null sequence E_0 and the leaves of the tree are complete $SFEs$ (length n sequences). The children of a node E_i at depth i are all subsequences of length $i+1$ that extend E_i by one feature that is not already in E_i. Based on this search space definition, the descendant leaves of a node E_i, denoted by $l(E_i)$, are all SFEs that have E_i as a prefix.

The key idea of branch and bound search is to view each node E_i as representing the set of possible solutions $l(E_i)$ and to compute upper and lower bounds on the objective value for those possible solutions. In particular, the upper (lower) bound

computed for a node E_i, denoted by $U(E_i)$ ($L(E_i)$), are values that bound the ESP of any SFE in $l(E_i)$ from above (below). Given these bounds, we can prune away a node E_i and all of its descendants if its lower bound $L(E_i)$ is greater than the upper bound $U(E')$ of some other node E' already considered during the search. This is a sound pruning strategy since it must be the case that $l(E')$ contains an SFE whose ESP is at least as good as the best SFE in $l(E_i)$ (recall that smaller ESP is better). Below we specify the search strategy for expanding nodes, followed by a description of the lower and upper bound computations. Pseudo code for the overall search is given in Algorithm 1.

Search Strategy. The search iteratively expands a set of fringe nodes, which is initialized to the single root node E_0 (null sequence). This set of nodes is maintained in a priority queue in Algorithm 1. At any point in the search, we maintain the upper bound for each fringe node as well as the best (smallest) upper bound U^* discovered so far during the search. Each iteration begins by selecting the fringe node E with the smallest upper bound and then computing the corresponding lower bound $L(E)$. If $L(E) \geq U^*$ then we remove/prune E from the set of fringe nodes, which effectively prunes all SFEs in $l(E)$ from consideration. If $L(E) < U^*$ then it is possible that an SFE in $l(E)$ is optimal and we expand E by adding all of its children to the set of fringe nodes and removing E. The upper bound for each newly added child is computed and the iteration proceeds. Importantly, as described below, our upper bound computation produces an SFE that achieves the upper bound value. Thus, at any point our search also keeps track of the best such SFE discovered so far.

The algorithm terminates and returns the best SFE found so far when a specified limit on the number of expanded nodes is reached. The algorithm will also terminate if it finds an SFE S whose ESP value is at least as good as the lower bound of any fringe node. In that case, S is guaranteed to be an optimal SFE. The algorithm will always terminate in finite time since there are a finite number of SFEs and each iteration expands a new node of the search tree.

Upper Bound Computation. We compute the upper bound for a node E_i by running the greedy sequential marginal algorithm (Section 3.5.2.1) starting with E_i in order to select the remaining features of a complete SFE. The ESP (Equation 3.2) computed for the resulting SFE is taken to be the upper bound since the optimal SFE under E_i will be no worse (smaller ESPs are better).

Lower Bound Computation. The lower bound for node E_i must bound the ESP of any SFE in the set $l(E_i)$. Our first step to such a bound is based on expanding the definition of ESP according to (3.2) and interchanging the order of minimization over SFEs and summation/expectation over α values.

$$\min_{E \in l(E_i)} \mathrm{ESP}(x, E) = \min_{E \in l(E_i)} \sum_{\alpha} \mathrm{SP}(x, E, \alpha) p(\alpha) \geq \sum_{\alpha} \min_{E_\alpha \in l(E_i)} \mathrm{SP}(x, E_\alpha, \alpha) p(\alpha) \tag{3.4}$$

The inequality follows due to the fact that the right-hand side of the inequality minimizes the SP for each value of α, rather than constraining the SFE to be the same for each value of α as in the original optimization problem. Unfortunately, finding the optimal E_α for each term of the right-hand side is still computationally intractable due to the need to enumerate over SFEs in $l(E_i)$. Thus, we instead compute an efficiently computable lower bound for each such term by restricting our attention to only the feature sequences in E_i. In particular, consider a single term

$$t_\alpha = \min_{E_\alpha \in l(E_i)} \mathrm{SP}(x, E_\alpha, \alpha) \tag{3.5}$$

noting that E_i is a prefix of all SFEs under consideration. If $t_\alpha = j \leq i$ then there is a $j \leq i$ such that $f(x_{E_j}) < \tau_j(E_i, \alpha)$ and we can easily compute this value by considering each $j \leq i$. If, on the other hand, $t_\alpha > i$ then there will not be any $j \leq i$ such that $f(x_{E_j}) < \tau_j(E, \alpha)$, which we can easily test. In this case, we can lower bound the value of t_α by i. Putting this all together, we use the following lower bound on each term:

$$t_\alpha \geq \hat{t}_\alpha = \min(\{j \ : \ j \leq i, f(x_{E_j}) < \tau_j(E, \alpha)\} \cup \{i\}) \tag{3.6}$$

and we compute the overall lower bound as $\sum_\alpha \hat{t}_\alpha p(\alpha)$.

It is important to note that our overall search procedure is guaranteed to do no worse than SeqMarg. This is because, the upper bound computation for the root node exactly corresponds to running SeqMarg starting from the empty sequence. Thus, the best SFE maintained by the search will always be at least as good as that produced by SeqMarg.

3.6 Framework for Evaluating Explanations

There are at least two challenges involved in evaluating anomaly-explanation methods. First, compared to supervised learning, the area of anomaly detection has many fewer established benchmark data sets, particularly benchmarks based on real-world data. Second, given a benchmark data set, it is not immediately clear how to quantitatively evaluate explanations, since the benchmarks do not come with either ground truth explanations or analysts.

Here we describe an evaluation framework that addresses both issues. We address the first issue by drawing on recent work on constructing large numbers of anomaly detection benchmarks based on real-world data. We address the second issue by using supervised learning to construct a simulated analyst that can be applied to quantitatively evaluate our explanations in terms of MFP. Below we expand on both of these points.

3.6.1 Anomaly Detection Benchmarks

Recent work [Emmott et al., 2013] described a methodology for systematically creating anomaly detection benchmarks from supervised learning benchmarks (either classification or regression). Given the huge number of real-world supervised learning benchmarks, this allows for a correspondingly huge number and diverse set of anomaly detection benchmarks. Further, these benchmarks can be created to have controllable and measurable properties, such as anomaly frequency and "clusteredness" of the normal and anomalous points. We briefly sketch the main idea. Given a supervised classification data set, called the *mother set*, the approach selects one or more of the classes to represent the anomaly class, with different choices giving rise to different properties of the anomaly class. The union of the other classes represents the normal class. Individual anomaly detection benchmarks are then created by sampling the normal and anomaly points at specified proportions.

Table 3.1 gives a summary of the benchmarks from Emmott et al.[2013] used in our experiments. For example, the UCI data set shuttle was used as a mother set to create 1600 distinct anomaly detection benchmarks. The number of points in the shuttle benchmarks range from 3570 to 9847. The number of anomalies ranges from 8 to 984.

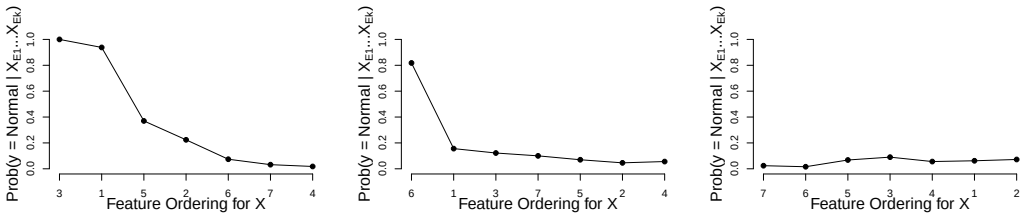

Figure 3.2: Analyst Certainty Curves. These are example curves generated using our simulated analyst on anomalies from the Abalone benchmark using SFEs produced by SeqMarg. The x-axis shows the index of the feature revealed at each step and the y-axis shows the analyst certainty about the anomalies being normal. The leftmost curve shows an example of where the analyst gradually becomes certain that the point is anomalous, while the middle curve shows more rapidly growing certainty. The rightmost curve is an example of where the analyst is certain of the anomaly after the first feature is revealed and remains certain.

3.6.2 Simulated Analyst

We consider modeling an analyst as a conditional distribution of the normal class given a subset of features from a data point. More formally we model the analyst as a function $A(x, S) = P(\text{normal} \mid x_S)$, which returns the probability that point x is normal considering only the features specified by the set S. We describe how we obtain this function in our experiments below. Given this function, a point x, and an SFE E for x, we can generate an *analyst certainty curve* that plots the analyst's certainty after revealing i features, that is, $A(x, E_i)$ versus i. Figure 3.2 shows an example of three analyst curves from our experiments using our simulated analysts on a benchmark computed from the UCI Abalone dataset. Each curve corresponds to a different anomaly in the data set using explanations computed by SeqMarg. We see that the different anomalies lead to different rates at which the analyst becomes certain of the anomaly, that is, certain that the point is not normal.

Recall that our proposed quality metric $\text{MFP}(x, a, E)$ measures the number of features that must be revealed to analyst a according to SFE E in order for a to detect an anomaly x. Evaluating this metric requires that we define the conditions under which the analyst detects x. We model this by associating an analyst with a detection threshold $\tau \in [0, 0.5]$ and saying that a detection occurs if $A(x, E_i) \leq \tau$,

that is, the probability of normality becomes small enough. We will denote this analyst by $a(\tau)$. Given an $a(\tau)$ we can then compute the MFP for any anomaly point by recording the number of features required for the analyst certainty curve to first drop below τ.

Of course, there is no a priori basis for selecting a value of τ. Thus, in our experiments, we consider a discrete distribution over values for τ, $P(\tau)$, which models a range of reasonable thresholds. Given this distribution, we report the expected MFP—the expected value of $\text{MFP}(x, a(\tau), E)$—as the quantitative measure of SFE E for anomaly x. In our experiments we define $P(\tau)$ to be uniform over the values 0.1, 0.2, and 0.3, noting that our results are consistent across a variety of reasonable choices for this distribution.

It remains to specify how we obtain the analyst function $A(x, S)$. Since our anomaly detection benchmarks are each derived from a mother classification data set [Emmott et al., 2013], we can construct a training set over those points for the anomaly and normal classes. Basically, a mother training set is a conversion from some well known classification and regression datasets by following some reasonable criteria described in [Emmott et al., 2013]. An anomaly detection benchmark is then created by subsampling instances that satisfy some criteria such as difficulty level, anomaly rate etc. Hence, the mother training set actually contains all the information for that particular dataset. Given this training set, one approach to obtaining the analyst would be to learn a generative model, or joint distribution $P(\text{normal}, x)$, which could be used to compute $A(x, S)$ by marginalizing out features not included in s. However, such generative models tend to be much less accurate in practice compared to discriminative models. On the other hand, learning a discriminative model $P(\text{normal} \mid x)$ does not directly support computing the probability for arbitrary subsets of x as we require. While heuristics have been proposed for this purpose (e.g. Robnik-Sikonja and Kononenko [Robnik-Sikonja and Kononenko, 2008]) we have found them to be unreliable when applied widely. Thus, in this work we follow a brute-force approach. We simply pre-learn an individual discriminative model for each possible subset of features up to a maximum size k. Evaluating $A(x, S)$ then simply requires evaluating the model associated with the subset S.

When the number of features or number of data points is very large, it may not be possible to pre-learn all possible subsets. In such cases, one option is to learn

Table 3.1: Summary of the benchmark datasets

Mother Set	Original Problem Type	# of Features	# of Benchmarks	# of Points per Benchmark (range)	# of Anomalies per Benchmark (range)
magic.gamma	Binary	10	1600	600 - 6180	5 - 618
skin	Binary	3	1200	10 - 9323	1 - 932
shuttle	Multiclass	9	1600	3570 - 9847	8 - 984
yeast	Multiclass	8	1600	70 - 1000	1 - 97
abalone	Regression	7	1600	580 - 2095	1 - 209
concrete	Regression	8	1200	190 - 1000	1 - 51
landsat	Multiclass	36	1600	561 - 5102	4 - 510
particle	Binary	50	400	371 - 8563	7 - 857

and cache models on the fly as they are needed during evaluation (each model would be learned only once). We used this approach for the KDD-Cup results reported in our experiments.

In all of our experiments, we use the Regularized Random Forests (RRFs) [Deng, 2013] as the classifier for the analyst model. The RRF model was selected for two primary reasons. First, RRFs are well known to provide high accuracies that are competitive with the state of the art across a wide range of classification problems [Fernández-Delgado et al., 2014]. Second, RRFs are relatively efficient to train, which is important to our study, since we must train one RRF for each possible subset of features (up to some maximum size). We trained RRFs composed of 500 trees using 10-fold cross-validation in order to tune the RRF regularization parameters.

It is worth noting that our evaluation framework is potentially sensitive to the choice of analyst model, since different models will have different biases. It was beyond the practical scope of this first study to replicate all experiments using a qualitatively different model.

3.7 Empirical Evaluation

We now present our empirical evaluation on anomaly detection benchmarks from Emmott et al. [Emmott et al., 2013] and the commonly used KDDCup anomaly detection benchmark.

3.7.1 Anomaly Detector

For all of our experiments, we have chosen to use the Ensemble Gaussian Mixture Model (EGMM) as the anomaly detector. This detector was first described in Emmott et al. [Emmott et al., 2013] and was shown to be a competitive density-based approach across a wide range of benchmarks. EGMM is based on learning a density function $f(x)$ represented as an ensemble of Gaussian mixture models (GMMs). The approach independently learns M GMM models by training each one using the Expectation-Maximization (EM) procedure on bootstrap replicates of the data set. Then it discards the low-likelihood GMMs (if any) and retains others based on a pre-specified threshold. The number of components of the GMMs is varied across the ensemble. In our experiments, the ensembles included 45 GMMs, 15 each using 3, 4, and 5 components. The final EGMM density $f(x)$ is simply a uniform mixture of the densities of the retained GMMs. The EGMM approach addresses at least two pitfalls of using single GMM models. First, EM training can sometimes produce poor models due to bad local optima. Second, it is difficult to select the best number of components to use for a single model. EGMM gains robustness by performing model averaging over the variations.

One advantage of using the EGMM model is that it is straightforward to derive closed forms for the marginal density computations required by our explanation methods. In particular, the overall EGMM density f can be viewed as a single large GMM model containing a mixture of all components across the ensemble. Since individual Gaussians have simple closed forms for marginal densities [Bishop et al., 2006], we can easily obtain closed forms for the mixture. It is worth noting that closed forms can also be derived for EGMM marginals when the data points are transformed by linear projections to reduce dimensionality (e.g. principle component analysis).

Another point to note is that we didn't consider discrete-valued features directly, primarily because we used anomaly detectors that expected numeric fea-

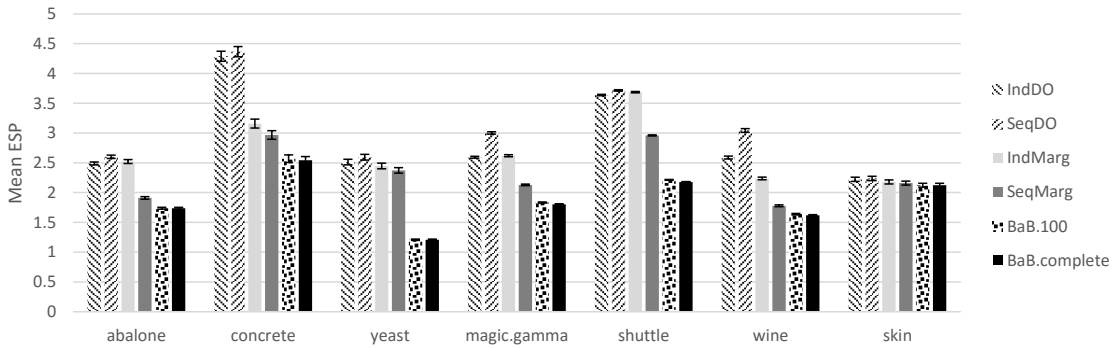

Figure 3.3: The bars show the Expected Smallest Prefix (ESP) achieved by different methods averaged across top 10% highly ranked ground truth anomalies in all the benchmarks. 95% confidence intervals are also shown.

tures. However, our approaches are only dependent on the marginal distributions of the original density f. If f is a joint distribution over both continuous and discrete variables and we can get any marginal of f on demand, our methods apply directly without modification.

3.7.2 Empirical Comparison of Greedy vs. BaB

Before presenting our full evaluation, we first compare the performance of greedy versus BaB (Branch and Bound from Section 3.5) with respect to optimizing ESP (Equation 3.3). For this experiment, we need to specify two parameters: the number of nodes to expand for BaB and a distribution over percentile values α (1 to 100) used to define ESP. We assign the number of nodes to expand as 100 (hence calling the method BaB.100), and choose an uniform prior probability over the first 10 values of α and others as 0, i.e.,

$$P(\alpha) = \begin{cases} 1/10 & \text{if } 1 \leq \alpha \leq 10 \\ 0 & \text{otherwise} \end{cases} .$$

The idea is to put higher emphasis on the low-percentile values as they should convey more accurate signals than others. Finally, we compute the best ESP (BaB.100) and the optimal ESP (BaB.complete) for each of the highly-ranked anomaly points and average across the datasets respectively.

Similarly, we compute the ESP of equation 3.2 using the SFEs obtained from the greedy methods (SeqMarg, IndMarg SeqDrop and IndDrop) of Section 3.5.2.1. The average of these ESPs are then plotted along with the ESPs of the BaB methods in Figure 3.3. We first observe that BaB.complete and BaB.100 achieve nearly identical results. This shows that a relatively small amount of search is needed by our BaB procedure to achieve nearly optimal ESP. Thus, in our full evaluation (Section 3.7.3) we will focus on the computationally cheaper BaB.100 as the representative BaB method. Second, we observe that the greedy sequential Marginal (SeqMarg) method achieved very close ESP to the BaB methods with the exception of the Yeast and Shuttle benchmarks. The other greedy methods more frequently perform significantly worse than the BaB methods. This shows that SeqMarg could be expected to be a computationally-efficient and effective alternative to BaB for optimizing ESP.

3.7.3 Evaluation on Benchmark Data Sets

We run our evaluation on anomaly detection benchmarks from Emmott et al. [Emmott et al., 2013], derived from seven UCI mother sets. A summary of the benchmarks are given in Table 3.1. There are over 10,000 benchmarks in total, which contain a number of points ranging from 10 to 9847 and a number of anomalies ranging from 1 to 984. An EGMM model was fit for each of the benchmarks to serve as the anomaly detector, and RRF models were trained for each mother set on all possible feature subsets. For this first part of our evaluation, we have chosen to focus on benchmarks with relatively small dimensionality in order to allow for a large scale study, which requires training large numbers of EGMM models (over 10,000) and RRF analyst models. All data from these experiments, including the analysts' models, will be made publicly available.

We evaluated seven methods for computing SFEs. These included the five methods from Section 3.5: SeqMarg, IndMarg, SeqDO, IndDO and BaB.100 (branch-and-bound was restricted to 100 nodes of exploration). In addition, we evaluated a random explanation method. In the random case, we report the average performance across 100 randomly generated SFEs. Finally, in order to provide a lower bound on attainable performance (lower MFP is better) we consider an optimal-oracle method, *OptOracle*. This method is allowed access to the simulated analyst

and for each number of features i computes the optimal feature subset of size i. More formally, for each value of i, OptOracle finds the feature subset S_i that minimizes the analyst's conditional probability $P(\text{normal} \mid x_{S_i})$. The MFP achieved by OptOracle for an anomaly x, given a particular analyst threshold τ (recall Section 3.6), is the minimum value of i such that $P(\text{normal} \mid x_{S_i}) < \tau$. Note that OptOracle is not constrained to produce "sequential explanations"—rather, OptOracle can produce an S_i that does not necessarily contain S_{i-1}. This gives OptOracle an additional advantage compared to the other methods which are constrained to produce SFEs. Clearly, OptOracle represents an optimistic bound on the performance of any SFE method that is evaluated with respect to the simulated analyst.

For each of the 10,000 benchmarks, we used the corresponding EGMM model to rank the points. For the true anomaly points ranked in the top 10%, we computed SFEs using each of the six methods. This choice is an attempt to model the fact that, in actual operation, only highly ranked anomalies will be presented to the expert. Note that while developing and evaluating explanation methods that target false positives is certainly an interesting direction, it is a large enough problem to warrant its own entire study. For that problem, the objective is presumably one of exoneration where the goal for the explanation system is to accurately determine when revealing additional features of an SFE is unlikely to raise further suspicion according to its model. This requires new developments and new approaches for evaluation. Finally, the expected MFP was computed for each SFE using a distribution over analyst thresholds that was uniform over the values 0.1, 0.2, and 0.3. For each mother set, we then report the average MFP across the anomalies derived from that mother set. These average MFPs are shown in Figure 3.4 along with 95% confidence intervals.

We first note that our observations below are not sensitive to the choice of what percentage of the top anomalies to focus on. While these results focus on the top 10%, we have also compiled results for other percentage points including using all anomalies. The main observations are qualitatively similar across all of these choices.

Comparison to Random and OptOracle. We observe in Figure 3.4 that all of the SFE methods outperform random explanations and often do so by a large margin. Comparing to OptOracle we see that, for three benchmarks—CONCRETE, YEAST, and WINE—the lower bound provided by OptOracle is significantly better

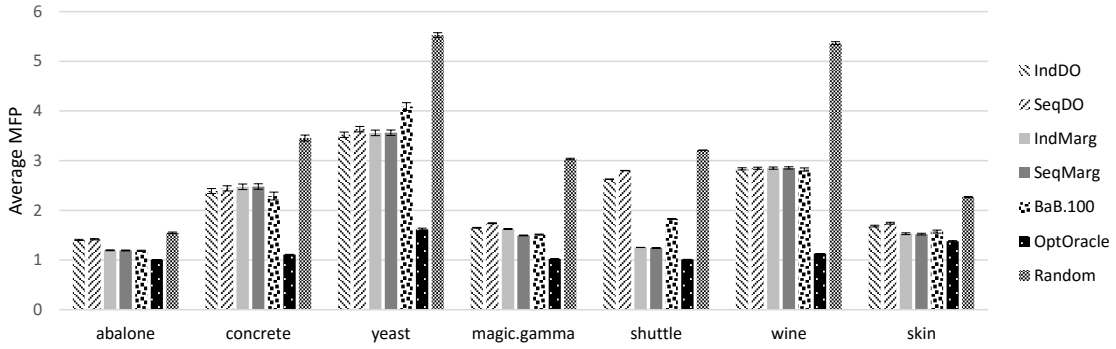

Figure 3.4: Performance of explanation methods on benchmarks. Each group of bars shows the performance of the six methods on benchmarks derived from a single mother set. The bars show the expected MFP averaged across anomalies in benchmarks for the corresponding mother set. 95% confidence intervals are also shown.

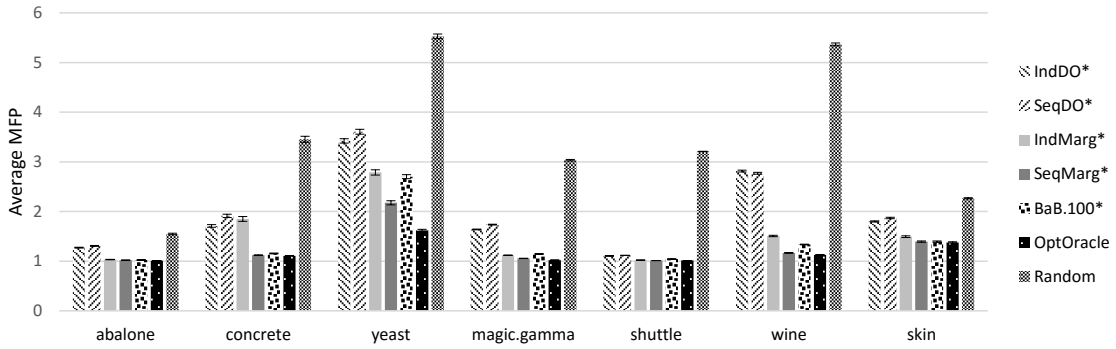

Figure 3.5: Performance of explanation methods on benchmarks when using an oracle anomaly detector. Each group of bars shows the performance of the six methods on benchmarks derived from a single mother set. The bars show the expected MFP averaged across anomalies in benchmarks for the corresponding mother set. 95% confidence intervals are also shown.

than our best SFE method. This gap could be due to either: 1) suboptimal SFE computations, 2) a poor match between the anomaly detector's notion of outlier versus the analyst's notion of anomaly, or 3) the fact that OptOracle is not constrained to output sequential explanations. We will investigate this further

below (Section 3.7.4).

For the remaining four mother sets, we see that the marginal methods are quite close to the lower bound of OptOracle, though there is still some room for improvement. Finally, it is worth noting that OptOracle is able to achieve MFPs of close to 1 for most of the mother sets. Thus, on average, for these data sets, a single feature is sufficient to allow for correct analyst detections.

Comparison to Branch and Bound. We now compare the greedy methods to BaB.100, which as described above attempts to optimize ESP as a surrogate for the true-but-unknown MFP objective. Figure 3.4 shows the performance of these methods and interestingly we see that the best greedy method significantly outperforms BaB.100 on the Yeast and Shuttle benchmarks. A potential reason for this is due to a mismatch between the ranking of points by the anomaly detector and the ranking of points by the simulated analysts. That is, there is a mismatch between statistical outliers and semantic anomalies. Since BaB.100 works harder than the greedy methods to optimize the ESP objective, the impact of this mismatch can be amplified. This hypothesis is supported by Figure 3.3 where we see that Yeast and Shuttle are the two benchmarks where BaB.100 has a significantly better ESP than the greedy methods. We investigate the issue further in Section 3.7.4.

Independent versus Sequential. It is reasonable to expect that the sequential version of the marginal and dropout methods will outperform the independent versions. This is because the sequential versions attempt to account more aggressively for feature interaction when computing SFEs, which requires additional computation time. However, we see that overall there is very little difference in performance between the independent and sequential methods. That is, SeqMarg and IndMarg (as well as SeqDO and IndDO) achieve nearly identical performance. The only exception is in magic.gamma where there is a small—but statistically significant—advantage (according to a paired t-test) of SeqMarg over IndMarg. One possible explanation for these results is that feature interactions are not critical in these domains for detecting anomalies. This explanation is supported by the fact that OptOracle is able to achieve average MFPs close to one.

Marginal versus Dropout. Recall that the marginal and dropout methods are dual approaches. Marginal evaluates a set of features in terms of how abnormal those features alone make a point appear, while dropout evaluates a set by the increase in normality score when the features are removed. We see that overall

the marginal methods are never significantly worse than dropout and significantly better on ABALONE, MAGIC.GAMMA, SHUTTLE, and SKIN. The difference is particularly large on SHUTTLE, where the marginal methods are close to OptOracle and the dropout methods are closer to random.

One possible explanation is that we have observed that often dropout will produce a "weaker signal" compared to marginal when making early decisions. For example, when considering single features, the differences in scores produced by dropout for those features are often much smaller than the differences produced by marginal. This can make dropout less robust for early decisions, which are the most important ones for achieving small MFP scores. Recall that the dropout method was inspired by prior work on explanations for supervised learning. The results here suggest that it is worth investigating adaptations of marginal to the supervised setting.

3.7.4 Comparing Methods with Oracle Detectors

Since the SFE methods make their decisions based on the anomaly detector's density function f, the results above reflect both the SFE methods and the quality of the detector. Here we attempt to factor out the performance of the SFE methods themselves by supplying the methods with an oracle anomaly detector. To do this we simply replace the use of f with the simulated analyst's conditional probability function $P(\text{normal} \mid x_S)$, which we can compute for any feature subset S. For example, the first feature selected by SeqMarg is the x_i that minimizes $P(\text{normal} \mid x_i)$. Note that this is also the first feature that would be selected by OptOracle. Unlike OptOracle, however, SeqMarg is sequentially constrained and will select the second feature as the one that works best when combined with the first selected feature.

Figure 3.5 shows results for all methods using the oracle detectors. We use a '*' to indicate that a method is using an oracle detector, for example, SeqMarg* is the oracle version of SeqMarg.

Comparison to OptOracle. The primary observation is that SeqMarg* performs nearly identically to OptOracle in all but one domain. Any difference between SeqMarg* and OptOracle would reflect the loss in performance due to requiring sequential explanations, which is required for OptOracle, and/or the greedy

optimization. For these data sets, there is little to no loss. This is good news, since the motivation for considering sequential explanations is to reduce the analyst's effort. In particular, the sequential constraint means that the analyst is shown an incrementally growing set of information. Rather, without the constraint, OptOracle could potentially show completely different sets of features from step to step, which is arguably less desirable from a usability perspective.

Comparison to Branch and Bound. The performance of BaB.100 is significantly improved in comparison with greedy methods when using the oracle detectors. This provides evidence that the primary reason for the poor performances of BaB.100 above was the mismatch between the anomaly detector ranking of points and that of the analyst. However, we still see that BaB.100 is usually slightly outperformed by the best greedy method SeqMarg and significantly outperformed on Yeast. A likely reason for this is that BaB.100 is optimizing a surrogate objective ESP, rather than the true objective MFP. When there is a disparity between these objectives, more aggressive optimization can be counter productive. Note that in Section 3.7.2, we did validate that BaB.100 does optimize its surrogate objective effectively compared to the greedy methods.

Independent versus Sequential. Here, we see that SeqMarg* is often outperforming IndMarg* and sometimes by significant amounts. This is in contrast to the results obtained when using EGMM as the anomaly detector. This indicates that reasoning about feature interactions, as done by SeqMarg*, can be important with higher-quality anomaly detection models. This leaves open the question of whether we will be able to observe this advantage when using non-oracle anomaly detection models on realistic benchmarks.

Dropout versus Marginal. The marginal methods show consistently better performance when using oracle detectors. The performance gap is quite large in several of the benchmarks. This provides evidence that the marginal approach is generally a better way of computing SFEs. Again, we hypothesize that this is due to the "weak signal" during early decisions observed for the dropout method.

3.7.5 Evaluation on high dimensional Datasets

We now consider the effectiveness of the proposed methods on larger-dimensional datasets. For large dimensions, computing the analyst models by learning one

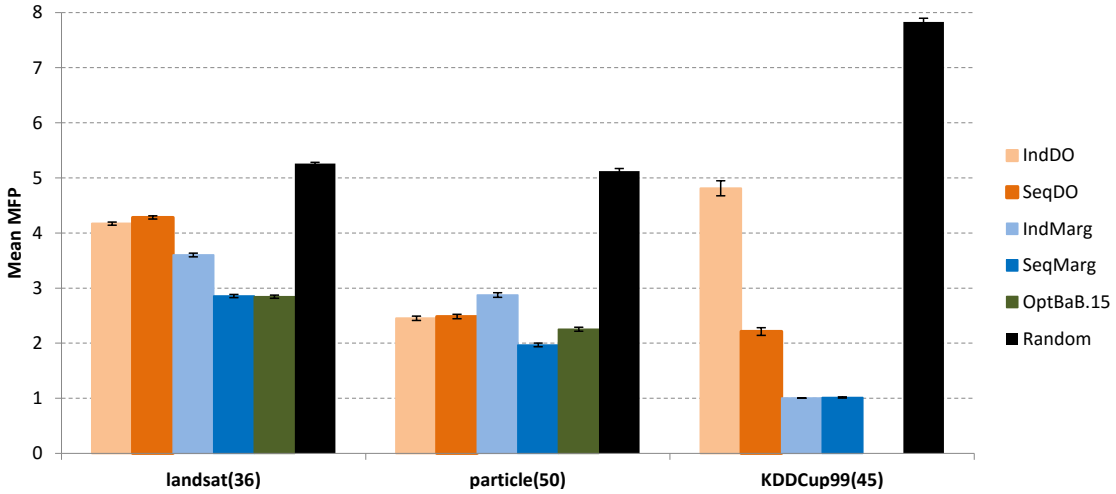

Figure 3.6: Performance of different explanation methods on some high dimensional benchmark datasets. 95% confidence intervals are also shown.

model for each feature subset (as done above) is computationally prohibitive due to the exponential number of feature subsets. However, most such subsets are not useful for evaluation especially those that contain many features since, in practice, it is rare for anomalous behavior to be a result of large numbers of interacting features. In addition, larger subsets are not easy to consume and process by human analysts. Hence, We focus on evaluating SFEs involving only the top $k \leq 10$ features instead of all d features.

In order to examine the effects of a large number of dimensions, we choose three datasets: LANDSAT, PARTICLE and KDDCUP99 having dimensions 36, 55, and 45, respectively. Even after imposing the constraint of SFE computation for $k \leq 10$ features, the restricted BaB method (BaB.100) took hours to finish and sometimes exceeded the memory limit. This is because at each node expansion during branch and bound search we need to calculate the upper bound of the node. This requires marginalizing the model which is costly due to the large number of parameters and having to compute the density for all the instances. Hence, we limit the execution of this method to 15 minutes (hence calling it BaB.15) and use the best SFE found after the time limit is reached.

For the KDDCUP99 intrusion detection benchmark [Hettich and Bay, 1999],

we use $k = 10$ and consider a subset of the data containing instances involving http service only. The resulting benchmark contains approximately 620K points with approximately 4K anomalous points representing network intrusions. We again employed EGMM as the anomaly detector. It was infeasible to train a simulated analyst on all feature subsets so we followed the adaptive approach described in Section 3.6 where only the subset of models required during the evaluation process was learned and cached. Overall this resulted in approximately 7.5K RFF models being trained. In this domain, the EGMM model was quite effective and ranked all anomalies very close to the top of the ranked list. Thus, we evaluate on all anomalies in this domain.

The results for KDDCUP99 are shown in Figure 3.6. It is clear that the marginal methods are significantly better than the dropout methods. In particular, both SeqMarg and IndMarg achieve an average MFP close to one, which is the smallest possible. This indicates that the combination of EGMM and marginal explanations is very effective in this domain. In particular, the simulated analyst only needed to be shown a single feature on average in order to correctly detect the anomalies.

For the LANDSAT and PARTICLE datasets we use $k = 6$ and evaluate with respect to anomalies ranked in the top 10%. The result is shown in Figure 3.6. The IndDO and SeqDO are similar for both of the datasets. SeqMarg is better than IndMarg in both cases and also better than IndDO and SeqDO. SeqMarg performs equal to or better than the BaB method. For the BaB method, 31% of the anomaly points from LANDSAT and 70% of the points from PARTICLE are solved for the optimal ESP value within the 15-minute time limit. Although the majority of the anomaly points in the PARTICLE dataset reached optimal solution, the performance is slightly worse than sequential marginal. This could be due to the reason already discussed in Section 3.7.3. Overall, the relative performance of the dropout and marginal methods are very similar to the relative performance observed earlier. The marginal methods tend to outperform the dropout methods.

We again hypothesize that the much weaker performance of the dropout methods is due to the "weak signal" they provide for early decisions. This problem is only amplified in the context of larger numbers of features as is the case for the KDDCup data.

3.8 Experiments on Real World Network Data

In this section we describe experiments performed on some real world network data [Siddiqui et al., 2019b]. Suppose we have collected some network activity data for a certain amount of time from M computers within an organization, where $x_{ij} \in \mathcal{R}^n$ is a set of count-based features representing the activities that occurred in the i^{th} interval of the j^{th} computer. Let, $\mathcal{X} = \{x_{11}, x_{12}, ..., x_{1j}, ..., x_{ij}, ..., x_{iM}\}$ be the collection of all the activity instances across all the computers. $\mathcal{X}$ contains both the benign (*i.e.*, regular) activity as well as the malicious (*i.e.*, attack) events. We assume that the time interval is large enough to capture a malicious event within that period, and if a malicious event is split between two consecutive time intervals, then one of the intervals should have enough data to capture the malicious activity. We also assume that the majority of the data consists of benign activity, while the malicious activity is only a small fraction of the entire dataset. Our goal is to detect all the time intervals containing the malicious events. We use anomaly detection over the dataset $\mathcal{X}$ to identify the time intervals that are anomalous compared to all other time intervals. We assume that during an attack the activities of the victim computer are somewhat different than typical or regular activity.

3.8.1 Data Collection

We collected data from over two millions of computers reporting during a two week period. To protect the identify of the users, we anonymized the user account and the computer name. We divided the data from computers into two classes, servers and clients, based on their operating system. Since servers tend to have significantly different types of activity than client computers, we filtered them out to focus the analyst's time on the majority computer type. For each of these client computers, we collected network activity containing logon events, RDP and SQL connections. The anomaly detector Isolation Forest [Liu et al., 2008] consumes numeric vectors as input. Thus, we chose a window size with a time resolution of 30 minutes and counted the different types of activity that occurred in that interval. In total, we used 23 count-based features (Table 3.2 [Siddiqui et al., 2019b]) that gave us a numeric vector of size 23 for each 30 minutes interval, *i.e.*, a total of 672 numeric vectors per computer we calculated over the two weeks of activity. We then filtered out the vectors that did not have any activity, *i.e.*, all of the values

Table 3.2: List of Features

SuccessfulLogonRDPPortCount
UnsuccessfulLogonRDPPortCount
RDPOutboundSuccessfulCount
RDPOutboundFailedCount
RDPInboundCount
SuccessfulLogonSQLPortCount
UnsuccessfulLogonSQLPortCount
SQLOutboundSuccessfulCount
SQLOutboundFailedCount
SQLInboundCount
NtlmCount
SuccessfulLogonTypeInteractiveCount
SuccessfulLogonTypeNetworkCount
SuccessfulLogonTypeUnlockCount
SuccessfulLogonTypeRemoteInteractiveCount
SuccessfulLogonTypeOtherCount
UnsuccessfulLogonTypeInteractiveCount
UnsuccessfulLogonTypeNetworkCount
UnsuccessfulLogonTypeUnlockCount
UnsuccessfulLogonTypeRemoteInteractiveCount
UnsuccessfulLogonTypeOtherCount
DistinctSourceIPCount
DistinctDestinationIPCount

were 0. After preprocessing, the final dataset contained approximately 300 million vectors for the client computers.

3.8.2 Anomaly Detector Setup

To create the Isolation Forest model we trained 500 trees, where each tree is grown with 10,000 unique instances (vectors), and the tree is grown until each instance is isolated to its own leaf. One major benefit of the Isolation Forest is that each tree can be trained independently. We leveraged this and constructed all the trees in parallel using the Cosmos, which is a distributed MapReduce platform. It took about 10 minutes in total to train the entire model. We then computed the anomaly score for each instance using the same distributed environment that allowed us to compute anomaly scores for different instances in parallel. For all the instances the scoring process took about 2 hours. We then selected the top 1000 anomalous

instances and produced the explanation for each. The explanation creation process took about an hour.

Table 3.3: Explanation examples along with anomaly scores

Anomaly Score	Top 3 features Explanation
0.839	UnsuccessfulLogonTypeOtherCount = 40.0 is unusual with score 0.56 SuccessfulLogonTypeNetworkCount = 800.0 is unusual with score 0.70 SuccessfulLogonTypeInteractiveCount = 120.0 is unusual with score 0.77
0.836	SQLOutboundSuccessfulCount = 44.0 is unusual with score 0.54 UnsuccessfulLogonTypeInteractiveCount = 21.0 is unusual with score 0.70 SQLInboundCount = 1568.0 is unusual with score 0.77
0.834	UnsuccessfulLogonTypeOtherCount = 40.0 is unusual with score 0.56 SuccessfulLogonTypeInteractiveCount = 69.0 is unusual with score 0.70 SuccessfulLogonTypeNetworkCount = 547.0 is unusual with score 0.76
1.24	UnsuccessfulLogonTypeNetworkCount = 59.0 is unusual with score 0.54 RDPInboundCount = 368.0 is unusual with score 1.13 DistinctDestinationIPCount = 11.0 is unusual with score 1.17
1.24	UnsuccessfulLogonTypeNetworkCount = 79.0 is unusual with score 0.54 RDPInboundCount = 145.0 is unusual with score 1.13 DistinctDestinationIPCount = 1.0 is unusual with score 1.16

3.8.3 Explanation Results

For each instance, we get the anomaly score from the Isolation Forest along with the sequential feature explanation computed as described in Section 3.5.2.1. Note that the Isolation Forest produces a score ($Score(x)$) instead of density estimate. So we compute the marginal score for an instance x under a feature subset s as $Score(x_s)$, which is the marginalized score computed by considering the threshold tests involving features only from s across all trees. Basically, the marginal anomaly score $Score(x_s)$ is obtained by traversing branches of all internal tree nodes whenever the corresponding threshold test on feature F does not belong to the set s, i.e., $F \notin s$. The resultant score is computed recursively by weighting the left and right subtree scores with the fraction of instances that went to left and right branches during training. This method of marginalization is very effective for the Isolation Forest as shown by [Dietterich and Zemicheal, 2018], where the

authors call the method "proportional distribution". They considered the set of features not present in the set s as missing value which allows one to compute an anomaly score with any arbitrary set of missing features. Finally, a desired length k explanation $E = (e_1, e_2, ..., e_k)$ can be computed as follows:

$$e_i = \underset{j \notin E_{1:(i-1)}}{\operatorname{argmin}} Score(x_{E_{1:(i-1)} \cup j}) \tag{3.7}$$

Table 3.3 [Siddiqui et al., 2019b] shows some of the example explanations, *i.e.*, SFE with length 3. Note that the raw anomaly scores are normalized according to [Liu et al., 2008] to make higher scores indicating more anomalousness. The explanations tell the analyst what features were most responsible for the high anomaly score. Also, we report the individual marginal anomaly score along with each accumulated feature, which gives a rough estimation of how much each additional feature contributes to the overall anomaly score. Note that the individual marginal scores are each independent since they have different set of features, and hence it should not be expected that the overall anomaly score somehow decomposes into them. The last two rows in Table 3.3 show examples of some actual RDP brute force attempts that correlate with the features shown in the explanations. These examples indicate that the explanations are inherently capturing some cause of the attack. Note that the anomaly scores for the two instances went above 1 because of the normalization constant being kept fixed. Explanation for these these two instances are computed after incorporating feedback (please see Chapter 4 for details), which caused the weights to be changed and the anomaly score to rise above 1.

3.9 Main Observations and Recommendation

The main observations from the above experiments can be summarized as follows.

- All of the introduced SFE methods significantly outperformed randomly generated SFEs.
- The marginal methods were generally no worse and sometimes significantly better than the dropout methods.
- When using the EGMM anomaly detector, we observed little to no difference

between the performance of sequential versus independent methods.

- When using the oracle anomaly detector, SeqMarg significantly outperformed IndMarg which suggests that, in general, sequential methods can outperform independent methods.
- While the BaB methods were more effective at optimizing ESP in many cases, this did not translate to outperforming the best greedy method with respect to MFP.

Overall, based on our results, SeqMarg is the recommended method for computing SFEs among the methods we studied.

3.10 Summary

This chapter introduced the concept of sequential feature explanations (SFEs) for anomaly detection. The main motivation was to reduce the amount of effort required of an analyst to correctly identify anomalies. We described several methods for computing SFEs and introduced a new framework that allows for large-scale quantitative evaluation of explanation methods. To the best of our knowledge, this is the first such large scale evaluation of explanation methods for anomaly detection. Our experiments indicated that, overall, the Sequential Marginal method for computing SFEs is the preferred method among those introduced in this work.

An interesting point of future work will be to explore this framework for a wider range of anomaly detection models and simulated-analyst models. This could include comparing the relative effectiveness of our generic approaches for computing SFEs with approaches that are specifically designed for particular anomaly detectors. One surprising observation from our results is that our more computationally intensive branch-and-bound approach never outperformed our best greedy approach in actual evaluations. We hypothesized that this was due to the potential mismatches between: 1) the anomaly detector and analyst model, and 2) the optimization objective, ESP, and the evaluation metric, MFP. Optimizing more aggressively in light of these mismatches appears to be counter productive. An interesting point of future work will be to investigate alternative optimization objectives and to better understand the surprising effectiveness of the greedy sequential marginal method.

Another important direction for future work will be to conduct qualitative evaluations using human subjects. This is a challenging direction due to the need to find subject matter experts that are available to take part in evaluation trials. Alternatively, human studies could be conducted using synthetic benchmarks that are constructed in a way that allows regular human subjects to be easily taught the domain properties. These subjects would then be our expert analysts. While this second approach is less realistic, it would allow for a wider-scale evaluation that could produce quantitative results in addition to the qualitative observations.

Chapter 4: Feedback-Guided Anomaly Discovery

4.1 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 chapter, 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 chapter, we demonstrate that significant benefits are possible on benchmark data and on a large-scale computer security application.

The main contribution of this chapter 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 their solid theoretical grounding, where convergence is understood. Prior state-of-the-art approaches, such as AAD [Das et al., 2016], 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 chapter we focus on the popular class of tree-based anomaly detectors, which includes the state-of-the-art Isolation Forest detector [Liu et al., 2008], 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.

4.2 Anomaly Detection Problem Setup

We reiterate the anomaly detection problem which can be 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 4.4). 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. [Jiang et al., 2017]).

4.3 Related Work

The general problems of active learning [Settles, 2012] 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 [Pelleg and Moore, 2005] 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 [Raginsky et al., 2012, Varadarajan et al., 2017]

that have considered incorporating feedback into a sequential setting for anomaly detection , 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 [Das et al., 2016, Das et al., 2017], where the same setting for incorporating feedback into anomaly detection is studied. In [Das et al., 2016] AAD was introduced as a way to incorporate feedback into a the LODA anomaly detector [Pevnỳ, 2016]. 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 [Das et al., 2016] 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 [Das et al., 2017]. 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) [Görnitz et al., 2013] and AI2 [Veeramachaneni et al., 2016]. 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 [Grill and Pevnỳ, 2016] uses some labeled data to estimate a convex combination of a set of anomaly detectors. In [Das et al., 2016], 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 chapter.

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 [Dong et al., 2017, Forrest et al., 1996, Gao et al., 2004, Sekar et al., 2001, Shu et al., 2015]. 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.

4.4 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.

4.4.1 Online Convex Optimization

Many classic and new online learning algorithms can be described and analyzed within the OCO framework. A good introduction to OCO framework can be found in [Shalev-Shwartz et al., 2012]. 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

$$\text{Regret}_T = \sum_{t=1}^{T} f_t(w_t) - \min_{w^* \in \mathcal{S}} \sum_{t=1}^{T} f_t(w^*), \tag{4.1}$$

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.

4.4.2 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 4.5 that tree-based anomaly detectors are easily modeled

[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 4.5.

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 [Das et al., 2016] parameterized the LODA anomaly detector [Pevnỳ, 2016] 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

$$f_t(w_t) = -y_t \text{SCORE}(x_t; w_t) = y_t w_t \cdot \phi(x_t), \tag{4.2}$$

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 $\text{SCORE}(x; w)$ 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:

$$P_a(x; w) = \exp\left(\text{SCORE}(x; w)\right)/Z = \exp\left(-w \cdot \phi(x)\right)/Z, \tag{4.3}$$

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:

$$f_t(w_t) = -y_t \log\left(P_a(x_t; w_t)\right). \tag{4.4}$$

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 [Korč and Förstner, 2008].

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.

4.4.3 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

$$w_t = \arg\min_{w \in \mathcal{S}} \sum_{j=1}^{t-1} f_j(w) + \frac{1}{\eta} R(w), \tag{4.5}$$

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.

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)* [Shalev-Shwartz et al., 2012], 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

[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.

Algorithm 2 Online Mirror Descent for Query-Guided Anomaly Discovery

Require: Data Instances D; Generalized-Linear Anomaly Detector with features ϕ 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**
  $w_t = \arg\min_{w \in S} \|w - \theta_t\|_2$
  $x_t = \arg\max_{x \in D} -w_t \cdot \phi(x)$ *;; 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. 4.2 or Eq. 4.4
  $\theta_{t+1} = \theta_t - \eta \partial f_t(w_t)$ *;; subtract η-weighted sub-gradient*
end for

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 2 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 4.5), 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" [Shalev-Shwartz et al., 2012]. 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.

4.5 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 [Liu et al., 2008, Siddiqui et al., 2016, Tan et al., 2011, Wu et al., 2014, Guha et al., 2016]. 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 [Liu et al., 2008], 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 [Siddiqui et al., 2016] 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 [Wu et al., 2014], 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 [Tan et al., 2011] and [Guha et al., 2016]. 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.

4.6 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 chapter. We note, however, that the baseline version of our system, which does not use feedback, corresponds to the Isolation Forest algorithm [Liu et al., 2008], which has been shown to be among the top performers on a large-scale evaluation [Emmott et al., 2013]. In addition, results in prior work on AAD [Das et al., 2017] have shown that the AAD approach, which we compare to here, significantly outperforms the state-of-the-art LODA [Pevnỳ, 2016] anomaly detector. For our comparison to AAD, we use the currently available implementation for tree-based detectors [Das et al., 2017] 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 [Das et al., 2017] and the LODA anomaly detector [Das et al., 2016].

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 sub-sample 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.

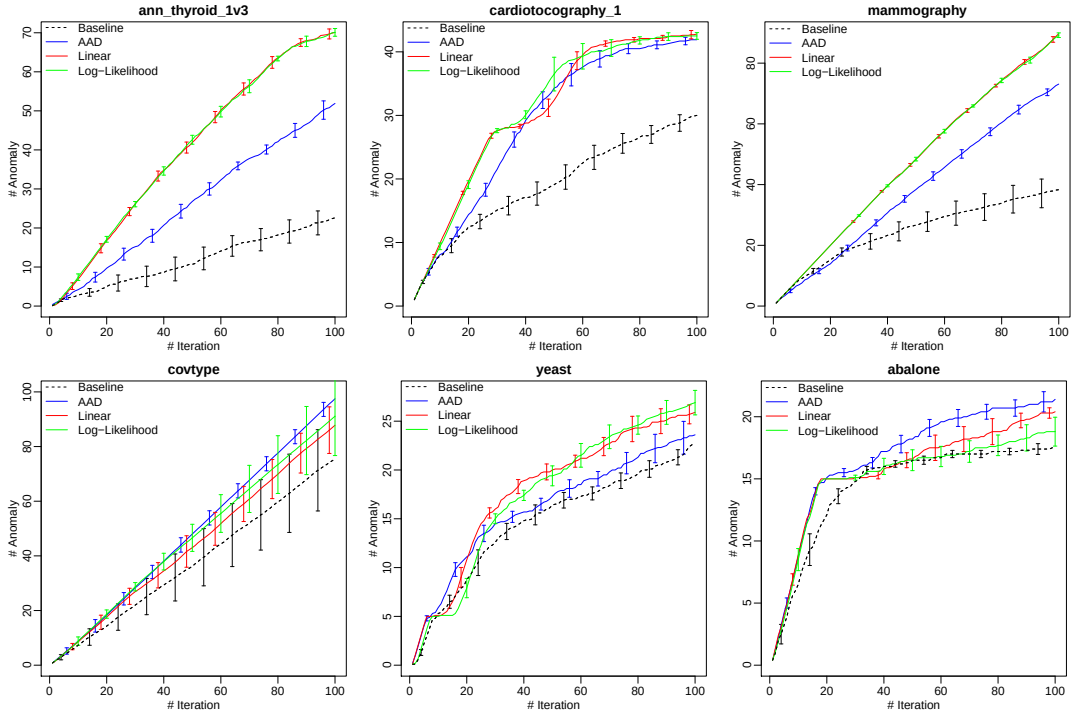

Figure 4.1: Feedback result comparison with AAD [Das et al., 2017] on some standard anomaly detection datasets [Das et al., 2017]. 95% confidence interval is also shown for some feedback iterations, others are similar, hence not shown to make the plot clear.

4.6.1 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., 2017]. Figure 4.1, 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.

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

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, 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., 2013]. 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 4.2. 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 4.3 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

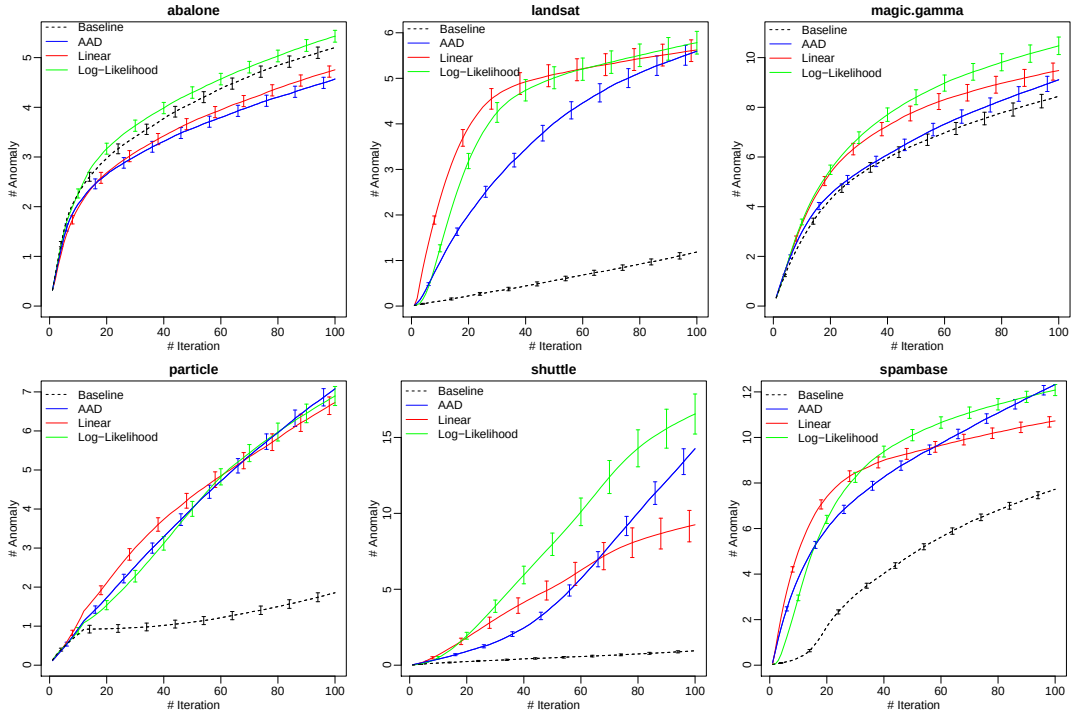

Figure 4.2: Average feedback iteration curves on 6 benchmark datasets from [Emmott et al., 2015]. Each curve is averaged over 120-300 benchmark datasets.

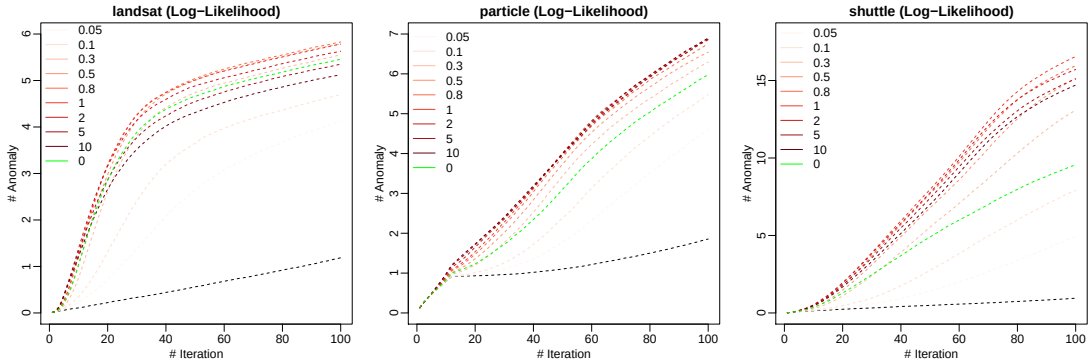

Figure 4.3: Sensitivity of the learning rate parameter in three different benchmark datasets. Learning rate 0 indicates variable learning rate. The black dotted line is the baseline performance without feedback.

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).

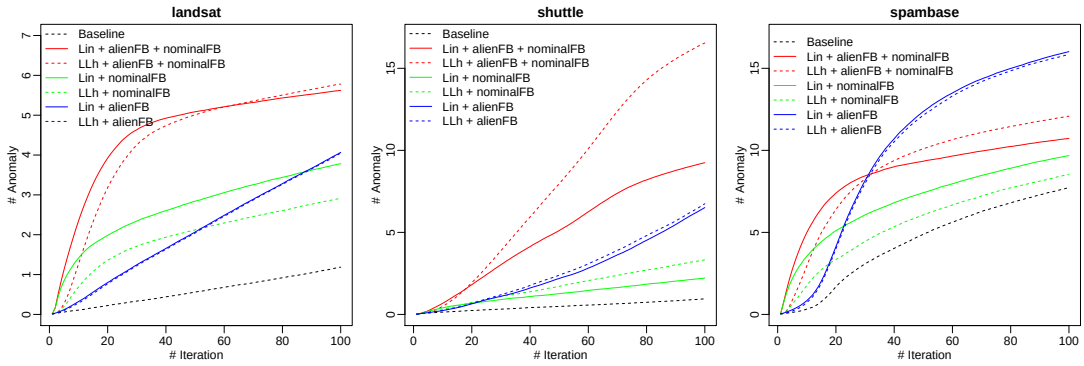

Figure 4.4: Effect of incorporating feedback from alien and/or nominal instances only in three benchmark datasets.

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.

Influence of Nominal and Alien Feedback. For Q4 we ran experiments where we restricted the types of feedback used for learning. Figure 4.4 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.

4.6.2 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.

Results. Figure 4.5 shows the number of malicious entities detected as a func-

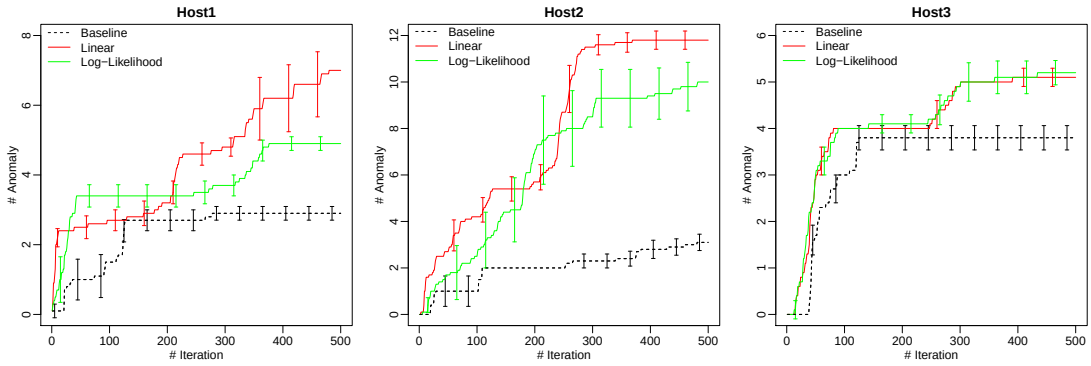

Figure 4.5: Feedback iteration results when applied to combine all 20 views on different hosts.

Table 4.1: 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**

tion 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 4.1 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

Table 4.2: Detection result after each feedback round

	Feedback Round 1			Feedback Round 2		
Iteration #	1	2	3	1	2	3
# TPs	1	9	20	0	1	2
# FPs	11	4	0	10	17	23

loss.

4.7 Experiments on Real World Network Data

In this section we describe results after applying the feedback approach on some real world nework data [Siddiqui et al., 2019b]. Please see the Section 3.8 for details about the experimental setup (datasets and anomaly detector setup).

4.7.1 Incorporating Analyst's Feedback

Once we have the top ranked outliers along with their explanations, we present this ranked list to the analyst. The analysts time is an extremely valuable resource, and we only had access to one professional analyst who could investigate about 20 instances per week in addition to their standard responsibilities. Analysts have access to much richer data logs than is available in the features in our dataset which allows them to make a confident assessment as to whether or not the anomalous activity is malicious. After each investigation the analyst give us their verdict as either "True Positive" or "False Positive". In some cases, the analyst is unable to decide if there is enough information to make a certain decision, so the analyst just labels that instance as "unknown". We only incorporated "True Positives" or "False Positives" in our feedback and ignored the unknowns.

4.7.2 Detection Results

We performed two rounds of feedback with the help of a professional analyst and repeated three iterations each time. Table 4.2 [Siddiqui et al., 2019b] shows the number of true positives and false positives detected after each iteration of the feedback round. In the first feedback round at iteration 1 we only detect 1 TP

and 11 FPs. After incorporating them we discovered 9 TPs and 4 FPs in the next iteration. After we included these new TPs in the third iteration the system became really good at detecting TPs and detected 20 TPs without any FPs. After further investigation, we found that all the TPs discovered in this round belong to the same type of attack known as RDP Brute Force Attempt. Next, we shifted our focus to detecting examples from a different type of attack. To that end, we started a second feedback round with the very first model and intentionally incorporated all the previous TPs and also all the FPs discovered so far as FPs. This strategy forced the model to ignore the RDP Brute Force Attempts. We continue this round for another three iterations and detected 1 and 2 TPs in iteration 2 and 3 respectively. During this round, we uncovered an additional attack behavior, that of network port scanning. These results show that in just a few iterations, the system can become very effective at discovering attacks with a small false positive rate for the top K results. We also found that providing TPs as feedback seems to be more effective at discovering similar type of attacks, and FPs as feedback helps to focus on finding something different than the type what the FPs belong to.

4.8 Summary

In this chapter, 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 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.

Chapter 5: Class Discovery

5.1 Introduction

In many real world situations we are interested in identifying all of the distinct classes of a given set of examples. For instance, in security settings we want to know how many different types of attacks a given dataset contains. In particular, events in the data are either benign or malicious, and each has a label reflecting the type of normal behavior or attack. Our goal is to identify all the distinct behaviors (e.g. classes) with the help of an oracle who can provide the class label of any queried example. So, we want to uncover all the distinct class labels by asking as few queries as possible to the oracle. The most related research to this idea is on active learning [Settles, 2012], which focuses on improving the underlying classifier with a small number of label queries to an expert. Rather, for the class discovery problem, our main focus is to discover at least one example from each class which makes is more related to the Rare Category Detection (RCD) [Pelleg and Moore, 2005, He and Carbonell, 2008] problem.

In the literature, there exist various approaches to solve the class discovery problem that are inheriting ideas from anomaly detection, rare class discovery [He and Carbonell, 2009], hierarchical clustering [Vatturi and Wong, 2009], active learning with classifiers [Haines and Xiang, 2014, Hospedales et al., 2011, Hospedales et al., 2012], etc. We didn't find any independent empirical comparison of these different methods within a single framework. So, our overall goal in this chapter is to provide an independent evaluation of these different approaches under a set of constructed benchmark datasets.

In the next Section 5.2, we define the class discovery problem more formally, and then in Section 5.3 we describe various approaches to address this problem. In Section 5.4, we describe some metrics to evaluate all the class discovery approaches under same framework. Finally, we summarize our findings in Section 5.5.

5.2 Problem Setup

Suppose we have a set of instances $\mathcal{X} = \{x_1, x_2, ..., x_n\}$, where $x_i \in \mathbb{R}^d$. Each instance x_i belongs to a class y_i, which can take one of k distinct class values, that is, $y_i \in \{1, 2, ...k\}$. We also have an oracle function $\mathcal{C}^*(x)$ which returns the true class label of an instance x on request. But, requesting a class label for an instance x_i is difficult, time consuming and sometime requires significant human resource and effort. The goal is to discover at least one instance from each class with a minimum number of queries to the oracle. Contrary to active classifier learning, where the number of classes is typically known, the number of classes k is unknown to the class discovery algorithm. For simplicity, we assume there exists one majority class, and the rest are minority classes (typically assumed in rare category detection problem [He and Carbonell, 2008]). This makes the problem more suitable in security applications, where the majority class usually represents the benign class, and each minority class may correspond to an attack.

5.3 Class Discovery Algorithms

In this section, we describe several algorithms for the class discovery problem that are based on concepts from anomaly detection, rare category discovery and hierarchical clustering, etc. All these algorithms have a general structure: each one typically starts with a base model (i.e., a classifier or an anomaly detector) trained on unlabeled data, and then uses the current model to heuristically select the next example to query and the result of the query is then used to update the model. Different methods differ primarily in the base model and the heuristic they use.

5.3.1 Anomaly Detection

One simple idea for class discovery is to use an anomaly detector. If we have one majority class and the rest are rare classes, the anomaly detector will ideally identify instances of those classes as anomalous. In this section, we describe two class discovery approaches based on the anomaly detection technique Isolation Forest [Liu et al., 2008].

5.3.1.1 Static Isolation Forest

We employ the anomaly detector Isolation Forest [Liu et al., 2008] which produces a ranking of the instances based on their anomaly scores (e.g., average isolation depth). The hope is that highly ranked instances will be from different rare classes. We simply query the oracle for the instances from most anomalous to least anomalous. One problem with this approach is that even if a class is rare, when there is an enormous amount of data in the overall dataset (e.g., the dataset used in Section 3.8.1), then even anomalous classes can have large numbers of instances. Those instances can all get high anomaly scores and be ranked close to the top. So, we might discover only one type of rare class. This problem can be handled by providing feedback for the earlier ranked instances so that the instances from the same class are then moved later in the ranking (see the next Section 5.3.1.2 for this approach). Another problem with this simple approach is that it doesn't exploit the answers to the oracle queries.

5.3.1.2 Isolation Forest with Feedback

In this approach, we leverage the oracle query to give feedback to the anomaly detector so that the detector can focus on discovering new classes. We provide negative feedback for the top ranking instance to the detector using the techniques described in Section 4.4. We specifically used the feedback approach with the linear loss function of Equation 4.2 for simplicity. This is an iterative process, that is, we pick the top ranking instance and provide feedback for it to adjust the parameters which gives a new ranking. We then again pick the top instance that was not previously queried and repeat the process. The idea is that with the negative feedback, the anomaly detector's weights will be adjusted in a way that forces the detector to focus on a different region from where the feedback is given. The instances that are similar to the previous queries will be ranked lower and the instances from different classes will be ranked higher.

5.3.2 Likelihood[1]

One general query criterion is to select points that are badly explained by a model, for example, a generative classifier or an anomaly detector. The low likelihood

points are usually good for discovering new classes. In Hospedales et al. [2011], they used this approach where the points with low likelihood are queried. They used incremental kernel density estimation [Sillito and Fisher, 2007] for likelihood computation:

$$i^* = \underset{i}{\operatorname{argmin}} \left(\max_{y_i} p(x_i|y_i;\theta) \right) \tag{5.1}$$

where, $p(x|y;\theta)$ is the class conditional distribution and θ is the parameter of the model. This approach aims to identify the data point that is "least explained" by the current model. Queries are then added to the model to increase their likelihood and ideally the likelihood of similar instances from the same class.

5.3.3 Entropy[1]

Entropy is a well known query criterion for refining the class boundary. The main idea is to query uncertain points; this can improve discrimination between classes. In Hospedales et al. [2011] they used uncertainty which is quantified by posterior entropy and the point with maximum uncertainty is queried:

$$i^* = \underset{i}{\operatorname{argmax}} \left(-\sum_{y_i} p(y_i|x_i;\theta) \log p(y_i|x_i;\theta) \right) \tag{5.2}$$

They used the support vector machine (SVM) [Deselaers et al., 2010] model with Gaussian kernel for classification. The probability $p(y|x)$ can be approximated based on the distance to the margin at each point [Settles, 2009].

5.3.4 DPEA[1]

Hospedales et al. [2012] define a criterion called Dirichlet Process Expected Accuracy (DPEA) to select a query point that maximizes the expected utility $\tilde{\mathcal{U}}_{D_i}(\theta)$:

$$i^* = \underset{i}{\operatorname{argmax}}\, \tilde{\mathcal{U}}_{D_i}(\theta) \tag{5.3}$$

[1]Code obtained from: http://homepages.inf.ed.ac.uk/thospeda/downloads.html

The expected utility is defined as:

$$\tilde{\mathcal{U}}_{D_i}(\theta) = \sum_{y_i} p(y_i|x_i) \frac{1}{N} \left(\sum_{j \in L \cup i} p_{\theta+i}(y_j|x_j) + \sum_{j \in U \setminus i} \sum_{y_j} p(y_j|x_j) p_{\theta+i}(y_j|x_j) \right) \quad (5.4)$$

Where, L is the labeled dataset, U is the unlabeled dataset, $D_i = \{L \cup (x_i, y_i), U \setminus x_i)\}$ denotes the dataset with the i^{th} example labeled. $\theta + i$ reflects the updated classifier parameters after adding the i^{th} example to the training set with its putative label y_i, $p_\theta(y_i|x_i)$ is the classifier distribution learned from the labeled set L and $p(y_i|x_i)$ is the true distribution that also considers an additional new class. Note that the true distribution $p(y_i|x_i)$ is unknown hence estimated using Dirichlet Process assumption [Antoniak, 1974, Rasmussen, 2000] to account for x_i belonging to a new class. The iteration over y_i and y_j includes both known classes and a new class.

5.3.5 PWrong[1]

Haines and Xiang [2014] proposed a method for querying instances based on misclassification probability:

$$P(Wrong|x) = 1 - P_n(c'|x) \quad (5.5)$$

$$c' = \underset{c \in C}{\operatorname{argmax}}\ P_c(c|x) \quad (5.6)$$

Where, $P_n(c'|x)$ is the probability that x belongs to a class c', where c' could be a new class. They used a Dirichlet Process [Ferguson, 1973] to account for a new unknown class. $P_c(c|x)$ is the probability calculated by a classifier, and does not involve a new class. The intuition for this heuristic is that we want to query instances that are not represented well by the current model. This can be measured in at least two ways: the likelihood that an instance belongs to a new class and how confident the current classifier is about the instance. If the probability of the instance belonging to a new class is high, the probability of being wrong will also go high. If probability of belonging to a new class is low but the classifier is wrong about the prediction, the probability of being wrong will also go high. Choosing an instance for which the current model is most wrong will provide the model the

maximum information.

5.3.6 SVM and KDE Fusion[1]

Generative classifiers often perform well with small numbers of training examples. On the other hand, discriminative classifiers require large numbers of examples; that is, they perform well asymptotically. Hospedales et al. [2011] leveraged this idea with KDE as the generative classifier and SVM as the discriminative classifier. For query criteria, they used uncertainty and likelihood for SVM and KDE respectively. They maintain a weight vector w for these two query criteria and sample one query from the multinomial distribution $k \sim Mult(w)$ during query time. After each query, they update the multinomial weights of the classifiers and also update the corresponding classifier model.

5.3.7 Clustering

Another natural approach for querying instances for class discovery is based on clustering techniques. Each cluster may correspond to a class. So, querying the representative points from different clusters may reveal all of the classes quickly. Here we present two approaches from Vatturi and Wong [2009].

5.3.7.1 HMS Outlierness

Vatturi and Wong [2009] used the Hierarchical Mean Shift procedure for rare category detection. They repeatedly applied the Mean Shift algorithm with increasing bandwidth to select cluster modes. Cluster modes are then used to query the oracle for new class discovery. They used two different criteria for selecting which cluster to query. The first one is based on outlierness. Outlierness criteria measures the lifetime of a cluster, which is the range of the logarithmic bandwidth scale over which a cluster survives (e.g., the difference between the logarithmic scale when the cluster is formed and when the cluster is merged). For a cluster C_i, the outlierness is measured as

$$Outlierness_i = \frac{lifetime\ of\ C_i}{\#\ of\ datapoints\ in\ C_i}. \tag{5.7}$$

The idea is that a cluster with long lifetime and a small number of points is more likely to be an outlier cluster which may represent a rare class. The most representative point in a cluster is queried, which is the point that moved the least from the cluster center in terms of mean shift distance.

5.3.7.2 HMS Compactness and Isolation

The second criterion is based on the compactness and isolation [Leung et al., 2000] of a cluster. Intuitively a cluster is well defined if withinness distance among points of a cluster is small (e.g., the cluster is compact) and the betweenness distance is large (e.g., the cluster is well isolated from other clusters):

$$isolation = \frac{\sum_{x \in C_i} exp(-||x - p_i||^2/2h^2)}{\sum_x exp(-||x - p_i||^2/2h^2)} \tag{5.8}$$

$$compactness = \frac{\sum_{x \in C_i} exp(-||x - p_i||^2/2h^2)}{\sum_{x \in C_i} \sum_j exp(-||x - p_j||^2/2h^2)}. \tag{5.9}$$

Vatturi and Wong [2009] combined isolation and compactness by adding them together and used that as a single criterion to pick a cluster with the highest compactness-isolation value. The query point is selected as the point with the minimum mean shift distance from the cluster center. In both of these criteria, a tie can happen, that is, we might have more than one cluster having same criterion value. They used a metric called highest average distance (HAD) to break ties. The idea is to calculate the distance between each cluster mode and the set of points that are already labeled and query the cluster with the highest HAD.

5.3.8 Random[1]

To compare how different query criteria perform with a random method, we included a random query criterion which picks instances uniformly from the pool of instances that haven't been queried. We ran this method 30 times and report the average.

Table 5.1: Summary of the original datasets

Dataset	# of Features	# of Classes	# of examples	Smallest class (%)	Largest class (%)
Abalone	7	28	4177	0.02	16.50
Covertype	10	7	5000	3.58	24.98
Glass	10	6	214	4.21	35.51
Leaf	14	30	340	2.35	4.71
Letter Recognition	16	26	3092	0.26	24.58
Yeast	8	10	1484	0.34	31.20

Table 5.2: Summary of the benchmark datasets

Dataset	# of Benchmarks	# of examples (range)	Normalized Entropy (range)
Abalone	10	1351 - 2373	0.55 - 0.85
Covertype	10	1265 - 3685	0.09 - 1.00
Glass	10	54 - 191	0.53 - 1.00
Leaf	10	203 - 268	0.97 - 0.99
Letter Recognition	10	594 - 2094	0.50 - 0.89
Yeast	10	375 - 1131	0.22 - 0.95

5.4 Empirical Evaluation

In this section we evaluate all the algorithms described in Section 5.3 on some standard datasets along with a suite of benchmarks. The benchmarks are constructed by varying some dataset properties. The analogous approach [Emmott et al., 2015] exists in the anomaly detection literature, where the authors systematically constructed a large number of benchmarks by varying different properties of datasets that have potential effect on anomaly detection problem. Our goal is to observe if

there is any property in the algorithms that also correlates with a dataset property.

5.4.1 Benchmark

We create a suite of benchmarks starting with several seed datasets from the UCI machine learning repository [Lichman, 2013]. Table 5.1 shows the seed datasets and their properties. Most of the datasets have been commonly used in the past on similar tasks. We also choose some additional datasets that have a large number of classes, since our main goal is to discover classes.

If a dataset contains instances uniformly from different classes, the entropy of the dataset is high, and it's easier for a class discovery algorithm to identify the classes than within a dataset where class instances are imbalanced. It might also depend on what underlying structure a class discovery algorithm is using, for example, the clustering-based methods might detect classes with many examples easily. On the other hand, anomaly detection-based methods might identify the classes with few instances easily. This motivated us to pick the entropy as a property of the datasets.

We seek to create a set of benchmakrs with normalized entropy that varies from 0.1 to 1 with an increment of 0.1. Normalized entropy is computed by dividing the entropy ($-\sum_i p_i \log(p_i)$ where p_i is the fraction of instances from class i) of a dataset by the maximum possible entropy (when all the classes have same number of instances). Assume that we have a dataset D, and C is the vector where each element C_i represents the number of instances available from the i^{th} class. We want to construct a benchmark with a normalized entropy τ. We solve the following constrained optimization problem to create a benchmark dataset where the number instances from class i is N_i:

$$\underset{N}{\operatorname{argmin}} \left| \frac{E(N)}{E_{max}(N)} - \tau \right| - \lambda \sum_i N_i \tag{5.10}$$

such that

$$\min(C_i, 5) \leq N_i \leq C_i, \tag{5.11}$$

where $E(N)$ is the entropy with the number of instances N (N_i is the number

Table 5.3: Normalized score under different evaluation metrics

	25%	50%	75%	100%	AUC	Reward
IF (No Feedback)	0.630	0.479	0.473	0.572	0.880	0.797
IF (Feedback)	**0.919**	0.686	0.546	0.570	0.910	0.847
SVM Entropy	0.640	0.418	0.413	0.564	0.863	0.720
KDE Likelihood	0.774	**0.810**	**0.780**	0.687	**0.953**	**0.918**
DPEA	0.682	0.598	0.622	0.665	0.875	0.787
PWrong	0.674	0.683	0.745	0.750	0.935	0.901
SVM/KDE FUSION	0.668	0.603	0.571	0.642	0.892	0.800
Random	0.637	0.491	0.456	0.576	0.820	0.700
HMS-Out	0.877	0.766	0.658	**0.794**	0.950	0.902
HMS-CI	0.859	0.713	0.613	0.708	0.944	0.879

of instances from class i), $E_{max}(N)$ is the maximum entropy (e.g. when all the classes have equal number of instances), λ is the parameter that balances the error to achieve the desired entropy τ vs. the number of examples we use from the total number of instances that are available. We used a very small value for the parameter $\lambda = 1/(100 * M)$, where M is the total number of instances. This ensures that we achieve normalized entropy very close to the τ and only after that do we maximize the number of instances. We created a set of 10 benchmarks from each dataset by sampling N_i number of examples from class i by varying τ from 0.1 to 1 with an increment of 0.1. Table 5.2 shows the summary of the benchmarks created. Note that, due to the limitations of number of instances from each class, we were not able to exactly achieve the target normalized entropy for all of the datasets.

5.4.2 Experimental Setup

We employed the algorithms described in Section 5.3 on the original datasets shown in Table 5.1 and on the constructed benchmarks shown in Table 5.2. All the algorithms were allowed to make up to 200 oracle queries. Among these algorithms, isolation forest is the only randomized algorithm, so we averaged over 10 runs on each dataset. We ran isolation forest with 100 trees with a subsample size

of 256. The HMS algorithms were run according to the setup described in Vatturi and Wong [2009]. For the rest of the algorithms the parameters were set according to Hospedales et al. [2011] and Hospedales et al. [2012].

5.4.3 Results

Class Discovery Curve: To evaluate different algorithms, we computed the number of classes an algorithm discovers within a given number of queries. We plot this as a class discovery curve (CDC) where on the X axis we have the number of queries to the oracle and on the Y axis, the number of classes discovered after making that many queries. Figures 5.1, 5.2 and 5.2 show the results of the algorithms on the original datasets described in Table 5.1. In general, the hierarchical mean shift algorithms are performing well in Abalone, Letterrec, Glass and Covertype datasets and DPEA is performing better on the other two datasets Leaf and Yeast. The HMS-Out and PWrong comes consistently near the top in four of the datasets. The SVM-Entropy and the IF(No Feedback) methods come near the bottom in four and two datasets respectively along with Random, which comes bottom in three datasets. The Random query criteria perform very well in the Leaf dataset, where the number of examples from different classes are almost equal, and in the Covertype, dataset where the number of classes is very small. In most of the datasets, the Random criterion seems equal or better than the SVM Entropy criterion. One reason for this is that the SVM Entropy is probably trying to learn the class boundary very well by making many queries from the same class which eventually hurts its class discovery performance.

Area Under the Class Discovery Curve (AUC): To get a single metric from a class discovery curve, we computed the area under the CDC curve and normalized it with the best area under the curve possible by an oracle. The oracle discovers one class on each query until all the classes are discovered. This gives a score ranging from 0 to 1, where a score close to one means performance is close to the oracle. In Figures 5.4, 5.5 and 5.6, we show results from the all 60 benchmark datasets listed in Table 5.2, where on the X axis we plotted the normalized entropy and on the Y axis the corresponding AUC achieved by each algorithm. To see if performance increase correlates with increasing entropy, we fit a regression line for each curve and report the slope of the lines in Table 5.4. We observe that most

Table 5.4: Slopes of the fitted regression lines of AUC vs. entropy curves shown in Figures 5.4, 5.5 and 5.6

	Leaf	Abalone	Yeast	Letter Recognition	Glass	Covertype
IF (No Feedback)	-1.38	0.34	0.20	0.35	0.08	-0.02
IF (Feedback)	-0.37	0.34	0.25	0.33	0.07	0.04
SVM Entropy	-1.25	0.60	0.30	0.34	-0.01	-0.14
KDE Likelihood	0.66	0.64	0.25	0.28	0.01	-0.06
DPEA	2.39	0.86	0.24	1.02	0.06	0.54
PWrong	0.31	0.68	0.20	0.28	0.00	-0.03
SVM/KDE FUSION	0.69	0.53	0.34	0.66	0.07	0.12
Random	1.03	0.94	0.51	0.94	0.11	0.58
HMS-Out	-0.43	0.43	0.31	-0.14	0.11	0.07
HMS-CI	-2.32	0.60	0.31	-0.12	0.12	-0.02

of the slopes are positive, which indicates increasing trend with the increasing entropy. In the most of the datasets almost all of the algorithms show increasing trend (e.g., in Abalone, Yeast, Letter Recognition and Glass datasets). With high entropy, the datasets become easier since the number of instances for each class become similar. The Random algorithm consistently shows the increasing trend across all of the datasets, since the Random strategy tends to perform better when class instances are balanced.

Discovering percentage of classes: To get a better picture of which algorithm performs better, we compute several metrics: number of queries required to discover 25%, 50%, 75% and 100% of the classes. We normalize them by the best performing algorithm (e.g., the one that asked minimum number of queries to discover a certain fraction of classes). This gives us a score of 1 for the best algorithm and score relative to 1 for the others. This score is then averaged over 60 benchmark datasets, and the average is shown in Table 5.3. Isolation forest with feedback performs best in discovering 25% of the classes. KDE Likelihood performs better in discovering 50% and 75% of the classes, and HMS-out is best for discovering all the classes. Since the isolation forest algorithm is an anomaly

detection based algorithm, it might be finding the rare classes earlier but can't get to find the majority classes. This is hurting the performance later in the iterations. A similar conclusion can be made for the KDE likelihood, which is another anomaly detection-based approach. HMS-Out is finding all the classes much faster than others, since it uses the clustering approach along with anomaly detection criteria. It seems the clustering property is helping it to find the additional classes quickly.

Normalized AUC: We normalized the area under the class discovery curve by the best AUC and then plotted the average normalized AUC over all 60 benchmark datasets in the AUC column. KDE likelihood performs better than other algorithms for this metric. HMS-out placed second.

Rare Class Discovery: The class discovery curve always gives a reward of 1 whenever an algorithm discovers a new class regardless of the class being rare or not. To give more credit whenever a rare class is discovered we produce a different class discovery curve where we give a reward of $-\log_2(p)$ which is the number of bits of information uncovered by discovering a class which has a fraction of instances p. Similarly, we normalize this by the best performing algorithm to get a score from 0 to 1. The Reward column shows the corresponding scores. KDE likelihood performs better under this metric as well. We don't observe much change in the ranking of algorithms under the AUC column and the Reward column except that PWrong becomes 3^{rd} in the Reward column and 4^{th} in the AUC column topping over HMS-CI. This might be the cause of using clustering approach by the HMS-CI which has an inherent bias to choose majority classes first.

The overall winner is the KDE likelihood algorithm, which wins 4 out of these 6 metrices. HMS-Out wins on discovering all the classes. HMS-out is also the second place in 4 of the metrics. Interestingly, all the top performing algorithms under different metrics are actually based on some kind of anomaly detection technique which indicates that likelihood criteria are quite effective for discovering new classes.

5.5 Summary

In this chapter we discussed the class discovery problem. We described several algorithms for the class discovery problem that were borrowed from some other area

of research like anomaly detection, clustering and rare category detection, etc. We evaluated these algorithms on some standard datasets along with some constructed benchmarks. Our evaluation showed that in general the likelihood criteria usually used in anomaly detection techniques perform better than other approaches. In future work, it would be interesting to include some other class discovery techniques and evaluate them with a richer set of benchmarks that could be constructed by some other appropriate dataset properties like isolation, compactness and so on.

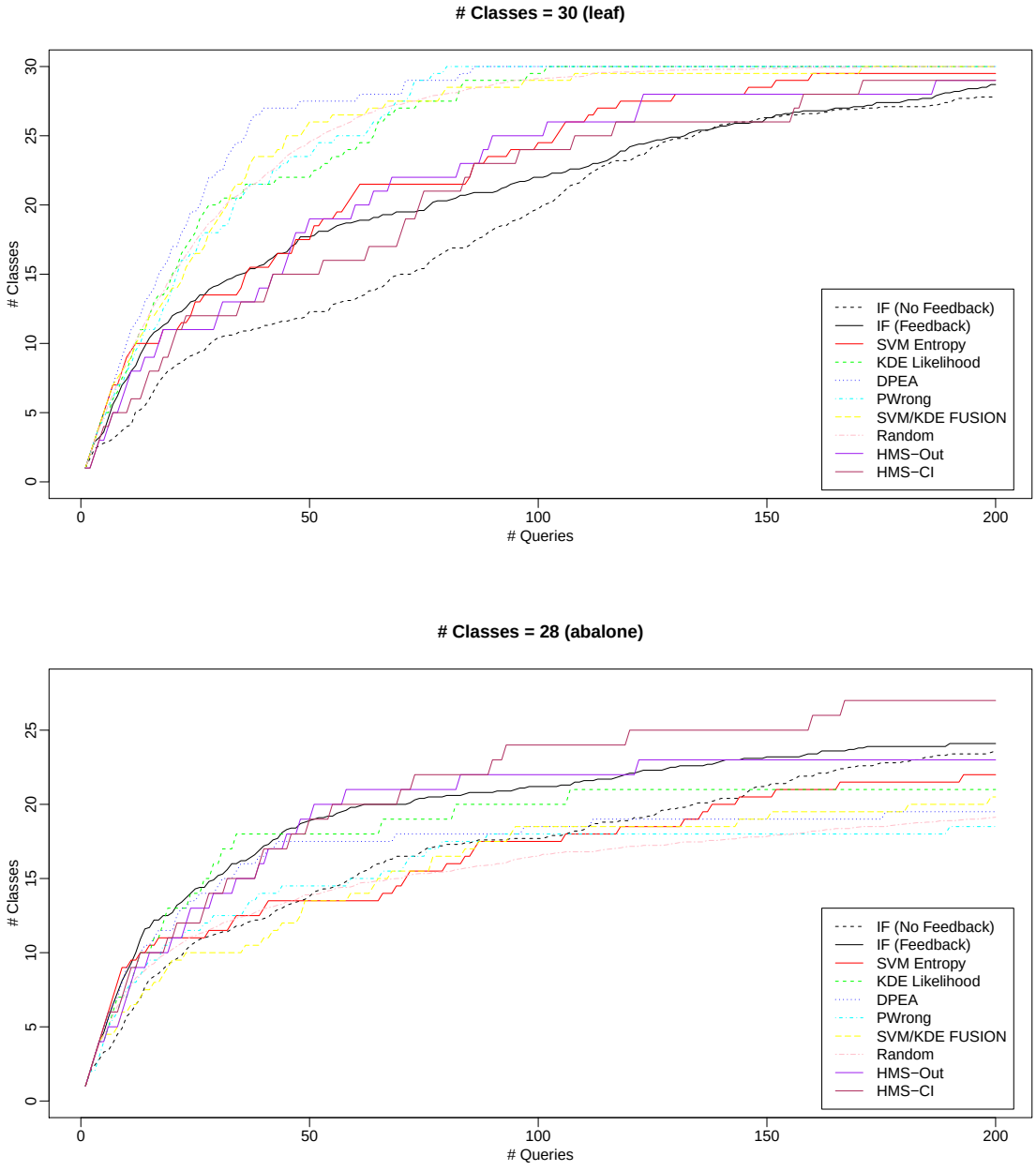

Figure 5.1: Class discovery curve for different algorithms on the Leaf and Abalone datasets

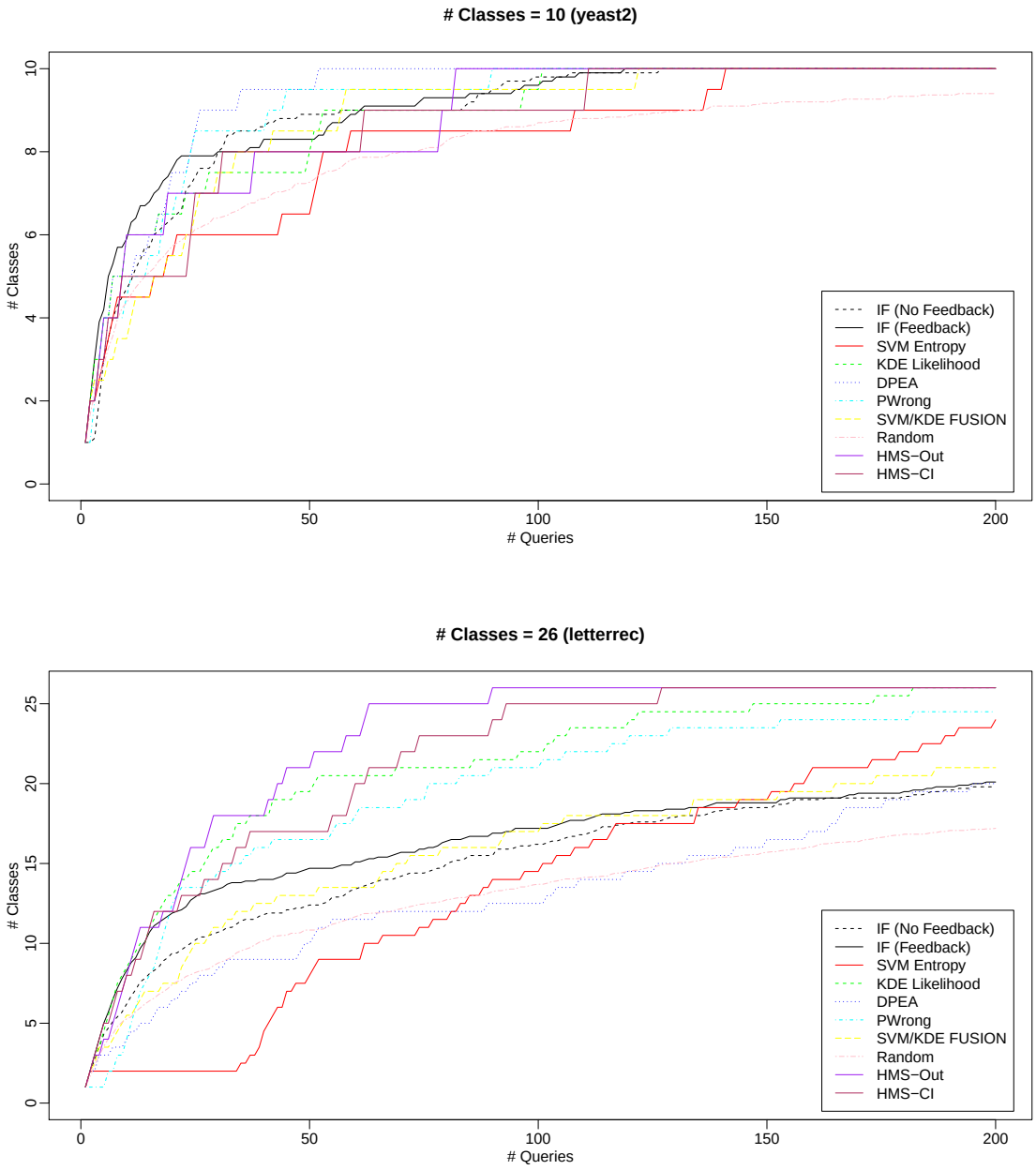

Figure 5.2: Class discovery curve for different algorithms on the Yeast and Letter recognition datasets

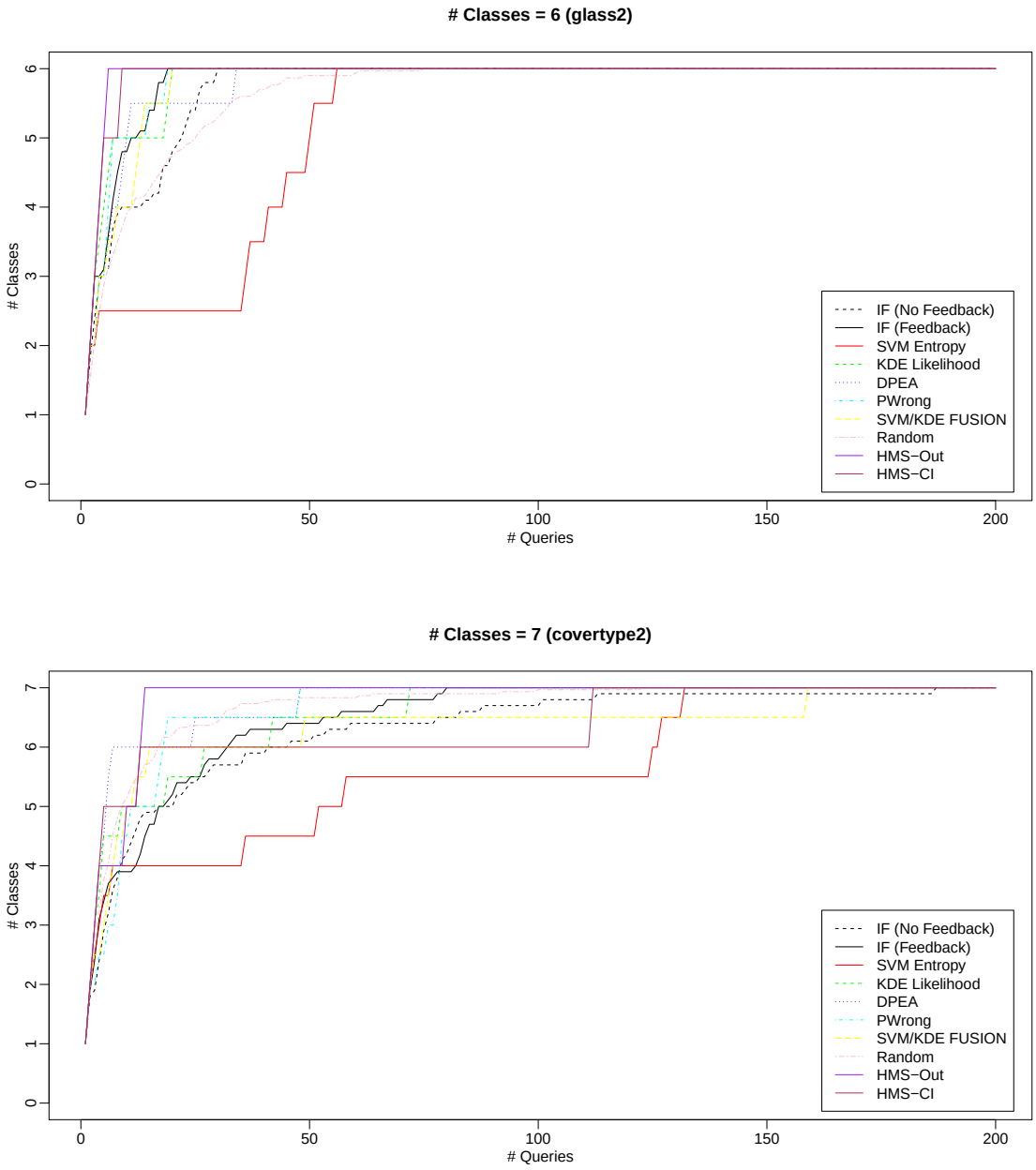

Figure 5.3: Class discovery curve for different algorithms on the Glass and Covertype datasets

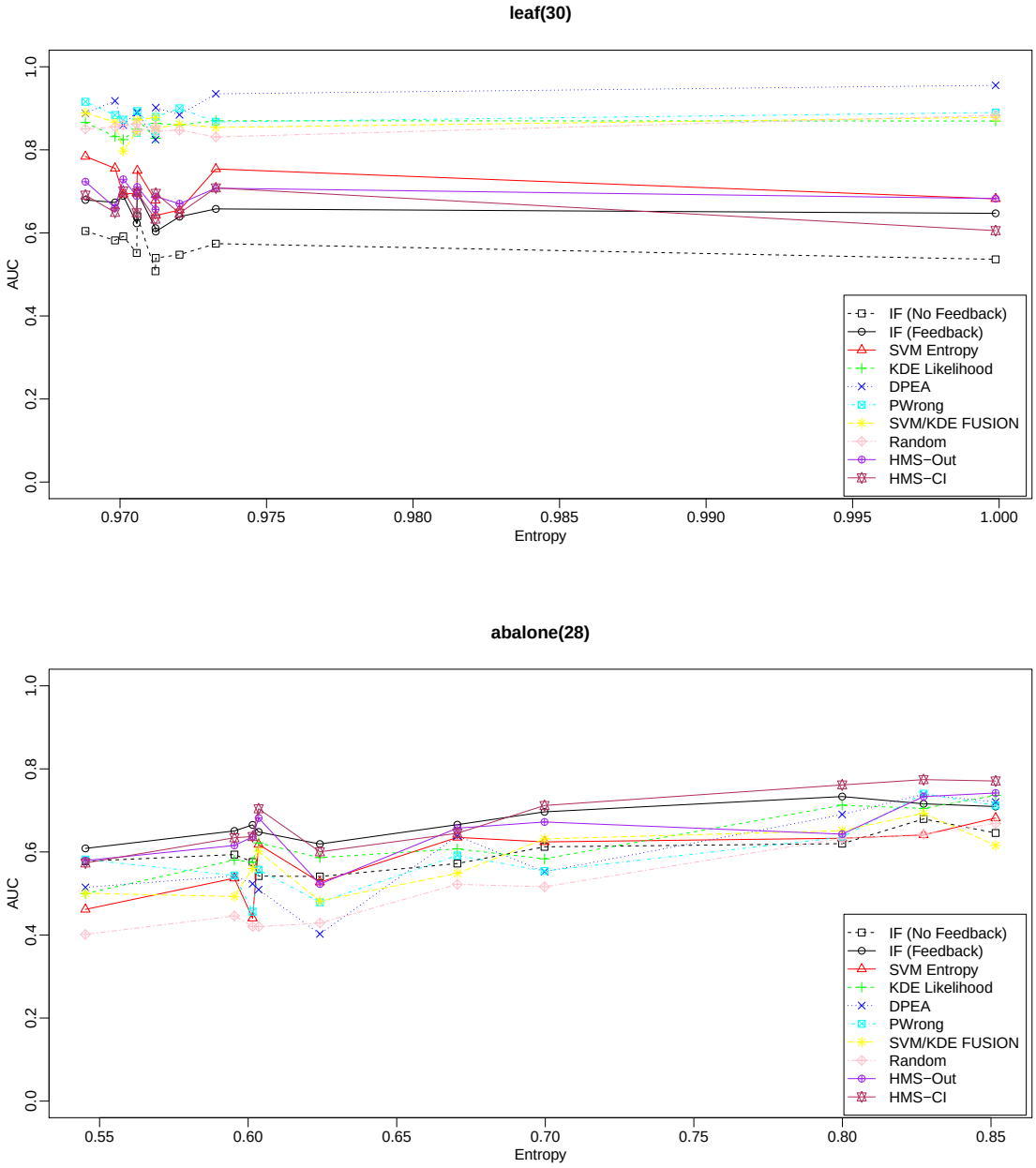

Figure 5.4: Entropy vs. normalized AUC (area under the class discovery curve) for different algorithms on Leaf and Abalone benchmark datasets

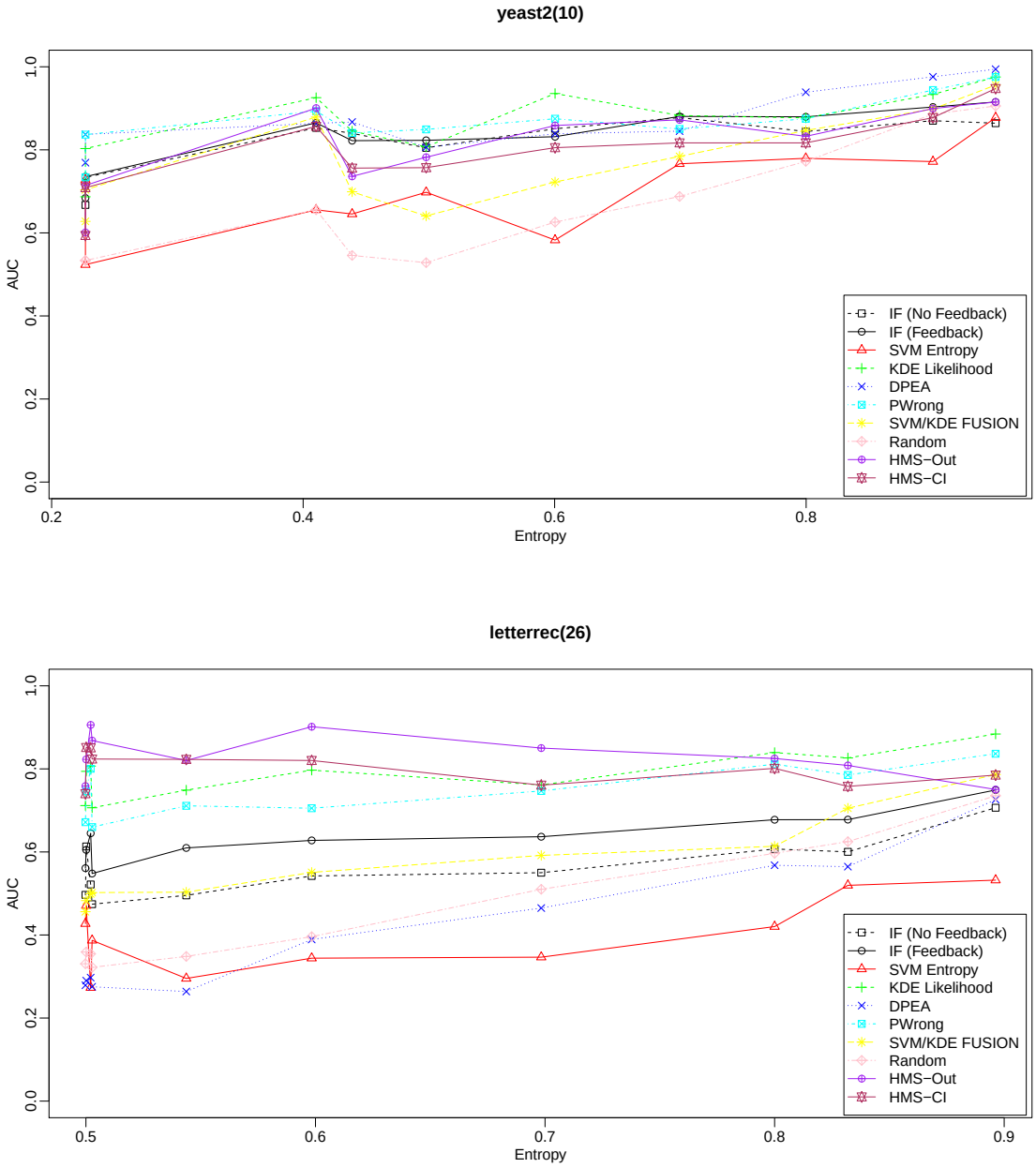

Figure 5.5: Entropy vs. normalized AUC (area under the class discovery curve) for different algorithms on Yeast and Letter recognition benchmark datasets

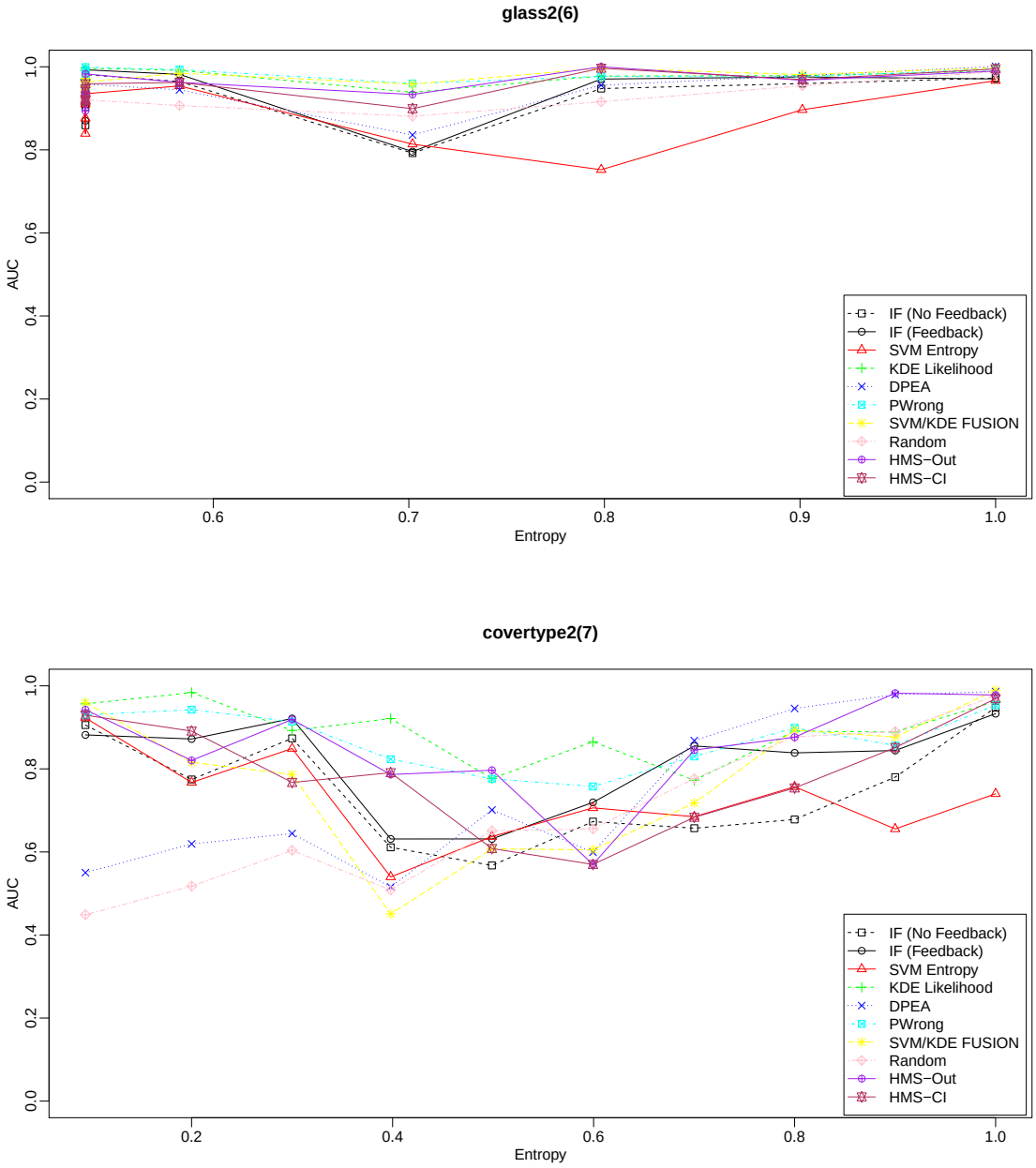

Figure 5.6: Entropy vs. normalized AUC (area under the class discovery curve) for different algorithms on the Glass and Covertype benchmark datasets

Chapter 6: Conclusions and Future Work

In this dissertation we focused on the anomaly detection problem in general and its potential application in computer security. We developed sample complexity results for anomaly detection to understand why many detectors perform very well with a small number of examples. This analysis unified a number of existing state-of-the-art anomaly detectors into a single framework underlying different pattern spaces, that is, all these algorithms works under the same principle except that they focus on different pattern spaces.

We proposed explanation ideas in terms of input features that explain why an anomaly detector choose a specific point as anomalous. Explanation helps the end-user analysts in their investigation of anomalies and also provide some reasons in justifying whether the detector picked the anomalies for the right reasons.

We then developed some approaches to incorporate the analyst's feedback to improve the performance of the anomaly detector. We showed the effectiveness of the explanation and feedback approaches on some red team attack data as well as on some real world network security data. We also measured their performance on a large number of benchmark datasets to show their superior performance over other state of the art algorithms.

Finally, we explored the class discovery problem and described different approaches from the literature and analyzed their empirical performance on a suite of benchmark datasets that are constructed by varying some dataset properties. We then compare the performance of the various algorithms and identify which one performs better empirically.

In the future, it would be interesting to discover some other pattern spaces where other state-of-the-art anomaly detectors operate and provide a thorough comparison in terms of their sample complexities. For the explanation work, it would be interesting to provide some more fine-grained explanation than the features along with their quantitative contribution. That is, if we could decompose the anomaly score into the combination of multiple explanations, where each explanation is intuitive to a human. Clustering anomalies based on their explanations

would also be interesting, since the resulting cluster could represent a class of anomalies which might correspond to a specific security attack.

For the feedback work, it would be interesting to handle noisy feedback, since in practice the analysts sometimes are not entirely sure of an entity whether it's actually part of an attack due to lack of information or time constraints. The incorporation of feedback could be extended to other anomaly detectors whose score function can't be represented as a linear function. For the class discovery algorithms, it would be interesting to get more insight into which algorithm performs better in what situation or under what properties of a dataset. To that purpose, a large number of benchmark datasets can be constructed by identifying some additional key dataset properties like compactness, isolation, and so on.

Bibliography

[Adda et al., 2007] Adda, M., Wu, L., and Feng, Y. (2007). Rare itemset mining. In *Machine Learning and Applications, 2007. ICMLA 2007. Sixth International Conference on*, pages 73–80. IEEE.

[Antoniak, 1974] Antoniak, C. E. (1974). Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The annals of statistics*, pages 1152–1174.

[Auger and Doerr, 2011] Auger, A. and Doerr, B. (2011). *Theory of randomized search heuristics: Foundations and recent developments*, volume 1. World Scientific.

[Baehrens et al., 2010] Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., and Müller, K.-R. (2010). How to explain individual classification decisions. *The Journal of Machine Learning Research*, 11:1803–1831.

[Bishop et al., 2006] Bishop, C. M. et al. (2006). *Pattern recognition and machine learning*, volume 4. springer New York.

[Blumer et al., 1989] Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1989). Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965.

[Chandola et al., 2009] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15.

[Chen et al., 2015] Chen, B., Ting, K. M., Washio, T., and Haffari, G. (2015). Half-space mass: a maximally robust and efficient data depth method. *Machine Learning*, 100(2-3):677–699.

[Dang et al., 2014] Dang, X. H., Assent, I., Ng, R. T., Zimek, A., and Schubert, E. (2014). Discriminative features for identifying and interpreting outliers. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 88–99. IEEE.

[Dang et al., 2013] Dang, X. H., Micenková, B., Assent, I., and Ng, R. T. (2013). Local outlier detection with interpretation. In *Machine Learning and Knowledge Discovery in Databases*, pages 304–320. Springer.

[Das et al., 2016] Das, S., Wong, W.-K., Dietterich, T. G., Fern, A., and Emmott, A. (2016). Incorporating expert feedback into active anomaly discovery. In *Proceedings of the IEEE ICDM*, pages 853–858.

[Das et al., 2017] Das, S., Wong, W.-K., Fern, A., Dietterich, T. G., and Siddiqui, M. A. (2017). Incorporating feedback into tree-based anomaly detection. *arXiv preprint arXiv:1708.09441*.

[Deng, 2013] Deng, H. (2013). Guided random forest in the rrf package. *arXiv:1306.0237*.

[Deselaers et al., 2010] Deselaers, T., Heigold, G., and Ney, H. (2010). Object classification by fusing svms and gaussian mixtures. *Pattern Recognition*, 43(7):2476–2484.

[Devroye et al., 2013] Devroye, L., Györfi, L., and Lugosi, G. (2013). *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media.

[Dietterich and Zemicheal, 2018] Dietterich, T. G. and Zemicheal, T. (2018). Anomaly detection in the presence of missing values. *arXiv preprint arXiv:1809.01605*.

[Dong et al., 2017] Dong, B., Chen, Z., Wang, H. W., Tang, L.-A., Zhang, K., Lin, Y., Li, Z., and Chen, H. (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.*

[Duan et al., 2014] Duan, L., Tang, G., Pei, J., Bailey, J., Campbell, A., and Tang, C. (2014). Mining outlying aspects on numeric data. *Data Mining and Knowledge Discovery*, pages 1–36.

[Emmott et al., 2015] Emmott, A., Das, S., Dietterich, T. G., Fern, A., and Wong, W. (2015). Systematic construction of anomaly detection benchmarks from real data. *CoRR*, abs/1503.01158.

[Emmott et al., 2013] Emmott, A. F., Das, S., Dietterich, T., Fern, A., and Wong, W.-K. (2013). Systematic construction of anomaly detection benchmarks from real data. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, pages 16–21. ACM.

[Ferguson, 1973] Ferguson, T. S. (1973). A bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230.

[Fernández-Delgado et al., 2014] Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems. *J. Mach. Learn. Res*, 15(1):3133–3181.

[Forrest et al., 1996] Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A. (1996). A sense of self for unix processes. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 120–128. IEEE.

[Gao et al., 2004] Gao, D., Reiter, M. K., and Song, D. (2004). Gray-box extraction of execution graphs for anomaly detection. In *Proc. of the 11th ACM conference on Computer and communications security*, pages 318–329.

[Görnitz et al., 2013] Görnitz, N., Kloft, M. M., Rieck, K., and Brefeld, U. (2013). Toward supervised anomaly detection. *Journal of Artificial Intelligence Research*.

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

[Guha et al., 2016] Guha, S., Mishra, N., Roy, G., and Schrijvers, O. (2016). Robust Random Cut Forest Based Anomaly Detection On Streams. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48.

[Haines and Xiang, 2014] Haines, T. S. and Xiang, T. (2014). Active rare class discovery and classification using dirichlet processes. *International Journal of Computer Vision*, 106(3):315–331.

[He and Carbonell, 2009] He, J. and Carbonell, J. (2009). Prior-free rare category detection. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 155–163. SIAM.

[He and Carbonell, 2008] He, J. and Carbonell, J. G. (2008). Nearest-neighbor-based active learning for rare category detection. In *Advances in neural information processing systems*, pages 633–640.

[Hettich and Bay, 1999] Hettich, S. and Bay, S. (1999). The UCI KDD archive [http://kdd.ics.uci.edu]. Irvine, CA: University of california. *Department of Information and Computer Science*, page 152.

[Hodge and Austin, 2004] Hodge, V. J. and Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126.

[Hospedales et al., 2011] Hospedales, T. M., Gong, S., and Xiang, T. (2011). Finding rare classes: Active learning with generative and discriminative models. *IEEE transactions on knowledge and data engineering*, 25(2):374–386.

[Hospedales et al., 2012] Hospedales, T. M., Gong, S., and Xiang, T. (2012). A unifying theory of active discovery and learning. In *European conference on computer vision*, pages 453–466. Springer.

[Jiang et al., 2017] Jiang, S., Malkomes, G., Converse, G., Shofner, A., Moseley, B., and Garnett, R. (2017). Efficient nonmyopic active search. In *International Conference on Machine Learning*, pages 1714–1723.

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

[Krause and Golovin, 2014] Krause, A. and Golovin, D. (2014). Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*, pages 71–104. Cambridge University Press.

[Leung et al., 2000] Leung, Y., Zhang, J.-S., and Xu, Z.-B. (2000). Clustering by scale-space filtering. *IEEE Transactions on pattern analysis and machine intelligence*, 22(12):1396–1410.

[Lichman, 2013] Lichman, M. (2013). UCI Machine Learning Repository. *[http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.*

[Liu et al., 2008] Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 413–422. IEEE.

[Micenková et al., 2013] Micenková, B., Ng, R. T., Dang, X.-H., and Assent, I. (2013). Explaining outliers by subspace separability. In *IEEE 13th International Conference on Data Mining (ICDM)*, pages 518–527. IEEE.

[Pelleg and Moore, 2005] Pelleg, D. and Moore, A. W. (2005). Active learning for anomaly and rare-category detection. In *Advances in neural information processing systems*.

[Pevnỳ, 2016] Pevnỳ, T. (2016). Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102(2):275–304.

[Raginsky et al., 2012] Raginsky, M., Willett, R. M., Horn, C., Silva, J., and Marcia, R. F. (2012). Sequential anomaly detection in the presence of noise and limited feedback. *IEEE Transactions on Information Theory*, 58(8):5544–5562.

[Rasmussen, 2000] Rasmussen, C. E. (2000). The infinite gaussian mixture model. In *Advances in neural information processing systems*, pages 554–560.

[Robnik-Sikonja and Kononenko, 2008] Robnik-Sikonja, M. and Kononenko, I. (2008). Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):589–600.

[Schölkopf et al., 2001] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471.

[Scott and Nowak, 2006] Scott, C. D. and Nowak, R. D. (2006). Learning minimum volume sets. *The Journal of Machine Learning Research*, 7:665–704.

[Sekar et al., 2001] Sekar, R., Bendre, M., Dhurjati, D., and Bollineni, P. (2001). A fast automaton-based method for detecting anomalous program behaviors. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 144–155. IEEE.

[Settles, 2009] Settles, B. (2009). Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

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

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

[Shalizi, 2016] Shalizi, C. (2016). *Advanced Data Analysis from an Elementary Point of View*. Cambridge University Press.

[Shu et al., 2015] Shu, X., Yao, D., and Ramakrishnan, N. (2015). Unearthing stealthy program attacks buried in extremely long execution paths. In *Proc. of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 401–413.

[Siddiqui et al., 2016] Siddiqui, M. A., Fern, A., Dietterich, T., and Das, S. (2016). Finite sample complexity of rare pattern anomaly detection. In *Conference on Uncertainty in Artificial Intelligence*, UAI.

[Siddiqui et al., 2015] Siddiqui, M. A., Fern, A., Dietterich, T. G., and Wong, W.-K. (2015). Sequential feature explanations for anomaly detection. In *Proceedings of the ACM SIGKDD workshop on outlier detection and description*. ACM.

[Siddiqui et al., 2019a] Siddiqui, M. A., Fern, A., Dietterich, T. G., and Wong, W.-K. (2019a). Sequential feature explanations for anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(1):1.

[Siddiqui et al., 2018a] Siddiqui, M. A., Fern, A., Dietterich, T. G., Wright, R., Theriault, A., and Archer, D. W. (2018a). Feedback-guided anomaly discovery via online optimization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2200–2209. ACM.

[Siddiqui et al., 2018b] Siddiqui, M. A., Fern, A., Wright, R., Theriault, A., Archer, D., and Maxwell, W. (2018b). Detecting cyberattack entities from audit data via multi-view anomaly detection with feedback. *Artificial Intelligence for Cyber Security, AAAI Workshop*.

[Siddiqui et al., 2019b] Siddiqui, M. A., Stokes, J. W., Seifert, C., Argyle, E., McCann, R., Neil, J., and Carroll, J. (2019b). Detecting cyber attacks using anomaly detection with explanations and expert feedback. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2872–2876. IEEE.

[Sillito and Fisher, 2007] Sillito, R. R. and Fisher, R. B. (2007). Incremental one-class learning with bounded computational complexity. In *International Conference on Artificial Neural Networks*, pages 58–67. Springer.

[Strumbelj and Kononenko, 2010] Strumbelj, E. and Kononenko, I. (2010). An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, 11:1–18.

[Štrumbelj and Kononenko, 2013] Štrumbelj, E. and Kononenko, I. (2013). Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, pages 1–19.

[Szathmary et al., 2007] Szathmary, L., Napoli, A., and Valtchev, P. (2007). Towards rare itemset mining. In *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, volume 1, pages 305–312. IEEE.

[Tan et al., 2011] Tan, S. C., Ting, K. M., and Liu, T. F. (2011). Fast anomaly detection for streaming data. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence - Volume Two*, pages 1511–1516.

[Vapnik and Vapnik, 1998] Vapnik, V. N. and Vapnik, V. (1998). *Statistical learning theory*, volume 1. Wiley New York.

[Varadarajan et al., 2017] Varadarajan, J., Subramanian, R., Ahuja, N., Moulin, P., and Odobez, J.-M. (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*, pages 615–623.

[Vatturi and Wong, 2009] Vatturi, P. and Wong, W.-K. (2009). Category detection using hierarchical mean shift. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–856. ACM.

[Veeramachaneni et al., 2016] Veeramachaneni, K., Arnaldo, I., Korrapati, V., Bassias, C., and Li, K. (2016). AI2: 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*, pages 49–54. IEEE.

[Vinh et al., 2015] Vinh, N. X., Chan, J., Bailey, J., Leckie, C., Ramamohanarao, K., and Pei, J. (2015). Scalable outlying-inlying aspects discovery via feature ranking. In *Advances in Knowledge Discovery and Data Mining*, pages 422–434. Springer.

[Wu et al., 2014] Wu, K., Zhang, K., Fan, W., Edwards, A., and Yu, P. S. (2014). Rs-forest: a rapid density estimator for streaming anomaly detection. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 600–609. IEEE.

APPENDICES

Appendix A: Proofs

A.1 NP-hardness of SFE-Decide

We now prove that the decision problem, SFE-Decide, which corresponds to the optimization problem in Equation 3.3, is NP-hard.

Proof. To prove that SFE-Decide is NP-hard we reduce the well known NP-Complete problem Vertex Cover to SFE-Decide.

Vertex Cover: *Does there exist a vertex cover of size* $\leq k$ *in graph* $G = (\mathcal{V}, \mathcal{E})$

SFE-Decide: *Does there exist an SFE* E *for instance* x *and density* f *that satisfies*

$$\sum_{\alpha} \min\{i : f(x_{E_i}) < \tau(E_i, \alpha)\} p(\alpha) \leq t.$$

Reduction. The basic idea of the reduction is to encode the graph $G = (\mathcal{V}, \mathcal{E})$ into the instance x as a vertex-edge incidence matrix. We view x as a $|\mathcal{V}|$ dimensional feature vector with each feature denoted by x_i. Each feature is an $|\mathcal{E}|$-bit integer where bit j of feature i, denoted by x_{ij}, is defined as:

$$x_{ij} = \begin{cases} 1 & if\ vertex\ \mathcal{V}_i\ is\ incident\ on\ edge\ \mathcal{E}_j \\ 0 & otherwise \end{cases}$$

Since x is integer-valued, we construct the marginal distribution function $f(x_s)$ as a discrete distribution (or probability function) as follows:

$$f(x_s) = \begin{cases} \frac{1}{C} & if \sum_j \mathbb{I}(\sum_{i \in s} x_{ij} > 0) < |\mathcal{E}| \\ 0 & otherwise \end{cases}$$

Here, C is a normalizing constant and $\mathbb{I}$ is an indicator function. Intuitively, $f(x_s)$ is defined to be a uniform distribution over all possible values of x_s that correspond to a vertex cover of the graph. To see this, note that the summation

$\sum_j \mathbb{I}(\sum_{i \in s} x_{ij} > 0)$ is computing the number of edges covered by the set of vertices in s. We say a value of x_s is valid if this summation exactly equals the number of edges.

Continuing the reduction we set $t = k$, $\tau(s, \alpha) = \frac{1}{C}$ and

$$p(\alpha) = \begin{cases} 1 & if\ \alpha = \alpha_0 \\ 0 & otherwise \end{cases}$$

where α_0 is a constant. Now, we observe that by the construction of $p(\alpha)$ as a deterministic distribution, SFE-DECIDE no longer involves a summation over α and can be simplified to the following problem:

$$\text{Does there exist an SFE } E \text{ satisfying } min\{i : f(x_{E_i}) < \tfrac{1}{C}\} \leq t$$

Now, we first show that a vertex cover of size $\leq k$ in $G \rightarrow$ existence of an SFE E satisfying $min\{i : f(x_{E_i}) < \frac{1}{C}\} \leq t$. Suppose, $|s| \leq k$ is the set of vertices forming the vertex cover in G. Then, by construction $\sum_j \mathbb{I}(\sum_{i \in s} x_{ij} > 0) = |\mathcal{E}|$ i.e. $f(x_s) = 0$. We can now construct an SFE E by setting the prefix $E_i = s$ and filling the rest of the features arbitrarily with the remaining features. Its easy to see that such an SFE E satisfies $min\{i : f(x_{E_i}) < \frac{1}{C}\} \leq t$ since $|E_i| \leq t$. Hence, E is an SFE corresponding to the vertex cover s.

We now prove the other direction: The existence of an SFE E satisfying $min\{i : f(x_{E_i}) < \frac{1}{C}\} \leq t \rightarrow$ existence of a vertex cover of size $\leq k$ in G. Since $min\{i : f(x_{E_i}) < \frac{1}{C}\} \leq t$, the inequality $f(x_{E_i}) < \frac{1}{C}$ is satisfied for some $i \leq t$. Since $t = k$, we have a feature subset E_i such that $|E_i| \leq k$. Now, we show that E_i is also a vertex cover in G. Since $f(x_{E_i}) < \frac{1}{C}$ we have $f(x, E_i) = 0$, i.e., $\sum_j \mathbb{I}(\sum_{p \in E_i} x_{pj} > 0) = |\mathcal{E}|$. Hence, for each j the indicator function is true which implies edge $\mathcal{E}_j$ is covered by some vertex in E_i. Hence, E_i is a vertex cover in G corresponding to SFE E with $|E_i| \leq k$. □