Sequential Feature Explanations for Anomaly Detection

MD AMRAN SIDDIQUI, ALAN FERN, THOMAS G. DIETTERICH, and
WENG-KEEN WONG, Oregon State University

In many applications, an anomaly detection system presents the most anomalous data instance to a human analyst, who then must determine whether the instance is truly of interest (e.g., a threat in a security setting). Unfortunately, most anomaly detectors provide no explanation about why an instance was considered anomalous, leaving the analyst with no guidance about where to begin the investigation. To address this issue, we study the problems of computing and evaluating sequential feature explanations (SFEs) for anomaly detectors. An SFE of an anomaly is a sequence of features, which are presented to the analyst one at a time (in order) until the information contained in the highlighted features is enough for the analyst to make a confident judgement about the anomaly. Since analyst effort is related to the amount of information that they consider in an investigation, an explanation's quality is related to the number of features that must be revealed to attain confidence. In this article, we first formulate the problem of optimizing SFEs for a particular density-based anomaly detector. We then present both greedy algorithms and an optimal algorithm, based on branch-and-bound search, for optimizing SFEs. Finally, we provide a large scale quantitative evaluation of these algorithms using a novel framework for evaluating explanations. The results show that our algorithms are quite effective and that our best greedy algorithm is competitive with optimal solutions.

CCS Concepts: • **Computing methodologies** → **Anomaly detection**;

Additional Key Words and Phrases: Anomaly detection, anomaly explanation, anomaly interpretation, explanation evaluation

ACM Reference format:
Md Amran Siddiqui, Alan Fern, Thomas G. Dietterich, and Weng-Keen Wong. 2019. Sequential Feature Explanations for Anomaly Detection. *ACM Trans. Knowl. Discov. Data* 13, 1, Article 1 (January 2019), 22 pages.
https://doi.org/10.1145/3230666

1 INTRODUCTION

Anomaly detection is the problem of identifying anomalies in a dataset, 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

This work is partially supported by DARPA under Contracts N66001-17-2-4030 and FA8650-15-C7557, and the Future of Life Institute under Contract W911NF-11-C-0088.
Authors' addresses: Md A. Siddiqui, A. Fern, T. G. Dietterich, and W.-K. Wong, School of EECS, 1148 Kelley Engineering Center, Oregon State University, Corvallis, Oregon 97331-5501; emails: {siddiqmd, afern, tgd, wong}@eecs.oregonstate.edu.

1556-4681/2019/01-ART1 $15.00
https://doi.org/10.1145/3230666

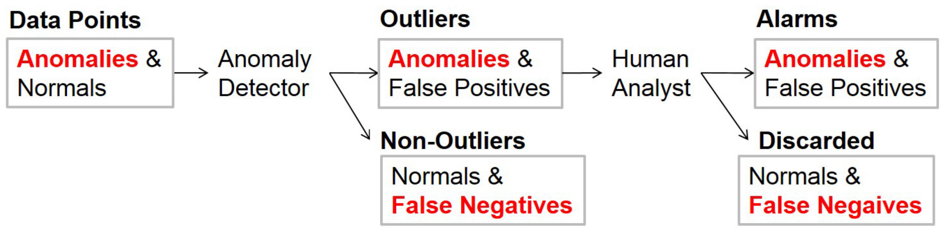

Fig. 1. The anomaly detection pipeline addressed by this article (from left to right). The original dataset contains both normal points and anomalies. Our goal is to detect the true semantic anomalies. First, an anomaly detector is applied to identify a set of statistical outliers, which are further 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 the following 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 article. 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 article is to compute explanations of statistical outliers that reduce the effort required to detect such true anomalies.

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 1.

In this article, 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 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 article, 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 article is organized as follows. Section 2 reviews related work on explanations for both supervised learning and anomaly detection. Next, Section 3 presents the anomaly-detection framework used in this article. Section 4 then more formally presents the concept of SFEs and possible quality metrics. Section 5 formulates an optimal SFE computation method, and then describes two greedy methods for computing SFEs. Section 7.2 shows a comparison between the optimal and greedy methods in achieving the optimal objective value. Section 6 introduces our quantitative evaluation framework for SFEs, and finally Section 7 presents experiments evaluating the introduced methods within the framework.

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 [12], or by considering the local gradient of the classifier's prediction score with respect to the features for a particular instance [1].

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 [13, 14]. 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 [12].

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. [11] 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" [6, 15]. For example, Duan et al. developed a method called OAMiner [6], 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 [15], 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 article are sensitive to the particular anomaly detector.

Other work on computing feature-subset explanations [4] 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. [3], 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 [1, 3] or to test whether a computed explanation collectively conforms to some known concept in the dataset [1], 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 ANOMALY DETECTION FORMULATION

We consider anomaly detection problems 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 that can be costly in many domains.

4 SEQUENTIAL FEATURE EXPLANATIONS

In order to reduce the analyst's effort for detecting anomalies, we propose to provide the analyst with *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 do not study that case in this article.

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 6 addresses this issue and describes an approach for conducting wide evaluations in terms of MFP.

5 EXPLANATION METHODS

We now consider methods for computing SFEs for anomaly detectors. Prior work on computing explanations for anomaly detectors has either computed explanations that do not depend on the particular anomaly detector used (e.g., [11]) or used methods that were specific to a particular anomaly detector (e.g., [4]). 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 mismatch 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*.

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

[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 article, where the density function is used to compute scores.

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.,

$$\mathrm{SP}(x, E, \alpha) = \min\{i : f(x_{E_i}) < \tau_i(E, \alpha)\}. \tag{1}$$

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,

$$\mathrm{ESP}(x, E) = \sum_{\alpha} \mathrm{SP}(x, E, \alpha) p(\alpha), \tag{2}$$

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:

$$\operatorname*{argmin}_{E} \mathrm{ESP}(x, E). \tag{3}$$

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 5.1. SFE-Decide *is NP-hard.*

The proof is in Appendix A. This hardness result motivates us to consider two optimization approaches described later. 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.

5.2 Greedy Algorithms

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 [10]. 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.

5.2.2 *Dropout Methods.*

The next two methods are inspired by the work of Robnik-Sikonja and Kononenko [12] 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:

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

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 $f \in (1:d)$ $AND$ $f \notin node.RankedFeatureList$ do
            $child := copy(node)$
            $child.AppendFeature(f)$
            $child.UpdateBounds()$
            if $BestNode.UB > child.UB$ then
                $BestNode := child$
            end
            $Q.Insert(child)$
        end
    end
end
$return$ $BestNode.RankedFeatureList$
```

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.

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 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). Later, 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 SeqMarg algorithm (Section 5.2) starting with E_i in order to select the remaining features of a complete SFE. The ESP (Equation (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 (2) and interchanging the order of minimization over SFEs and summation/expectation over α values

$$\min_{E \in l(E_i)} \text{ESP}(x, E) = \min_{E \in l(E_i)} \sum_{\alpha} \text{SP}(x, E, \alpha) p(\alpha) \geq \sum_{\alpha} \min_{E_\alpha \in l(E_i)} \text{SP}(x, E_\alpha, \alpha) p(\alpha). \tag{4}$$

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

$$t_\alpha = \min_{E_\alpha \in l(E_i)} \text{SP}(x, E_\alpha, \alpha) \tag{5}$$

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

Table 1. Summary of the Benchmark Datasets

Mother set	Original problem type	# of features	# of benchmarks	# of points per benchmark (range)	# of anomaly 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

lower bound on each term:

$$t_\alpha \geq \hat{t}_\alpha = \min(\{j \ : \ j \leq i, f(x_{E_j}) < \tau_j(E, \alpha)\} \cup \{i\}), \tag{6}$$

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.

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 datasets, particularly benchmarks based on real-world data. Second, given a benchmark dataset, 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. We expand on both of these points as follows.

6.1 Anomaly Detection Benchmarks

Recent work [7] 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 dataset, 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.

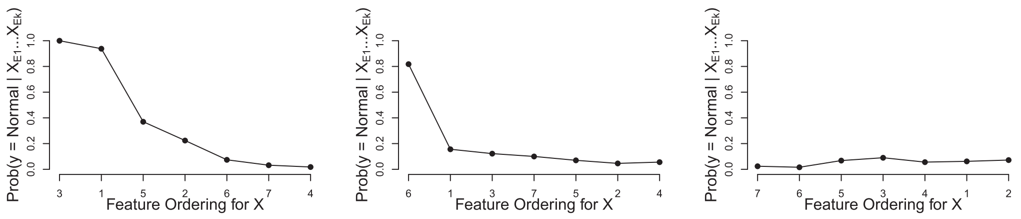

Fig. 2. Analyst Certainty Curves. These are example curves generated using our simulated analyst on anomalies from the Abalone benchmark using SFEs produced by SeqMarg. The x-axis shows the index of the feature revealed at each step and the y-axis shows the analyst certainty about the anomalies being 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.

Table 1 gives a summary of the benchmarks from Emmott et al. [7] used in our experiments. For example, the UCI dataset 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.

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 later. 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 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 dataset 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 dataset [7], 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 [7]. An anomaly detection benchmark is then created by subsampling instances that satisfy some criteria, such as difficulty level, anomaly rate, and so on. 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 [12]) 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 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) [5] 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 [8]. 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.

7 EMPIRICAL EVALUATION

We now present our empirical evaluation on anomaly detection benchmarks from Emmott et al. [7] and the commonly used KDDCup anomaly detection benchmark.

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. [7] 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 dataset. 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

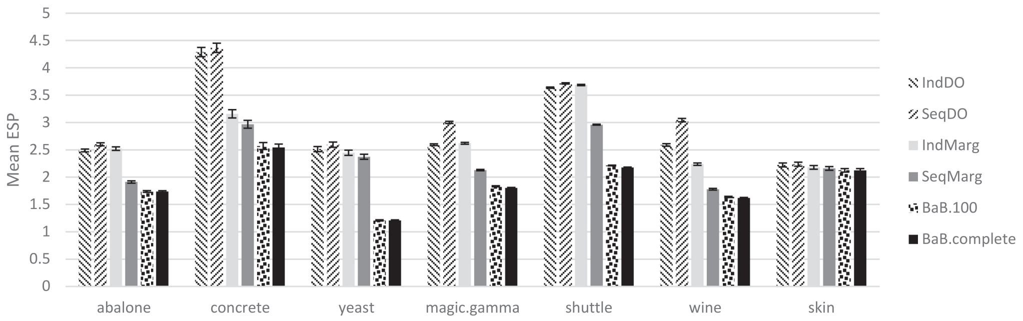

Fig. 3. The bars show the Expected Smallest Prefix (ESP) achieved by different methods averaged across top 10% highly ranked ground truth anomalies in all the benchmarks; 95% confidence intervals are also shown.

all components across the ensemble. Since individual Gaussians have simple closed forms for marginal densities [2], 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 did not consider discrete-valued features directly, primarily because we used anomaly detectors that expected numeric features. 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.

7.2 Empirical Comparison of Greedy Versus BaB

Before presenting our full evaluation, we first compare the performance of greedy versus BaB (Branch and Bound from Section 5) with respect to optimizing ESP (Equation (3)). For this experiment, we need to specify the following two parameters: the number of nodes to expand for BaB and a distribution over percentile values α (1–100) 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 (2) using the SFEs obtained from the greedy methods (SeqMarg, IndMarg SeqDrop, and IndDrop) of Section 5.2. The average of these ESPs are then plotted along with the ESPs of the BaB methods in Figure 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 7.3), we will focus on the computationally cheaper BaB.100 as the representative BaB method. Second, we observe that the greedy 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.

7.3 Evaluation on Benchmark Datasets

We run our evaluation on anomaly detection benchmarks from Emmott et al. [7], derived from seven UCI mother sets. A summary of the benchmarks are given in Table 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 following five methods from Section 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 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 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 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 than our best SFE method. This gap could be due to either of the following: (1) sub-optimal 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 as follows (Section 7.4).

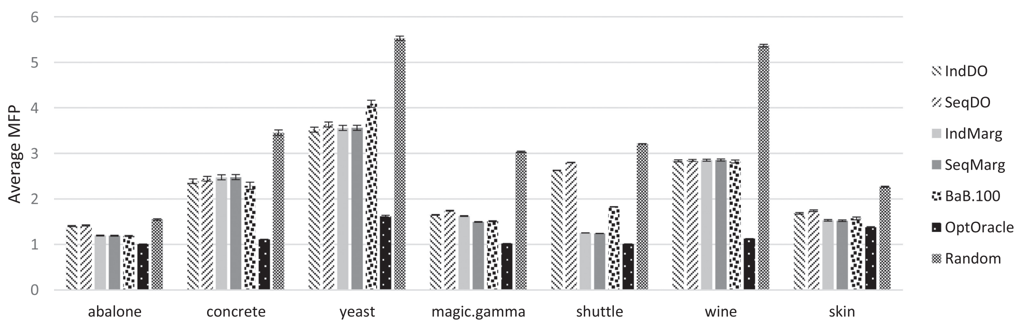

Fig. 4. Performance of explanation methods on benchmarks. Each group of bars shows the performance of the six methods on benchmarks derived from a single mother set. The bars show the expected MFP averaged across anomalies in benchmarks for the corresponding mother set; 95% confidence intervals are also shown.

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 datasets, 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 previously attempts to optimize ESP as a surrogate for the true-but-unknown MFP objective. Figure 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, 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 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.

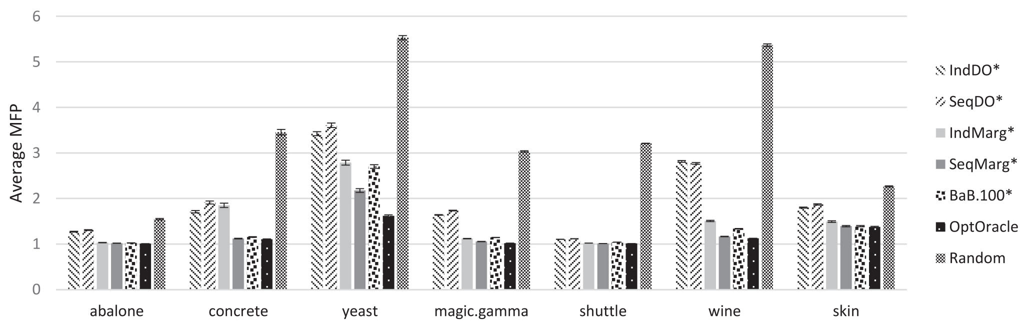

Fig. 5. Performance of explanation methods on benchmarks when using an oracle anomaly detector. Each group of bars shows the performance of the six methods on benchmarks derived from a single mother set. The bars show the expected MFP averaged across anomalies in benchmarks for the corresponding mother set; 95% confidence intervals are also shown.

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.

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 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 datasets, 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 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.

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 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 [9], 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 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 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

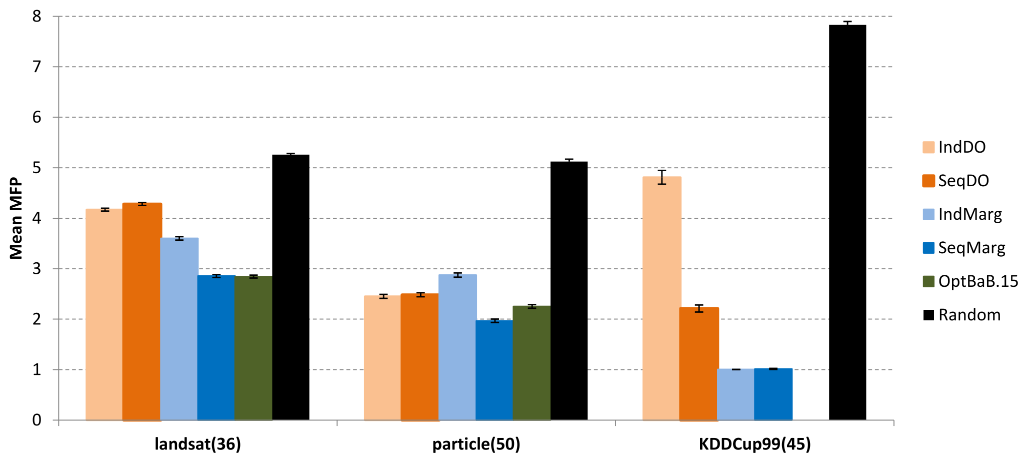

Fig. 6. Performance of different explanation methods on some high-dimensional benchmark datasets; 95% confidence intervals are also shown.

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 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 SeqMarg. This could be due to the reason already discussed in Section 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.

8 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.

9 SUMMARY AND FUTURE WORK

This article introduced the concept of 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 SeqMarg method for computing SFEs is the preferred method among those introduced in this article.

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, first, the anomaly detector and analyst model, and second, 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 SeqMarg 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.

APPENDIX

A NP-HARDNESS OF SFE-DECIDE

We now prove that the decision problem, SFE-Decide, which corresponds to the optimization problem in Equation (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

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

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

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

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

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

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:

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

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: 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 $|E_i| \leq k$. □

ACKNOWLEDGMENTS

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the DARPA, the Army Research Office, or the US government. Also, this material is based upon work while W.-K. Wong was serving at the National Science Foundation. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors, and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. How to explain individual classification decisions. *Journal of Machine Learning Research* 11 (2010), 1803–1831.

[2] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*, Vol. 4. Springer, New York.

[3] Xuan Hong Dang, Ira Assent, Raymond T. Ng, Arthur Zimek, and Erich Schubert. 2014. Discriminative features for identifying and interpreting outliers. In *Proceedings of the IEEE 30th International Conference on Data Engineering (ICDE'14)*. 88–99.

[4] Xuan Hong Dang, Barbora Micenková, Ira Assent, and Raymond T. Ng. 2013. Local outlier detection with interpretation. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 304–320.

[5] Houtao Deng. 2013. Guided random forest in the RRF package. arXiv:1306.0237.

[6] Lei Duan, Guanting Tang, Jian Pei, James Bailey, Akiko Campbell, and Changjie Tang. 2015. Mining outlying aspects on numeric data. *Data Mining and Knowledge Discovery* 29, 5 (2015), 1116–1151.

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

[8] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. 2014. Do we need hundreds of classifiers to solve real world classification problems. *Journal of Machine Learning Research* 15, 1 (2014), 3133–3181.

[9] Seth Hettich and S. D. Bay. 1999. The UCI KDD Archive. Department of Information and Computer Science, University of California, Irvine, CA. Retrieved from http://kdd.ics.uci.edu.

[10] Andreas Krause and Daniel Golovin. 2014. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*, Lucas Bordeaux, Youssef Hamadi, and Pushmeet Kohli (Eds.). Cambridge University Press, 71–104.

[11] Barbora Micenková, Raymond T. Ng, Xuan-Hong Dang, and Ira Assent. 2013. Explaining outliers by subspace separability. In *Proceedings of the IEEE 13th International Conference on Data Mining (ICDM'13)*. 518–527.

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

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

[14] Erik Štrumbelj and Igor Kononenko. 2014. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems* 41, 3 (2014), 647–665.

[15] Nguyen Xuan Vinh, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Jian Pei. 2015. Scalable outlying-inlying aspects discovery via feature ranking. In *Advances in Knowledge Discovery and Data Mining*. Springer, 422–434.

Received December 2016; revised April 2018; accepted April 2018