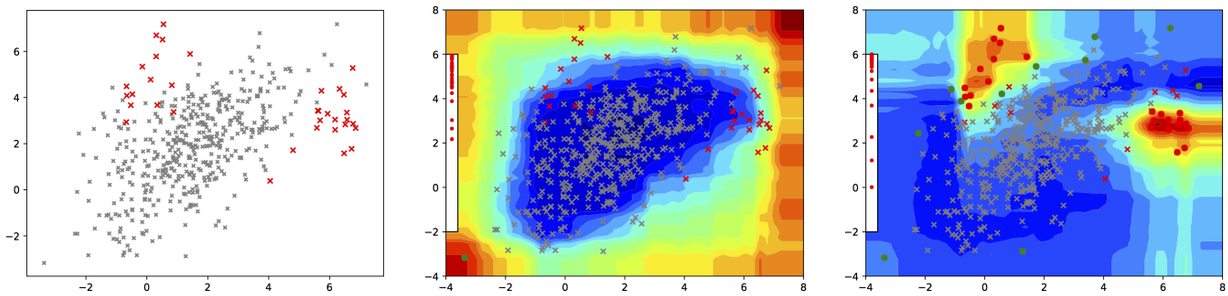


(a) Dataset (b) Initial score contours (c) After 35 feedback

Figure 1: Dataset and score contours. Figure 1a shows a synthetic toy dataset. Figure 1b shows the initial Isolation Forest score contours. Figure 1c shows the score contours after 35 feedback iterations from the Oracle. The red dots are discovered anomalies (true positives). The green dots are discovered nominals (false positives). The red checks are undiscovered anomalies (false negatives).

1 Compact Description For Anomalies

Providing explanations for anomalies is important and some recent work (among others) can be found in [1]. We illustrate a different and yet very simple approach when AAD is used with *Isolation Forest*.

The Isolation Forest version of AAD works by partitioning the space into smaller subspaces and weights each subspace to fit the expert feedback. The end result is: (1) a set of default (unsupervised) anomaly scores for each subspace and, (2) weights for each subspace inferred from feedback. We can generate compact descriptions for the anomalies using this information. The idea is to select some combination of (preferably) the smallest subspaces that **cover** all **discovered** anomalies. Therefore, we treat this as an instance of the *set covering* problem. We illustrate this on a toy dataset below.

Assume that there are total m subspaces across all Isolation Forest trees. We represent weights by $\mathbf{w} \in \mathbb{R}^m$, and the corresponding unsupervised anomaly scores by $\mathbf{d} \in \mathbb{R}^m$.

Now, the final anomaly scores for the **subspaces** (not instances), after incorporating feedback, is $\mathbf{a} = \mathbf{w} \circ \mathbf{d} \in \mathbb{R}^m$ where $\circ$ denotes the element wise (*Hadamard*) product.

We sort the scores in $\mathbf{a}$ in descending order and select (say) 30 top ranked subspaces. Next, we retain only those subspaces from this set which contain at least one anomaly **discovered** by the analyst. Let us denote the resulting set of subspaces by $\mathcal{S}$. Further, let $\mathcal{Z}$ be the set of discovered anomalies that belong to one or more subspaces in $\mathcal{S}$, and $|\mathcal{Z}| = n$. (Note that some of the discovered anomalies might not belong to any of the top ranked subspaces we selected.)

Let $|\mathcal{S}| = k$. Denote the *volumes* of the subspaces in $\mathcal{S}$ by the vector $\mathbf{v} \in \mathbb{R}^k$. Now, assume that a binary vector $\mathbf{x} \in \{0,1\}^k$ contains 1 in locations corresponding to the subspaces in $\mathcal{S}$ which are included in the covering set, and 0 otherwise. Let $\mathbf{u}_z \in \{0,1\}^k$ denote a vector for each anomaly $z \in \mathcal{Z}$ which contains 1 in all locations corresponding to the subspaces in $\mathcal{S}$ that z belongs to. Let $\mathbf{U} \in \{0,1\}^{[n \times k]}$ represent the matrix of all $\mathbf{u}$s.

The selection of the compact set of subspaces to describe all discovered anomalies can be formulated as:

$$\begin{aligned} &\arg\min_{\mathbf{x} \in \{0,1\}^k} \mathbf{x} \cdot \mathbf{v}^p \\ &\text{s.t.} \mathbf{U} \cdot \mathbf{x} \geq \mathbf{1} \\ &\text{where, } \mathbf{1} \text{ is the column vector of } n \text{ 1s, and} \\ &p \text{ is an integer} \geq 1 \text{ (more below)} \end{aligned} \tag{1}$$

The parameter p determines how severely to penalize larger volumes. This is usually 1. However, a larger value will strongly discourage bigger volume subspaces from being selected.

We apply this idea on a toy data shown in Figure 1a. The compact descriptions are shown in Figure 2c.

Note that we have not considered labeled nominals in the optimization objective, but a more sophisticated objective might include them too. For instance, we could add another term in the objective which penalizes whenever a nominal is present in the selected subspace.

A different approach to generate descriptions might train a cost-based decision tree with the labeled anomalies and nominals. Once trained, the extracted rules would correspond to 'descriptions'. This approach is possibly more principled because the decision tree splits are [usually] based on information gain or gini-index and are therefore more informative. The downside is that this would be separate from the anomaly detector and therefore it would not offer much insight into the anomaly detector's behavior – ideally we would like the explanation or description of anomalies to reflect the detector's internals.

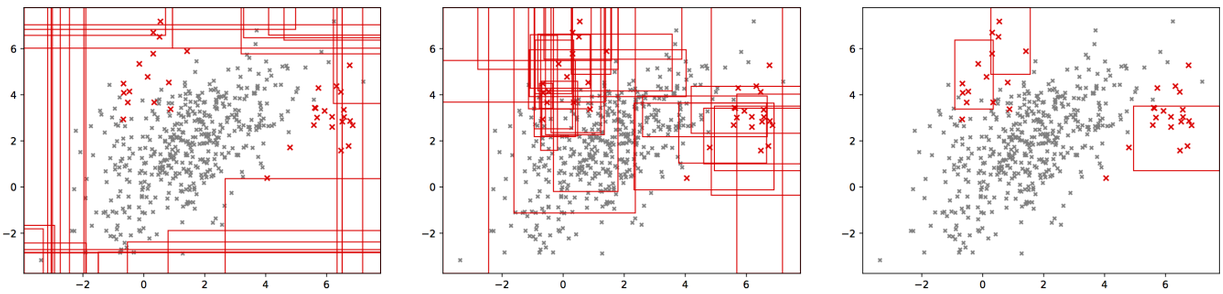
2 Querying Diversity

We can use the anomaly description to diversify our queries when we have the option to query labels for more than one instance per round of feedback. Following is how we go about it:

1. Select the top ranked m (say, $m = 15$) instances. Denote this set by $\mathcal{C}$ (points in blue in Figure 3a).
2. Next, select the top (say) 5 most anomalous regions for each instance in $\mathcal{C}$. Let $\mathcal{F}$ be the union of all these regions (rectangles in red in Figure 3a).
3. Now, *compactly* describe all instances in $\mathcal{C}$ using the regions in $\mathcal{F}$ (rectangles in red in Figure 3b).
4. Now, start with the most anomalous instance, and select one-by-one k (say, $k = 5$) instances which have the fewest overlapping regions for querying (points circled in green in Figures 3b and 3c).

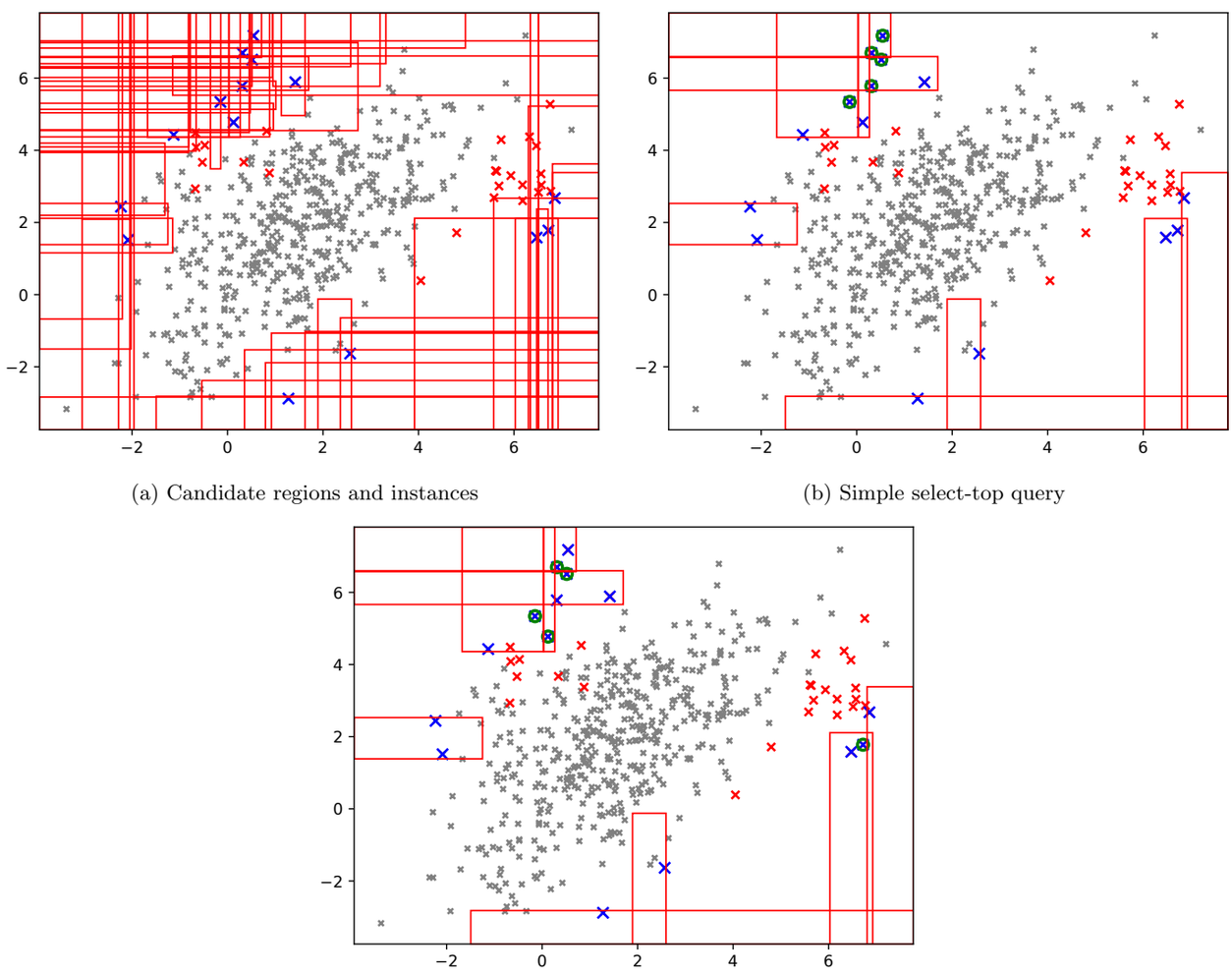
2.1 Does the diverse querying strategy help?

Whatever querying strategy we use, it should ideally not lower the number of anomalies discovered within a budget. For the diverse query strategy, we need to



(a) Baseline (b) AAD (c) Compact AAD

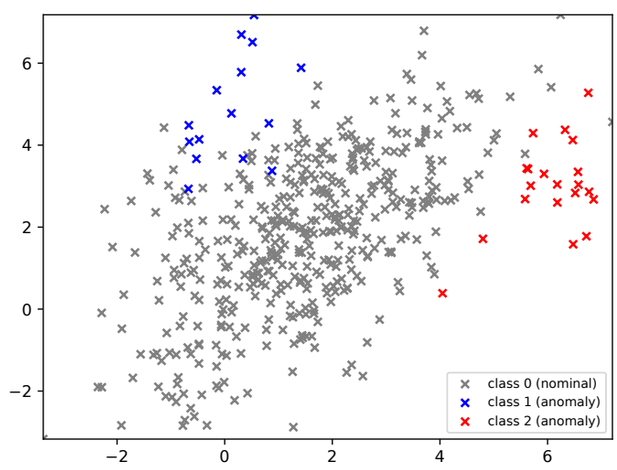
Figure 2: Top 30 subspaces ranked by $\mathbf{w} \circ \mathbf{d}$. Figure 2a shows the top 30 most important subspaces (w.r.t their *anomalousness*) without any feedback. We can see that initially, these simply correspond to the exterior regions of the dataset. AAD **learns the true importance of subspaces** automatically with feedback. For example, after incorporating the labels of 35 instances, the subspaces around the labeled anomalies have emerged as the most important (Figure 2b). Figure 2c shows the most compact set of subspaces (for AAD) which cover all labeled anomalies. These are computed with Equation 1. We might even think of this as a non-parametric clustering. Note that the compact subspaces only cover anomalies that were discovered in the 35 feedback iterations. Anomalies which were not detected are likely to fall outside these compact subspaces.



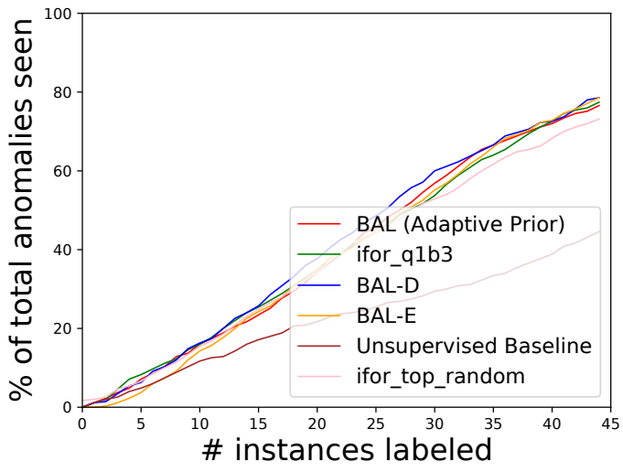
(a) Candidate regions and instances (b) Simple select-top query

(c) Select diverse query

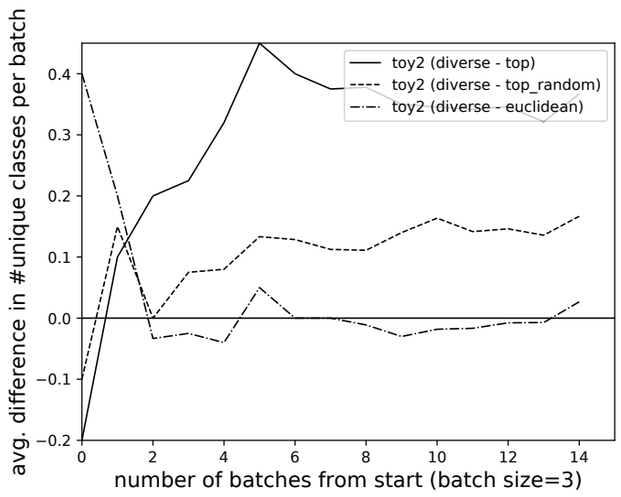
Figure 3: Diverse querying strategy using sub-space descriptions. These figures show the 15 most anomalous instances (ranked by score) in blue. We assume that the true labels are not known for these instances. We find the top 5 most anomalous regions for each of the 15 instances and the union of all these regions is shown as the red rectangles in Figure 3a. Figure 3b and Figure 3c show the regions which compactly contain (describe) the top-ranked instances. The instances circled in green are the ones selected for querying. The instances (in green) in Figure 3b are merely the top-ranked 5 instances without taking into account any diversity. Figure 3c shows that when we select instances which have different descriptions (minimum region overlap), they are more diverse.



(a) Underlying Processes

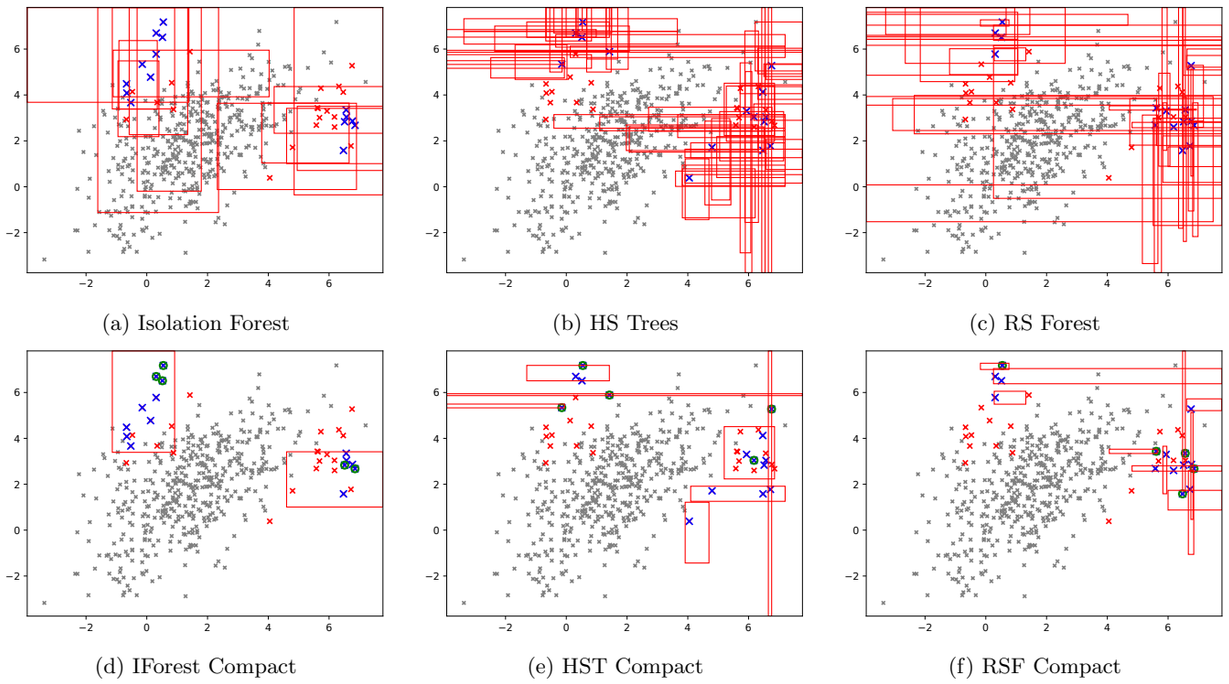


(b) Anomalies discovered



(c) Difference in class diversity per batch

Figure 4: Diversity in the classes shown to an analyst per batch. Here we assume that the data contains **three** classes which represent three underlying processes (Figure 4a). Class 0 is *nominal*, while classes 1 and 2 are *anomalous*. In Figure 4b, we plot the anomaly discovery curves for five Isolation Forest-based algorithms: (a) **BAL (Adaptive Prior)** – label only the single most anomalous instance (per iteration), (b) **ifor_q1b3** – label the top three most anomalous instances, (c) **BAL-D** – label the top three **most diverse** instances among the top ten anomalous instances (employing the diversity strategy), (d) **BAL-E** – label the top three instances **farthest in euclidean space** among the top ten anomalous instances, (e) **Unsupervised Baseline** – unsupervised Isolation Forest, and (f) **ifor_top_random** – label three instances selected uniformly at random from the top ten anomalous instances. We see that all algorithms other than the baseline have similar performance, and all active labeling algorithms perform better than the baseline. In Figure 4c, the solid line `(diverse - top)` shows the average difference in the number of unique classes shown to the analyst per batch between **BAL-D** and **ifor_q1b3** (these differences are averaged over 10 runs). The dashed line `(diverse - top_random)` in Figure 4c shows the average difference in the number of unique classes shown to the analyst per batch between **BAL-D** and **ifor_top_random**. Since both these differences are mostly positive, we conclude that the diversity strategy indeed presents more diverse instances to the analyst to label. Moreover, as seen in Figure 4b, this diversity does not lower the anomaly detection accuracy over a strictly select-top-anomalous query strategy. **We also see this pattern in most real-world datasets we experimented with.** The dash-dot line `(diverse - euclidean)` shows the difference between **BAL-D** and **BAL-E**. **BAL-E** is a standard way to find diverse instances. The performance of **BAL-D** and **BAL-E** are very similar; however, **BAL-D** is more user-friendly because it provides descriptions which can characterize multiple anomalies.



(a) Isolation Forest (b) HS Trees (c) RS Forest

(d) IForest Compact (e) HST Compact (f) RSF Compact

Figure 5: Top subspaces after 35 feedback iterations ranked by $\mathbf{w} \circ \mathbf{d}$ which cover the top ranked 15 instances. We find the top 5 most anomalous regions for each of the 15 instances and the union of all these regions is shown as the red rectangles (**top row**). The corresponding compact subspaces which describe the 15 instances are shown in the **bottom row**. The points circled in green (in the bottom row) are the 5 instances (out of the 15) selected for query using a diverse querying strategy. The Isolation Forest subspaces tend to be larger and each subspace usually contains more instances. This lets the feedback in Isolation Forest to be shared across more instances than in HS Trees or RS Forest.

3 Comparison between Isolation Forest, HS Trees, RS Forest

There are fundamental differences between Isolation Forest and the other two: HS Trees and RS Forest. These differences directly influence the effectiveness of incorporating feedback. Since the depths in HS Trees and RS Forest are fixed and they repeatedly split a dimension, there are $O(2^H)$ subspaces represented by the leaf nodes of these detectors and most are very small in volume. We see this property in Figures 5b and 5c. The implication is that more number of subspaces will be required to generate a compact representation for a set of instances. This also means that feedback at instance level is going to be shared by a much fewer set of instances. In contrast, in Isolation forest (Figure 5a), the depths of leaf nodes are not fixed; they are usually **much** shallower than in HS Trees and RS Forest. As a result, the subspaces represented by the leaf nodes are larger and cover more number of instances, thereby requiring fewer subspaces to compactly represent a set of instances. Moreover, this results in feedback being shared by more instances in Isolation Forest. One remedy for HS Trees and RS Forest might be to dynamically determine when to stop splitting based on (maybe) the number of samples at a node. This might be a motivation for future research. My personal opinion right now is to stick to Isolation Forest because of its 'nicer' properties.

References

[1] Meghanath Macha and Leman Akoglu *X-PACS: eXPlaining Anomalies by Characterizing Subspaces*, 2017, http://arxiv.org/abs/1708.05929.