Human in-the-Loop Learning for Anomaly Detection

Jana Doppa
Huie-Rogers Endowed Chair Associate Professor
School of EECS, Washington State University

Joint work w/ Rakibul Islam and Shubhomoy Das
Motivating App #1: Credit Card Fraud

- Unusual transactions
- False alarms are common
- High false positive rate $\Rightarrow$ wasted human effort
Motivating App #2: ICU Patient Monitoring

- Monitor patients' condition
- False alarms are common
- High cost of wrong decisions
Motivating App #3: Android Malware

Malwares can potentially
- Spy
- Steal person information
- Make fake calls
- ....

Anti-Malware Industry
Anomaly Detection: The Problem

Input → Computational Algorithms & Tools → Output
Anomaly Detection: Challenges

- # of anomalies $<<$ # of nominals
- Knowledge discovery task involving humans
- High false-positive rate $\rightarrow$ wasted human effort
Prior work on Anomaly Detection: Unsupervised Methods

- **Density-Based Approaches**
  - RKDE: Robust Kernel Density Estimation (Kim & Scott, 2008)
  - EGMM: Ensemble Gaussian Mixture Model

- **Quantile-Based Methods**
  - OCSVM: One-class SVM (Schoelkopf, et al., 1999)
  - SVDD: Support Vector Data Description (Tax & Duin, 2004)

- **Neighbor-Based Methods**
  - LOF: Local Outlier Factor (Breunig, et al., 2000)
  - ABOD: kNN Angle-Based Outlier Detector (Kriegel, et al., 2008)

- **Projection-Based Methods**
  - IFOR: Isolation Forest (Liu, et al., 2008)
  - LODA: Lightweight Online Detector of Anomalies (Pevny, 2016)
Unsupervised methods: when they may not work?

▸ Underlying model assumptions are violated

▸ Nature/type of anomalies is not known beforehand (unknown unknowns)

▸ Only a subset of identified anomalies are relevant for real-world task

High false positive rate and wasted effort from human analysts
Key Research Question

- How can we use human analyst efficiently to improve anomaly detection rate of unsupervised approaches?

Discover all the anomalies with very small number of queries to human analyst (minimize human effort)
Aside: Isolation Tree

- Key Idea
  - Anomalies can be easily isolated from nominals
  - Degree of anomaly is inversely proportional to depth

Random feature and random split point

Deeper leaf

Shallow leaf

Nodes Score is a function of path length

Nominals

Partitioned subspaces

Depth Colored subspaces

Anomalies

Tree Depth Color Code
Aside: Isolation Forest

- State of the art unsupervised approach
- Assumptions
  - # of anomalies are small
  - Features are distinguishable
- Assigns uniform weights for subspace score
Isolation Forest

Random feature and random split point

Shallow leaf

Deeper leaf

Isolation Tree

Isolation Forest

Space partition for 100 Trees

Contour Plot for 100 Trees
Model Definition

- Isolation forest generates high number of subspaces/leaves (say $m$)

- We can map each data instance to this representation
  - $x \mapsto IFor(x) \in R^m$
  - If the data point falls in a leaf, its feature value is equal to the adjusted depth based score. Otherwise, its feature value is zero

- All subspaces are equally weighted ($m$ weights)
  - $w_1 = w_2 = ... = w_m$
Making Predictions

- Score all the data points
  - \( \text{score}(x) = w \cdot \text{IFor}(x) \)
- Rank them based on the scores
- Pick top \( \tau \) candidates as anomalies

Anomaly detection accuracy depends on the weights of the model. How can we learn the optimal weights?
HiLAD: Overview

Use **human feedback** to improve the anomaly detection performance
Anomaly Detector

Anomalies are pushed higher in score space

Greedily select the highest scoring unlabeled example for feedback
Query strategy to discover diverse anomalies

- What if we want diverse anomalies?
  - The inherent structure of Isolation Forest can be exploited
  - Subspaces are selected to produce diverse queries

- Optimally pick queries from the most relevant compact subspaces
  - related to set covering problem
Learning Goal

- Learn weights so that anomalies are ranked higher than the nominals
- Score(anomaly) > Score(nominal)

Optimization to Update weights

- After feedback, nominals are ranked lower than top most $\tau$ fraction
- A small change in the weights to satisfy the constraints
Framework Illustration

$q_t^t$  $q_t^{t+1}$  $q_t^{t+2}$

Nominal  Anomaly
Experimental Setup

▸ Baselines

<table>
<thead>
<tr>
<th>Approach</th>
<th>Feedback</th>
<th>Prior</th>
<th>Influence Of prior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsupervised</td>
<td>NO</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>HiLAD (no prior-rand)</td>
<td>YES</td>
<td>Random</td>
<td>NO</td>
</tr>
<tr>
<td>HiLAD (no prior-uniform)</td>
<td>YES</td>
<td>Uniform</td>
<td>NO</td>
</tr>
<tr>
<td>HiLAD (Adaptive Prior)</td>
<td>YES</td>
<td>Uniform</td>
<td>YES</td>
</tr>
</tbody>
</table>

▸ Metrics

▸ What percentage of anomalies are discovered as a function of the number of queries?
▸ AUC (Area Under ROC Curve)
Results for Single Queries

- Small amount of feedback significantly improves the accuracy
  - Updating weights help in finding new anomalies
Higher diversity
- # of unique classes in query is higher than regular
- Little or no loss in anomaly detection accuracy
Streaming Anomaly Detection

- Goal: Identify “anomalies” from the continuous stream of data
Streaming Anomaly Detection: Challenges

- Detecting concept drift?
- When and how to update the model?
- Memory constraints to store data?
Streaming Anomaly Detection: Illustration
Streaming AD via Drift Detection

- Identify the data distribution of current window from ensembles
- Compare it with the data distribution of new window (via KL divergence)

Not much drift was observed

Drift detected
Updating Ensemble of Trees for Streaming Data

- Limited memory and **minor concept drift**
  - Same as batch data setting and only requires retraining

- Limited memory with **major concept drift**
  - Replace a fraction of older trees
  - Include new trees from current window
  - Initialize the weights for the new trees with default (uniform) weights
## Experimental Setup

- **When there is no concept drift**
- **Baselines**

<table>
<thead>
<tr>
<th>Approach</th>
<th>Feedback</th>
<th>Prior</th>
<th>Query Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsupervised</td>
<td>NO</td>
<td>N/A</td>
<td>Select Top</td>
</tr>
<tr>
<td>HiLAD-Batch</td>
<td>YES</td>
<td>Uniform</td>
<td>Select Top</td>
</tr>
<tr>
<td>HiLAD-Stream</td>
<td>YES</td>
<td>Uniform</td>
<td>Select Top</td>
</tr>
<tr>
<td>HiLAD-Stream-D</td>
<td>YES</td>
<td>Uniform</td>
<td>Select Diversified</td>
</tr>
</tbody>
</table>
Results for “No Drift”

- Limited memory settings

- With feedback, HiLAD-Batch is consistently the best performer
- Performance of HiLAD-Stream is close to HiLAD-Batch (upper-bound)
Results for

- Updating ensemble of trees in a principled manner
  - Fixed threshold (10%, 20%, …)
HiLAD-Stream(KL adaptive) model update is stable without knowing the drift amount prior.
Summary

- Unsupervised ensembles and Isolation Forest in particular is a state-of-the-art approach for anomaly detection
  - High false-positive rate => wasted effort from human analyst

- Human feedback can be used to tune ensembles to reduce false-positives
  - Exploit the structure of Isolation forest to discover diverse anomalies
  - Handle streaming data via drift detection and updating ensembles