ANOMALY DETECTION WITH MACHINE LEARNING: AN INTRODUCTION

Anomaly detection plays an instrumental role in robust distributed software systems. Anomaly detection can:

- Enhance communication around system behavior
- Improve root cause analysis
- Reduce threats to the software ecosystem

Traditional anomaly detection is manual. However, machine learning techniques are improving the success of anomaly detectors. Of course, with anything machine learning, there are upstart costs—data requirements and engineering talent.

In this article, we'll take a look at:

- Anomaly detection
- Machine learning AD
- Benchmarks
- How to begin

What is anomaly detection?

Anomaly detection is any process that finds the outliers of a dataset; those items that don't belong.

These anomalies might point to unusual network traffic, uncover a sensor on the fritz, or simply identify data for cleaning, before analysis.

In today's world of distributed systems, managing and monitoring the system's performance is a chore—albeit a necessary chore. With hundreds or thousands of items to watch, anomaly detection can help point out where an error is occurring, enhancing root cause analysis and quickly getting tech support on the issue. Anomaly detection helps the monitoring cause of chaos engineering by detecting outliers, and informing the responsible parties to act.

In enterprise IT, anomaly detection is commonly used for:

- Data cleaning
- Intrusion detection
- Fraud detection
- Systems health monitoring
- Event detection in sensor networks
- Ecosystem disturbances

The challenge of anomaly detection

But even in these common use cases, above, there are some drawbacks to anomaly detection. From a conference paper by Bram Steenwinckel:

> *"Anomaly detection (AD) systems are either manually built by experts setting thresholds on data or constructed automatically by learning from the available data through machine learning (ML)."*

It is tedious to build an anomaly detection system by hand. This requires domain knowledge and—even more difficult to access—foresight.

For an ecosystem where the data changes over time, like fraud, this cannot be a good solution. Building a wall to keep out people works until they find a way to go over, under, or around it. When the system fails, builders need to go back in, and manually add further security methods.

Under the lens of chaos engineering, manually building anomaly detection is bad because it creates a system that cannot adapt (or is costly and untimely to adapt).

Anomaly detection with ML

Machine learning, then, suits the engineer's purpose to create an AD system that:

- Works better
- Is adaptive and on time
- Handles large datasets

Despite these benefits, anomaly detection with machine learning can only work under certain conditions.

Unstructured data: what's the anomaly?

Applying machine learning to anomaly detection requires a good understanding of the problem, especially in situations with unstructured data.

Structured data already implies an understanding of the problem space. Anomalous data may be easy to identify because it breaks certain rules. If a sensor should never read 300 degrees Fahrenheit and the data shows the sensor reading 300 degrees Fahrenheit—there's your anomaly. There is a clear threshold that has been broken.

Fraud detection in the early anomaly algorithms could work because the data carried with it meaning. The data came structured, meaning people had already created an interpretable setting for collecting data. Their data carried significance, so it was possible to create random trees and look for fraud.

However, dark data and unstructured data, such as images encoded as a sequence of pixels or language encoded as a sequence of characters, carry with it little interpretation and render the old algorithms useless...until the data becomes structured. Structure can be found in the last layers of a convolutional neural network (CNN) or in any number of sorting algorithms.

Large datasets needed

Second, a large data set is necessary. A founding principle of any good machine learning model is that it requires datasets. Like law, if there is no data to support the claim, then the claim cannot hold in court.

Machine learning requires datasets; inferences can be made only when predictions can be validated. Anomaly detection benefits from even larger amounts of data because the assumption is that anomalies are rare.

Scarcity can only occur in the presence of abundance.

Talent required

Third, machine learning engineers are necessary. Obvious, but sometimes overlooked. Machine learning talent is not a commodity, and like car repair shops, not all engineers are equal.

It requires skill and craft to build a good Machine Learning model. The cost to get an anomaly detector from 95% detection to 98% detection could be a few years and a few ML hires.

Anomaly detection in three settings

In a 2018 lecture, Dr. Thomas Dietterich and his team at Oregon State University explain how anomaly detection will occur under three different settings. They all depend on the condition of the data. The three settings are:

1. Supervised
2. Clean
3. Unsupervised

Let's look at each setting in depth.

1. Supervised

Training data is labeled with "nominal" or "anomaly".

The supervised setting is the ideal setting. It is the instance when a dataset comes neatly prepared for the data scientist with all data points labeled as anomaly or nominal. In this case, all anomalous points are known ahead of time. That means there are sets of data points that are anomalous, but are not identified as such for the model to train on.

Popular ML algorithms for structured data:

- Support vector machine learning
- k-nearest neighbors (KNN)
- Bayesian networks
- Decision trees

2. Clean

In the Clean setting, all data are assumed to be "nominal", and it is contaminated with "anomaly" points.

The clean setting is a less-ideal case where a bunch of data is presented to the modeler, and it is clean and complete, but all data are presumed to be nominal data points. Then, it is up to the modeler to detect the anomalies inside of this dataset.

3. Unsupervised

In Unsupervised settings, the training data is unlabeled and consists of "nominal" and "anomaly" points.

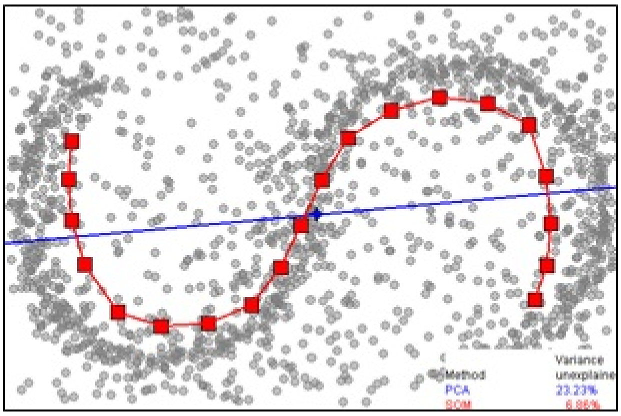
The hardest case, and the ever-increasing case for modelers in the ever-increasing amounts of dark data, is the unsupervised instance. The datasets in the unsupervised case do not have their parts labeled as nominal or anomalous. There is no ground truth from which to expect the outcome to be. The model must show the modeler what is anomalous and what is nominal.

> *"The most common tasks within unsupervised learning are clustering, representation learning, and density estimation. In all of these cases, we wish to learn the inherent structure of our data without using explicitly-provided labels."- Devin Soni*

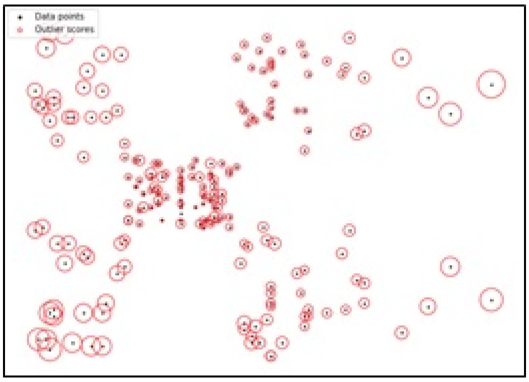
In the Unsupervised setting, a different set of tools are needed to create order in the unstructured data. In unstructured data, the primary goal is to create clusters out of the data, then find the few groups that don't belong. Really, all anomaly detection algorithms are some form of approximate density estimation.

Popular ML Algorithms for unstructured data are:

- Self-organizing maps (SOM)
- K-means
- C-means
- Expectation-maximization meta-algorithm (EM)
- Adaptive resonance theory (ART)
- One-class support vector machine

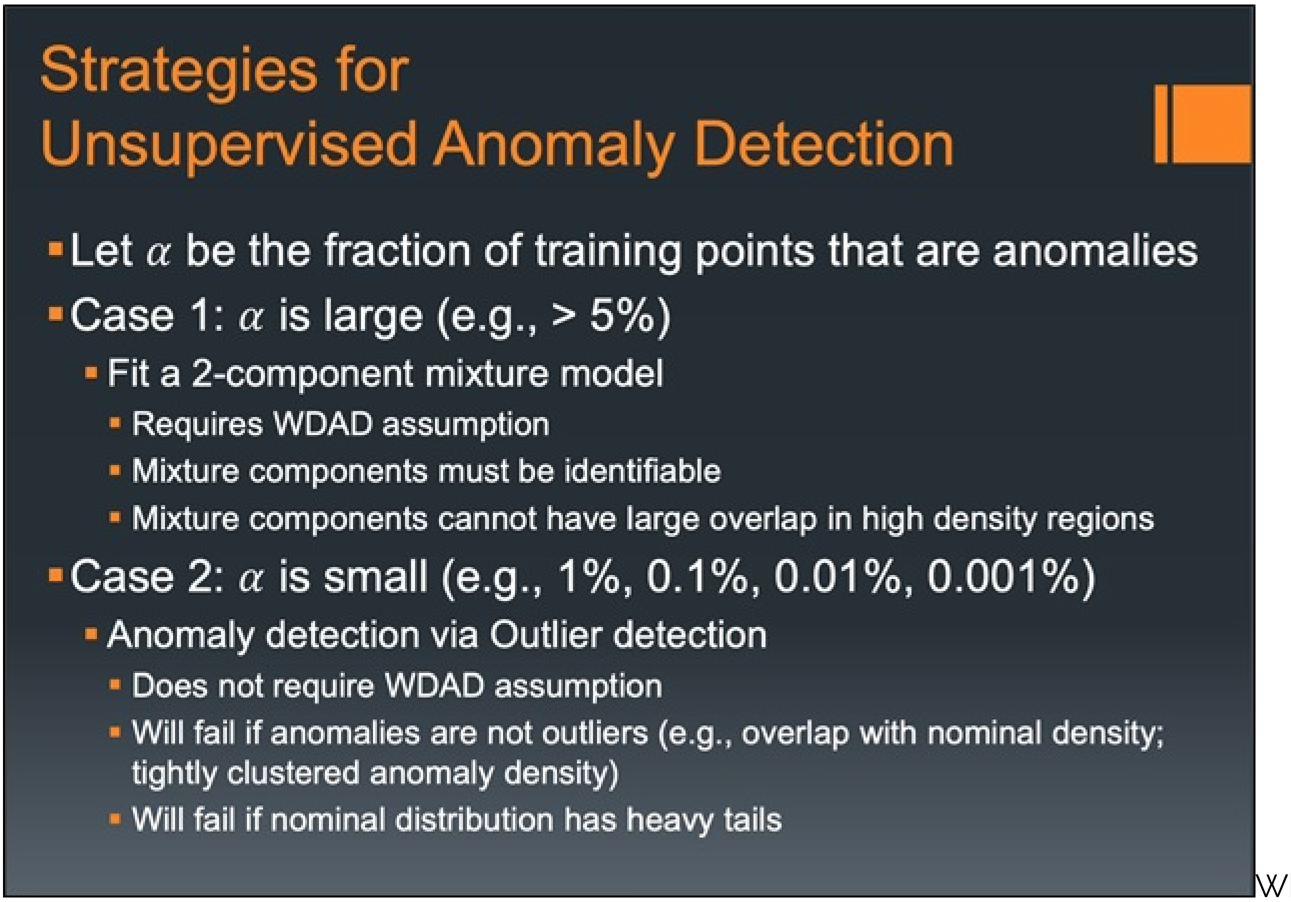


SOM Detection (Source)



K-Means Clustering (Source)

From Dr. Dietterich's lecture slides, the strategies for anomaly detection in the case of the unsupervised setting are broken down into two cases:



Strategies for Unsupervised Anomaly Detection

- Let α be the fraction of training points that are anomalies
- Case 1: α is large (e.g., > 5%)
  - Fit a 2-component mixture model
    - Requires WDAD assumption
    - Mixture components must be identifiable
    - Mixture components cannot have large overlap in high density regions
- Case 2: α is small (e.g., 1%, 0.1%, 0.01%, 0.001%)
  - Anomaly detection via Outlier detection
    - Does not require WDAD assumption
    - Will fail if anomalies are not outliers (e.g., overlap with nominal density; tightly clustered anomaly density)
    - Will fail if nominal distribution has heavy tails

Where machine learning isn't appropriate, top non-ML detection algorithms include:

- IFOR: Isolation Forest (Liu, et al., 2008)
- LODA: Lightweight Online Detector of Anomalies (Pevny, 2016)

Benchmarking anomaly detection

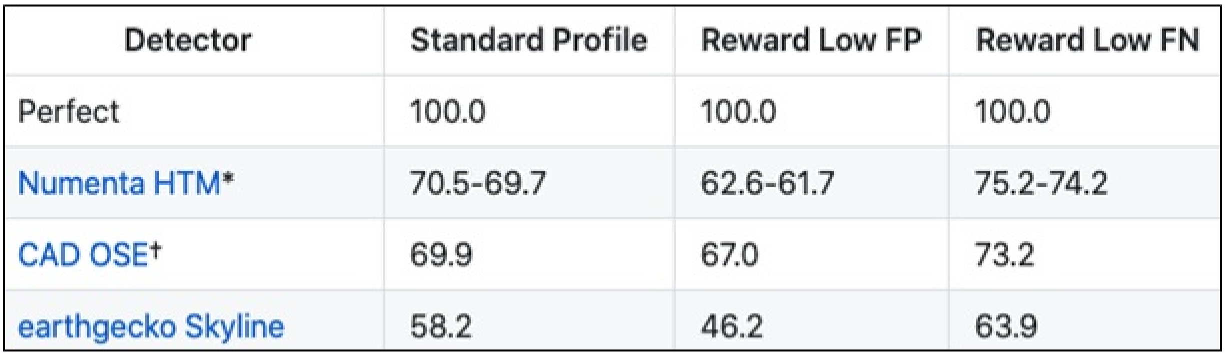
Engineers use benchmarks to be able to compare the performance of one algorithm to another's. Different kinds of models use different benchmarking datasets:

- Image classification has MNIST and IMAGENET.
- Language modelling has Penn TreeBank and Wiki Text-2.

In anomaly detection, no one dataset has yet become a standard. This has to do, in part, with how varied the applications can be. However, one body of work is emerging as a continuous presence—the Numenta Anomaly Benchmark. From the GitHub Repo:

> *"NAB is a novel benchmark for evaluating algorithms for anomaly detection in streaming, real-time applications. It is composed of over 50 labeled real-world and artificial time series data files plus a novel scoring mechanism designed for real-time applications."*

Thus far, on the NAB benchmarks, the best performing anomaly detector algorithm catches 70% of anomalies from a real-time dataset.



Detector	Standard Profile	Reward Low FP	Reward Low FN
Perfect	100.0	100.0	100.0
Numenta HTM*	70.5-69.7	62.6-61.7	75.2-74.2
CAD OSE†	69.9	67.0	73.2
earthgecko Skyline	58.2	46.2	63.9

Get started with ML anomaly detection

If you want to get started with machine learning anomaly detection, I suggest started here:

- Building a real-time anomaly detection system for time series at Pinterest (software engineers Kevin Chen and Brian Overstreet)
- Anomaly Detection Techniques in Python (Christopher Jose)
- Outlier and Anomaly Detection with scikit-learn Machine Learning (Walker Rowe)

Additional resources

For more on this and related topics, explore these resources:

- BMC Machine Learning & Big Data Blog
- Top Machine Learning Frameworks To Use in 2020
- Guide to Machine Learning with TensorFlow & Keras
- scikit-learn Guide
- Python vs Java: Why Python is Becoming More Popular than Java