Carnegie Mellon University

SUOD: Toward Scalable Unsupervised Outlier Detection*

Yue Zhao, Xueying Ding, Jianing Yang, Haoping Bai
Carnegie Mellon University

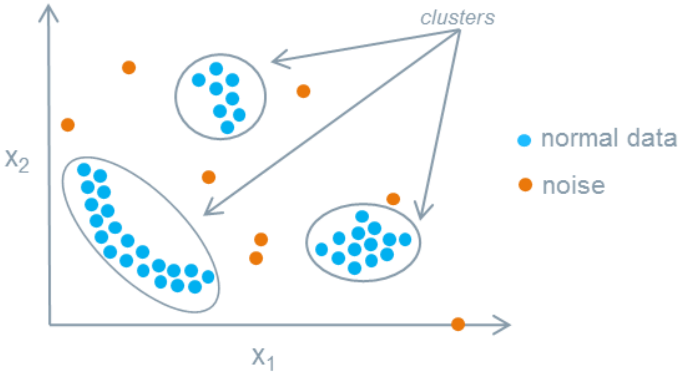
   

* An extended version is under review at *AAAI 2020 Workshop*. Will revise and resubmit for *KDD 2020 (ADS track)*.

Outlier Detection

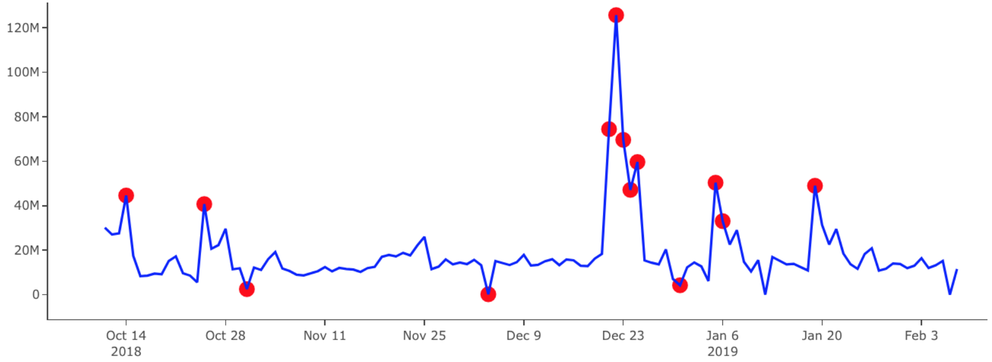
Outlier detection, also known as anomaly detection, aims to identify the data samples that are **deviant** from **the general distribution.**

Applications: fraud detection, intrusion defense, fake news identification



Anomalies in Tabular Data

Source: https://developer.mindsphere.io/apis/analytics-anomalydetection/api-anomalydetection-overview.html



Anomalies in Time-series

Source: https://towardsdatascience.com/anomaly-detection-with-isolation-forest-visualization-23cd75c281e2

Challenges in Outlier Detection

- **Limited number of labeled data** – unsupervised models are used in practice
- **Extreme data skew** – number of outliers << number of inliers
- **Complex patterns** – outliers may be well hidden in certain subspaces or only identifiable under specific assumptions

unsupervised approaches + extremely skewed data + **complex patterns =**

A challenging learning task!

Remedy: Ensemble Learning

Ensemble learning is a technique to **combine/fuse/aggregate** multiple base models [1]. It shows two key advantages:

- **Stability Enhancement**: independent trials in empirical studies
- **Performance Improvement**: boosted trees, random forests, even neural nets may be considered as a form of ensembling

[1] Zhao, Y., Wang, X., Cheng, C. and Ding, X., 2020. Combining Machine Learning Models and Scores using combo library. *Thirty-Fourth AAAI Conference on Artificial Intelligence.*

Challenges while Using Many Outlier Detectors

For effective outlier ensembles, **a large group of diverse base models** are needed.

However, training and prediction with many heterogeneous unsupervised outlier detectors shows the following limitations:

- **Computationally expensive:** density estimation and distance calculation
- **Inefficient in parallelization:** heterogeneous models, varying cost, hard to schedule
- **Limited in interpretability:** unsupervised (non-parametric) models

Research Objective

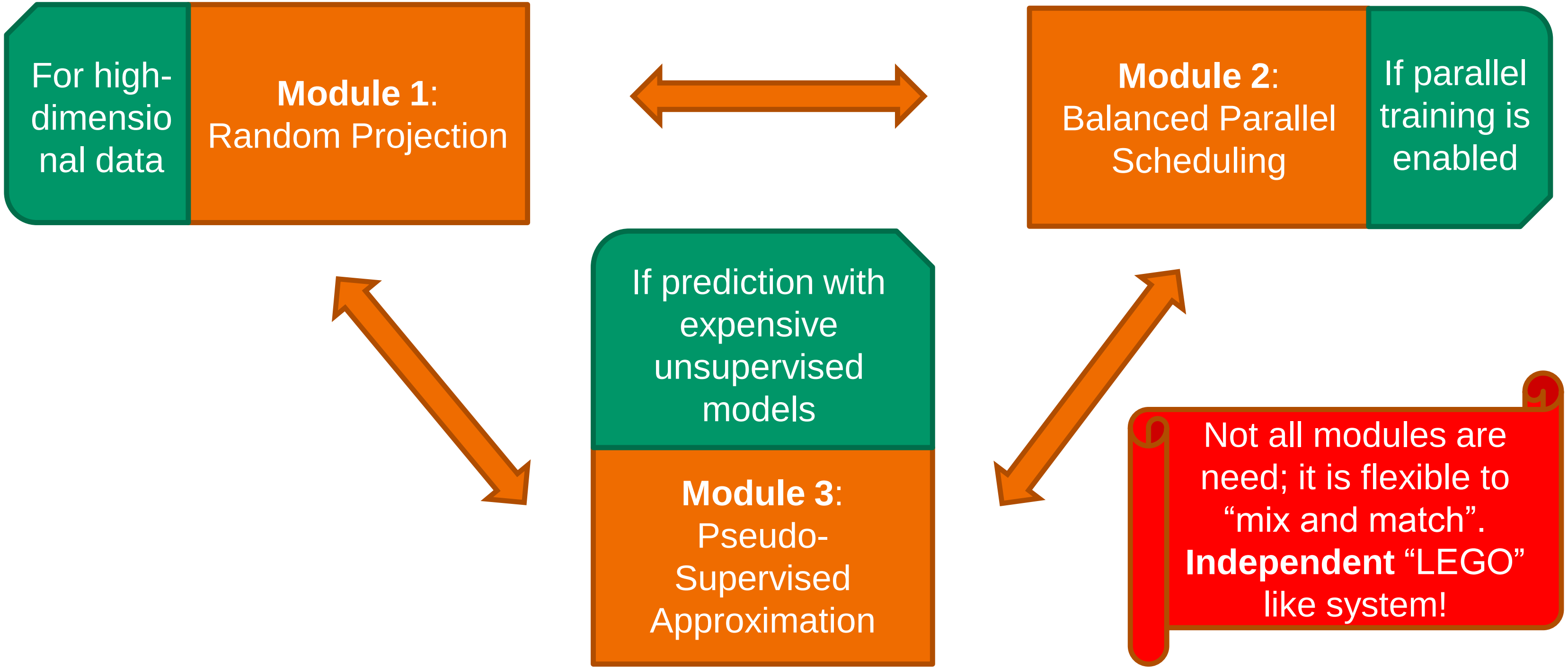
Most outlier ensembles (unsupervised [1], semi-supervised [2], and supervised, need to build a large group of unsupervised detectors first.

The proposed SUOD framework focuses on **accelerating the training and prediction when many detectors are used**.

[1] Lazarevic, A. and Kumar, V. 2005. Feature bagging for outlier detection. *ACM SIGKDD*. (2005), 157.

[2] Zhao, Y. and Hryniewicki, M.K. 2018. XGBOD: Improving Supervised Outlier Detection with Unsupervised Representation Learning. *IJCNN*. (2018).

SUOD: A Three-module **Acceleration** System



Module I: Random Projection

For high-dimensional datasets, Johnson-Lindenstraus (JL) projection (covered in Nov 4th lecture) is leveraged to **reduce dimensionality** and **induce diversity** (by its randomness). Refer to our paper for details (proof and flowchart).

1. Four variants are used: (i) $basic$ (ii) $discrete$ (iii) $circulant$ (iv) $toeplitz$
2. Compared with: (i) $original$ (no projection) (ii) PCA (iii) RS (random subsets)
3. Run with three outlier detectors, e.g., ABOD, LOF, *k*NN

Module I: Random Projection

(a) ABOD on **MNIST**

Method	Time	ROC	PRN
original	12.89	0.80	0.39
PCA	8.93	0.81	0.37
RS	8.27	0.74	0.32
basic	8.94	0.80	0.38
discrete	8.86	0.80	0.39
circulant	9.33	0.80	0.38
toeplitz	8.96	0.80	0.38

(b) ABOD on **Satellite**

Method	Time	ROC	PRN
original	4.03	0.59	0.41
PCA	3.01	0.62	0.44
RS	3.53	0.63	0.44
basic	3.10	0.64	0.45
discrete	3.12	0.65	0.46
circulant	3.14	0.66	0.48
toeplitz	3.14	0.66	0.47

(c) ABOD on **Satimage-2**

Method	Time	ROC	PRN
original	3.68	0.85	0.28
PCA	2.70	0.88	0.30
RS	3.20	0.89	0.28
basic	2.78	0.91	0.29
discrete	2.79	0.91	0.31
circulant	2.85	0.91	0.29
toeplitz	2.83	0.92	0.30

(d) LOF on **MNIST**

Method	Time	ROC	PRN
original	7.64	0.68	0.29
PCA	4.92	0.67	0.27
RS	3.65	0.63	0.23
basic	4.87	0.70	0.31
discrete	5.21	0.70	0.32
circulant	5.06	0.69	0.31
toeplitz	4.97	0.71	0.31

(e) LOF on **Satellite**

Method	Time	ROC	PRN
original	0.82	0.55	0.38
PCA	0.23	0.54	0.36
RS	0.39	0.54	0.37
basic	0.31	0.54	0.37
discrete	0.32	0.54	0.37
circulant	0.39	0.55	0.37
toeplitz	0.37	0.54	0.37

(f) LOF on **Satimage-2**

Method	Time	ROC	PRN
original	0.79	0.54	0.07
PCA	0.20	0.52	0.04
RS	0.37	0.53	0.08
basic	0.29	0.52	0.08
discrete	0.30	0.53	0.07
circulant	0.43	0.59	0.11
toeplitz	0.32	0.54	0.09

(g) *k*NN on **MNIST**

Method	Time	ROC	PRN
original	7.13	0.84	0.42
PCA	3.92	0.84	0.40
RS	3.33	0.77	0.34
basic	4.17	0.84	0.42
discrete	4.11	0.84	0.41
circulant	4.13	0.84	0.41
toeplitz	4.11	0.84	0.42

(h) *k*NN on **Satellite**

Method	Time	ROC	PRN
original	0.71	0.67	0.49
PCA	0.18	0.67	0.50
RS	0.31	0.68	0.49
basic	0.24	0.68	0.49
discrete	0.25	0.69	0.50
circulant	0.33	0.70	0.50
toeplitz	0.30	0.70	0.51

(i) *k*NN on **Satimage-2**

Method	Time	ROC	PRN
original	0.68	0.94	0.39
PCA	0.15	0.94	0.39
RS	0.29	0.94	0.38
basic	0.23	0.94	0.38
discrete	0.20	0.95	0.37
circulant	0.36	0.96	0.37
toeplitz	0.25	0.96	0.39

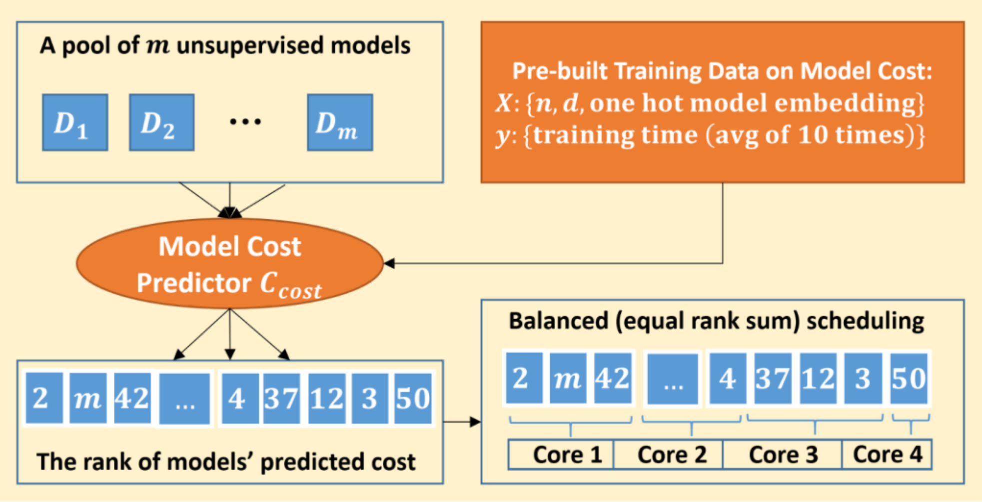
All projection methods are faster than operating on the full space ($original$).

$JL\ circulant$ and $JL\ Toeplitz$ work best across all projection methods regarding both accuracy (outperforming) and time complexity (on par).

Module II: Balanced Parallel Scheduling (BPS)

For parallel/distributed systems, **the scheduled task load among workers** (e.g., CPU cores) may be **imbalanced**.

A **model cost predictor C_{cost}** is built to **forecast each model's running time**, so a balanced parallel scheduling system may be achieved by **minimizing the worker load imbalance.**



$$\min_{\mathcal{W}} \sum_{i=1}^{t} \left| \sum_{D_j \in \mathcal{W}_i} C_{cost}(\mathcal{D}_i) - \sum_{l=1}^{m} C_{cost}(\mathcal{D}_l) \right|$$

Module II: Balanced Parallel Scheduling (BPS)

Table 2: Comparison between simple scheduling and BPS

n	d	m	t	**Simple**	**BPS**	**% RED**
1831	21	100	2	26.33	19.85	24.61
1831	21	100	4	17.93	13.69	23.65
1831	21	100	6	19.16	15.23	20.51
1831	21	500	2	100.51	72.16	28.21
1831	21	500	4	80.38	39.46	50.91
1831	21	500	6	55.3	32.78	40.72
5393	10	100	2	51.11	35.17	31.19
5393	10	100	4	42.49	16.23	61.80
5393	10	100	6	38.45	16.97	55.86
5393	10	500	2	197.84	137.46	30.52
5393	10	500	4	167.36	76.14	54.51
5393	10	500	6	127.08	66.29	47.84
6870	16	100	2	80.89	67.23	16.89
6870	16	100	4	68.31	35.29	48.34
6870	16	100	6	49.19	24.18	50.84
6870	16	500	2	336.54	286.14	14.98
6870	16	500	4	345.69	164.02	52.55
6870	16	500	6	211.34	113.13	46.47

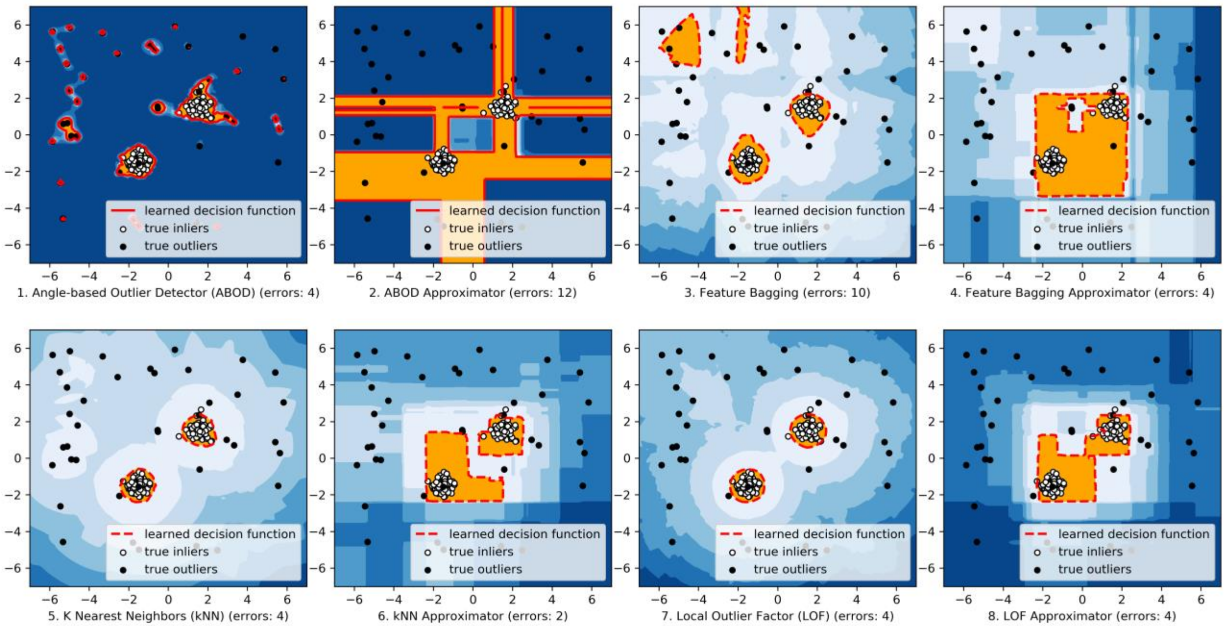
- BPS leads to 15%-60% execution time reduction with $\#\, models = \{100, 500\}$ and $\#\, workers = \{2,4,6\}$ on three datasets.
- The performance improvement is more apparent with **more models** to be scheduled against **more workers**.

Module III: Pseudo-supervised Approximation (PSA)

Approximate and replace each of **costly unsupervised model** by a faster **supervised regression model** for predicting on unseen samples.

- Unsupervised outlier detectors can be slow ($O(nd)$), e.g., *k*NN and LOF. Can be approximated by supervised regressors like ensemble trees ($O(dp)$).
- This may be viewed as distilling knowledge from slow unsupervised models, although it has multiple key differences from “knowledge distillation”.
- It may lead to faster inference, smaller storage cost, and better interpretability.

Module III: Pseudo-supervised Approximation (PSA)



PSA works well for **proximity-based models** operating on Euclidean space, but not **linear models**.

The visual on synthetic data shows some **regularization effects** on decision boundary.

See paper for extensive quantitative analysis.

Conclusion

A three-module acceleration framework, SUOD, is proposed for the training and prediction with many unsupervised outlier detectors: $\{Random\ Projection\}$, $\{Balanced\ Parallel\ Scheduling\}$, $\{Pseudo\ Supervised\ Approximination\}$. They are independent but can be mixed and match for flexibility.

Future Directions:

1. Demonstrate the module effectiveness as a complete framework.
2. Investigate the performance with the parallelization system with many workers.
3. Further analyze why and when will pseudo-supervised approximation work.

Model Reproducibility and Accessibility

- SUOD's code, experiment results, and figures are openly shared:

  https://github.com/yzhao062/SUOD

- **Production level implementation** will appear in **Python Outlier Detection Toolbox (PyOD)** [1]:

  https://github.com/yzhao062/pyod

- The extended version under review*:

  https://www.andrew.cmu.edu/user/yuezhao2/papers/19-preprint-suod.pdf

[1] Zhao, Y., Nasrullah, Z. and Li, Z., 2019. PyOD: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research (JMLR)*, 20(96), pp.1-7.

* An extended version is under review at *AAAI 2020 Workshop*. Will revise and resubmit for *KDD 2020 (ADS track)*.

SUOD: Toward Scalable Unsupervised Outlier Detection*

Yue Zhao, Xueying Ding, Jianing Yang, Haoping Bai
Carnegie Mellon University




* An extended version is under review at *AAAI 2020 Workshop*. Will revise and resubmit for *KDD 2020 (ADS track)*.