Journal of Machine Learning Research 20 (2019) 1-7 Submitted 1/19; Revised 4/19; Published 5/19

PyOD: A Python Toolbox for Scalable Outlier Detection

Yue Zhao ZHAOY@CMU.EDU
Carnegie Mellon University[*]
Pittsburgh, PA 15213, USA

Zain Nasrullah ZNASRULLAH@CS.TORONTO.EDU
University of Toronto
Toronto, ON M5S 2E4, Canada

Zheng Li JK_ZHENGLI@HOTMAIL.COM
Northeastern University Toronto
Toronto, ON M5X 1E2, Canada

Editor: Alexandre Gramfort

Abstract

`PyOD` is an open-source Python toolbox for performing scalable outlier detection on multivariate data. Uniquely, it provides access to a wide range of outlier detection algorithms, including established outlier ensembles and more recent neural network-based approaches, under a single, well-documented API designed for use by both practitioners and researchers. With robustness and scalability in mind, best practices such as unit testing, continuous integration, code coverage, maintainability checks, interactive examples and parallelization are emphasized as core components in the toolbox's development. `PyOD` is compatible with both Python 2 and 3 and can be installed through Python Package Index (PyPI) or `https://github.com/yzhao062/pyod`.

Keywords: anomaly detection, outlier detection, outlier ensembles, neural networks, machine learning, data mining, Python

1. Introduction

Outlier detection, also known as anomaly detection, refers to the identification of rare items, events or observations which differ from the general distribution of a population. Since the ground truth is often absent in such tasks, dedicated outlier detection algorithms are extremely valuable in fields which process large amounts of unlabelled data and require a means to reliably perform pattern recognition (Akoglu et al., 2012). Industry applications include fraud detection in finance (Ahmed et al., 2016), fault diagnosis in mechanics (Shin et al., 2005), intrusion detection in network security (Garcia-Teodoro et al., 2009) and pathology detection in medical imaging (Baur et al., 2018).

To help approach these problems, established outlier detection packages exist in various programming languages such as `ELKI Data Mining` (Achtert et al., 2010) and `RapidMiner` (Hofmann and Klinkenberg, 2013) in Java and `outliers` (Komsta and Komsta, 2011) in R. However, Python, one of the most important languages in machine learning, still lacks a dedicated toolkit for outlier detection. Existing implementations either stand as single

∗. Work initialized while at University of Toronto.

Method	Category	JIT Enabled	Multi-core
LOF (Breunig et al., 2000)	Proximity	No	Yes
kNN (Ramaswamy et al., 2000)	Proximity	No	Yes
AvgkNN (Angiulli and Pizzuti, 2002)	Proximity	No	Yes
CBLOF (He et al., 2003)	Proximity	Yes	No
OCSVM (Schölkopf et al., 2001)	Linear Model	No	No
LOCI (Papadimitriou et al., 2003)	Proximity	Yes	No
PCA (Shyu et al., 2003)	Linear Model	No	No
MCD (Hardin and Rocke, 2004)	Linear Model	No	No
Feature Bagging (Lazarevic and Kumar, 2005)	Ensembling	No	Yes
ABOD (Kriegel et al., 2008)	Proximity	Yes	No
Isolation Forest (Liu et al., 2008)	Ensembling	No	Yes
HBOS (Goldstein and Dengel, 2012)	Proximity	Yes	No
SOS (Janssens et al., 2012)	Proximity	Yes	No
AutoEncoder (Sakurada and Yairi, 2014)	Neural Net	Yes	No
AOM (Aggarwal and Sathe, 2015)	Ensembling	No	No
MOA (Aggarwal and Sathe, 2015)	Ensembling	No	No
SO-GAAL (Liu et al., 2019)	Neural Net	No	No
MO-GAAL (Liu et al., 2019)	Neural Net	No	No
XGBOD (Zhao and Hryniewicki, 2018b)	Ensembling	No	Yes
LSCP (Zhao et al., 2019)	Ensembling	No	No

Table 1: Select outlier detection models in `PyOD`

algorithm tools like `PyNomaly` (Constantinou, 2018) or exist as part of a general-purpose framework like `scikit-learn` (Pedregosa et al., 2011) which does not cater specifically to anomaly detection. To fill this gap, we propose and implement `PyOD`—a comprehensive Python toolbox for scalable outlier detection.

Compared to existing libraries, `PyOD` has six distinct advantages. Firstly, it contains more than 20 algorithms which cover both classical techniques such as local outlier factor and recent neural network architectures such as autoencoders or adversarial models. Secondly, `PyOD` implements combination methods for merging the results of multiple detectors and outlier ensembles which are an emerging set of models. Thirdly, `PyOD` includes a unified API, detailed documentation and interactive examples across all algorithms for clarity and ease of use. Fourthly, all models are covered by unit testing with cross platform continuous integration, code coverage and code maintainability checks. Fifthly, optimization instruments are employed when possible: just-in-time (JIT) compilation and parallelization are enabled in select models for scalable outlier detection. Lastly, `PyOD` is compatible with both Python 2 and 3 across major operating systems (`Windows`, `Linux` and `MacOS`). Popular detection algorithms implemented in `PyOD` (version 0.7.0) are summarized in Table 1.

2. Project Focus

Build robustness. Continuous integration tools (*Travis CI*, *AppVeyor* and *CircleCI*) are leveraged to conduct automated testing under various versions of Python and operating systems. Tests are executed daily, when commits are made to the development and master branches, or when a pull request is opened.
Quality assurance. The project follows the `PEP8` standard; maintainability is ensured by *CodeClimate*, an automated code review and quality assurance tool. Additionally, code blocks with high cognitive complexity are actively refactored and a standard set of unit tests exist to ensure 95% overall code coverage. These design choices enhance collaboration and this standard is enforced on all pull requests for consistency.
Community-based development. `PyOD`'s code repository is hosted on GitHub[1] to facilitate collaboration. At the time of this writing, eight contributors have helped develop the code base and others have contributed in the form of bug reports and feature requests.
Documentation and examples. Comprehensive documentation is developed using `sphinx` and `numpydoc` and rendered using *Read the Docs*[2]. It includes detailed API references, an installation guide, code examples and algorithm benchmarks. An interactive Jupyter notebook is also hosted on *Binder* allowing others to experiment prior to installation.
Project relevance. `PyOD` has been used in various academic and commercial projects (Zhao and Hryniewicki, 2018a; Ramakrishnan et al., 2019; Krishnan and Wu, 2019). The GitHub repository receives more than 10,000 monthly views and its PyPI downloads exceed 6,000 per month.

3. Library Design and Implementation

`PyOD` is compatible with both Python 2 and 3 using `six`; it relies on `numpy`, `scipy` and `scikit-learn` as well. Neural networks such as autoencoders and SO_GAAL additionally require `Keras`. To enhance model scalability, select algorithms (Table 1) are optimized with JIT using `numba`. Parallelization for multi-core execution is also available for a set of algorithms using `joblib`. Inspired by `scikit-learn`'s API design (Buitinck et al., 2013), all implemented outlier detection algorithms inherit from a base class with the same interface: (i) `fit` processes the train data and computes the necessary statistics; (ii) `decision_function` generates raw outlier scores for unseen data after the model is fitted; (iii) `predict` returns a binary class label corresponding to each input sample instead of the raw outlier score and (iv) `predict_proba` offers the result as a probability using either normalization or Unification (Kriegel et al., 2011). Within this framework, new models are easy to implement by taking advantage of inheritance and polymorphism. Base methods can then be overridden as necessary.

Once a detector has been fitted on a training set, the corresponding train outlier scores and binary labels are accessible using its `decision_scores_` and `labels_` attributes. Once fitted, the model's `predict`, `decision_function` and `predict_proba` methods may be called for use on new data. An example showcasing the ease of use of this API is shown in Code Snippet 1 with an angle-based outlier detector (ABOD).

1. `https://github.com/yzhao062/pyod`
2. `https://pyod.readthedocs.io`

```
>>> from pyod.models.abod import ABOD
>>> from pyod.utils.data import generate_data
>>> from pyod.utils.data import evaluate_print
>>> from pyod.utils.example import visualize
>>>
>>> X_train, y_train, X_test, y_test = generate_data(\
...    n_train=200, n_test=100, n_features=2)
>>>
>>> clf = ABOD(method="fast")                          # initialize detector
>>> clf.fit(X_train)
>>>
>>> y_test_pred = clf.predict(X_test)                  # binary labels
>>> y_test_scores = clf.decision_function(X_test)      # raw outlier scores
>>> y_test_proba = clf.predict_proba(X_test)           # outlier probability
>>>
>>> evaluate_print("ABOD", y_test, y_test_scores)      # performance evaluation
ABOD Performance; ROC: 0.934; Precision at n: 0.902
>>>
>>> visualize(y_test, y_test_scores)                   # prediction visualization
```

Code Snippet 1: Demo of PyOD API with the ABOD detector

In addition to the outlier detection algorithms, a set of helper and utility functions (`generate_data`, `evaluate_print` and `visualize`) are included in the library for quick model exploration and evaluation. The two-dimensional artificial data used in the example is created by `generate_data` which generates inliers from a Gaussian distribution and outliers from a uniform distribution. An example of using `visualize` is shown in Figure 1.

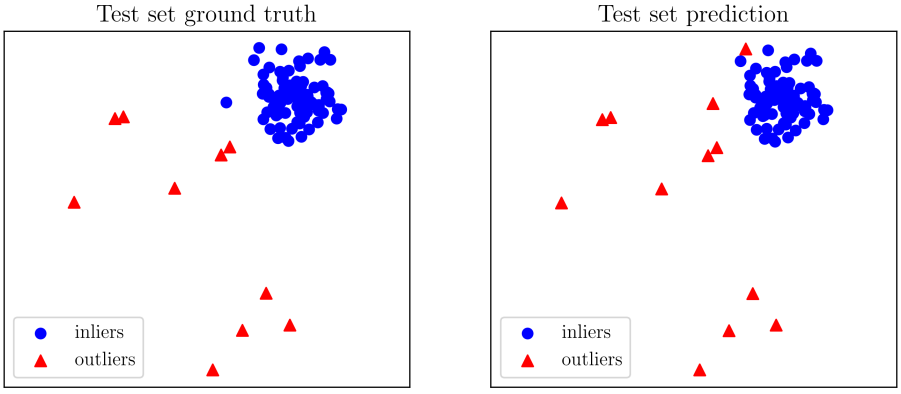

Figure 1: Demonstration of using `PyOD` in visualizing prediction result

4. Conclusion and Future Plans

This paper presents `PyOD`, a comprehensive toolbox built in Python for scalable outlier detection. It includes more than 20 classical and emerging detection algorithms and is being used in both academic and commercial projects. As avenues for future work, we plan to enhance the toolbox by implementing models that work well with time series and geospatial data, improving computational efficiency through distributed computing and addressing engineering challenges such as handling sparse matrices or memory limitations.

Acknowledgments

We thank the editor and anonymous reviewers for their constructive comments. This work was partly supported by Mitacs through the Mitacs Accelerate program.

References

Elke Achtert, Hans-Peter Kriegel, Lisa Reichert, Erich Schubert, Remigius Wojdanowski, and Arthur Zimek. Visual evaluation of outlier detection models. In *International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 396–399. Springer, 2010.

Charu C Aggarwal and Saket Sathe. Theoretical foundations and algorithms for outlier ensembles. *ACM SIGKDD Explorations Newsletter*, 17(1):24–47, 2015.

Mohiuddin Ahmed, Abdun Naser Mahmood, and Md Rafiqul Islam. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, 55:278–288, 2016.

Leman Akoglu, Hanghang Tong, Jilles Vreeken, and Christos Faloutsos. Fast and reliable anomaly detection in categorical data. In *International Conference on Information and Knowledge Management (CIKM)*, pages 415–424. ACM, 2012.

Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 15–27. Springer, 2002.

Christoph Baur, Benedikt Wiestler, Shadi Albarqouni, and Nassir Navab. Deep autoencoding models for unsupervised anomaly segmentation in brain mr images. *arXiv preprint arXiv:1804.04488*, 2018.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM SIGMOD Record*, volume 29, pages 93–104, 2000.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, et al. Api design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*, 2013.

Valentino Constantinou. Pynomaly: Anomaly detection using local outlier probabilities (LoOP). *The Journal of Open Source Software (JOSS)*, 3:845, 2018.

Pedro Garcia-Teodoro, Jesus Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2):18–28, 2009.

Markus Goldstein and Andreas Dengel. Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm. *Annual German Conference on Artificial Intelligence (KI-2012)*, pages 59–63, 2012.

Johanna Hardin and David M Rocke. Outlier detection in the multiple cluster setting using the minimum covariance determinant estimator. *Computational Statistics & Data Analysis*, 44(4):625–638, 2004.

Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.

Markus Hofmann and Ralf Klinkenberg. *RapidMiner: Data mining use cases and business analytics applications*. CRC Press, 2013.

JHM Janssens, Ferenc Huszár, EO Postma, and HJ van den Herik. Stochastic outlier selection. Technical report, Technical report TiCC TR 2012-001, Tilburg University, Tilburg Center for Cognition and Communication, Tilburg, The Netherlands, 2012.

Lukasz Komsta and Maintainer Lukasz Komsta. *Package outliers*, 2011. URL `https://cran.r-project.org/web/packages/outliers/outliers.pdf`. R package version 0.14.

Hans-Peter Kriegel, Arthur Zimek, et al. Angle-based outlier detection in high-dimensional data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 444–452. ACM, 2008.

Hans-Peter Kriegel, Peer Kroger, Erich Schubert, and Arthur Zimek. Interpreting and unifying outlier scores. In *SIAM International Conference on Data Mining (SDM)*, pages 13–24. SIAM, 2011.

Sanjay Krishnan and Eugene Wu. Alphaclean: Automatic generation of data cleaning pipelines. *arXiv preprint arXiv:1904.11827*, 2019.

Aleksandar Lazarevic and Vipin Kumar. Feature bagging for outlier detection. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 157–166. ACM, 2005.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *International Conference on Data Mining (ICDM)*, pages 413–422. IEEE, 2008.

Yezheng Liu, Zhe Li, Chong Zhou, Yuanchun Jiang, Jianshan Sun, Meng Wang, and Xiangnan He. Generative adversarial active learning for unsupervised outlier detection. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2019.

Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *International Conference on Data Engineering (ICDE)*, pages 315–326. IEEE, 2003.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830, 2011.

Jagdish Ramakrishnan, Elham Shaabani, Chao Li, and Mátyás A Sustik. Anomaly detection for an e-commerce pricing system. *arXiv preprint arXiv:1902.09566*, 2019.

Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *ACM SIGMOD Record*, volume 29, pages 427–438, 2000.

Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Pacific Rim International Conference on Artificial Intelligence (PRICAI), Workshop on Machine Learning for Sensory Data Analysis (MLSDA)*, 2014.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

Hyun Joon Shin, Dong-Hwan Eom, and Sung-Shick Kim. One-class support vector machinesan application in machine fault detection and classification. *Computers & Industrial Engineering*, 48(2):395–408, 2005.

Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. A novel anomaly detection scheme based on principal component classifier. Technical report, Department of Electrical and Computer Engineering, University of Miami, 2003.

Yue Zhao and Maciej K Hryniewicki. DCSO: dynamic combination of detector scores for outlier ensembles. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), Outlier Detection De-constructed Workshop*, London, UK, August 2018a.

Yue Zhao and Maciej K Hryniewicki. XGBOD: improving supervised outlier detection with unsupervised representation learning. In *International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro, Brazil, July 2018b. IEEE.

Yue Zhao, Zain Nasrullah, Maciej K Hryniewicki, and Zheng Li. LSCP: locally selective combination in parallel outlier ensembles. In *SIAM International Conference on Data Mining (SDM)*, pages 585–593, Calgary, Canada, May 2019. SIAM.