LOTUS: Learning to learn with Optimal Transport in Unsupervised Scenarios

Anonymous Author(s)
Affiliation
Address
email

Abstract

Automated machine learning has been widely researched and adopted for supervised tasks such as classification and regression. Unsupervised scenarios, lacking a ground truth to optimize on, are much harder to automate. We propose a novel zero-shot meta-learning approach that recommends which algorithms and hyperparameters to use on new unsupervised tasks by learning from prior supervised proxy datasets. Our premise is that the selection of optimal unsupervised algorithms depends on the inherent properties of the data distribution. We first build a large meta-dataset evaluating many algorithms and hyperparameter settings on prior datasets, leverage optimal transport to find the prior datasets with the most similar underlying distribution, and then recommend the (tuned) algorithm that proved to work best for that data distribution. We evaluate the robustness of our approach on one particular task, i.e. outlier detection, and find that it outperforms state of the art methods in unsupervised outlier detection.

1 Introduction

An open problem in Automated Machine Learning (AutoML) is how to select algorithms for unsupervised tasks, or how to efficiently optimize pipelines that include unsupervised preprocessing steps, such as outlier detection or dimensionality reduction. We propose a meta-learning framework for unsupervised machine learning which leverages optimal transport distances [14, 17] to recommend which unsupervised algorithms and hyperparameters to use based on how well they performed on proxy tasks with similar data distributions. Such recommendations can be used as smart defaults, or to warm-start or reduce the search space of AutoML techniques.

In this work we evaluate this approach specifically for outlier detection. Outlier detection (OD) is the process of identifying data points that are significantly different from the rest of the data. These data points can be caused by errors in the data collection process, incorrect values, or unusual events. Hence, it can be used to improve the quality of the data or to find unusual events that require special attention. We also introduce **GAMAOD**, an outlier detection extension to the AutoML framework GAMA [5], to collect rich meta-data to learn from.

2 Background

Many AutoML [8] tools leverage meta-learning schemes [23] to find good configurations to warm-start optimization. For instance, AutoSklearn-2.0 [4] learns pipeline portfolios, FLAML [24] uses

Submitted to the 6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans. Do not distribute.

meta-learned defaults, and MetaBu [15] uses optimal transport (Fused Gromov Wasserstein with proximal gradients) to learn better meta-features to find similar prior tasks.

AutoML for outlier detection is a much harder problem since it lacks an objective metric to guide optimization. MetaOD [26] uses collaborative filtering [21] to build a recommender system for predicting the best outlier detection techniques and leverage meta-learning based on landmark and model-based meta-features.[1] However, it's unclear if these unsupervised meta-features capture sufficient information about the underlying data distribution.

Optimal transport Optimal transport (OT) theory aims to find an optimal transport map between two probability measures, often on different metric spaces. We focus on the Gromov Wasserstein (GW) distance between two discrete probability distributions. Gromomv Wasserstein allows us to match points taken within different metric spaces. Hence, they can be used to measure the similarity between two numeric datasets. To speed up computation and use it in a realistic AutoML setting we use the Low-Rank Gromov-Wasserstein (GW-LR) approximation [18, 17, 19], which reduces the computational cost from cubic to linear time. [19] consider the GW problem with low-rank couplings, linked by a common marginal g. Therefore, the set of possible transport plans is restricted to those adopting the factorization of the form $P_r = Qdiag(1/g)R^T$. In this form Q and R are thin matrices with dimensionality of $n \times r$, $r \times m$ respectively and g is an $r-$dimensional probability vector. The GW-LR distance is be described as:

$$\text{GW-LR}^{(r)}((a, A), (b, B)) := \min_{(Q,R,g)\in\mathcal{C}_{a,b,r}} \mathcal{Q}_{A,B}(Qdiag(1/g)R^T) \tag{1}$$

3 Methodology

We introduce LOTUS, Learning to learn with Optimal Transport for Unsupervised Scenarios, which is summarized in Algorithm 1. LOTUS meta-learns how well different unsupervised algorithms work on prior *labeled* datasets. These can be datasets where the correct labels are known, or proxy tasks. For instance, for outlier detection we can use extremely imbalanced classification tasks where examples of the smallest class are considered outliers. More formally, we require:

- A collection of n prior labeled datasets $\mathcal{D}_{meta} = \{D_1, ..., D_n\}$ with test and train splits such that $D_i = (X_i^{train}, y_i^{train}), (X_i^{test}, y_i^{test})$.
- A collection of n optimized algorithms A_i^* with associated hyperparameters λ_i^* for every dataset in $\mathcal{D}_{meta}$; $\mathcal{A} = \{A^*_{\lambda_1^*}, ..., A^*_{\lambda_n^*}\}$

Given a new input dataset (i.e., outlier detection task) $D_{new} = (X_{new})$ without any labels, we aim to select a model $A^*_{\lambda^*} \in \mathcal{A}$ to employ on X_{new}, where $A^*_{\lambda^*}$ is a tuned model for a dataset similar to X_{new}.

Our premise is that, if a prior dataset exists that is very similar to the new dataset, then its optimal algorithms will likely work well on the new dataset. We consider two datasets similar if they have the same underlying data distribution, which we measure using (unsupervised) Optimal Transport.

We first require a transformation function ϕ to map the dataset to a metric space. Next, we calculate the dataset similarity $\mathcal{O}$ based on some distance metric ψ in equation 2. Because our distributions lie on different metric spaces, and we require computationally efficient similarity estimates, we adopt the Low Rank Gromov-Wasserstein distance from equation 1 on these transformed distributions, as summarized in equation 3, where r is the selected rank.

$$\mathcal{O} = \psi(\phi(D_a)\phi(D_b)) \tag{2}$$

$$\mathcal{O} = \text{GW-LR}^{(r)}(\phi(D_a)\phi(D_b)) \tag{3}$$

[1] PyODDS [11] also claims to automate outlier detection but uses a supervised metric, without meta-learning.

Algorithm 1 Pseudocode for LOTUS

Inputs: $D_{new}, \mathcal{D}_{meta}, \mathcal{A}$

while $D_i \in \mathcal{D}_{meta}$ **do**
    $\mathcal{O}_i \leftarrow \psi(\phi(D_{new}, D_i))$ ▷ Distance calculation
$s \leftarrow \underset{i}{\text{argmin}}\{\mathcal{O}_1, ..., \mathcal{O}_n\}$ ▷ Retrieval of most similar dataset
$A^*_{\lambda^*_{new}} \leftarrow A^*_{\lambda^*_s}$ ▷ Model Selection

The most similar prior dataset $D_s \in \mathcal{D}_{meta}$ is then the dataset with the smallest distance to the new dataset D_{new}. LOTUS then assigns the optimal configuration from $\mathcal{A}$: $A^*_{\lambda^*_{new}} = A^*_{\lambda^*_s}$ where $A^*_{\lambda^*_s}$ is predicted as the optimal configuration for D_{new}.

GAMAOD: Automated supervised learning for outlier detection To populate our meta-data we develop an extension on top of GAMA [5], whose search space consists of all outlier detectors (and their hyperparameters) from PyOD [25], a Python library for detecting outlying objects in multivariate data. GAMAOD can use different metrics to optimise for the given task, such as AUC and PRC.

4 Experiments on ADBench

For our experiments, we use ADBench [7] and retrieve all tabular datasets. This collection consists of 46 datasets. As we do not have access to multiple benchmarks we use a leave-one-out strategy for the evaluation of our system, i.e., we take out one dataset at a time from ADBench and use only the other datasets in the meta-data. This ensures independent meta-training on the following datasets. We compare our approach against 7 outlier detection algorithms available in PyOD [25]: IForest [12], ABOD [10], OCSVM [20], LODA [13], KNN [1, 16], HBOS [6], and COF [22]. We also compare it against the current state of the art meta-learner for outlier detection, MetaOD [26]. Based on preliminary experiments, we used ICA [9] as our transformation function and run LR-GW with rank 6.

5 Results and Discussion

5.1 LOTUS vs MetaOD

For pairwise comparison of LOTUS and MetaOD, we use the Bayesian Wilcoxon signed-rank test (or ROPE test [2, 3]). We use AUC as our performance measure and set the ROPE value to 1%.[2] Results are shown in Figure 1. We find that, based on experiments over the 46 datasets, there is a 74.2 % probability that LOTUS will outperform MetaOD. Since $p(LOTUS) > p(MetaOD)$ LOTUS proves to be more robust. We show the per-dataset performances in Appendix A.1.

5.2 LOTUS vs individual methods

The results of the ROPE test comparing LOTUS with individual outlier detection techniques are shown in Table 1. LOTUS proves to be significantly better than other techniques, with default parameters. In this case $P(LOTUS) >> P(Estimator)$. We also include the critical difference plot of LOTUS vs PyOD estimators in Figure 2, again showing that it performs significantly better. The detailed experimental results are reported in appendix A.1 table 3 and Figure 3.

5.3 Limitations

First, LOTUS depends on the quality of meta-data, i.e. the variety of datasets and algorithms. If there are no similar datasets in $\mathcal{D}_{meta}$, LOTUS can recommend a dataset which is not sufficiently similar to new dataset. On the other hand, it is expected to improve as more benchmarks and datasets

[2]We use the baycomp library [2] to run and visualize the analysis

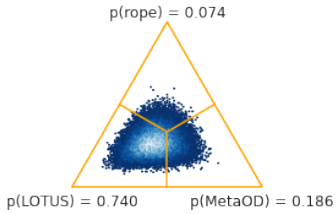


Figure 1: ROPE plot, showing the probability distribution of LOTUS outperforming MetaOD.

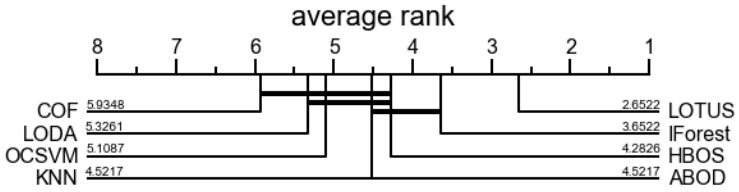


Figure 2: Comparison of average rank (lower is better) of outlier detection methods w.r.t. performance across datasets in ADBench

with different properties become available. Second, the computation cost of GW-LR on really large datasets can still be very high. In these cases we recommend using an appropriate subsampling technique. Finally, tuning the rank of GW-LR can be tricky. A low rank results in faster computation but high loss, and vice versa. [19] studies this effect for GW-LR. This rank can also be tuned by minimizing the loss between GW and GW-LR.

6 Conclusion and Future Work

In this work, we propose LOTUS, a meta-learning technique which uses optimal transport distances to estimate the similarity between datasets and uses this to recommend tuned algorithms on unsupervised tasks. We evaluate this technique on outlier detection and developed a new AutoML tool, **GAMAOD**, to collect the required metadata. We demonstrate that LOTUS outperforms MetaOD and other built-in estimators in PyOD. The LOTUS approach enables researchers to use a simplified meta-learning framework as compared to other methods based on hand-crafted meta-features, and can be used to warm-start various AutoML approaches. Finally, we believe that this approach can be extended to perform model selection in other unsupervised machine learning tasks as well. These include clustering, distance metric learning, density estimation and covariance estimation.

Estimator name	p(LOTUS)	p(rope)	p(Estimator)
IForest	0.99954	0.0	0.00046
ABOD	1.0	0.0	0.0
OCSVM	1.0	0.0	0.0
LODA	1.0	0.0	0.0
KNN	1.0	0.0	0.0
HBOS	0.99982	0.0	0.00018
COF	1.0	0.0	0.0

Table 1: Rope testing results with LOTUS vs PyOD estimators with rope=1%

References

[1] F. Angiulli and C. Pizzuti. "Fast Outlier Detection in High Dimensional Spaces". In: *PKDD*. 2002.

[2] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon. "Time for a Change: a Tutorial for Comparing Multiple Classifiers Through Bayesian Analysis". In: *Journal of Machine Learning Research* 18.77 (2017), pp. 1–36.

[3] A. Benavoli, G. Corani, F. Mangili, M. Zaffalon, and F. Ruggeri. "A Bayesian Wilcoxon signed-rank test based on the Dirichlet process". In: *Proceedings of the 31st International Conference on Machine Learning*. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China, 2014, pp. 1026–1034.

[4] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter. "Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning". In: *arXiv:2007.04074 [cs.LG]* (2020).

[5] P. Gijsbers and J. Vanschoren. "GAMA: A General Automated Machine Learning Assistant". In: *Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track*. Cham, 2021, pp. 560–564.

[6] M. Goldstein and A. R. Dengel. "Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm". In: 2012.

[7] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao. "ADBench: Anomaly Detection Benchmark". In: *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2022.

[8] F. Hutter, L. Kotthoff, and J. Vanschoren. "Automated Machine Learning: Methods, Systems, Challenges". In: *Automated Machine Learning* (2019).

[9] A. Hyvärinen and E. Oja. "Independent component analysis: algorithms and applications". In: *Neural networks : the official journal of the International Neural Network Society* 13 4-5 (2000), pp. 411–30.

[10] H.-P. Kriegel, M. Schubert, and A. Zimek. "Angle-Based Outlier Detection in High-Dimensional Data". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2008, pp. 444–452.

[11] Y. Li, D. Zha, N. Zou, and X. Hu. "PyODDS: An End-to-end Outlier Detection System with Automated Machine Learning". In: *Companion Proceedings of the Web Conference 2020* (2020).

[12] F. T. Liu, K. M. Ting, and Z.-H. Zhou. "Isolation Forest". In: *2008 Eighth IEEE International Conference on Data Mining* (2008), pp. 413–422.

[13] T. Pevný. "Loda: Lightweight on-line detector of anomalies". In: *Machine Learning* 102 (2015), pp. 275–304.

[14] G. Peyré and M. Cuturi. "Computational Optimal Transport". In: *Found. Trends Mach. Learn.* 11 (2019), pp. 355–607.

[15] H. Rakotoarison, L. Milijaona, A. RASOANAIVO, M. Sebag, and M. Schoenauer. "Learning meta-features for AutoML". In: *International Conference on Learning Representations*. 2022.

[16] S. Ramaswamy, R. Rastogi, and K. Shim. "Efficient Algorithms for Mining Outliers from Large Data Sets". In: *SIGMOD Rec.* 29.2 (2000), pp. 427–438.

[17] M. Scetbon and M. Cuturi. "Low-rank Optimal Transport: Approximation, Statistics and Debiasing". In: *NeurIPS 2022* abs/2205.12365 (2022).

[18] M. Scetbon, M. Cuturi, and G. Peyré. "Low-Rank Sinkhorn Factorization". In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. 2021, pp. 9344–9354.

[19] M. Scetbon, G. Peyré, and M. Cuturi. "Linear-Time Gromov Wasserstein Distances using Low Rank Couplings and Costs". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. 2022, pp. 19347–19365.

[20] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. C. Platt. "Support Vector Method for Novelty Detection". In: *NIPS*. 1999.

[21] D. Stern, R. Herbrich, T. Graepel, H. Samulowitz, L. Pulina, and A. Tacchella. "Collaborative Expert Portfolio Management". In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence AAAI-10 (to appear)*. 2010.

[22] J. Tang, Z. Chen, A. W.-c. Fu, and D. W. Cheung. "Enhancing Effectiveness of Outlier Detections for Low Density Patterns". In: *Advances in Knowledge Discovery and Data Mining*. Berlin, Heidelberg, 2002, pp. 535–548.

[23] J. Vanschoren. "Meta-Learning: A Survey". In: *ArXiv* abs/1810.03548 (2018).

[24] C. Wang, Q. Wu, M. Weimer, and E. Zhu. "FLAML: A Fast and Lightweight AutoML Library". In: *MLSys*. 2021.

[25] Y. Zhao, Z. Nasrullah, and Z. Li. "PyOD: A Python Toolbox for Scalable Outlier Detection". In: *J. Mach. Learn. Res.* 20 (2019), 96:1–96:7.

[26] Y. Zhao, R. Rossi, and L. Akoglu. "Automatic Unsupervised Outlier Model Selection". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021, pp. 4489–4502.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] They are in the code
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] No, there was no random seed, though the metadata will be changed after running any other automl system on it
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] These experiments can be ran on CPU cluster in parallel for meta-data but for main approach of the paper a single cpu is enough
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] ADBench, OTTJAX
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] All asset links are in URL
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Dataset	LOTUS	MetaOD
19_landsat	**0.7902**	0.5931
25_musk	**0.9895**	0.9655
24_mnist	**1.0000**	1.0000
32_shuttle	**0.9216**	**0.9163**
23_mammography	**0.6434**	0.6477
42_WBC	0.8521	**0.8655**
15_Hepatitis	**0.9353**	**0.9353**
43_WDBC	0.8548	**0.9671**
12_fault	**0.9246**	0.9043
10_cover	**0.9463**	**0.9436**
34_smtp	0.2744	**0.5212**
11_donors	**0.8064**	**0.8049**
29_Pima	**0.8804**	0.7197
37_Stamps	**0.9275**	**0.9339**
44_Wilt	**0.7765**	0.5327
40_vowels	0.8491	**0.9355**
8_celeba	**0.9908**	0.9906
1_ALOI	**0.8954**	0.8957
30_satellite	**0.8913**	0.7890
26_optdigits	**0.9996**	0.9997
2_annthyroid	**0.8472**	0.8445
41_Waveform	**0.9758**	0.9413
28_pendigits	0.8597	**0.9265**
4_breastw	**0.7466**	0.7438
21_Lymphography	0.9441	**0.9861**
20_letter	0.9701	**0.9891**
39_vertebral	0.7634	**0.8424**
47_yeast	**0.9089**	**0.9097**
3_backdoor	**1.0000**	1.0000
13_fraud	**0.9646**	0.8904
45_wine	**0.9841**	0.9481
22_magic.gamma	**0.9322**	0.8122
9_census	0.9819	**1.0000**
7_Cardiotocography	**0.9392**	**0.9378**
35_SpamBase	**0.9446**	0.9015
46_WPBC	0.7811	**0.8088**
36_speech	**1.0000**	0.4344
6_cardio	**0.9794**	**0.9793**
31_satimage-2	**0.9552**	0.8100
18_Ionosphere	0.8072	**0.8338**
27_PageBlocks	0.7164	**0.7668**
5_campaign	**0.9922**	**0.9996**

Table 2: AUC scores of MetaOD vs LOTUS on ADBench

A Appendix

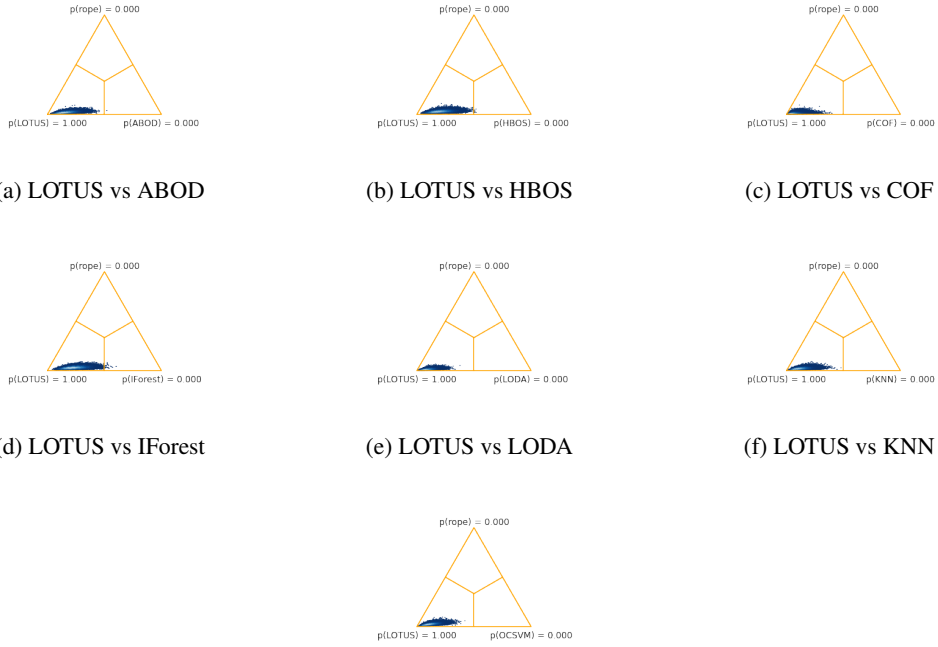
A.1 Performances

Table 2 contains the performances of LOTUS and MetaOD on 42 datasets, We had to eliminate 4 datasets from this experiment because MetaOD returned invalid models for these datasets(i.e. models with invalid values). Scores are in bold where AUC of LOTUS $>$ MetaOD or differ by less than a %. The dataset names are as they were in ADBench [7].

Table 3 reports the auc scores over datasets from ADBench. The bold number shows scores where LOTUS is better than **all** other estimators in PyOD.

Dataset	IForest	ABOD	OCSVM	LODA	KNN	HBOS	COF	LOTUS
44_Wilt	0.471963	0.568222	0.301310	0.408280	0.472095	0.281412	0.544269	**0.7765**
6_cardio	0.943738	0.498576	0.939676	0.892753	0.741544	0.865343	0.544550	**0.9794**
43_WDBC	0.987241	0.987241	0.989655	0.987586	0.960345	**0.998966**	0.771034	0.8548
4_breastw	0.976321	0.976321	0.778694	**0.981964**	0.947386	0.969329	0.381366	0.7466
42_WBC	**0.993567**	**0.993567**	**0.994103**	**0.995980**	0.911954	**0.991691**	0.754757	0.8521
47_yeast	0.431011	0.417114	0.448353	0.492504	0.413668	0.410032	0.428639	**0.9089**
45_wine	0.735205	0.735205	0.681612	0.923158	0.471241	0.891757	0.412289	**0.9841**
5_campaign	0.692549	0.642977	0.645556	0.566477	0.696817	0.771387	0.564588	**0.9922**
46_WPBC	0.522489	0.522489	0.475911	0.562133	0.419170	0.555259	0.495170	**0.7811**
7_Cardiotocography	0.752439	0.539423	0.810433	0.785916	0.582569	0.623355	0.572511	**0.9392**
8_celeba	0.757810	0.757810	0.761861	0.718291	0.632204	0.805965	0.393545	**0.9908**
9_census	0.598140	0.598140	0.523211	0.325589	0.650628	0.633393	0.413254	**0.9819**
39_vertebral	0.377788	0.377788	0.427308	0.284423	0.417163	0.282356	0.321923	**0.7634**
41_Waveform	0.669757	0.698172	0.474443	0.611266	0.782120	0.639714	0.804121	**0.9758**
38_thyroid	**0.979620**	**0.979620**	0.867786	0.699534	0.951152	0.952834	0.871991	0.7910
40_vowels	0.708373	0.956714	0.532701	0.655924	**0.971722**	0.646130	0.849763	0.8491
3_backdoor	0.734361	0.734361	0.802264	0.708914	0.738679	0.665487	0.728995	**1.0000**
32_shuttle	0.996250	0.618768	0.987461	0.951075	0.678578	0.994925	0.557606	0.9216
31_satimage-2	0.996844	0.762625	0.983527	0.987126	0.909884	0.985936	0.451384	0.9552
26_optdigits	0.771433	0.525541	0.527237	0.623480	0.398194	0.852822	0.423611	**0.9996**
1_ALOI	0.501898	0.609567	0.532848	0.549594	0.555634	0.478001	0.635583	**0.8954**
35_SpamBase	0.657074	0.390792	0.520510	0.273952	0.515358	0.651507	0.416468	**0.9446**
36_speech	0.469975	0.729473	0.462061	0.448529	0.473192	0.476358	0.553156	**1.0000**
34_smtp	0.696899	0.670223	0.018006	0.372124	0.744582	0.878626	0.890630	0.2744
22_magic.gamma	0.704407	0.799144	0.594241	0.635940	0.823228	0.681717	0.663549	**0.9322**
23_mammography	0.859409	0.859409	0.854704	0.814810	0.859614	0.871755	0.792004	0.6434
24_mnist	0.794443	0.750330	0.834765	0.743575	0.828259	0.619057	0.733384	**1.0000**
20_letter	0.581556	0.880889	0.485185	0.627407	0.867111	0.540593	0.829704	**0.9701**
30_satellite	0.707795	0.538013	0.605468	0.609243	0.646056	0.768130	0.556999	**0.8913**
19_landsat	0.495534	0.500057	0.374050	0.382382	0.577134	0.556768	0.542057	**0.7902**
37_Stamps	0.909527	0.909527	0.878255	0.944582	0.746473	0.928582	0.636364	0.9275
18_Ionosphere	0.867847	0.867847	0.765359	0.858325	0.862297	0.667416	0.850478	0.8072
21_Lymphography	0.997003	0.997003	0.993506	0.667582	0.512862	0.995005	0.934316	0.9441
25_musk	0.999923	0.085936	0.818675	0.959047	0.701124	1.000000	0.400387	0.9895
17_InternetAds	0.700473	0.673305	0.710028	0.580881	0.712320	0.704318	0.693902	**1.0000**
16_http	1.000000	1.000000	0.995308	0.000000	0.001340	0.994638	0.583110	0.7106
15_Hepatitis	0.742736	0.742736	0.722262	0.772817	0.467871	0.813292	0.425388	**0.9353**
14_glass	0.818496	0.818496	0.459264	0.632274	0.740799	0.791758	0.882668	0.8374
13_fraud	0.934023	0.941569	0.914391	0.751185	0.916394	0.941169	0.914591	**0.9646**
11_donors	0.794215	0.794215	0.723436	0.260784	0.829936	0.763981	0.720262	**0.8064**
12_fault	0.571477	0.676490	0.494426	0.436072	0.713079	0.479224	0.612146	**0.9246**
2_annthyroid	0.824922	0.824922	0.606069	0.305845	0.730291	0.691522	0.704828	**0.8472**
27_PageBlocks	0.889696	0.684494	0.892650	0.753280	0.769997	0.788657	0.673234	0.7164
28_pendigits	0.949714	0.673023	0.938642	0.951140	0.705836	0.921169	0.475639	0.8597
29_Pima	0.660016	0.660016	0.580166	0.606169	0.685681	0.713573	0.566752	**0.8804**
10_cover	0.914310	0.767605	0.886407	0.866889	0.899776	0.795243	0.870260	**0.9463**

Table 3: AUC Scores: LOTUS vs PyOD estimators with default configuration



(a) LOTUS vs ABOD (b) LOTUS vs HBOS (c) LOTUS vs COF

(d) LOTUS vs IForest (e) LOTUS vs LODA (f) LOTUS vs KNN

(g) LOTUS vs OCSVM

Figure 3: ROPE test result of LOTUS vs (a) ABOD (b) HBOS (c) COF (d) IForest (e) LODA (f) KNN (g) OCSVM

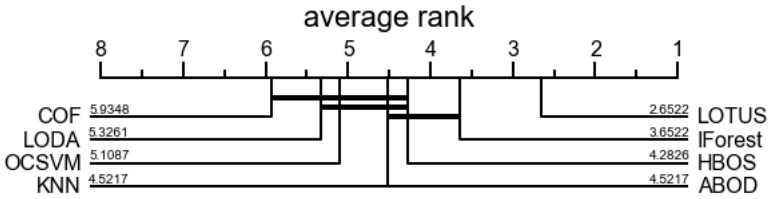


Figure 4: Comparison of average rank (lower is better) of methods w.r.t. performance across datasets in ADBench.

A.2 Baselines

The 8 baslines estimators and frameworks are listed below with brief description from PyOD's [25] documentation for reference here:

1. **MetaOD**: MetaOD is the first automated tool for outlier detection. MetaOD use collaborative filtering, landmark and model based meta-features to recommend the model for given task.
2. **IForest**: IsolationForest 'isolates' observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.
3. **LOF**:The anomaly score of each sample is called Local Outlier Factor. It measures the local deviation of density of a given sample with respect to its neighbors. It is local in that the anomaly score depends on how isolated the object is with respect to the surrounding neighborhood. More precisely, locality is given by k-nearest neighbors, whose distance is used to estimate the local density. By comparing the local density of a sample to the local

densities of its neighbors, one can identify samples that have a substantially lower density than their neighbors. These are considered outliers.

4. **ABOD**:For an observation, the variance of its weighted cosine scores to all neighbors could be viewed as the outlying score.
5. **HBOS**: Histogram- based outlier detection assumes the feature independence and calculates the degree of outlier by building histograms.
6. **KNN**: kNN class for outlier detection. For an observation, its distance to its kth nearest neighbor could be viewed as the outlying score.
7. **COF**: Connectivity-Based Outlier Factor uses the ratio of average chaining distance of data point and the average of average chaining distance of k nearest neighbor of the data point, as the outlier score for observations.
8. **LDOA**: Lightweight on-line detector of anomalies detects anomalies in a dataset by computing the likelihood of data points using an ensemble of one-dimensional histograms.
9. **OCSVM**: One class support vector machines unsupervised outlier Detection. Estimate the support of a high-dimensional distribution.

A.3 LOTUS+GAMAOD search space and MetaOD reproducibility

We implement the same searchspace as MetaOD's github repository for a fair comparison [3], MetaOD also uses all the existing datasets from ADbench. We believe that we have fairly evaluated MetaOD against out baseline. We believe that our Benchmark setting was more challenging than the one evaluated in [26] where it take child and parent datasets. [4]

A.4 Architecture

An overview of LOTUS system can be found in Figure 5. An overview of GAMAOD system can be

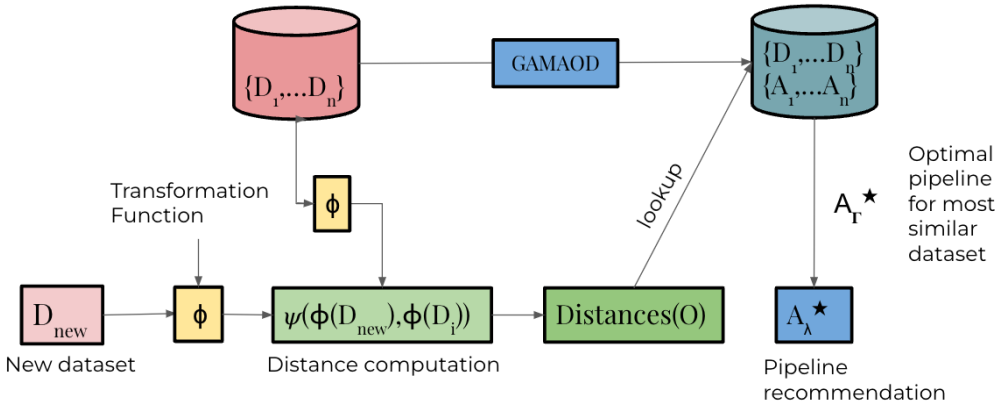
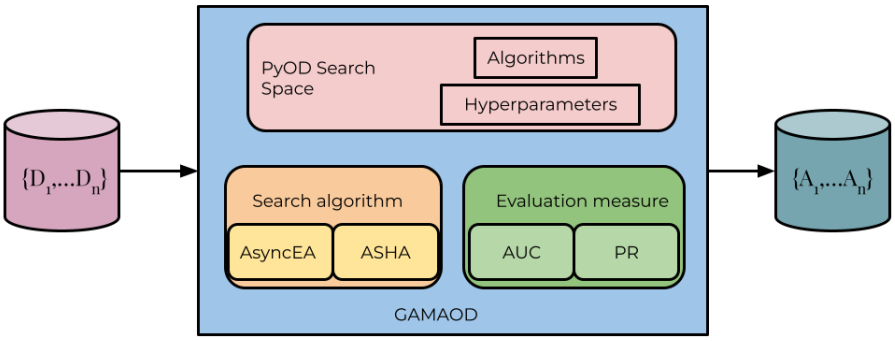


Figure 5: An overview of LOTUS

found in Figure 6.

[3] https://github.com/yzhao062/MetaOD/blob/master/metaod/models/base_detectors.py

[4] https://github.com/yzhao062/MetaOD/blob/2a8ed2761468d2f8ee2cd8194ce36b0f817576d1/metaod/models/train_metaod.py#L44



A.5 Experimental Implementation

Implementation details: We use Independent Component Analysis(ICA) from scikit-learn as our transformation function ϕ. We use OTT-JAX library to implement Low Rank Gromov Wassersstein distance. For this experiment, we set the rank parameter of Low Rank Gromov Wasserstein to 6. The model selection phase of LOTUS in our experiments is as follows: First the datasets are transformed via ICA and then converted into JAX pointclouds geometry objects [a] and then we turn these distributions into a quadratic regularized optimal transport problem. We input this quadratic problem to our Gromov Wasserstein Low Rank solver which returns us the distance(cost) between two datasets. When a new dataset is given to LOTUS, the pipeline corresponding to the dataset with the lowest distance(except the new dataset itself) is chosen from the optimal pipeline database.

Figure 6: An overview of GAMAOD

[a] `https://ott-jax.readthedocs.io/en/latest/_autosummary/ott.geometry.pointcloud.PointCloud.html`