Angle-Based Dictionary Learning for Outlier Detection

Denis C. Ilie-Ablachim, Bogdan Dumitrescu,

National University of Science and Technology Politehnica Bucharest
Department of Automatic Control and Computers
313 Spl. Independenţei, 060042 Bucharest, Romania
Emails: denis.ilie_ablachim@upb.ro, bogdan.dumitrescu@upb.ro

***Abstract*—We propose an extension of the Angle-Based Outlier Detection (ABOD) technique by combining it with a Dictionary Learning (DL) problem. The ABOD method benefits from this change by constructing an inlier base with the vector atoms obtained from the DL problem. Our method computes the dictionary D and calculates the angles between the data points in the feature space and the obtained atoms. We show that using a dictionary in the ABOD method can improve the results in anomaly detection tasks. Generally, the atoms can capture the direction of the inliers, thus better isolating the outliers.**

***Index Terms*—angle-based outlier detection, dictionary learning, anomaly detection, clustered anomalies**

I. Introduction

Dictionary Learning (DL) is a representation learning technique that aims to find a sparse approximation for a collection of N signals, $\boldsymbol{Y}$. Signals are organized in a matrix with N columns (samples), each with a size of m. The problem is solved by computing a dictionary $\boldsymbol{D}$ of size $m \times n$ and a sparse representation matrix $\boldsymbol{X}$ of size $n \times N$ such that a satisfactory approximation $\boldsymbol{Y} \approx \boldsymbol{D}\boldsymbol{X}$ is achieved. The problem can be formulated as follows

$$\begin{array}{ll} \min\limits_{\boldsymbol{D},\boldsymbol{X}} & \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 \\ \text{s.t.} & \|\boldsymbol{x}_\ell\|_0 \leq s, \ell = 1:N \\ & \|\boldsymbol{d}_j\| = 1, j = 1:n, \end{array}$$

where we define $\|\cdot\|_0$ as L0 norm (the number of nonzero entries) and s represents the sparsity constraint.

The standard dictionary learning problem can be effectively solved using established strategies. To address the challenges posed by its non-convex nature and high dimensionality, the optimization problem is organized in two steps: sparse coding and dictionary update. The procedure is known as DL by Alternate Optimization. The approach can yield satisfactory local solutions by alternating between these two stages for a specific number of iterations. An iteration involves first computing the sparse representation $\boldsymbol{X}$, while the dictionary $\boldsymbol{D}$ is fixed. In the next stage, the dictionary is updated, fixing the coefficients representation. An Orthogonal Matching Pursuit (OMP) [1] method can be employed for the sparse coding step. For the dictionary update stage, there are several well-known methods [2]. A usual choice can be the K-SVD [3] algorithm or its approximate version, named AK-SVD [4].

Anomaly detection, also termed outlier detection, represents the task of identifying samples that diverge from the general representation in the dataset. This task can be solved in both a supervised and unsupervised manner. In this paper, we concentrate on the unsupervised one. In the literature, there are several well-known libraries dedicated to outlier detection problems, such as the Python Outlier Detection (PyOD) toolbox [5], which contains a variety of unsupervised algorithms. Some of the most notable algorithms are: Isolation Forest (IForest) [6], k-Nearest Neighbors (kNN), One-Class SVM (OCSVM) [7], Minimum Covariance Determinant (MCD) [8], [9] and Angle-Based Outlier Detection (ABOD) [10]. A more comprehensive anomaly detection benchmark is ADBench [11], which stands out for its extensive coverage. This toolbox evaluates the performance of 30 different anomaly detection algorithms over 57 datasets (47 widely used real-world datasets and 10 more complex datasets from Computer Vision and Natural Language Processing). Compared to PyOD, ADBench introduces supervised and semi-supervised methods for anomaly detection tasks.

This paper presents an adaptation of the unsupervised algorithm named ABOD, also available in PyOD. We demonstrate that this method is not invariant to clustered anomalies. Since the anomalies are organized in clusters, there is a risk that the outlier detection method will classify the cluster as a cluster of normal data points. To overcome this bottleneck, we propose using a Dictionary Learning algorithm to isolate the outliers better. The paper is organized as follows. Section II introduces the anomaly detection problem and is divided into two subsections corresponding to different approaches to the problem. Subsection II-A presents several strategies of dictionary learning methods adapted for anomaly detection. Subsection II-B presents the Angle-Based Outlier Detection method and its approximation forms. Section III contains our main contribution, which is a mix of the previously mentioned methods; specifically, instead of computing angles between a sample vector and its neighbors, we calculate angles between the sample vector and the atoms of a learned dictionary. In section IV, we present the obtained results and algorithm

This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS - UEFISCDI, project number PN-III-P4-PCE-2021-0154, within PNCDI III.

performances over the PyOD benchmark datasets. We give our main conclusions in Section V.

II. Anomaly Detection

A. Anomaly Detection via Dictionary Learning

Dictionary Learning (DL) problems can be adapted for anomaly detection tasks. A natural way to do this is to solve the problem using all available data $\boldsymbol{Y}$, then compute the representation error

$$\boldsymbol{E} = \boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X},$$

and identify the outliers based on the magnitude of the error norm, $\|e_i\|$, which is more likely to be larger for anomalies than for normal signals. The assumption behind this strategy is that the dictionary should be able to capture the representation direction of the inliers since the number of outliers is much smaller. Due to their dissimilarity, we expect to obtain high errors for representing the outliers as a natural trade-off to minimize the objective. Effective anomaly detection should use under-complete dictionaries with small sparsity constraints. This ensures that the representations are optimized to favor similarity for normal signals. We note that this approach is unsupervised; no label information is needed.

To improve the representation of inliers, a selection procedure can also be employed (Selective DL [12]) during optimization. First, during the sparse coding stage, we only use a subsample of available signals to reduce the number of anomalies during training. In the dictionary update stage, the samples with the worst representation errors are eliminated, and the matrix $\boldsymbol{D}$ is updated only with the rest of the signals. Selective DL demonstrates good behavior in anomaly detection problems.

B. Angle-Based Outlier Detection

ABOD is based on the Angle-Based Outlier-Factor (ABOF) method of mining high-dimensional data to identify outliers. In a high-dimensional space, the distances between data points become less meaningful. This method demonstrates that the angles between distance vectors of points in a vector space are more suitable for anomaly detection tasks in a high-dimensional space. The discrimination between the inliers and the outliers is made by comparing the angles to other points. The motivation of the ABOF method is that inliers would be organized in clusters of data points. The angles between the lines formed by joining an inlier with two other points should have quite different values, since other points usually surround an inlier; this leads to significant variance of the angles. On the other side, a similar construction for outliers should lead to more uniform angles, since the outliers are positioned outside clusters. Therefore, we expect the variance of the angles to be small.

The ABOF approach provides a way to quantify the divergence in directions of objects relative to each other, helping to identify outliers in high-dimensional data based on the observed angles between distance vectors. The method can be formulated as follows.

Considering $\mathcal{D} \subseteq \mathbb{R}^m$, the space of the data points, and three relevant points $\boldsymbol{x}_{\vec{A}}, \boldsymbol{x}_{\vec{B}}$ and $\boldsymbol{x}_{\vec{C}}$, we express the Angle-Based Outlier-Factor $ABOF(x_{\vec{A}})$ with respect to the anchor $\boldsymbol{x}_{\vec{A}}$ as the variance over the angles between the difference vectors of $\boldsymbol{x}_{\vec{A}}$ to all the other pairs, weighted by the distance between the points

$$VAR_{\boldsymbol{x}_{\vec{B}},\boldsymbol{x}_{\vec{C}} \in \mathcal{D}} \left(\frac{\left(\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{B}}\right)^\top \left(\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{C}}\right)}{\|\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{B}}\|^2 \cdot \|\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{C}}\|^2} \right).$$

Taking into consideration the construction of triplet pairs, the current method has a time complexity of $O(n^3)$. An approximation algorithm named FastABOD can be used to address this issue. This method approximates the Angle-Based Outlier Factor (ABOF) using only a small subsample of the available data points. In the FastABOD method, only the points with the strongest weights are considered for the variance computation. Typically, the k-nearest neighbors method is used to identify relevant data points. This approximation is more effective, especially in low-dimensionality datasets, where the distance is more meaningful. Using nearest neighbors provides a better approximation of ABOF. The FastABOD method is more suitable for large datasets, reducing the computation to $O(n^2 + nk^2)$, where k represents the number of nearest neighbors. On the other hand, the algorithm's performance depends on the selected neighbors. Using a large number of neighbors improves the quality of the results but also affects the time complexity. Given the large nature of the available datasets, in practice, the method used is FastABOD. However, this method is not invariant to clustered anomalies; Figure 1 shows an example of this type of anomaly. Since the anomalies are structured in a cluster, this might be considered a cluster of standard samples during classification. In the spirit of the ABOF method, the supposed anomalies will be the inliers having significant deviation from the center of the inliers clusters. Clearly, these kinds of points will obtain small variances in terms of angle distances.

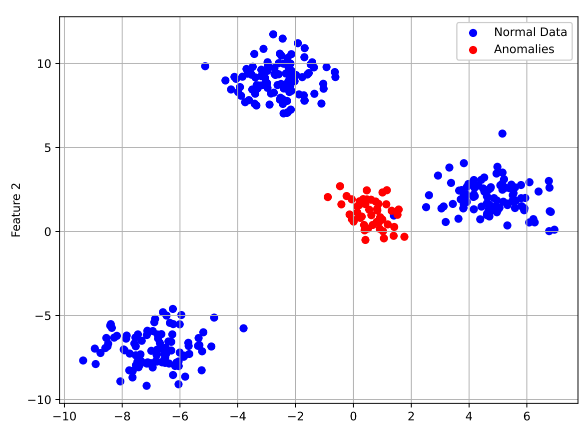

Fig. 1. Synthetic Dataset with Clustered Anomalies

This method cannot identify the outliers for scenarios where the anomalies are organized in data clusters larger than the number k. In this case, the data points within the anomalous clusters, the angles between them, and other points in the same cluster will have a consistent variance, similar to the angles between the inliers. Clustered anomalies might not show low-angle variability.

There are other approximation methods available. For example, in [13], the authors use a random projection-based technique to estimate the ABOF for the data points. This method achieves near-linear time complexity and can be easily parallelized. On the other hand, other adaptations exist, such as the concept of depth L1 [14]. This one only requires a quadratic time, making it more efficient than the original form. L1 depth can be estimated using a sampling method named SamDepth. These methods obtain competitive results, significantly reducing the execution complexity.

The ABOD function can be adapted using different distance metrics. For example, kernel methods can be used by substituting the scalar product with a kernel function. In this way, the data points are reprojected into the kernel space. The kernel enables new representation spaces, which might favor outlier identification. The initial definition of ABOF can be adapted by extending the distance vectors using a nonlinear function $\varphi(\cdot)$. Furthermore, the formulation is computed following the kernel trick, in which the scalar product $\varphi(x)^\top \varphi(y)$ is changed with a suitable kernel function

$$VAR_{\boldsymbol{x}_{\vec{B}},\boldsymbol{x}_{\vec{C}}\in\mathcal{D}} \left(\frac{\varphi\left(\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{B}}\right)^\top \varphi\left(\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{C}}\right)}{\|\varphi\left(\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{B}}\right)\|^2 \cdot \|\varphi\left(\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{C}}\right)\|^2} \right).$$

The squared norm $\|\varphi\left(\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{B}}\right)\|^2$ can be rewritten in the following form $\varphi\left(\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{B}}\right)^\top \varphi\left(\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{B}}\right)$, which also leads to a kernel function $k\left(\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{B}}, \boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{B}}\right)$. For the RBF kernel, since the denominator is equal to 1 (it becomes negligible), we choose to normalize the cosine distance between the directions of the representation with the squared norm of the original vectors:

$$VAR_{\boldsymbol{x}_{\vec{B}},\boldsymbol{x}_{\vec{C}}\in\mathcal{D}} \left(\frac{\varphi\left(\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{B}}\right)^\top \varphi\left(\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{C}}\right)}{\|\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{B}}\|^2 \cdot \|\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{C}}\|^2} \right).$$

For any other kernel function, the denominator is computed according to the kernel function, $\|\varphi\left(\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{B}}\right)\|^2 = k(\boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{B}}, \boldsymbol{x}_{\vec{A}} - \boldsymbol{x}_{\vec{B}})$ We name this method Kernel Angle-Based Outlier Detection (KABOD).

On the other hand, we propose adapting the kernel version by extending the features into the kernel space. Compared to the original formulation, the data points are first lifted into the kernel space. After that, we calculate the distance vectors, which will be further used for the angle orientation

$$VAR_{\boldsymbol{x}_{\vec{B}},\boldsymbol{x}_{\vec{C}}\in\mathcal{D}} \left(\frac{\left(\varphi\left(\boldsymbol{x}_{\vec{A}}\right) - \varphi\left(\boldsymbol{x}_{\vec{B}}\right)\right)^\top \left(\varphi\left(\boldsymbol{x}_{\vec{A}}\right) - \varphi\left(\boldsymbol{x}_{\vec{C}}\right)\right)}{\|\varphi\left(\boldsymbol{x}_{\vec{A}}\right) - \varphi\left(\boldsymbol{x}_{\vec{B}}\right)\|^2 \cdot \|\varphi\left(\boldsymbol{x}_{\vec{A}}\right) - \varphi\left(\boldsymbol{x}_{\vec{C}}\right)\|^2} \right).$$

Since the data points are lifted to an infinite space, we name this method an extension of the previous, Extended Kernel Angle-Based Outlier Detection (EKABOD). A good choice for the kernel function might be the Radial Basis Function (RBF) $k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\gamma\|\boldsymbol{x} - \boldsymbol{y}\|_2^2\right)$ or the Polynomial Kernel (Poly) $k(\boldsymbol{x}, \boldsymbol{y}) = (\gamma \boldsymbol{x}^\top \boldsymbol{y} + \alpha)^\beta$, where γ, α and β are hyperparameters.

III. Angle-Based Dictionary Learning

This section introduces our main contribution, Angle-Based Dictionary Learning (ABDL). This is a combination of the prior methods. We first solve a Dictionary Learning problem to generate a basis for the ABOF method. Then, we use the atom vectors to calculate the distance vectors and angle variations. Additionally, the approximation is determined by the k-nearest neighbors in relation to the dictionary atoms. As the dictionary should capture the direction of representation of the inliers, we anticipate a small variance for the outliers.

During the optimization process, the dictionary atoms will mostly capture the pertinent directions in relation to the inliers. It is important to note that we guarantee that the atoms are aligned with the current anchor for calculating the vector distances. For example, for a given data point $\boldsymbol{x}_A$ and two atoms $\boldsymbol{d}_1$, $\boldsymbol{d}_2$ we first align the atoms with the current anchor

$$\hat{\boldsymbol{d}}_1 = sign(\boldsymbol{d}_1^\top \boldsymbol{x}_A) \cdot \boldsymbol{d}_1$$
$$\hat{\boldsymbol{d}}_2 = sign(\boldsymbol{d}_2^\top \boldsymbol{x}_A) \cdot \boldsymbol{d}_2$$

and after this stage, we compute the ABOF as follows

$$ABDL(x_{\vec{A}}) = VAR_{x_{\boldsymbol{d}_1}, x_{\boldsymbol{d}_2}\in\mathcal{D}} \left(\frac{\boldsymbol{x}_{\boldsymbol{d}_1 A}^\top \boldsymbol{x}_{\boldsymbol{d}_2 A}}{\|\boldsymbol{x}_{\boldsymbol{d}_1 A}\|^2 \cdot \|\boldsymbol{x}_{\boldsymbol{d}_2 A}\|^2} \right),$$

where we denote

$$x_{\boldsymbol{d}_1 A} = \boldsymbol{x}_A - \hat{\boldsymbol{d}}_1$$
$$x_{\boldsymbol{d}_2 A} = \boldsymbol{x}_A - \hat{\boldsymbol{d}}_2.$$

This interpretation is in agreement with the kernel versions presented in the previous subsections, which use the atom vectors reprojected into the nonlinear space. The kernel version calculates the variances on the basis of the kernel distances relative to the dictionary atoms. The names used previously are adapted to Kernel Angle-Based Dictionary Learning (KABDL) for the standard form and Extended Kernel Angle-Based Dictionary Learning (EKABDL) for the extended kernel form.

IV. Experiments

In this section, we present the results of our experimental tests. For our examinations, we followed the PyOD framework [5], which is a comprehensive anomaly detection toolbox containing 17 real-world datasets from a large collection of outlier detection datasets of different domains (Outlier Detection DataSets - ODDS)[1] and 10 unsupervised methods for anomaly detection. Following the structure of this framework[2], we implemented 5 new strategies in the spirit of PyOD: KABOD, EKABOD, ABDL, KABDL, and EKABDL. Figure I summarizes the datasets used. In all of our tests, each dataset was divided into 60% for training and 40% for testing. To

[1] https://odds.cs.stonybrook.edu/

[2] https://pyod.readthedocs.io/en/latest/benchmark.html

Data	#Samples	# Dimensions	Outlier Perc
arrhythmia	452	274	14.6
cardio	1831	21	9.61
glass	214	9	4.2
ionosphere	351	33	35.89
letter	1600	32	6.25
lympho	148	18	4.05
mnist	7603	100	9.2
musk	3062	166	3.16
optdigits	5216	64	2.87
pendigits	6870	16	2.27
pima	768	8	34.89
satellite	6435	36	31.63
satimage-2	5803	36	1.22
shuttle	49097	9	7.15
vertebral	240	6	12.5
vowels	1456	12	3.43
wbc	378	30	5.55

TABLE I
DATA SUMMARY

measure the performance of the methods, we computed the receiver operating characteristic area under the curve (ROC AUC) and the precision @ rank n score over ten independent rounds with different initialization seeds. The presented results are calculated as the mean of the ten rounds. We also measured the execution time, but we will only briefly compare the original ABOD method and the new ones.

We performed our tests on a Desktop PC, including a processor with a base frequency of 2.90 GHz (Max Turbo Frequency of 4.80 GHz) and 80GB RAM. To select the values of the hyperparameters, we performed a grid search. More precisely, we have conducted tests with different numbers of atoms $n \in \{10, 20, 50\}$, sparsity levels $s \in \{2, 3, 5\}$, and number of neighbors $k \in \{3, 5, 8\}$. All dictionary learning methods involve the AK-SVD method for the dictionary training stage. For the update of the coefficient matrix, we used the OMP method. During our evaluation, we concluded that a maximum number of 10 iterations is enough to obtain reasonable results. We conclude that small dictionaries can better produce the representation directions of the inliers. Moreover, fewer atoms are favorable for computing an appropriate basis for the Angle-Based Outlier Detection method. We used $\gamma = 1/m$ for the kernel methods that depend on the number of features. For the polynomial kernel, we used $\alpha = 1$ and $\beta = 3$. Here, we only include the results obtained with the RBF kernel function. The hyperparameters for these results are $n = 50$ atoms, a sparsity level of $s = 3$, and $k = 5$ neighbors. We make public the full results and the source code of our implementations at https://asydil.upb.ro/software/.

The obtained results demonstrate the excellent behavior of our methods. We successfully improved the results on both the ABDL and DL anomaly detectors for several datasets. In Table II, we include the ROC AUC performance for all PyOD methods, including ours. The precision rank score is available in Table III. We do not include the execution times, but we conclude that our methods depend on the execution time of the DL problem. Since the dictionary learning problem is computationally costly, most of the time is spent during dictionary training. Moreover, this depends on the dictionary configuration and the dimensions of the dataset.

The results show that the mix between the ABOD and DL methods is beneficial. ABDL usually outperforms its ancestor methods by combining their strengths. DL is able to produce atoms with good representation power for inliers. Furthermore, ABOD can be successfully applied by computing the angles of the directions between data points and dictionary atoms, which appears to succeed in identifying many of the outliers.

The most relevant results, compared to the original ABOD algorithm in terms of ROC AUC performance, are obtained for the following datasets: cardio (improvement with 36%), musk (improvement with 70%), pendigits (improvement with 24%), and shuttle (improvement with 36%). For several datasets, our methods are efficient enough to obtain results in the top three of the ranking. For example, KABDL obtains third place for cardio and second place for pendigits dataset, while ABDL obtains the second place for the wbc dataset. Regarding the precision rank score, we also observed an improvement in the results. This shows that the proposed methods are invariant to the imbalanced nature of the anomalies or the clustered structure.

We performed some specific tests to understand better the roles of the sparsity level and the number of neighbors in our problem. We first fixed the parameters of our problem to $n = 50$ atoms and $k = 5$ neighbors. We then computed the ROC AUC performance at different levels of sparsity $s \in \{2, 3, 5, 8, 10, 15, 20, 25\}$. In Figure 2, we show the results obtained on the cardio dataset. We observe here that a sparsity level larger than the number of neighbors facilitates better identification of the outliers. Since the dictionary basis is over-specialized in the representation of the inliers, we expect to obtain better results with large sparsity levels. In this case, we can easily identify a threshold value, $s = 10$, after which the results do not improve. Moreover, the DL performance starts to decrease.

Another interesting experiment is shown in Figure 3 for the pendigit dataset. This time, we fixed the sparsity level to $s = 5$ and performed tests with different numbers of neighbors $k \in \{3, 5, 8, 10, 15, 20, 25, 50\}$. The results demonstrate that the number of neighbors should be comparable with the sparsity level. A few numbers of neighbors ensure good results, but the most relevant results are obtained when the number is near the sparsity level. For ABOD, KABOD, and EKABOD, it is clear that the best results are obtained when the complete dataset is used for variance computation; ROC AUC performance improves with the number of neighbors. On the other hand, for ABDL, KABDL, and EKABDL, it is clear that a small number of neighbors is sufficient to obtain good results. This demonstrates the excellent behavior of the dictionary matrix, which captures the inliers representation.

Overall, the mean ROC AUC score for all the datasets is 70% for ABOD, 69% for DL and 75% for ABDL. The ABDL has been shown to outperform its rivals in terms of performance.

Data	CBLOF	FB	HBOS	IForest	KNN	LOF	MCD	OCSVM	PCA	ABOD	KABOD	EKABOD	DL	ABDL	KABDL	EKABDL
arrhythmia	0.7838	0.7780	0.8219	0.7996	0.7861	0.7786	0.7789	0.7811	0.7815	0.7687	0.7777	0.7674	0.7717	0.7586	0.7662	0.7653
cardio	0.8100	0.5867	0.8351	0.9184	0.7236	0.5735	0.8165	0.9348	0.9503	0.5691	0.5494	0.5736	0.6725	0.9191	0.9317	0.9233
glass	0.8412	0.8726	0.7388	0.7497	0.8507	0.8644	0.7900	0.6323	0.6747	0.7950	0.7869	0.7988	0.7313	0.6748	0.6794	0.67427
ionosphere	0.8971	0.8730	0.5614	0.8541	0.9267	0.8753	0.9556	0.8419	0.7962	0.9247	0.9289	0.9212	0.9302	0.8373	0.8407	0.83754
letter	0.7830	0.8660	0.5926	0.6401	0.8765	0.8593	0.8074	0.6118	0.5283	0.8782	0.9077	0.8644	0.8370	0.6076	0.6066	0.61004
lympho	0.9673	0.9752	0.9956	0.9928	0.9745	0.9770	0.9104	0.9758	0.9846	0.9109	0.9411	0.9238	0.9647	0.9643	0.9782	0.92982
mnist	0.8404	0.7204	0.5741	0.8067	0.8481	0.7160	0.8666	0.8528	0.8526	0.7815	0.7770	0.7895	0.8053	0.8147	0.8397	0.82144
musk	1	0.5262	0.9999	0.9998	0.7985	0.5286	0.9999	1	0.9999	0.1844	0.2141	0.1783	0.7948	0.8491	0.9560	0.89403
optdigits	0.7692	0.4433	0.8732	0.7060	0.3707	0.4500	0.3979	0.4997	0.5085	0.4667	0.4695	0.4339	0.3692	0.4597	0.4121	0.48013
pendigits	0.8930	0.4595	0.9238	0.9496	0.7486	0.4697	0.8343	0.9303	0.9352	0.6877	0.6676	0.6913	0.6625	0.9240	0.9378	0.91931
pima	0.6578	0.6234	0.6999	0.6779	0.7078	0.6270	0.6752	0.6215	0.6481	0.6793	0.6832	0.6908	0.5682	0.6541	0.6446	0.65366
satellite	0.7494	0.5571	0.7581	0.6937	0.6836	0.5572	0.8030	0.6622	0.5988	0.5713	0.5829	0.5776	0.3894	0.5290	0.5792	0.49618
satimage-2	0.9992	0.4570	0.9804	0.9938	0.9536	0.4577	0.9959	0.9978	0.9821	0.8189	0.8107	0.8321	0.4744	0.8907	0.9546	0.78554
shuttle	0.6272	0.4723	0.9854	0.9971	0.6537	0.5263	0.9903	0.9917	0.9898	0.6234	0.6149	0.6234	0.7511	0.9888	0.9895	0.98764
vertebral	0.4330	0.4165	0.3262	0.3927	0.3816	0.4081	0.3985	0.4430	0.4026	0.4261	0.3692	0.4122	0.3425	0.4089	0.4067	0.41868
vowels	0.9222	0.9425	0.6726	0.7596	0.9680	0.9409	0.8076	0.7802	0.6026	0.9605	0.9517	0.9629	0.8608	0.6740	0.7045	0.66398
wbc	0.9200	0.9325	0.9516	0.9307	0.9366	0.9348	0.9210	0.9318	0.9158	0.9047	0.9186	0.9183	0.8821	0.9483	0.9420	0.94284

TABLE II
ROC AUC

Data	CBLOF	FB	HBOS	IForest	KNN	LOF	MCD	OCSVM	PCA	ABOD	KABOD	EKABOD	DL	ABDL	KABDL	EKABDL
arrhythmia	0.4538	0.4229	0.5110	0.4999	0.4463	0.4334	0.3995	0.4614	0.4612	0.3807	0.4390	0.3786	0.4311	0.4026	0.4258	0.4209
cardio	0.4296	0.1690	0.4476	0.4918	0.3322	0.1540	0.4156	0.5011	0.6090	0.2374	0.2219	0.2476	0.2928	0.4723	0.5284	0.5063
glass	0.0726	0.1476	0	0.0726	0.0726	0.1476	0	0.1726	0.0726	0.1702	0.0869	0.0869	0.1869	0.0726	0.0726	0.0392
ionosphere	0.7748	0.7055	0.3295	0.6474	0.8602	0.7063	0.8806	0.7000	0.5728	0.8441	0.8601	0.8480	0.8035	0.6707	0.6564	0.6694
letter	0.2396	0.3641	0.0715	0.0882	0.3311	0.3641	0.1932	0.1509	0.0874	0.3800	0.4253	0.3433	0.3083	0.1363	0.1243	0.1127
lympho	0.7516	0.7516	0.8466	0.8766	0.7516	0.7516	0.5183	0.7516	0.7516	0.4483	0.7516	0.6683	0.7433	0.7183	0.7516	0.1900
mnist	0.4023	0.3298	0.1188	0.3034	0.4204	0.3342	0.3462	0.3961	0.3846	0.3555	0.3618	0.3774	0.3686	0.3494	0.3793	0.3658
musk	1	0.2229	0.9783	0.9806	0.2733	0.1695	0.9888	1	0.9799	0.0507	0.0807	0.0478	0.2781	0.1872	0.4242	0.2267
optdigits	0	0.0244	0.2194	0.0271	0	0.0233	0	0	0	0.0060	0.0073	0.0060	0.0075	0.0046	0	0
pendigits	0.2397	0.0657	0.2979	0.3550	0.0984	0.0652	0.0892	0.3286	0.3186	0.0812	0.0574	0.0775	0.1041	0.2719	0.2775	0.2469
pima	0.4837	0.4480	0.5423	0.5023	0.5413	0.4555	0.4962	0.4703	0.4942	0.5192	0.5200	0.5296	0.4080	0.5069	0.4866	0.5035
satellite	0.5797	0.3901	0.5690	0.5576	0.4994	0.3892	0.6845	0.5345	0.4784	0.3902	0.4061	0.3995	0.2296	0.3968	0.4696	0.3393
satimage-2	0.9375	0.0555	0.6939	0.8775	0.3808	0.0555	0.6481	0.9355	0.8040	0.2130	0.2761	0.2570	0.0119	0.1502	0.4485	0.0090
shuttle	0.2944	0.0697	0.9551	0.9508	0.2183	0.1423	0.7500	0.9541	0.9501	0.1977	0.1900	0.1978	0.2980	0.9395	0.9500	0.9052
vertebral	0.0338	0.0643	0.0071	0.0533	0.0238	0.0505	0.0142	0.0238	0.0226	0.0600	0.0329	0.0600	0.0100	0	0	0.0071
vowels	0.3642	0.3224	0.1297	0.1960	0.5092	0.3550	0.2186	0.2790	0.1363	0.5710	0.5746	0.6173	0.3769	0.1187	0.1708	0.1236
wbc	0.4806	0.5187	0.5816	0.5087	0.4951	0.5187	0.4577	0.5124	0.4767	0.3060	0.4472	0.4304	0.5628	0.6465	0.5995	0.5882

TABLE III
RANK @N SCORE

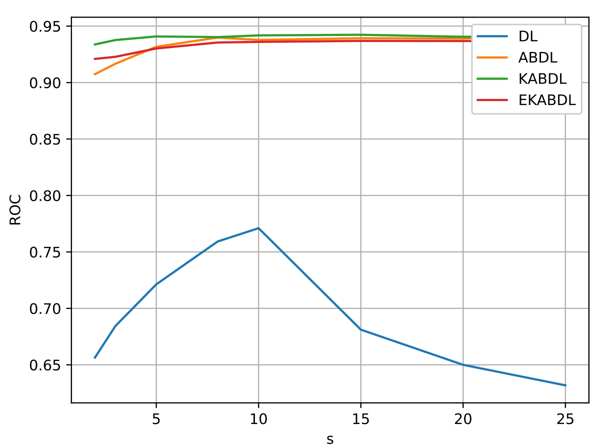

Fig. 2. ROC AUC evolution with sparsity

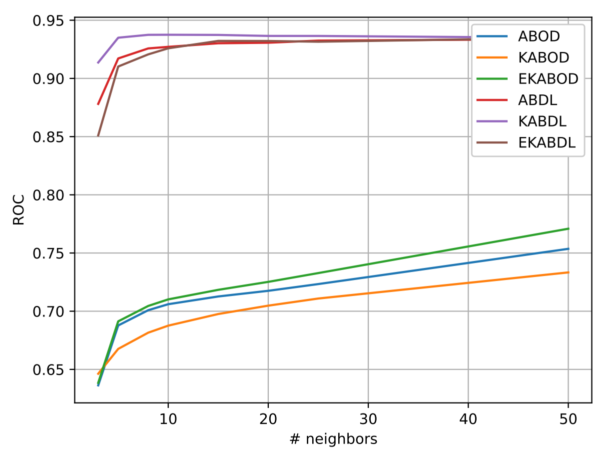

Fig. 3. ROC AUC evolution over # neighbors

V. CONCLUSIONS

This paper presents a novel algorithm for anomaly detection. It is based on a combination of the Angle-Based Outlier Detection technique and the sparse basis of the inliers obtained through Dictionary Learning. The proposed method is accompanied by two kernel versions, a classic and a lifted one. The results show that the algorithms perform well in terms of ROC AUC and precision rank score. Compared to the original ABOD method, the ROC AUC performance was improved by 7%, but for some datasets the improvement can be significant, even up to 70%. On the PyOD benchmark, the algorithm achieved competitive rankings with the state-of-the-art.

REFERENCES

[1] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pages 40–44. IEEE, 1993.

[2] B. Dumitrescu and P. Irofti. *Dictionary learning algorithms and applications*. Springer, 2018.

[3] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[4] R. Rubinstein, M. Zibulevsky, and M. Elad. Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. Technical report, Computer Science Department, Technion, 2008.

[5] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *arXiv preprint arXiv:1901.01588*, 2019.

[6] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[7] 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.

[8] 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.

[9] Peter J Rousseeuw and Katrien Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

[10] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 444–452, 2008.

[11] Songqiao Han, Xiyang Hu, Hailiang Huang, Mingqi Jiang, and Yue Zhao. Adbench: Anomaly detection benchmark. *arXiv preprint arXiv:2206.09426*, 2022.

[12] Denis C Ilie-Ablachim and Bogdan Dumitrescu. Anomaly detection with selective dictionary learning. In *2022 8th International Conference on Control, Decision and Information Technologies (CoDIT)*, volume 1, pages 255–260. IEEE, 2022.

[13] Ninh Pham and Rasmus Pagh. A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 877–885, 2012.

[14] Ninh Pham. L1-depth revisited: A robust angle-based outlier factor in high-dimensional space. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 105–121. Springer, 2018.