# Anomaly Detection with Selective Dictionary Learning

Denis C. ILIE-ABLACHIM

*Faculty of Automatic Control and Computers*
*University Politehnica of Bucharest*
denis.ilie_ablachim@upb.ro

Bogdan DUMITRESCU

*Faculty of Automatic Control and Computers*
*University Politehnica of Bucharest*
bogdan.dumitrescu@upb.ro

***Abstract*—In this paper we present new methods of anomaly detection based on Dictionary Learning (DL) and Kernel Dictionary Learning (KDL). The main contribution consists in the adaption of known DL and KDL algorithms in the form of unsupervised methods, used for outlier detection. We propose a reduced kernel version (RKDL), which is useful for problems with large data sets, due to the large kernel matrix. We also improve the DL and RKDL methods by the use of a random selection of signals, which aims to eliminate the outliers from the training procedure. All our algorithms are introduced in an anomaly detection toolbox and are compared to standard benchmark results.**

## I. Introduction

Dictionary Learning (DL) is a representation learning method which aims to find a sparse representation for a set of signals $\boldsymbol{Y}$, represented as a matrix with $N$ columns (signals) of size $m$. The representation is achieved by computing a dictionary $\boldsymbol{D}$ of size $m \times n$ and a sparse representation $\boldsymbol{X}$ of size $n \times N$ such that a good approximation $\boldsymbol{Y} \approx \boldsymbol{DX}$ is obtained. Most applications with dictionary learning are in problems with image denoising, inpainting, signal reconstruction, clustering or classification.

In this paper we present novel methods for unsupervised learning, in particular outlier detection, using DL. The main idea is based on finding a suited dictionary, which is capable of well representing most signals in a dataset, while the outlier signals representation should obtain large errors. Considering the number of outliers significantly lower than the rest of the signals, we expect the dictionary optimization to generally follow the directions of the normal signals. Our developments cover both the standard and the nonlinear (kernel) DL.

Anomaly detection (outlier detection) is the identification of a subset of signals that have a different representation in relation to the rest of the data. There are several successful anomaly detection methods, such as Isolation Forest (IForest) [1], Minimum Covariance Determinant (MCD) [2], [3], One-class SVM detector (OCSVM) or Principal Component Analysis (PCA) Outlier Detector [4].

There are also several successful sparse coding algorithms used for anomaly detection. An idea was presented in [5], [6]. These methods consider the data representation as a joint sparse linear combination of training data. By following this technique, the authors try to achieve a direct correlation between all the available signals. Naturally, non-correlated signals are considered as being anomalies. Another example is given in [7], where the anomalies are identified in terms of deviation from a trained model. This method tries to achieve good sparse representation for jointly distributed signals, while the other independent signals should be isolated. An overview of DL can be found in [8].

The paper is organized as follows. Section II introduces a natural way of solving outlier detection problems using DL algorithms. Section III formulates a new DL algorithm, called Selective Dictionary Learning, which aims to improve the anomaly detection algorithm by randomly selecting signals for the training procedure in order to discourage dictionary adaptation to outliers. In Section IV we present a reduced kernel version of the DL problem and its Selective form. Section V contains the experimental results, obtained by running tests on multivariate data and comparing the results with those of methods available in a Python toolkit for outlier detection.

## II. Anomaly Detection via Dictionary Learning

The DL problem is formulated as following

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

where $\|\cdot\|_0$ represents the 0-pseudo-norm and $s$ is the sparsity level.

The standard dictionary learning problem can be solved by using simple strategies. In order to overcome the nonconvexity and the huge dimension of the problem, the optimization procedure is organized in two steps. This method is also known as DL by Alternate Optimization. In this way, the problem is divided in two subproblems: sparse coding and dictionary update. By alternating these two stages for a given number of iterations, the method can obtain good local solutions. An iteration consists of computing the sparse representation $\boldsymbol{X}$, while the dictionary $\boldsymbol{D}$ is fixed, and then successively updating the dictionary columns, named atoms, while the sparse representation is fixed. For sparse coding we use Orthogonal Matching Pursuit (OMP) [9]. For the dictionary update we use the Approximate version of the K-SVD algorithm (AK-SVD)

This work was supported by a grant of the Romanian Ministry of Research, Innovation and Digitization, CCCDI - UEFISCDI, project number PN-III-P2-2.1-PED-2019-3248, within PNCDI III.

[10], [11], which optimizes the atoms and their representations successively.

A simple strategy for anomaly detection is to compute the representation error

$$\boldsymbol{E} = \boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X} \tag{2}$$

and identify the signals that obtain bad representations. The score of signal $i$ is simply the norm $\|\boldsymbol{e}_i\|$ of the $i$-th column of $\boldsymbol{E}$. The largest the error norm, the more likely that signal is an outlier. The underlying assumption is that signals that are alike can be better represented by the dictionary designed when solving (1). However, the dictionary size $n$ and the sparsity level $s$ must be taken smaller than usual, otherwise the representation may be uniformly good for all signals and even outliers can be well represented. A small dictionary favors good representations for signals that are similar, tuning the atoms for this purpose; a bad representation of the outliers has little effect on the objective of (1), since they are few. This trade-off is naturally obtained during the optimization.

Of course, since sparse representation is linear, similarity and dissimilarity can be thought in terms of direction. Normal signals belong to a small number of low dimensional subspaces and the outliers lie on very different subspaces. This is a model that is appropriate for some anomaly detection problems but not suited for others.

## III. Selective Dictionary Learning

In the standard DL algorithm, during the training procedure, both stages could be affected by the presence of outliers in the training dataset. The problem of anomaly detection can be solved more easily if we could train the dictionary only on normal data. By neglecting the outliers from the training dataset, we expect to obtain higher representation errors for anomalies. However, this is not possible, since we do not know which signals are normal and which are outliers.

To describe our strategy for eliminating most of the outliers from the training process, we introduce two new parameters in the DL algorithm: $train\_perc$ (*training percent*) and $train\_drop\_prec$ (*training dropout percent*). The first one represents the percent of data that are used during the sparse coding stage. At each iteration, we first apply a random sampling on the training data and only $train\_perc\%$ of the signals are used for sparse coding. In the dictionary update stage, we further drop off $train\_drop\_perc\%$ of the signals, namely those having the worst representations (largest representation errors). Although the first random selection can eliminate both normal signals and outliers from a training iteration, the representation of normal signals is less likely to suffer, since there are still signals in the current training set that are similar to them. On the contrary, outliers are more likely to lack good proxies and so their representation will worsen. The second random selection, that of signals with bad representation, aims to directly remove outliers from the training process. The dictionary will be updated to better represent the signals that already have good representations. Hence, again, the outliers representation will worsen, but the representation of normal signals not present in the current selection will not be significantly altered.

The DL problem can be formulated by the use of a zero extended permutation matrix $\boldsymbol{P}$ that is modified at each stage and has the role of randomly selecting the signals:

$$\begin{array}{ll} \min\limits_{\boldsymbol{D},\boldsymbol{X}} & \|\boldsymbol{Y}\boldsymbol{P} - \boldsymbol{D}\boldsymbol{X}\|_F^2 \\ \text{s.t.} & \|\boldsymbol{x}_\ell\|_0 \le s, \ell = 1:N \\ & \|\boldsymbol{d}_j\| = 1, j = 1:n. \end{array} \tag{3}$$

## IV. Reduced Kernel Dictionary Learning

Linear spaces can usually hinder good representations. In order to overcome this problem, the standard DL can easily be extended to a nonlinear space. This method is called Kernel Dictionary Learning (KDL) and was introduced in [12] and [13]. By this, we reproject each signal $\boldsymbol{y}$ to a nonlinear space $\phi(\boldsymbol{y})$, where $\phi(\cdot)$ is a nonlinear function. The dictionary $\boldsymbol{D}$ is also extended to $\phi(\boldsymbol{Y})\boldsymbol{A}$, where $\boldsymbol{A}$ is a matrix with unknown coefficients, taking the role of dictionary. The KDL problem is formulated as

$$\begin{array}{ll} \min\limits_{\boldsymbol{A},\boldsymbol{X}} & \|\varphi(\boldsymbol{Y}) - \varphi(\boldsymbol{Y})\boldsymbol{A}\boldsymbol{X}\|_F^2 \\ \text{s.t.} & \|\boldsymbol{x}_\ell\|_0 \le s, \ell = 1:N \\ & \|\varphi(\boldsymbol{Y})\boldsymbol{a}_j\| = 1, j = 1:n. \end{array} \tag{4}$$

The KDL problem can be solved similarly to the DL problem (1) if Mercer kernels are used, which allows the substitution of a scalar product of feature vectors $\varphi(\boldsymbol{x})^\top\varphi(\boldsymbol{y})$ with a kernel function $k(\boldsymbol{x},\boldsymbol{y})$. However, the problem becomes difficult when using large datasets, due to the large kernel matrix $\varphi(\boldsymbol{Y})^\top\varphi(\boldsymbol{Y})$ that results. The size of the kernel matrix scales linearly with the volume of the data, which leads to a large volume of memory. Thus this strategy might not be tractable for problems with large datasets.

In order to overcome this limitation we extend the dictionary $\boldsymbol{D}$ to a smaller nonlinear space by $\varphi(\bar{\boldsymbol{Y}})\boldsymbol{A}$, where $\bar{\boldsymbol{Y}}$ represents a small batch of signals from the original dataset. Permuting the signals such that $\boldsymbol{Y} = [\bar{\boldsymbol{Y}}\ \tilde{\boldsymbol{Y}}]$, we can write

$$\varphi(\bar{\boldsymbol{Y}}) = \underbrace{[\varphi(\bar{\boldsymbol{Y}})\quad \varphi(\tilde{\boldsymbol{Y}})]}_{\boldsymbol{\varphi(Y)}} \underbrace{\begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{0} \end{bmatrix}}_{\boldsymbol{P}}. \tag{5}$$

The KDL problem becomes

$$\begin{array}{ll} \min\limits_{\boldsymbol{A},\boldsymbol{X}} & \|\varphi(\boldsymbol{Y}) - \varphi(\bar{\boldsymbol{Y}})\boldsymbol{A}\boldsymbol{X}\|_F^2 \\ \text{s.t.} & \|\boldsymbol{x}_\ell\|_0 \le s, \ell = 1:N \\ & \left\|\varphi(\bar{\boldsymbol{Y}})\boldsymbol{a}_j\right\| = 1, j = 1:n. \end{array} \tag{6}$$

From (5) and (6) we obtain a new optimization problem

$$\min_{\boldsymbol{A},\boldsymbol{X}} \quad \|\varphi(\boldsymbol{Y})(\boldsymbol{I} - \boldsymbol{P}\boldsymbol{A}\boldsymbol{X})\|_F^2. \tag{7}$$

We denote

$$\boldsymbol{E} = \boldsymbol{I} - \boldsymbol{P}\boldsymbol{A}\boldsymbol{X} \tag{8}$$

the representation error and

$$\boldsymbol{F} = \left[\boldsymbol{I} - \boldsymbol{P}\sum_{i\neq j}\boldsymbol{a}_i\boldsymbol{x}_i^T\right]_{I_j} \tag{9}$$

the representation error without the contribution of the current atom $\boldsymbol{a}_j$; by $I_j$ we denote the set of signal indices to whose representation $\boldsymbol{a}_j$ contributes. In order to solve the optimization problem (6), we update the current atom while the other atoms and the representation are fixed. Removing the index $j$ for a lighter notation, the atom update problem becomes

$$\min_{\boldsymbol{a}} \quad \left\|\varphi(\boldsymbol{Y})\left(\boldsymbol{F}-\boldsymbol{P}\boldsymbol{a}\boldsymbol{x}^\top\right)\right\|_F^2. \tag{10}$$

Using the trace form of the squared Frobenius norm, the objective function becomes

$$\begin{aligned}&\operatorname{Tr}\left[\left(\boldsymbol{F}^\top-\boldsymbol{x}\boldsymbol{a}^\top\boldsymbol{P}^\top\right)\varphi^\top(\boldsymbol{Y})\varphi(\boldsymbol{Y})\left(\boldsymbol{F}-\boldsymbol{P}\boldsymbol{a}\boldsymbol{x}^\top\right)\right]=\\&=\operatorname{Tr}\left[\boldsymbol{F}^\top\boldsymbol{K}\boldsymbol{F}\right]-2\boldsymbol{x}^\top\boldsymbol{F}^\top\boldsymbol{K}\boldsymbol{P}\boldsymbol{a}+\|\boldsymbol{x}\|^2\boldsymbol{a}^\top\boldsymbol{P}^\top\boldsymbol{K}\boldsymbol{P}\boldsymbol{a}.\end{aligned} \tag{11}$$

We compute the partial derivative of the objective function with respect to the current atom

$$\frac{\partial(\cdot)}{\partial\boldsymbol{a}}=2\|\boldsymbol{x}\|^2\underbrace{\boldsymbol{P}^\top\boldsymbol{K}\boldsymbol{P}}_{\bar{\boldsymbol{K}}}\boldsymbol{a}-2\underbrace{\boldsymbol{P}^\top\boldsymbol{K}}_{\hat{\boldsymbol{K}}^\top}\boldsymbol{F}\boldsymbol{x} \tag{12}$$

and so the optimal atom is

$$\boldsymbol{a}=\left(\|\boldsymbol{x}\|^2\bar{\boldsymbol{K}}\right)^{-1}\hat{\boldsymbol{K}}^\top\boldsymbol{F}\boldsymbol{x}. \tag{13}$$

The atom is normalized after each update; note that the normalizing factor is $\left(\boldsymbol{a}^\top\bar{\boldsymbol{K}}\boldsymbol{a}\right)^{\frac{1}{2}}$ in order to obtain $\|\boldsymbol{a}_j\|=1$, as required by the original DL problem.

We call Reduced Kernel Dictionary Learning using a Sampled kernel (RKDL-S) the method solving problem (6) and summarize its update step in Algorithm 1. The optimal representation from step 6 is computed by setting to zero the partial derivative of (11) with respect to $\boldsymbol{x}$. The sparse representation step, not listed here, is made using the Kernel OMP algorithm [13].

**Algorithm 1:** RKDL-S

**Data:** reduced kernel matrix $\bar{\boldsymbol{K}}\in\mathbb{R}^{p\times p}$
partial kernel matrix $\hat{\boldsymbol{K}}\in\mathbb{R}^{N\times p}$
current dictionary $\boldsymbol{A}\in\mathbb{R}^{N\times n}$
representation matrix $\boldsymbol{X}\in\mathbb{R}^{n\times N}$
**Result:** updated dictionary $\boldsymbol{A}$

1 Compute error $\boldsymbol{E}=\boldsymbol{I}-\boldsymbol{P}\boldsymbol{A}\boldsymbol{X}$
2 **for** $j=1$ **to** $n$ **do**
3 Modify error: $\boldsymbol{F}=\boldsymbol{E}_{\mathcal{I}_j}+\boldsymbol{P}\boldsymbol{a}_j\boldsymbol{X}_{j,\mathcal{I}_j}$
4 Update atom: $\boldsymbol{a}_j=\left(\|\boldsymbol{x}\|_2^2\bar{\boldsymbol{K}}\right)^{-1}\hat{\boldsymbol{K}}^\top\boldsymbol{F}\boldsymbol{X}_{j,\mathcal{I}_j}$
5 Normalize atom: $\boldsymbol{a}_j\leftarrow\boldsymbol{a}_j/\left(\boldsymbol{a}_j^\top\bar{\boldsymbol{K}}\boldsymbol{a}_j\right)^{\frac{1}{2}}$
6 Update representation: $\boldsymbol{X}_{j,\mathcal{I}_j}^\top\leftarrow\boldsymbol{F}^\top\hat{\boldsymbol{K}}\boldsymbol{a}_j$
7 Recompute error: $\boldsymbol{E}_{\mathcal{I}_j}=\boldsymbol{F}-\boldsymbol{P}\boldsymbol{a}_j\boldsymbol{X}_{j,\mathcal{I}_j}$

RKDL-S achieves good results, but nevertheless in the training process there are chances to use abnormal signals, by the use of random sampling extraction. This fact can lead to a decrease in accuracy and performance. A better strategy that could overcome this problem would be to use a trained dictionary instead of $\bar{\boldsymbol{Y}}$ signals. This can be achieved by using a dictionary, denoted $\bar{\boldsymbol{D}}$, obtained from the linear cases in the previous sections. The corresponding optimization problem is

$$\begin{aligned}\min_{\boldsymbol{A},\boldsymbol{X}}\quad&\|\varphi(\boldsymbol{Y})-\varphi(\bar{\boldsymbol{D}})\boldsymbol{A}\boldsymbol{X}\|_F^2\\\text{s.t.}\quad&\|\boldsymbol{x}_\ell\|_0\le s,\ell=1:N\\&\left\|\varphi(\bar{\boldsymbol{D}})\boldsymbol{a}_j\right\|=1,j=1:n.\end{aligned} \tag{14}$$

We name it RKDL-D, the last letter indicating the use of dictionary instead of sampled signals. In order to update the current atom, we rewrite the new optimization problem as follows

$$\min_{\boldsymbol{a}_j}\left\|\varphi(\boldsymbol{Y})-\varphi(\bar{\boldsymbol{D}})\sum_{i\neq j}\boldsymbol{a}_i\boldsymbol{x}_i^\top-\varphi(\bar{\boldsymbol{D}})\boldsymbol{a}_j\boldsymbol{x}_j^\top\right\|_F^2. \tag{15}$$

Expressing the Frobenius norm via its trace form, (15) becomes

$$\begin{aligned}\min_{\boldsymbol{a}_j}\operatorname{Tr}\Bigg[&\left(\varphi^\top(\boldsymbol{Y})-\sum_{i\neq j}\boldsymbol{x}_i\boldsymbol{a}_i^\top\varphi^\top(\bar{\boldsymbol{D}})-\boldsymbol{x}_j\boldsymbol{a}_j^\top\varphi^\top(\bar{\boldsymbol{D}})\right)\\&\left(\varphi(\boldsymbol{Y})-\varphi(\bar{\boldsymbol{D}})\sum_{i\neq j}\boldsymbol{a}_i\boldsymbol{x}_i^\top-\varphi(\bar{\boldsymbol{D}})\boldsymbol{a}_j\boldsymbol{x}_j^\top\right)\Bigg].\end{aligned} \tag{16}$$

After the substitution of scalar products with the kernel function and negleting the terms that do not depend on $\boldsymbol{a}_j$ the final optimization problem is

$$\begin{aligned}\min_{\boldsymbol{a}_j}\operatorname{Tr}\Bigg[&2\sum_{i\neq j}\boldsymbol{x}_i\boldsymbol{a}_i^\top K(\bar{\boldsymbol{D}},\bar{\boldsymbol{D}})\boldsymbol{a}_j\boldsymbol{x}_j^\top+\boldsymbol{x}_j\boldsymbol{a}_j^\top\underbrace{K(\bar{\boldsymbol{D}},\bar{\boldsymbol{D}})}_{\bar{K}_{\bar{\boldsymbol{D}}}}\boldsymbol{a}_j\boldsymbol{x}_j^\top\\&-2\underbrace{K(\boldsymbol{Y},\bar{\boldsymbol{D}})}_{\hat{K}_{\bar{\boldsymbol{D}}}}\boldsymbol{a}_j\boldsymbol{x}_j^\top\Bigg].\end{aligned} \tag{17}$$

Algorithm 1 can be easily modified for solving (17), following the same line of reasoning as above. In particular, the atom update relation is

$$\boldsymbol{a}_j=\left(\|\boldsymbol{x}\|_2^2\bar{\boldsymbol{K}}_{\bar{\boldsymbol{D}}}\right)^{-1}(\hat{\boldsymbol{K}}_{\bar{\boldsymbol{D}}}^\top+\bar{\boldsymbol{K}}_{\bar{\boldsymbol{D}}}R)\boldsymbol{X}_j$$

and the representation update is

$$\boldsymbol{X}_j^\top\leftarrow(\hat{\boldsymbol{K}}_{\bar{\boldsymbol{D}}}-R\bar{\boldsymbol{K}}_{\bar{\boldsymbol{D}}})\boldsymbol{a}_j,$$

where we denoted $\bar{\boldsymbol{K}}_D$ the reduced kernel matrix $k(\bar{\boldsymbol{D}},\bar{\boldsymbol{D}})$, $\hat{\boldsymbol{K}}_D$ the partial kernel matrix $k(\boldsymbol{Y},\bar{\boldsymbol{D}})$ and $\boldsymbol{R}=\boldsymbol{X}^\top\boldsymbol{A}^\top-\boldsymbol{X}_j\boldsymbol{a}_j^\top$ the transposition representation product with respect to the current atom $\boldsymbol{a}_j$. The new method is summarized in Algorithm 2.

Following the same strategy presented in Section III, the RKDL methods can easily be adapted to their Selective form. The Selective Reduced Kernel Dictionary Learning (SRKDL) problem is solved as the previous one, by introducing two additional steps for the randomly selection of signals, one for the kernel OMP subproblem and the second one for the matrix coefficients update subproblem. In both cases the random sampling selection is made according to the entire data set (including the abnormal signals).

**Algorithm 2:** RKDL-D

**Data:** reduced kernel matrix $\bar{\boldsymbol{K}}_{\bar{\boldsymbol{D}}} \in \mathbb{R}^{p \times p}$
partial kernel matrix $\hat{\boldsymbol{K}}_{\bar{\boldsymbol{D}}} \in \mathbb{R}^{N \times p}$
current dictionary $\boldsymbol{A} \in \mathbb{R}^{N \times n}$
representation matrix $\boldsymbol{X} \in \mathbb{R}^{n \times N}$
**Result:** updated dictionary $\boldsymbol{A}$

Compute sum $\boldsymbol{S} = \sum_{i=1}^{n} \boldsymbol{X}_i^\top \boldsymbol{a}_i^\top$
**for** $j = 1$ **to** $n$ **do**
  Modify sum: $\boldsymbol{R} = \boldsymbol{S} - \boldsymbol{X}_j \boldsymbol{a}_j^\top$
  Update atom:
  $\boldsymbol{a}_j = \left(\|\boldsymbol{x}\|_2^2 \bar{\boldsymbol{K}}_{\bar{\boldsymbol{D}}}\right)^{-1} (\hat{\boldsymbol{K}}_{\bar{\boldsymbol{D}}}^\top + \bar{\boldsymbol{K}}_{\bar{\boldsymbol{D}}} R) \boldsymbol{X}_j$
  Normalize atom: $\boldsymbol{a}_j \leftarrow \boldsymbol{a}_j / \left(\boldsymbol{a}_j^\top \bar{\boldsymbol{K}}_{\bar{\boldsymbol{D}}} \boldsymbol{a}_j\right)^{\frac{1}{2}}$
  Update representation: $\boldsymbol{X}_j^\top \leftarrow (\hat{\boldsymbol{K}}_{\bar{\boldsymbol{D}}} - R \bar{\boldsymbol{K}}_{\bar{\boldsymbol{D}}}) \boldsymbol{a}_j$
  Recompute error: $\boldsymbol{S} = \boldsymbol{R} + \boldsymbol{X}_j \boldsymbol{a}_j^\top$

## V. Experiments

In this section we present the main results obtained with the proposed DL algorithms for anomaly detection. All algorithms have been developed in Python and have been introduced in the framework of the PyOD [14] anomaly detection toolbox. For the evaluation, all vectors of a dataset were normalized and were split into two sets: 60% for training and 40% for testing. Each experiment was repeated ten times independently with random splits. In terms of performance, we measure and compute the mean of the area under the receiver operating characteristic (ROC) curve and the precision @ rank n score. We used 16 real-world datasets from different domains, more precisely those gathered in ODDS (Outlier Detection DataSets)[1] and used as benchmark in PyOD, and 2 synthetic datasets.

All the algorithms were implemented in Python on a Desktop PC with Ubuntu 20.04 as operating system, having a processor of base frequency of 2.90 GHz (Max Turbo Frequency 4.80 GHz) and 80GB RAM memory (although a 16 GB RAM memory is sufficient). During the experiments, ten different rounds were run. The execution time, receiver operating characteristic value and precision n score were measured based on the average of all rounds. For the nonlinear versions we used two different kernels: radial basis function kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\gamma ||\boldsymbol{x} - \boldsymbol{y}||_2^2\right)$ and polynomial kernel $k(\boldsymbol{x}, \boldsymbol{y}) = (\gamma \boldsymbol{x}^\top \boldsymbol{y} + \alpha)^\beta$. The hyperparameters of the kernel functions were chosen according to a grid search. Based on the average results on all the datasets, they were set as following: $\gamma = 1/m$, $\alpha = 1$ and $\beta = 3$, for the synthetic datasets, while for the rest we used $\gamma = 0.1/m$ for the rbf kernel and $\gamma = 10/m$ for the polynomial kernel; we remind that $m$ is the size of a signal. All the implementations are available at https://github.com/denisilie94/pyod-dl, including the two synthetic datasets.

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

The first synthetic dataset was generated based on two different sparse coded sets of signals. Using two dictionaries, $D_i$, the dictionary for inliers, and $D_o$, the dictionary for outliers, two sets of signals were generated having the sparsity constraint $s = 4$. For the numerical experiment we set the number of inliers $N_i = 512$ and number of outliers $N_o = 64$, while the dictionary size are $n_i = 50$ and $n_o = 400$. The signals size was set to $m = 64$. For the outliers signals we used an overcomplete dictionary, since its representation ability is much more diverse than in the case of the dictionary for inliers. The second dataset consists of random samples from two normal (Gaussian) distributions, of different mean and standard deviation. We kept the same number of normal and abnormal signals of size 64 as in the previous dataset. The two Gaussian distributions were generated so that the distribution of normal signals clearly overlaps with the distribution of abnormal signals. More exactly the inlier mean and variance are $\mu_i = 0$ and $\sigma_i = 0.5$, while the outliers parameters are $\mu_o = -0.1$ and $\sigma_o = 0.45$.

For the DL methods we used small dictionaries of size $n = 50$, while the sparsity constraint was $s = 5$. All the dictionaries were trained using 20 iterations using the AK-SVD method. For the SDL method the $train\_perc = 0.7$ and $train\_drop\_perc = 0.4$. For the RKDL method, the size of the matrices $\bar{\boldsymbol{Y}}$ from (6) and $\bar{\boldsymbol{D}}$ from (14) was set to $10\%$ of the number of signals. The selective version of RKDL used the parameters $train\_perc = 0.8$ and $train\_drop\_perc = 0.3$.

The results show the good behaviour of our algorithms in detecting outliers via sparse coding. In terms of performance, the DL methods obtain competitive results. The main results are summarized in Tables I, III for the public PyOD methods and in Tables II, IV for the DL methods. In all the tables we highlight the best three results from both sets of methods (PyOD and DL) taken together. For the synthetic datasets, we noticed that the PyOD methods do not obtain good results. The DL methods obtain better classification results for the dataset generated with sparse coding and the dataset with Gaussian distribution. For ODDS datasets, the overall results are predominantly better for PyOD methods. However, there are a few datasets where DL methods stand out as being better. For example, for the *cardio* dataset, the DL methods achieve the third place in top, while for the *ionosphere* and *satellite* datasets it occupies the second and third place. An interesting dataset is *vertebral* where the DL methods are the best, occupying all three positions of the top.

In general, the SDL method achieve better results than the DL method, but this is not always true. Depending on how the random selection of signals is made, there are chances that abnormal signals to be used during the training procedure. This is possible for the datasets with a very high percentage of outliers or small datasets. The same statement is valid for the KDL vs SKDL comparison. On the other hand, comparing the standard methods with the kernel methods, we notice that the second ones obtain better results. Moreover, the selective strategy improves the invariance of dictionaries to representing abnormal signals. The RKDL-D and SRKDL-D methods often

| Data | Samples | Dim. | Out. Perc. | ABOD | CBLOF | FB | HBOS | IForest | KNN | LOF | MCD | OCSVM | PCA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| dl_out | 576 | 64 | 11.1111 | 0.84496 | 0.52271 | 0.58823 | 0.48706 | 0.50225 | 0.56253 | 0.59419 | **0.8734** | 0.5106 | 0.49162 |
| 2gauss_out | 576 | 64 | 11.1111 | 0.00633 | 0 | 0.19077 | 0.47946 | 0.28293 | 0 | 0.26359 | 0.00015 | 0.36793 | 0.54262 |
| arrhythmia | 452 | 274 | 14.6018 | 0.76875 | 0.78382 | 0.77807 | **0.82193** | **0.7996** | **0.7861** | 0.77866 | 0.77897 | 0.78116 | 0.7815 |
| cardio | 1831 | 21 | 9.6122 | 0.56917 | 0.81003 | 0.58673 | 0.8351 | 0.91844 | 0.72363 | 0.57357 | 0.82715 | **0.93484** | **0.95038** |
| glass | 214 | 9 | 4.2056 | 0.79507 | 0.84125 | **0.87261** | 0.73887 | 0.74977 | **0.85076** | **0.8644** | 0.79006 | 0.63236 | 0.6747 |
| ionosphere | 351 | 33 | 35.8974 | 0.92476 | 0.89718 | 0.87304 | 0.56144 | 0.85411 | 0.92674 | 0.87535 | **0.95566** | 0.84192 | 0.7962 |
| letter | 1600 | 32 | 6.25 | **0.87825** | 0.78306 | **0.86605** | 0.59268 | 0.64011 | **0.87656** | 0.85935 | 0.8074 | 0.61182 | 0.5283 |
| lympho | 148 | 18 | 4.0541 | 0.91097 | 0.96731 | 0.97528 | **0.99569** | **0.99288** | 0.9745 | 0.97709 | 0.91125 | 0.97587 | **0.98467** |
| mnist | 7603 | 100 | 9.2069 | 0.78153 | 0.84041 | 0.72046 | 0.57419 | 0.80673 | 0.84813 | 0.71608 | **0.86661** | **0.85289** | **0.85266** |
| musk | 3062 | 166 | 3.1679 | 0.18444 | **1** | 0.52626 | **0.99998** | 0.99984 | 0.79857 | 0.52867 | 0.99997 | **1** | 0.99995 |
| optdigits | 5216 | 64 | 2.8758 | 0.46674 | **0.7692** | 0.44336 | **0.87325** | **0.70608** | 0.37076 | 0.45004 | 0.3979 | 0.49972 | 0.50856 |
| pendigits | 6870 | 16 | 2.2707 | 0.68776 | 0.89307 | 0.45953 | 0.92381 | **0.94964** | 0.74865 | 0.46975 | 0.83439 | **0.93031** | **0.93525** |
| pima | 768 | 8 | 34.8958 | **0.67938** | 0.65781 | 0.62345 | **0.69995** | 0.67798 | **0.70781** | 0.62705 | 0.67528 | 0.6215 | 0.64811 |
| satellite | 6435 | 36 | 31.6395 | 0.57137 | 0.74942 | 0.55717 | 0.75811 | 0.6937 | 0.68364 | 0.55727 | **0.80304** | 0.66224 | 0.59884 |
| satimage-2 | 5803 | 36 | 1.2235 | 0.81896 | **0.99922** | 0.45701 | 0.98042 | **0.99384** | 0.9536 | 0.45774 | **0.99593** | 0.9978 | 0.98218 |
| vertebral | 240 | 6 | 12.5 | 0.42615 | 0.43309 | 0.41658 | 0.32625 | 0.39276 | 0.38166 | 0.40811 | 0.39158 | 0.44308 | 0.40269 |
| vowels | 1456 | 12 | 3.4341 | **0.96059** | 0.92221 | **0.94252** | 0.67267 | 0.75966 | **0.968** | 0.94096 | 0.80761 | 0.78021 | 0.60267 |
| wbc | 378 | 30 | 5.5556 | 0.90473 | 0.92005 | 0.93254 | **0.95163** | 0.93073 | **0.93662** | **0.93488** | 0.92102 | 0.93189 | 0.91587 |

TABLE I
ROC PERFORMANCE - PYOD METHODS

| Data | DL | SDL | RKDL-S | | RKDL-D | | SRKDL-S | | SRKDL-D | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | rbf | poly | rbf | poly | rbf | poly | rbf | poly |
| dl_out | **0.89666** | **0.85336** | 0.36259 | 0.36424 | 0.39653 | 0.37279 | 0.32207 | 0.34155 | 0.3849 | 0.36028 |
| 2gauss_out | **0.91278** | **0.90274** | 0.05688 | 0.02432 | 0.0165 | 0.01193 | **1** | 0.82143 | 0.00569 | 0.0036 |
| arrhythmia | 0.77057 | 0.77194 | 0.68557 | 0.63535 | 0.70723 | 0.76356 | 0.72746 | 0.72333 | 0.72112 | 0.78032 |
| cardio | 0.70023 | 0.72884 | 0.63584 | 0.92797 | 0.60367 | 0.82092 | 0.69322 | **0.93747** | 0.63524 | 0.89624 |
| glass | 0.67484 | 0.64841 | 0.77257 | 0.65649 | 0.80463 | 0.68711 | 0.79033 | 0.68143 | 0.62142 | 0.72519 |
| ionosphere | **0.93401** | **0.93923** | 0.87097 | 0.62494 | 0.88993 | 0.80313 | 0.85409 | 0.60376 | 0.8578 | 0.77142 |
| letter | 0.82366 | 0.81978 | 0.72686 | 0.33016 | 0.74923 | 0.41022 | 0.72863 | 0.3328 | 0.71789 | 0.42632 |
| lympho | 0.91635 | 0.93615 | 0.54727 | 0.711 | 0.7846 | 0.95522 | 0.60362 | 0.82363 | 0.82788 | 0.9102 |
| mnist | 0.81029 | 0.79723 | 0.66476 | 0.80017 | 0.54424 | 0.57414 | 0.6443 | 0.71678 | 0.5917 | 0.57642 |
| musk | 0.88574 | 0.89346 | 0.69204 | 0.75747 | 0.55197 | 0.70492 | 0.77949 | 0.9375 | 0.70581 | 0.82653 |
| optdigits | 0.40227 | 0.4084 | 0.36615 | 0.41572 | 0.40347 | 0.56437 | 0.39185 | 0.43933 | 0.48264 | 0.61547 |
| pendigits | 0.62745 | 0.63382 | 0.8356 | 0.91127 | 0.78032 | 0.88282 | 0.83539 | 0.92798 | 0.84152 | 0.91862 |
| pima | 0.56365 | 0.55959 | 0.60599 | 0.64188 | 0.6131 | 0.65217 | 0.62009 | 0.6357 | 0.63752 | 0.65449 |
| satellite | 0.65351 | 0.64655 | 0.65338 | 0.66762 | 0.64236 | 0.59866 | **0.77197** | 0.68252 | **0.77682** | 0.69671 |
| satimage-2 | 0.59438 | 0.55493 | 0.85817 | 0.97076 | 0.90414 | 0.92801 | 0.98003 | 0.96482 | 0.98159 | 0.9646 |
| vertebral | **0.48265** | 0.46904 | 0.39465 | 0.41046 | **0.48752** | 0.39953 | 0.46184 | 0.40483 | **0.50182** | 0.38861 |
| vowels | 0.77689 | 0.80236 | 0.80882 | 0.519 | 0.78853 | 0.66121 | 0.86525 | 0.52215 | 0.84647 | 0.678 |
| wbc | 0.81705 | 0.84737 | 0.67586 | 0.88776 | 0.71915 | 0.9167 | 0.81133 | 0.89594 | 0.73305 | 0.90848 |

TABLE II
ROC PERFORMANCE - DL METHODS

| Data | Samples | Dim. | Out. Perc. | ABOD | CBLOF | FB | HBOS | IForest | KNN | LOF | MCD | OCSVM | PCA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| dl_out | 576 | 64 | 11.1111 | **0.47533** | 0.11883 | 0.1958 | 0.10082 | 0.07764 | 0.15684 | 0.19525 | 0.47469 | 0.11109 | 0.0999 |
| 2gauss_out | 576 | 64 | 11.1111 | 0 | 0 | 0 | 0.10901 | 0.03221 | 0 | 0.004 | 0 | 0.02137 | 0.10618 |
| arrhythmia | 452 | 274 | 14.6018 | 0.38076 | 0.45385 | 0.42297 | **0.51108** | **0.49992** | 0.44637 | 0.43343 | 0.39952 | **0.4614** | 0.46129 |
| cardio | 1831 | 21 | 9.6122 | 0.23743 | 0.42966 | 0.169 | 0.44761 | 0.49186 | 0.33227 | 0.15409 | 0.42084 | 0.50112 | **0.609** |
| glass | 214 | 9 | 4.2056 | **0.17023** | 0.07262 | **0.14762** | 0 | 0.07262 | 0.07262 | 0.14762 | 0 | **0.17262** | 0.07262 |
| ionosphere | 351 | 33 | 35.8974 | **0.84415** | 0.77489 | 0.70558 | 0.32951 | 0.64743 | **0.86021** | 0.70634 | **0.88065** | 0.70005 | 0.57286 |
| letter | 1600 | 32 | 6.25 | **0.38009** | 0.23969 | **0.36419** | 0.07152 | 0.08828 | 0.33117 | **0.36411** | 0.19327 | 0.15096 | 0.08747 |
| lympho | 148 | 18 | 4.0541 | 0.44834 | 0.75167 | 0.75167 | **0.84667** | **0.87667** | **0.75167** | 0.75167 | 0.56833 | 0.75167 | 0.75167 |
| mnist | 7603 | 100 | 9.2069 | 0.3555 | **0.40231** | 0.32986 | 0.11882 | 0.30346 | **0.42043** | 0.33429 | 0.3462 | **0.39619** | 0.38461 |
| musk | 3062 | 166 | 3.1679 | 0.05075 | **1** | 0.22297 | 0.97832 | **0.98069** | 0.2733 | 0.16955 | 0.98889 | **1** | 0.97994 |
| optdigits | 5216 | 64 | 2.8758 | 0.00602 | 0 | 0.02445 | **0.2194** | 0.0271 | 0 | 0.02335 | 0 | 0 | 0 |
| pendigits | 6870 | 16 | 2.2707 | 0.08125 | 0.23974 | 0.06579 | 0.29793 | **0.35505** | 0.09844 | 0.06529 | 0.08928 | **0.32866** | **0.31865** |
| pima | 768 | 8 | 34.8958 | **0.51929** | 0.48378 | 0.44802 | **0.54238** | 0.50233 | **0.54133** | 0.45552 | 0.49625 | 0.47035 | 0.49429 |
| satellite | 6435 | 36 | 31.6395 | 0.39023 | 0.57978 | 0.39016 | 0.56903 | 0.55766 | 0.49945 | 0.38929 | **0.68452** | 0.53455 | 0.47844 |
| satimage-2 | 5803 | 36 | 1.2235 | 0.21305 | **0.93759** | 0.0555 | 0.6939 | **0.8775** | 0.38087 | 0.05551 | 0.64813 | **0.93556** | 0.80408 |
| vertebral | 240 | 6 | 12.5 | 0.06005 | 0.03381 | 0.06439 | 0.00714 | 0.05337 | 0.02381 | 0.05059 | 0 | 0.02381 | 0.02262 |
| vowels | 1456 | 12 | 3.4341 | **0.57102** | **0.36427** | 0.3224 | 0.12974 | 0.19602 | **0.50929** | 0.35506 | 0.2186 | 0.27907 | 0.13636 |
| wbc | 378 | 30 | 5.5556 | 0.30604 | 0.48064 | 0.51879 | **0.58166** | 0.50879 | 0.49518 | 0.51879 | 0.45771 | 0.51249 | 0.47673 |

TABLE III
PRECISION @ N PERFORMANCES - PYOD METHODS

| Data | DL | SDL | RKDL-S | | RKDL-D | | SRKDL-S | | SRKDL-D | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | rbf | poly | rbf | poly | rbf | poly | rbf | poly |
| dl_out | **0.54116** | **0.53812** | 0.01536 | 0.02597 | 0.01429 | 0.02937 | 0.00385 | 0.01629 | 0.01087 | 0.01447 |
| 2gauss_out | **0.51047** | 0.49212 | 0.00333 | 0 | 0 | 0 | **1** | **0.78568** | 0 | 0 |
| arrhythmia | 0.42828 | 0.42783 | 0.32057 | 0.31022 | 0.35562 | 0.42643 | 0.38116 | 0.37517 | 0.37499 | 0.45478 |
| cardio | 0.30468 | 0.30867 | 0.22022 | **0.5416** | 0.19782 | 0.35779 | 0.24374 | **0.58545** | 0.1814 | 0.50126 |
| glass | 0.1369 | 0.04762 | 0.12262 | 0.09762 | 0.11429 | 0.07262 | 0.14762 | 0.03929 | 0.125 | 0.09762 |
| ionosphere | 0.8081 | 0.81393 | 0.73023 | 0.43417 | 0.78044 | 0.61622 | 0.70715 | 0.42508 | 0.72037 | 0.58191 |
| letter | 0.27955 | 0.26227 | 0.1753 | 0.01572 | 0.2086 | 0.03195 | 0.15485 | 0.02421 | 0.16185 | 0.02761 |
| lympho | 0.49833 | 0.45666 | 0.125 | 0.12833 | 0.25833 | 0.42833 | 0.22333 | 0.265 | 0.26667 | 0.465 |
| mnist | 0.37567 | 0.36151 | 0.23029 | 0.35623 | 0.13266 | 0.12771 | 0.18196 | 0.28904 | 0.16703 | 0.13247 |
| musk | 0.4075 | 0.37901 | 0.12873 | 0.23353 | 0.10472 | 0.17828 | 0.21692 | 0.68521 | 0.26918 | 0.0882 |
| optdigits | 0.00963 | 0.00696 | 0.0111 | 0 | 0.02303 | **0.04896** | 0.00474 | 0 | 0.0437 | **0.1143** |
| pendigits | 0.09973 | 0.09568 | 0.16584 | 0.28128 | 0.10907 | 0.27757 | 0.1476 | 0.27513 | 0.16232 | 0.25404 |
| pima | 0.41686 | 0.39683 | 0.43692 | 0.4744 | 0.44025 | 0.48196 | 0.45952 | 0.48992 | 0.4757 | 0.49287 |
| satellite | 0.46056 | 0.45341 | 0.47054 | 0.51423 | 0.45615 | 0.4217 | **0.61256** | 0.54732 | **0.62188** | 0.55516 |
| satimage-2 | 0.07119 | 0.04584 | 0.09132 | 0.64813 | 0.33605 | 0.38399 | 0.58768 | 0.64744 | 0.56207 | 0.62129 |
| vertebral | **0.09294** | 0.07273 | 0.06198 | 0.05927 | **0.10508** | 0.02143 | 0.05048 | 0.03214 | **0.07903** | 0.00667 |
| vowels | 0.28151 | 0.30329 | 0.22516 | 0.10201 | 0.21783 | 0.18607 | 0.25828 | 0.06196 | 0.22259 | 0.1181 |
| wbc | 0.41909 | 0.36207 | 0.18477 | **0.55319** | 0.24709 | 0.52414 | 0.27334 | **0.5301** | 0.15626 | 0.51664 |

TABLE IV
PRECISION @ N PERFORMANCES - DL METHODS

improve the results. In general, the trained dictionary, $\boldsymbol{D}$, is better adapted for the representation of the normal signals. However, it is likely that the trained dictionaries contain atoms that are beneficial in the nonlinear representation of all signals, including the outliers.

The execution time of DL methods is usually larger than that of the PyOD methods. For example, for the *musk* dataset, which is among the largest, DL and SDL take about 6 seconds, i.e., not much more than MCD, which needs about 4 seconds; RKDL algorithms take between 9 and 11 seconds, while SRKDL variant are slightly faster, with 7-10 seconds. The other PyOD algorithms are at least 10 times faster than the methods presented in the article.

## VI. CONCLUSIONS

In this paper we have presented a novel unsupervised method for outlier detection, based on Dictionary Learning and Kernel Dictionary Learning. We have introduced a reduced kernel DL version that is suitable for problems with large datasets. The kernel reduction technique is based on choosing a small sample of signals from the original dataset, which will further be used for the nonlinear extension. Another way to represent the kernel is to use a dictionary initially trained in with the standard DL algorithm. Both methods are accompanied by improved versions based on a random selection of the data used in the training procedure. This ensures invariance in the representation of normal signals, while the capabilities of the dictionaries for the representation of abnormal signals decrease.

Based on these results, we demonstrated that sparse learning can easily isolate the outliers from the normal signals, while obtaining competitive results with other unsupervised methods. All the developed algorithms were introduced in an outlier detection toolbox.

## REFERENCES


[1] 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.

[2] 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.

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

[4] 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.

[5] Nam H Nguyen, Nasser M Nasrabadi, and Trac D Tran. Robust multi-sensor classification via joint sparse representation. In *14th International Conference on Information Fusion*, pages 1–8. IEEE, 2011.

[6] Sumit Shekhar, Vishal M Patel, Nasser M Nasrabadi, and Rama Chellappa. Joint sparsity-based robust multimodal biometrics recognition. In *European Conference on Computer Vision*, pages 365–374. Springer, 2012.

[7] Amir Adler, Michael Elad, Yacov Hel-Or, and Ehud Rivlin. Sparse coding with anomaly detection. *Journal of Signal Processing Systems*, 79(2):179–188, 2015.

[8] Bogdan Dumitrescu and Paul Irofti. *Dictionary learning algorithms and applications*. Springer, 2018.

[9] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy 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.

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

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

[12] Jayaraman J Thiagarajan, Karthikeyan Natesan Ramamurthy, and Andreas Spanias. Multiple kernel sparse representations for supervised and unsupervised learning. *IEEE Transactions on Image Processing*, 23(7):2905–2915, 2014.

[13] Hien Van Nguyen, Vishal M Patel, Nasser M Nasrabadi, and Rama Chellappa. Design of non-linear kernel dictionaries for object recognition. *IEEE Transactions on Image Processing*, 22(12):5123–5135, 2013.

[14] Yue Zhao, Zain Nasrullah, and Zheng Li. PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019.