Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version 6 Sept, 2023.

*Digital Object Identifier 10.1109/XXXX.2022.1234567*

# Sparse representation with Gaussian atoms and its use in anomaly detection

**Denis C. Ilie-Ablachim[1], Andra Băltoiu[1], and Bogdan Dumitrescu[1]**

[1]Faculty of Automatic Control and Computers, National University of Science and Technology Politehnica of Bucharest, Romania

Corresponding author: Bogdan Dumitrescu (email: bogdan.dumitrescu@upb.ro).

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. It has also been funded by the Ministry of Investments and European Projects through the Human Capital Sectoral Operational Program 2014-2020, Contract no. 62461/03.06.2022, SMIS code 153735.

**ABSTRACT** We propose sparse representation and dictionary learning algorithms for dictionaries whose atoms are characterized by Gaussian probability distributions around some central atoms. This extension of the space covered by the atoms permits a better characterization of localized features of the signals. The representations are computed by trading-off representation error and the probability of the actual atoms used in the representation. We propose two types of algorithms: a greedy one, similar in spirit with Orthogonal Matching Pursuit, and one based on L1-regularization. We apply our new algorithms to unsupervised anomaly detection, where the representation error is used as anomaly score. The flexibility of our approach appears to improve more the representations of the many normal signals than those of the few outliers, at least for an anomaly type called dependency, thus improving the detection quality. Comparison with both standard dictionary learning algorithms and established anomaly detection methods is favorable.

**INDEX TERMS** dictionary learning, sparse representations, anomaly detection, normal distribution

## I. INTRODUCTION

Sparse coding is a technique used in signal processing and machine learning for representing signals using a set of features, named atoms. Each signal is represented as a linear combination of a small number of features that are relevant to the underlying structure. Sparse coding methods are employed in many applications such as image denoising, compression, signal reconstruction, clustering, or classification. There are many methods for solving sparse coding problems; remarkable representatives are Orthogonal Matching Pursuit (OMP) [1] and Least Absolute Shrinkage and Selection Operator (LASSO) [2]; other important methods can be found in [3].

Anomaly detection (AD) is the process of identifying patterns or data points that deviate significantly from expected behavior in a given dataset. Anomalies, also known as outliers, are observations that do not conform to the normal behavior or representation of the data. AD is applied to different types of signals, such as images (medical, satellite, etc.), time series data (server traffic, meteorology), or graphs (financial transactions, utility networks). The main requirement is identifying unusual representations that may indicate errors, defects, or fraud in the data. There are several well-designed algorithms for problems involving anomaly detection, such as Histogram-based Outlier Detection (HBOS) [4], Local Outlier Factor (LOF) [5], Connectivity-based Outlier factor (COF) [6], Subspace Outlier Degree (SOD) [7], and k-Nearest Neighbors (KNN) [8]. Other AD methods are reviewed in [9] and in [10]. The latter work, called ADBench, takes a practical approach, based on own implementations and extensive comparisons. Besides the AD methods listed above, it implements also some recent ones, based on deep networks [11], Gaussian mixture models [12], copula [13], empirical cumulative distribution functions [14].

Sparse coding and dictionary learning (DL) have been previously used for anomaly detection on numerous types of signals: target detection in hyperspectral images was performed in [15], by distinguishing it from the normal background; structural defects in solid media were identified in [16] with the aid of two distinct dictionaries (for local and global features respectively) in the representation of the ultrasonic wavefield response; identifying anomalies in telemetry time series data has been approached with DL in [17]; fault detection was cast as AD in water networks in [18].

The DL approach works in both supervised and unsupervised learning. In this paper, our focus is on unsupervised AD. The setup is very simple. A basic set of atoms, named dictionary, is trained with a DL algorithm, using all (unlabeled) data. The representation error is used as anomaly score. The rationale behind this method is that, since normal signals outnumber the anomalies and are more alike, the dictionary will be naturally trained to better represent them in order to minimize the overall representation error. On the contrary, the anomalies will have poorer representations.

In [19], a selective procedure is used, during the training stage of the representation basis, to attempt avoiding to include outliers when training the dictionary. In [20] a new formulation for the dictionary problem is presented with the main applicability in anomaly detection; the authors introduce a row sparsity regularization such that outliers are represented in a different subspace from the inliers. Other methods of sparse coding for anomaly detection are presented in [21] and [22] involving the detection of irregular heartbeats in ECG. The key idea presented in [22], [23] is the extension of the atom notion to an infinite set. The atoms are now represented as cones, offering the possibility of dynamic actual atoms around some central atoms. We refer generically to the associated algorithms as Cone-DL. Good results were obtained in an anomaly detection problem. Moreover, recent work has shown that sparse coding and DL methods are appropriate for performing anomaly detection in a wide range of applications, through a series of tests performed on an extensive anomaly detection benchmark [23].

*Our contribution.* In this paper, we present a probabilistic representation method suitable for applications involving outlier detection. The fundamental strategy is to see the atoms of a dictionary $\boldsymbol{D}$ as centers of Gaussian distributions. We find sparse representations using atoms near the centers by combined optimization of the representation error and the probability of the actual atoms. The motivation of our proposal is the following:

- Flexibility of the representation, provided by the extension of the space covered by the set of atoms that replaces the fixed-atom used in the standard approach.
- For this reason, the Gaussian atoms are effective for capturing gradual variations and localized features in data, usually associated with normal signals much more than with anomalies.
- In anomaly detection, we expect that the representation error of normal signals is improved much more than for anomalies, with respect to the usual DL setting. For normal signals, it is much easier than for anomalies to find better representations with atoms nearby the central ones (hence with high probability). For anomalies, a similar reduction cannot be achieved due to the low probability of the atoms suited for the representation.

The problem may be seen as a weighted total least squares (TLS) one [24], with the extra constraint of sparsity; however, it is posed and solved differently than other sparse TLS approaches [25]–[27]. We propose an algorithm in OMP style that greedily selects central atoms based on the probability of the residual to belong to the respective distributions; then, for the obtained support, we propose a coordinate descent algorithm for finding the representation that is optimal for the combined error-probability objective. Then, a DL algorithm is built on top of this Gauss version of OMP. The same problems are also solved by a L1 regularization approach, with algorithms that are both simple and effective.

The paper is organized as follows. In section II we present the general formulation of our proposed problem. Section III includes the method of computing the Gaussian atoms when the support is known. Section IV presents a natural way for finding the nearest Gaussian atom which should be used during the support update. Moreover, this procedure is necessary for the Gaussian version of OMP, which is presented in full. Section V is dedicated to the L1 regularization algorithm for sparse representation with Gaussian atoms. DL algorithms are presented in Section VI; the atom update rule is simply averaging. In section VII, we give the results of our methods in outlier detection tasks, using the ADBench [10] framework. For a specific type of anomalies, which pose a challenge to several other detection methods, we obtain better results than the state of the art.

## II. PROBLEM FORMULATION

The core of our proposal is to replace each atom of the dictionary $\boldsymbol{D} \in \mathbb{R}^{m \times n}$ with an infinite set of vectors characterized by a probability distribution. More precisely, instead of an atom $\boldsymbol{d}_i \in \mathbb{R}^m$ (column $i$ of the dictionary), we propose to use vectors $\boldsymbol{x}_i$ from the Gaussian distribution $\mathcal{G}_i(\boldsymbol{d}_i, \sigma_i)$ defined by the probability density function

$$p\left(\boldsymbol{x}_i; \boldsymbol{d}_i, \sigma_i^2\right) = \frac{1}{\sigma_i^m (2\pi)^{m/2}} \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{d}_i\|^2}{2\sigma_i^2}\right). \quad (1)$$

Here, $\boldsymbol{d}_i$ is the mean of the distribution and $\sigma_i$ is the standard deviation. Since the distribution is multivariate, we could use a covariance matrix different from $\sigma_i^2 \boldsymbol{I}$, but we prefer to start with the simplest configuration.

The sparse representation problem is reformulated such that both representation error and probability are taken into account. Given a signal $\boldsymbol{y} \in \mathbb{R}^m$, we seek a representation

$$\boldsymbol{y} \approx \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i, \quad (2)$$

where only $s$ coefficients $\alpha_i$ are nonzero and the atom $\boldsymbol{x}_i$ is associated with the Gaussian distribution corresponding to $\boldsymbol{d}_i$. We name $\boldsymbol{d}_i$ central atom and $\boldsymbol{x}_i$ actual atom.

Besides the approximation error in (2), we characterize the actual atoms $\boldsymbol{x}_i$ by their joint probability of belonging to $\mathcal{G}_i(\boldsymbol{d}_i, \sigma_i)$ defined by (1):

$$p(\boldsymbol{x}_i \in \mathcal{G}_i(\boldsymbol{d}_i, \sigma_i), i = 1:n) = \prod_{\alpha_i \neq 0} p\left(\boldsymbol{x}_i; \boldsymbol{d}_i, \sigma_i^2\right) \tag{3}$$

Since we will seek to maximize or to bound this probability, it is more convenient to work with the negative log-likelihood

$$-\log\left(p(\boldsymbol{x}_i \in \mathcal{G}_i(\boldsymbol{d}_i, \sigma_i), i = 1:n)\right) = \\ = \sum_{\alpha_i \neq 0} \left( \frac{\|\boldsymbol{x}_i - \boldsymbol{d}_i\|^2}{2\sigma_i^2} + m \log\left(\sigma_i \sqrt{2\pi}\right) \right). \tag{4}$$

In a broad sense, our goal is to compute a sparse approximation (2) with small error and high probability. More precisely, to make the problem more amenable to optimization, we propose an objective that trades off error and probability:

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}_i \in \mathbb{R}^m, \boldsymbol{\alpha} \in \mathbb{R}^n} & \sum\limits_{\alpha_i \neq 0} \frac{1}{\sigma_i^2} \|\boldsymbol{x}_i - \boldsymbol{d}_i\|^2 + \lambda \|\boldsymbol{y} - \sum\limits_{i=1}^{n} \alpha_i \boldsymbol{x}_i\|^2 \\ \text{s.t.} & \|\boldsymbol{x}_i\| = 1, \ i = 1:n \\ & \|\boldsymbol{\alpha}\|_0 = s \end{array} \tag{5}$$

where $\|\boldsymbol{\alpha}\|_0$ is the number of nonzero elements in the coefficients vector $\boldsymbol{\alpha}$ and $\lambda > 0$ is a trade-off factor.

## III. REPRESENTATION PROBLEM WITH KNOWN SUPPORT

We start by solving the problem with fixed support. For the sake of simplicity, we assume that the $s$ central atoms (the means of the Gaussian distributions described by (1)) are $\boldsymbol{d}_i$, $i = 1:s$; they are normalized: $\|\boldsymbol{d}_i\| = 1$. Let $\boldsymbol{y} \in \mathbb{R}^m$ be the signal to be represented. The problem (5) becomes

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}_i \in \mathbb{R}^m, \alpha_i \in \mathbb{R}} & \sum\limits_{i=1}^{s} \frac{1}{\sigma_i^2} \|\boldsymbol{x}_i - \boldsymbol{d}_i\|^2 + \lambda \|\boldsymbol{y} - \sum\limits_{i=1}^{s} \alpha_i \boldsymbol{x}_i\|^2 \\ \text{s.t.} & \|\boldsymbol{x}_i\| = 1, \ i = 1:s \end{array} \tag{6}$$

This is a weighted TLS [24] problem. We attempt to solve (6) by successive optimization (block coordinate descent), initialized with $\boldsymbol{x}_i = \boldsymbol{d}_i$. The coefficients $\alpha_i$ are initialized by solving the least-squares problem

$$\min_{\alpha_i} \|\boldsymbol{y} - \sum_{i=1}^{s} \alpha_i \boldsymbol{d}_i\|. \tag{7}$$

Assume that all variables are fixed, excepting $\boldsymbol{x}_1$; for convenience, we remove the index 1 from all variables and constants. The objective of (6) becomes

$$f(\boldsymbol{x}, \alpha) = \frac{1}{\sigma^2} \|\boldsymbol{x} - \boldsymbol{d}\|^2 + \lambda \|\tilde{\boldsymbol{y}} - \alpha \boldsymbol{x}\|^2 \tag{8}$$

where

$$\tilde{\boldsymbol{y}} = \boldsymbol{y} - \sum_{i=2}^{s} \alpha_i \boldsymbol{x}_i. \tag{9}$$

**Proposition 1:** The function

$$h(\boldsymbol{x}) = \min_{\alpha} f(\boldsymbol{x}, \alpha) \tag{10}$$

is unimodal on the chord defined by the intersection of the hypersphere $\|\boldsymbol{x}\| = 1$ with the plane defined by $\boldsymbol{d}$ and $\tilde{\boldsymbol{y}}$, between $\boldsymbol{d}$ and $\tilde{\boldsymbol{y}}$.

*Proof.* See the appendix. ■

The function (8) is convex in each of the variables $\boldsymbol{x}$ and $\alpha$ (but not in both). If $\boldsymbol{x}$ is fixed, then the optimal value of $\alpha$ is

$$\alpha = \frac{\boldsymbol{x}^T \tilde{\boldsymbol{y}}}{\|\boldsymbol{x}\|^2}. \tag{11}$$

Replacing in (8) we get

$$h(\boldsymbol{x}) = \frac{1}{\sigma^2} \|\boldsymbol{x} - \boldsymbol{d}\|^2 + \lambda \left\| \tilde{\boldsymbol{y}} - \frac{\boldsymbol{x}^T \tilde{\boldsymbol{y}}}{\|\boldsymbol{x}\|^2} \boldsymbol{x} \right\|^2. \tag{12}$$

Prop. 1 says that the minimum of (12) is between $\boldsymbol{d}$ and $\tilde{\boldsymbol{y}}$, at the point where moving towards $\boldsymbol{d}$ or $\tilde{\boldsymbol{y}}$ increases the objective. So, essentially, we have to search for solutions of the form

$$\boldsymbol{x} = \gamma \boldsymbol{d} + (1 - \gamma) \tilde{\boldsymbol{y}}, \tag{13}$$

with $\gamma \in [0, 1]$. Normalization is necessary and also alignment, i.e., a change of sign of $\boldsymbol{d}$ or $\tilde{\boldsymbol{y}}$, if their scalar product is negative.

So, we find the minimum of (12) by line search on the chord joining $\boldsymbol{d}$ and $\tilde{\boldsymbol{y}}/\|\tilde{\boldsymbol{y}}\|$ on the hypersphere, thus enforcing the constraint $\|\boldsymbol{x}\| = 1$. Several techniques can be used; we have chosen golden section, which is simple and robust, although not the fastest. The above described algorithm for minimizing (12) is named Atom_update for further reference.

Returning now to the coordinate descent algorithm for solving (6), of which minimizing (12) is just a step, we note that once $\boldsymbol{x}_1$ and $\alpha_1$ have been updated, the coefficients $\alpha_i$ are no longer optimal. Their optimal value for the current $\boldsymbol{x}_i$ can be found by solving the least squares problem similar to (7):

$$\min_{\alpha_i} \|\boldsymbol{y} - \sum_{i=1}^{s} \alpha_i \boldsymbol{x}_i\|. \tag{14}$$

We can solve it after the update of each $\boldsymbol{x}_i$, noting that a fast algorithm is possible since we update only the $i$-th column of the matrix involved in the least squares problem. Thus, we update an orthogonal triangularization.

*Remark.* The algorithm described above converges. Due to Prop. 1, the minimum of (12) can be computed with good approximation by the golden section algorithm. So, each coordinate descent step indeed decreases the objective of (6). The same happens when solving (14).

## IV. OMP WITH GAUSSIAN ATOMS

In this section, we present the OMP method adapted for Gaussian atoms, which solves problem (5).

The main difficulty is identifying the support of the representation, since once the support is available, we know to compute the actual atoms and their coefficients from the previous section. We assume that the central atoms $\boldsymbol{D}$ and the associated standard deviations are given.

In the spirit of the classical OMP, the atoms are chosen sequentially. In order to add a new atom to the support, we search for the central atom that is the nearest to the current residual. However, the distance is measured in a probabilistic sense, as the probability of the residual to belong to a distribution (1).

Let us assume that $k$ atoms have been selected and the current residual is $\tilde{\boldsymbol{r}} = \boldsymbol{y} - \sum_{i\in\mathcal{S}} \alpha_i \boldsymbol{x}_i$, where $\mathcal{S}$ is the current support. Since the central atoms can have directions that point away from the current residual, we adjust them based on the scalar product between the atoms and the residual, $\boldsymbol{\varphi} = \text{sign}(\tilde{\boldsymbol{r}}^\top \boldsymbol{D})$. If a scalar product is negative, the atom is reversed, such that it is nearest to the residual (this implies also reversing the sign of the coefficient in the linear representation). The nearest central atom is identified by the highest probability (1) of the residual, which amounts to the minimum of (compare with (4))

$$\frac{\|\tilde{\boldsymbol{r}} - \varphi_i \boldsymbol{d}_i\|^2}{2\sigma_i^2} + m \log\left(\sigma_i \sqrt{2\pi}\right). \tag{15}$$

Once the central atom is found, the current support is updated and the method continues with the search for the actual atom belonging to the Gaussian distribution around the chosen central atom. This implies changing all the actual atoms of the support and is done as described in the previous section, using coordinate descent. A predetermined number $T$ of iterations is used, in which we sweep the current support and sequentially update the atoms and their coefficients; alternatively, one can iterate until atom changes fall below a given tolerance. We summarize the steps of the Gauss-OMP method in Algorithm 1. Since Atom_update is a coordinate descent algorithm, the representation error always decreases. Although Gauss-OMP converges, the chosen support is not necessarily optimal, hence attaining the minimum of (5) is not guaranteed.

The complexity of Gauss-OMP is $O(Tms^2 + smn)$. The operations in steps 10-13 are $O(m)$, although the hidden coefficient of $m$ can be large, due to the golden section search made by Atom_update. The three loops (steps 2, 7, 8) account for the $Ts^2$ part of the complexity. The $smn$ part of the complexity is due to steps 4 and 5. So, considering $T$ a constant, the complexity is similar to that of OMP, which is $O(ms(n+s)+s^3)$ [28]. However, Gauss-OMP is much slower due to Atom_update.

**Algorithm 1:** Gauss-OMP

```
Data: central atoms $d_j \in \mathbb{R}^m$ with standard
      deviations $\sigma_j \in \mathbb{R}$; ($d_j$ and $\sigma_j$ define Gaussian
      atoms $\mathcal{G}_j(d_j, \sigma_j)$, $j = 1:n$)
      vector $y \in \mathbb{R}^m$
      sparsity level $s$
      number of search iterations $T$
Result: support $\mathcal{S} \subset 1:n$ of sparse representation
      actual atoms $x_j \in \mathbb{R}^m$, $j \in \mathcal{S}$
      representation coefficients $\alpha_j \in \mathbb{R}$, $j \in \mathcal{S}$
1  Initialize $\tilde{r} = y$, $\mathcal{S} = \emptyset$, $X = D$
2  for $k = 1$ to $s$ do
3      Normalize residual: $\tilde{r} = \tilde{r}/\|\tilde{r}\|$
4      Compute projection orientations: $\varphi = \text{sign}(r^\top D)$
5      Decide next index based on probabilities:
         $j = \text{argmin}_i \|\tilde{r} - \varphi_i d_i\|^2/(2\sigma_i^2) + m\log\left(\sigma_i\sqrt{2\pi}\right)$
6      Increase support: $\mathcal{S} \leftarrow \mathcal{S} \cup \{j\}$
7      for $t = 1$ to $T$ do
8          for $i = 1$ to $k$ do
9              Index in full dictionary $j = \mathcal{S}(i)$
10             Remove current atom from residual:
                 $\tilde{y} \leftarrow \tilde{r} + x_j \alpha_j$
11             Update atom:
                 $x_j = \text{Atom\_update}(d_j, \sigma_j, \tilde{y}, \lambda, \varepsilon)$
12             Update $\alpha_j$ with (11) (with $x = x_j$)
13             Update residual: $\tilde{r} \leftarrow \tilde{y} - x_j \alpha_j$
```

## V. CONVEX RELAXATION

Using a standard L1 relaxation, we replace the objective of (5) with

$$\sum_{i=1}^{n} \frac{1}{\sigma_i^2} \|\boldsymbol{x}_i - \boldsymbol{d}_i\|^2 + \lambda \|\boldsymbol{y} - \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i\|^2 + \gamma \|\boldsymbol{\alpha}\|_1 \tag{16}$$

where $\gamma > 0$ is a constant. Of course, the constraints $\|\boldsymbol{x}_i\| = 1$, $i = 1:n$, need to be enforced.

We adopt a block coordinate descent approach and optimize successively the actual atoms $\boldsymbol{x}_j$ and then immediately the corresponding representation coefficient $\alpha_j$.

Assume first that all variables are fixed except $\boldsymbol{x}_j$. Denote $\hat{\boldsymbol{X}}$ the matrix having the vectors $\boldsymbol{x}_i$, $i \neq j$, as columns. Similarly, $\hat{\boldsymbol{\alpha}}$ is the vector $\boldsymbol{\alpha}$ from which $\alpha_j$ was removed. Denoting $\hat{\boldsymbol{y}} = \boldsymbol{y} - \hat{\boldsymbol{X}}\hat{\boldsymbol{\alpha}}$, the terms of (16) that depend on $\boldsymbol{x}_j$ are

$$f(\boldsymbol{x}_j) = \frac{1}{\sigma_j^2}\|\boldsymbol{x}_j - \boldsymbol{d}_j\|^2 + \lambda\|\hat{\boldsymbol{y}} - \alpha_j \boldsymbol{x}_j\|^2$$

By setting the gradient of $f$ to zero, we obtain the new atom

$$\boldsymbol{x}_j = \frac{\sigma_j^2}{\lambda \alpha_j^2 \sigma_j^2 + 1}\left(\alpha_j \hat{\boldsymbol{y}} + \frac{1}{\sigma_j^2}\boldsymbol{d}_j\right). \tag{17}$$

We proceed to update coefficient $\alpha_j$, assuming all other variables are fixed. The function to be optimized is

$$g(\alpha_j) = \lambda\|\hat{\boldsymbol{y}} - \alpha_j \boldsymbol{x}_j\|^2 + \gamma|\alpha_j| \tag{18}$$

This is a convex function whose minimum is given by

$$\alpha_j = \text{soft\_thr}\left(\boldsymbol{x}_j^T \hat{\boldsymbol{y}}, \frac{\gamma}{2\lambda}\right), \tag{19}$$

where soft_thr is the soft thresholding operator.

Algorithm 2 implements the above approach. The initial representations are computed by running the FISTA algorithm [29] with the dictionary of central atoms. This gives a

**Algorithm 2:** Gauss-L1

```
Data: central atoms $d_j \in \mathbb{R}^m$ with standard
      deviations $\sigma_j \in \mathbb{R}$, $j = 1 : n$
      vector $y \in \mathbb{R}^m$
      number of iterations $n_{\text{it}}$
      parameters $\lambda$, $\gamma$ from (16)
      number of FISTA iterations $n_{\text{FISTA}}$
      step parameter $t$ for FISTA
Result: actual atoms $x_i \in \mathbb{R}^m$, $i = 1 : n$
        representation coefficients $\alpha \in \mathbb{R}^m$
1  Initialize $X = D$, $\alpha = 0$
2  Compute initial representations:
     $\alpha = \text{FISTA}(y, X, \alpha, \gamma/\lambda, n_{\text{FISTA}}, t)$
3  for $i = 1$ to $n_{\text{it}}$ do
4      Compute residual: $r = y - X\alpha$
5      for $j$ = random permutation of $1 : n$ do
6          Remove current atom from residual:
             $\hat{y} \leftarrow r + \alpha_j x_j$
7          Update actual atom $x_j$ with (17)
8          Update coefficient $\alpha_j$ with (19)
9          Update residual: $r \leftarrow \hat{y} - \alpha_j x_j$
10     Normalize atoms (columns of $X$)
```

good start of the algorithm. The atoms are updated in random order in each iteration, to prevent stalling. The residual is efficiently computed by updating it with the current atom; full recomputation is made only in each master iteration. Otherwise, the algorithm is simple: formulas (17) and (19) are repeatedly used. Note that normalization is made only at the end of an iteration. Besides the data and parameters from (16), the other input parameters are related to FISTA: the number of iterations is $n_{\text{FISTA}}$, which should not be too large, since an approximate solution is sufficient; the step size $t$ is specific to proximal gradient methods and ideally should be the inverse of the largest eigenvalue of $X^T X$; finally, the regularization constant is $\gamma/\lambda$, as visible from (16).

The complexity of Gauss-L1 is $O(n_{\text{it}} mn)$. Each individual atom update (steps 6–9 of Algorithm 2) requires $O(m)$ operations. So, while the complexity per iteration is light, the number of iterations $n_{\text{it}}$ is an important factor that is hard to quantify. The experiments in Section VII suggest that 100 iterations are more than enough for convergence.

It is worth mentioning that the algorithm does not guarantee the decrease of the objective. Although (17) and (19) decrease the objective, the normalization in each iteration may increase it. However, the results shown in Section VII suggest that each iteration decreases the objective.

## VI. DL WITH GAUSSIAN ATOMS

We aim now to solve the DL problem associated with dictionaries of Gaussian atoms. Given a matrix $Y \in \mathbb{R}^{m \times N}$, whose columns are the training signals, we want to find the dictionary $D$ of central atoms such that the sum over all signals of objective functions like those in (5) or (16) is minimum. We assume that the standard deviations $\sigma_i$ are given.

**Algorithm 3:** DL-Gauss (-OMP or -L1)

```
Data: training signals $Y \in \mathbb{R}^{m \times N}$
      initial central atoms $D \in \mathbb{R}^{m \times n}$
      standard deviations $\sigma_j \in \mathbb{R}$, $j = 1 : n$
      number of iterations $n_{\text{it}}$
Result: trained dictionary $D \in \mathbb{R}^{m \times n}$
1  for $i = 1$ to $n_{\text{it}}$ do
2      Initialize $\tilde{D} = 0_{m \times N}$
3      for $\ell = 1$ to $N$ do
4          Compute representation $\alpha$ and actual atoms
             $X$ for $y_\ell$ with Gauss-OMP or Gauss-L1
5          for $j = 1$ to $n$, $\alpha_j \neq 0$ do
6              $\tilde{d}_j \leftarrow \tilde{d}_j + x_j$
7      Normalize columns of $\tilde{D}$
8      $D \leftarrow \tilde{D}$
```

We adopt the standard technique of alternating sparse representation steps (using Algorithms 1 or 2) and dictionary update steps. To see how the latter can be done, consider the representation of signal $y_\ell$, which is $\sum_{i=1}^{n} \alpha_{i\ell} x_{i\ell}$. We note that in both (5) and (16), the only term that directly depends on the central atoms is the first. In the DL context, it becomes

$$\sum_{\ell=1}^{N} \sum_{i=1}^{n} \frac{1}{\sigma_i^2} \|x_{i\ell} - d_i\|^2 = \sum_{i=1}^{n} \frac{1}{\sigma_i^2} \sum_{\ell, \alpha_{i\ell} \neq 0} \|x_{i\ell} - d_i\|^2. \quad (20)$$

The second sum contains only the actual atoms effectively used in the representation. This is explicitly the case in (5). For (16), this is justified by the update formula (17): when $\alpha_j = 0$, it results that $x_j = d_j$; so, the terms for which $\alpha_{i\ell} = 0$ naturally disappear from (20).

The update rule for the central atoms is clear from (20). Once the actual atoms have been computed, the central atoms are computed with

$$d_i = \sum_{\ell, \alpha_{i\ell} \neq 0} x_{i\ell}, \quad (21)$$

followed by normalization. This formula is not only simple, but allows on the fly computation of the future central atoms, without the necessity of storing the actual atoms. Algorithm 3 presents the structure of DL for dictionaries with Gaussian atoms.

We note that the central atoms update (21) is a very light operation and hence the complexity of DL-Gauss is essentially given by the representation algorithm.

*Remark.* Regarding convergence, DL-Gauss shares the behavior of other DL algorithms. Although Gauss-OMP gives the optimal solution for the chosen support, in the context of DL-Gauss-OMP there is no guarantee that the supports that are chosen are better than those at the previous iteration. Also, although the update (21) of central atoms decreases the objective, there is no guarantee that the next computed representations have smaller error. Similar comments apply to DL-Gauss-L1. So, there is no guarantee that DL-Gauss decreases the objective of (5) or (16) at each iteration. However, like for most DL algorithms lacking such guarantees, the practical behavior, illustrated in the next section, is very good.

## VII. ANOMALY DETECTION EXPERIMENTS

In this section, we present the main results obtained with the proposed methods for dictionaries with Gaussian atoms. Our focus on anomaly detection is supported by the possibility of trade-off between representation error and likelihood, which should favor the many normal samples more than the few anomalies.

### A. Evaluation setup

The anomaly detection procedure is carried out in two stages. The first one is the training of a dictionary $\boldsymbol{D}$ capable of representing all samples of the dataset. The second stage represents the anomaly classifications, in which we identify the outliers as the samples that obtain the worst representation score.

*Anomaly scores.* Due to the combined nature of the objectives of (5) and (16), we can use for anomaly detection several scores related to a signal $\boldsymbol{y}$.

- The first choice is the error

$$S_e(\boldsymbol{y}) = \|\boldsymbol{y} - \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i\|. \tag{22}$$

  Error was used in several other DL applications to anomaly detection, like [17], [21], [30].
- Another candidate is the probability (1) of the representation or the corresponding negative log-likelihood

$$S_p(\boldsymbol{y}) = \sum_{\alpha_i \neq 0} \frac{1}{\sigma_i^2} \|\boldsymbol{x}_i - \boldsymbol{d}_i\|^2 + m \log(\sigma_i). \tag{23}$$

  The score (23) is especially suited for the OMP versions of Gaussian DL, as the atoms are chosen based on it in (15).
- Finally, the value of the objectives (5) or (16) can also be used.

*Competing methods.* We compare our algorithms to the 14 ADBench methods suitable for unsupervised anomaly; see [10] for the complete list; most of them are cited in Section I. ADBench appears to be the most comprehensive framework and benchmark for anomaly detection currently available. It is a continuator of Python Outlier Detection (PyOD) [31] and it contains the most used algorithms in anomaly detection tasks. Moreover, it contains a benchmark with datasets proposed as a reference.

Standard dictionary learning is performed using the Approximate version of the K-SVD algorithm [32] [33]; for the coefficients update during the learning phase, the OMP algorithm is used. The dictionaries were trained with 40 iterations. Standard DL was used for generating the dictionaries used in Gauss-OMP and Gauss-L1, but also for anomaly detection, as the baseline DL method; the anomaly scores are the representation errors given by OMP.

The Gauss-OMP and DL-Gauss-OMP algorithms have been developed in Python and are compatible with the open-source anomaly detection framework PyOD [31]. Gauss-L1 and DL-Gauss-L1 have been implemented in MATLAB; the Python version will be soon available. The code of our algorithms is available at https://asydil.upb.ro/software/.

*Performance indicators.* We classify the methods using two performance indicators:

- The ROC AUC (Receiver Operating Characteristic Area Under Curve) values for each dataset and their average over all datasets, based on the scores described above.
- The average *rank*, in terms of ROC AUC, of each of our methods in the context of the 14 ADBench methods, over all used datasets. The ROC AUC results of the 15 methods for each dataset are sorted; the best rank is 1 and the worst is 15; tied values are given equal, averaged ranks. The ranks obtained by each method are then averaged.

*Datasets.* ADBench drew attention on the influence of anomaly types on the performance of the detection methods and proposed four anomaly classes that differ with respect to their resemblance with normal samples: local, global, dependency and cluster anomalies; refer to [10] for a detailed description. The benchmark also provides a procedure for synthetically generating these types of anomalies starting from the normal signals in each dataset. We exclude local and cluster anomalies from our tests, as they are not suited for the error criterion that we use for distinguishing between the two types of samples. The global anomalies differ from the normal samples to such an extent that many of the benchmark methods perfectly detect them and we therefore exclude them as well. Dependency anomalies, which are signals with independent samples, unlike those of normal data, appear to be well suited for our methods.

Out of the 57 datasets in the benchmark, we perform the tests on 30 sets that span numerous domains of applications including healthcare, astronautics, linguistics, biology, chemistry and others. We generate the dependency variants of all 30 datasets using the software from [10]. The proportion of anomalies in the datasets ranges from 1% to 40%, however 60% of datasets contain less than 10% anomalies. The list of datasets and the associated ROC AUC results for both the ADBench methods and standard DL algorithm can be found in [23, Table C.7].

All the simulations were performed in the spirit of an unsupervised method. We split each dataset in two: 70% for training and 30% for testing, respecting the global proportion of anomalies in both sets (stratified sampling). The labels of the training set are not otherwise used. Z-score normalization was performed on all data. For each dataset, we ran 5 independent rounds, with random initializations. The ROC AUC values that we report are mean values over the 5 runs.

*Parameters.* Our methods require the use of some hyperparameters, which we explored with a partial grid search. In order to further broaden the representation accuracy between normal samples and anomalies, we impose that the samples be represented using a few dictionary atoms, i.e., set the sparsity level to a lower value than the usual DL setup. More precisely, we took $s \in \{2, 3\}$. We used dictionaries with overcompleteness factor $c = n/m = 2$. These choices are also supported by complexity reasons. An important hyperparameter is the trade-off factor $\lambda$ from (5) and (16); we took $\lambda \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$. The sparsity enforcing parameter from (16) was generally taken $\gamma = \lambda$. Finally, we adopted two sets of standard deviation values; in one, the value $\sigma = 0.05$ is associated with all Gaussian atoms; in the other, the standard deviations have uniformly random values in the interval $[0.01, 0.1]$.

The other parameters are set as follows. The number of refinement iterations in Gauss-OMP is $T = 10$. The parameters of FISTA in Gauss-L1 are $n_{\text{FISTA}} = 50$, $t = 0.0001$. The number of iterations is $n_{\text{it}} = 100$ for both Gauss-L1 and DL-Gauss-L1, but only 40 for DL-Gauss-OMP. The dictionary $\boldsymbol{D}$ used in Gauss-L1 is learned on the training data with 40 iterations of AK-SVD with a sparsity level $s = 2$ and random initialization. The same dictionary is used as initialization for DL-Gauss-L1.

## B. Results

*Convergence curves.* We start by illustrating the evolution of our methods during the optimization process. The behavior of Gauss-L1 is shown in Figures 1 and 2. The first shows the individual behavior for 100 randomly chosen test signals from the dataset *landsat*, while the second shows the evolution of the mean values and standard deviation intervals for all test signals. In both cases, the parameters are $\lambda = \gamma = 1$, $c = 2$, $\sigma \in [0.01, 0.1]$. From left to right, we see the error (22), the quadratic term of (23) (reflecting the distance between the actual and the central atoms), and the objective of the optimization problem (16). First of all, we note that the objective of the optimization problem decreases not only in average, but for all signals, although not necessarily by much, a sign that the initialization is good. The error has also a decreasing tendency, although more erratic (since not directly optimized). The distance has very small values for many signals, especially for the normal ones, which means that the actual and central atoms are nearly identical; however, for some signals, the distance increases and has relatively large values. Finally, we note that, in this case, the error and the objective value are larger for the anomalies. An important remark is that the decrease of the error and of the objective is larger for normal signals than for anomalies, showing that our approach indeed attains one of its goals. In particular, for a few normal signals whose error and objective are initially larger than those of some anomalies (initial values correspond to AK-SVD), our algorithm manages to decrease the error and the objective below those of all anomalies. So, there are signals misclassified by AK-SVD that are correctly classified by Gauss-L1. Figure 2 suggests that indeed, in this case, the error gives the best separation of normal and abnormal signals, while the distance would be a unreliable anomaly score. The behavior of DL-Gauss-L1 is similar. For other datasets the situation is not necessarily so neat, but the trends are similar.

The evolution of DL-Gauss-OMP is shown in Figure 3, on 500 randomly chosen signals from the dataset *cardio*, with $\lambda = 1000$, $c = 2$, $s = 2$, $\sigma = 0.05$. This time, the probability score (23) gives the best anomaly detection; since $\lambda$ is large, the error is very well optimized for all signals; the small number of anomalies in *cardio* makes it less costly to use actual atoms that are farther from the central ones, hence the good detection performance of the probability score. The values of the objective are generally decreasing, although individual curves (not shown) are not decreasing as nicely as for the L1 algorithm. We also note that, although at initialization (with AK-SVD), the normal and abnormal signals are not well separated (their average scores are about the same), our algorithm is able to separate them fairly well.

*Anomaly detection.* The rank and ROC AUC results of Gauss-OMP and DL-Gauss-OMP are presented in Table 1 and those of Gauss-L1 and DL-Gauss-L1 in Table 2; this is a selection containing the best results. In Table 1, the anomaly score that gives the best results varies; it is indicated in the first column as 'err' for (22), 'prob' for (23) and 'obj' for the objective of (5). For the L1 methods, the best score is always the error (22). The explanations of Figures 1–3 have a larger support here: indeed, for small $\lambda$ the error is the best score, while for large $\lambda$, likelihood is best. The optimization objective is also good in the midrange. Somewhat surprisingly (possibly due to our inability to find the right parameters), the OMP methods give poor results with small $\lambda$ and the L1 methods are bad for large $\lambda$. DL-Gauss-OMP gives usually a significant improvement over Gauss-OMP. The improvement of DL-Gauss-L1 over Gauss-L1 is rather marginal.

To put the numbers in context, here are some values for comparison. Whenever the rank is $\leq 2.50$, our methods have a better rank than all the 14 from ADBench. Extensive results of AK-SVD and Cone-DL methods can be found in [23]; we give here only the best. AK-SVD obtains the best rank of 1.93, while a Cone-DL method has a best of 1.67. So, DL-Gauss-OMP is at the level of AK-SVD, while DL-Gauss-L1 narrowly beats Cone-DL.

The situation is more clear for ROC AUC. The best method from ADBench, COF, has 0.9274. AK-SVD has a

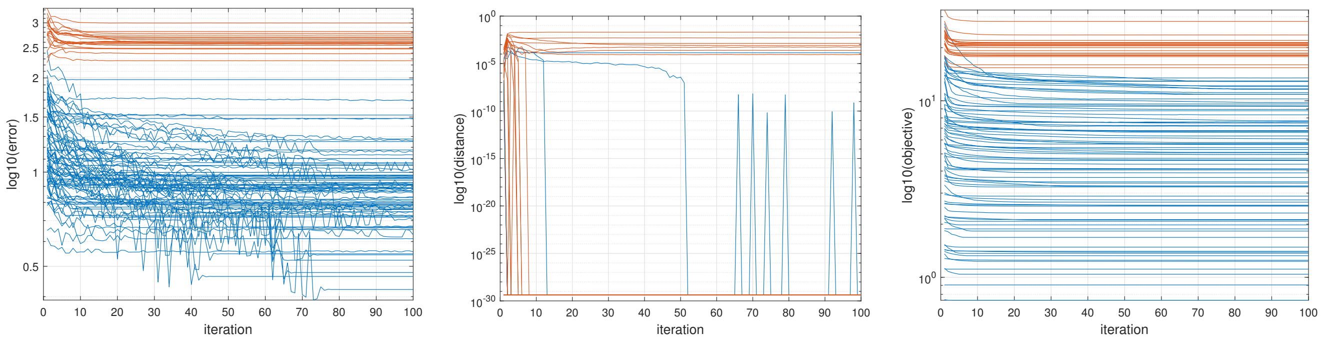

**FIGURE 1.** **Gauss-L1 evolution for 100 test signals from the dataset *landsat*. Left: error. Middle: distance between actual and central atoms. Right: objective function. Blue: normal signals. Red: anomalies.**

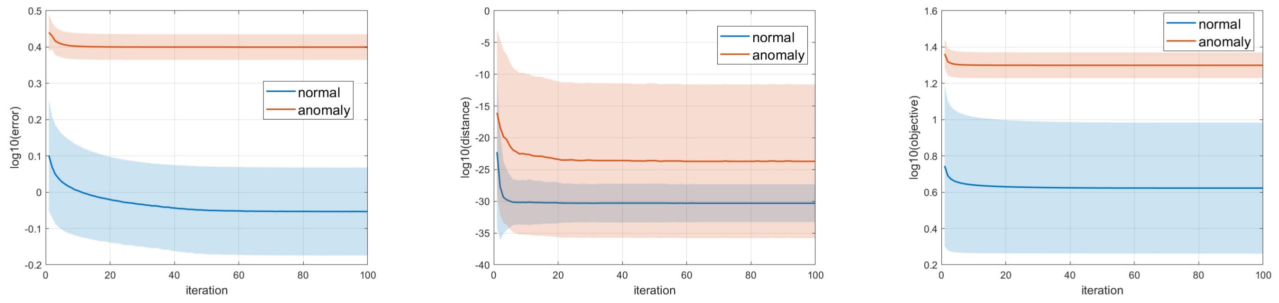

**FIGURE 2.** **Gauss-L1 evolution for all test signals from the dataset *landsat*. Left: error. Middle: distance between actual and central atoms. Right: objective function.**

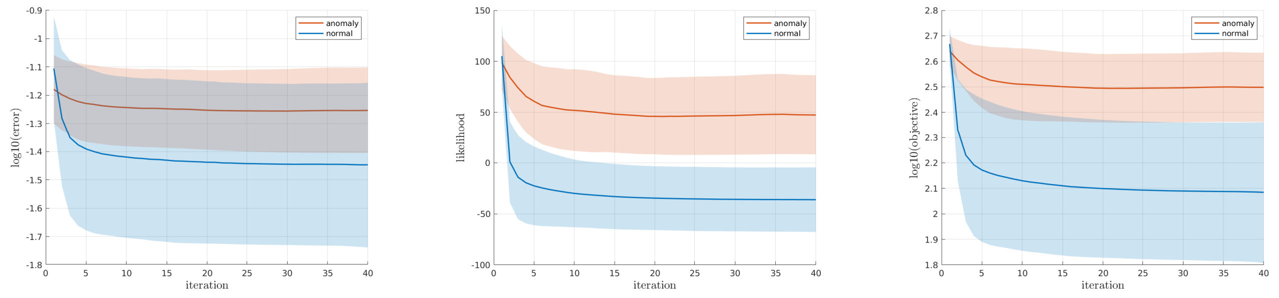

**FIGURE 3.** **DL-Gauss-OMP evolution for signals from the dataset *cardio*. Left: error. Middle: log-likelihood** (23)**. Right: objective function.**

best of $0.9329$. Our best results are clearly better. The best of Cone-DL methods is $0.9443$. Gauss-L1 and DL-Gauss-L1 achieve better results.

Our tests were run on a Lenovo laptop with Intel i7 processor and 32 GB of RAM. The running times of our algorithm for processing all 30 datasets are shown in Table 3. The training time for Gauss-OMP and Gauss-L1 is actually that of AK-SVD; note the differences between the MATLAB and Python implementations. Testing is made with exactly the same operations for the OMP methods, so the times should be identical in ideal conditions; same remark holds for the L1 methods. It is clear that DL is a costly procedure in this context and so efforts to speed it up are necessary. Note however that AK-SVD uses a highly optimized pre-compiled OMP implementation, while we have plainly implemented our methods.

**TABLE 1.** **Results for Gauss-OMP methods**

| parameters | | | | Gauss-OMP | | DL-Gauss-OMP | |
|---|---|---|---|---|---|---|---|
| score | $s$ | $\lambda$ | $\sigma$ | rank | ROC AUC | rank | ROC AUC |
| err | 2 | 1 | 0.05 | 4.07 | 0.8676 | 3.13 | 0.8967 |
| err | 2 | 1 | $\in [0.01, 1]$ | 6.77 | 0.8092 | 4.10 | 0.8817 |
| obj | 2 | 10 | 0.05 | 4.27 | 0.8643 | 2.20 | 0.9271 |
| obj | 2 | 10 | $\in [0.01, 1]$ | 6.70 | 0.8176 | 3.37 | 0.9192 |
| prob | 2 | 100 | 0.05 | 3.77 | 0.8742 | 3.57 | 0.8802 |
| prob | 2 | 100 | $\in [0.01, 1]$ | 5.67 | 0.8267 | 2.50 | 0.9212 |
| prob | 2 | 1000 | 0.05 | 3.70 | 0.8685 | 1.93 | 0.9401 |
| prob | 2 | 1000 | $\in [0.01, 1]$ | 6.80 | 0.8042 | 2.63 | 0.9357 |
| prob | 3 | 1000 | 0.05 | 4.40 | 0.8464 | 2.10 | 0.9283 |
| prob | 3 | 1000 | $\in [0.01, 1]$ | 6.60 | 0.8110 | 2.47 | 0.9344 |

At this moment, Gauss-L1 appears to give the best trade-off: very good results (better than state of the art) and affordable execution time.

**TABLE 2. Results for L1 methods using the error score** (22)**.**

| parameters | $\sigma$ | Gauss-L1 rank | Gauss-L1 ROC AUC | DL-Gauss-L1 rank | DL-Gauss-L1 ROC AUC |
|---|---|---|---|---|---|
| $\lambda = \gamma = 0.001$ | 0.05 | 1.77 | 0.9479 | 1.77 | 0.9479 |
| | $\in [0.01, 1]$ | 1.73 | 0.9503 | 1.73 | 0.9503 |
| $\lambda = \gamma = 0.01$ | 0.05 | 1.80 | 0.9453 | 1.90 | 0.9454 |
| | $\in [0.01, 1]$ | 1.80 | 0.9457 | 1.80 | 0.9458 |
| $\lambda = \gamma = 0.1$ | 0.05 | 1.83 | 0.9465 | 1.83 | 0.9468 |
| | $\in [0.01, 1]$ | 1.73 | 0.9471 | 1.60 | 0.9476 |
| $\lambda = \gamma = 1$ | 0.05 | 1.73 | 0.9492 | 1.70 | 0.9513 |
| | $\in [0.01, 1]$ | 1.70 | 0.9498 | 1.63 | 0.9519 |

**TABLE 3. Execution times (in seconds) of our algorithms for all 30 datasets.**

| algorithm | training | testing |
|---|---|---|
| Gauss-OMP | 80.8 | 332.4 |
| DL-Gauss-OMP | 3528 | 331.6 |
| Gauss-L1 | 22.8 | 157.7 |
| DL-Gauss-L1 | 1585 | 157.4 |

## VIII. CONCLUSIONS

In this paper, we present a novel perspective on the sparse representation problem and derive an algorithm inspired by Orthogonal Matching Pursuit, named Gaussian OMP. This method obtains a sparse representation for a given input signal by taking into account a Gaussian distribution for each atom, which is thus extended to an infinite set. The optimization objective combines the representation error and the probability of the actual atoms belonging to the distributions defined by the central atoms.

Moreover, we propose an L1 regularization method suited for sparse representations with Gaussian atoms that uses a block coordinate descent approach. We also present a solution for performing the dictionary update step in the DL problem when working with Gaussian atoms, that works for both our proposed Gauss-OMP and the L1 variant, Gauss-L1.

These strategies can be used in anomaly detection tasks. To represent outliers, attempting to find actual atoms with a high probability is more likely to decrease the representation error more than for normal signals, hence Gauss-DL can potentially obtain better results than standard DL.

We show that Gaussian atoms representation methods can obtain competitive results in the context of anomaly detection open-source benchmarks.

Further research will be focused on methods to choose different variances for different Gaussian atoms and on speeding up the algorithms.

## APPENDIX: PROOF OF PROPOSITION 1

1. We first prove that the solution is in the plane defined by $\boldsymbol{d}$ and $\tilde{\boldsymbol{y}}$. Multiplying with $\sigma^2$ (which does not change unimodality, hence we abusively keep the notation $h$), the function (10) can be written as

$$h(\boldsymbol{x}) = h_1(\boldsymbol{x}) + \beta h_2(\boldsymbol{x}), \tag{24}$$

where $h_1(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{d}\|^2$, $h_2(\boldsymbol{x}) = \min_\alpha \|\tilde{\boldsymbol{y}} - \alpha\boldsymbol{x}\|^2$, and $\beta = \sigma^2\lambda$.

*Lemma.* If $\boldsymbol{x}, \boldsymbol{\xi} \in \mathbb{R}^m$ are such that $\|\boldsymbol{x}\| = \|\boldsymbol{\xi}\| = 1$, $\boldsymbol{x}^T\tilde{\boldsymbol{y}} > 0$ and $\boldsymbol{\xi}^T\tilde{\boldsymbol{y}} > 0$ (changing the sign of $\boldsymbol{x}$ does not change the value of $h_2$) and $\|\tilde{\boldsymbol{y}}-\boldsymbol{x}\| < \|\tilde{\boldsymbol{y}}-\boldsymbol{\xi}\|$, then $h_2(\boldsymbol{x}) < h_2(\boldsymbol{\xi})$.

*Proof.* With the optimal form (11) of $\alpha$, we get

$$h_2(\boldsymbol{x}) = \|\tilde{\boldsymbol{y}} - (\boldsymbol{x}^T\tilde{\boldsymbol{y}})\boldsymbol{x}\|^2 = \|\tilde{\boldsymbol{y}}\|^2 - (\boldsymbol{x}^T\tilde{\boldsymbol{y}})^2.$$

Since $\|\tilde{\boldsymbol{y}} - \boldsymbol{x}\|^2 < \|\tilde{\boldsymbol{y}} - \boldsymbol{\xi}\|^2$ implies $\boldsymbol{x}^T\tilde{\boldsymbol{y}} > \boldsymbol{\xi}^T\tilde{\boldsymbol{y}}$ and these scalar products are positive, the conclusion is obvious. ■

Assume now that $\boldsymbol{\xi}$ is not in the plane defined by $\boldsymbol{d}$ and $\tilde{\boldsymbol{y}}$. By (spherical) projection, we can find $\boldsymbol{x}$ on that plane, with $\|\boldsymbol{d} - \boldsymbol{x}\| < \|\boldsymbol{d} - \boldsymbol{\xi}\|$ and $\|\tilde{\boldsymbol{y}} - \boldsymbol{x}\| < \|\tilde{\boldsymbol{y}} - \boldsymbol{\xi}\|$ (meaning that $\boldsymbol{x}$ is closer to both $\boldsymbol{d}$ and $\tilde{\boldsymbol{y}}$). The first inequality means that $h_1(\boldsymbol{x}) < h_1(\boldsymbol{\xi})$; through the above Lemma, the second inequality implies $h_2(\boldsymbol{x}) < h_2(\boldsymbol{\xi})$. We conclude that $h(\boldsymbol{x}) < h(\boldsymbol{\xi})$. Hence, the solution is on the plane defined by $\boldsymbol{d}$ and $\tilde{\boldsymbol{y}}$.

2. Since the problem has been reduced to a plane, we can solve it in 2D after an appropriate rotation. Since $\|\boldsymbol{x}\| = 1$, we take $x_1 = \cos\theta$, $x_2 = \sin\theta$. The optimal value (11) of $\alpha$ is

$$\alpha = \boldsymbol{x}^T\tilde{\boldsymbol{y}} = \tilde{y}_1\cos\theta + \tilde{y}_2\sin\theta.$$

Using this value, after some straightforward calculation it results that (the variable is now $\theta$ instead of $\boldsymbol{x}$):

$$h_2(\theta) = (\tilde{y}_1\sin\theta - \tilde{y}_2\cos\theta)^2.$$

Since a rotation does not change the properties of the function, we can take $\tilde{y}_2 = 0$. Also, since only the angle between the directions of $\boldsymbol{d}$ and $\tilde{\boldsymbol{y}}$ matters, it is enough to take $\theta \in [0, \pi/2]$ and examine only the case $d_1 \geq 0, d_2 \geq 0$. So, the function (24) becomes

$$h(\theta) = (\cos\theta - d_1)^2 + (\sin\theta - d_2)^2 + \beta\tilde{y}_1^2\sin^2\theta.$$

Its derivative is

$$h'(\theta) = 2\left(d_1\sin\theta - d_2\cos\theta + \beta\tilde{y}_1^2\sin\theta\cos\theta\right). \tag{25}$$

We note that $h'(0) < 0$, hence $h$ is decreasing in the neighborhood of zero. Let $\theta_0 > 0$ be the smallest for which $h'(\theta_0) = 0$, which also implies that $h'(\theta) < 0$ for $\theta \in [0, \theta_0)$. The first two terms of (25) are increasing functions of $\theta \in [0, \pi/2]$; the third is positive on the same interval. Thus, it results that $h'(\theta) > 0$ for $\theta \in (\theta_0, \pi/2]$. So, the function $h(\theta)$ is unimodal on $[0, \pi/2]$ and it has a single minimum. The angle $\theta = 0$ corresponds to $\tilde{\boldsymbol{y}}$, while the angle corresponding to $\boldsymbol{d}$ is somewhere in $(0, \pi/2]$. Hence, the function (10) is unimodal on the chord between the directions $\boldsymbol{d}$ and $\tilde{\boldsymbol{y}}$.

## 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] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.
[3] Zheng Zhang, Yong Xu, Jian Yang, Xuelong Li, and David Zhang. A survey of sparse representation: algorithms and applications. *IEEE Access*, 3:490–530, 2015.
[4] Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: poster and demo track*, 1:59–63, 2012.
[5] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.
[6] Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David W Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Advances in Knowledge Discovery and Data Mining: 6th Pacific-Asia Conference, PAKDD 2002 Taipei, Taiwan, May 6–8, 2002 Proceedings 6*, pages 535–548. Springer, 2002.
[7] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Outlier detection in axis-parallel subspaces of high dimensional data. In *Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 Proceedings 13*, pages 831–838. Springer, 2009.
[8] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 427–438, 2000.
[9] L. Ruff, J.R. Kauffmann, R.A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T.G. Dietterich, and K.R. Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021.
[10] Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. ADBench: Anomaly detection benchmark. *Advances in Neural Information Processing Systems*, 35:32142–32159, 2022.
[11] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International Conference on Machine Learning*, pages 4393–4402. PMLR, 2018.
[12] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.
[13] Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. Copod: copula-based outlier detection. In *2020 IEEE international conference on data mining (ICDM)*, pages 1118–1123. IEEE, 2020.
[14] Zheng Li, Yue Zhao, Xiyang Hu, Nicola Botta, Cezar Ionescu, and George Chen. ECOD: Unsupervised outlier detection using empirical cumulative distribution functions. *IEEE Transactions on Knowledge and Data Engineering*, pages 12181–12193, 2022.
[15] Yang Xu, Zebin Wu, Jun Li, Antonio Plaza, and Zhihui Wei. Anomaly detection in hyperspectral images based on low-rank and sparse representation. *IEEE Transactions on Geoscience and Remote Sensing*, 54(4):1990–2000, 2016.
[16] Jeffrey M. Druce, Jarvis D. Haupt, and Stefano Gonella. Anomaly-sensitive dictionary learning for structural diagnostics from ultrasonic wavefields. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 62(7):1384–1396, 2015.
[17] B. Pilastre, L. Boussouf, S. d'Escrivan, and J.Y. Tourneret. Anomaly detection in mixed telemetry data using a sparse representation and dictionary learning. *Signal Processing*, 168:107320, 2020.
[18] P. Irofti, F. Stoican, and V. Puig. Fault handling in large water networks with online dictionary learning. *Journal of Process Control*, 94:46–57, 2020.
[19] 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.
[20] Paul Irofti, Cristian Rusu, and Andrei Pătraşcu. Dictionary learning with uniform sparse representations for anomaly detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3378–3382, 2022.
[21] Amir Adler, Michael Elad, Yacov Hel-Or, and Ehud Rivlin. Sparse coding with anomaly detection. *Journal of Signal Processing Systems*, 79:179–188, 2015.
[22] Denis C. Ilie-Ablachim, Andra Băltoiu, and Bogdan Dumitrescu. Sparse representations with cone atoms. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.
[23] A. Băltoiu, D.C. Ilie-Ablachim, and B. Dumitrescu. Dictionary learning with cone atoms and application to anomaly detection. submitted, August 2023. http://asydil.upb.ro/wp-content/uploads/2023/08/Cone_DL_extended.pdf.
[24] Ivan Markovsky and Sabine Van Huffel. Overview of total least-squares methods. *Signal Processing*, 87(10):2283–2302, 2007.
[25] M.A. Herman and T. Strohmer. General Deviants: An Analysis of Perturbations in Compressed Sensing. *IEEE J. Sel. Topics Signal Proc.*, 4(2):342–349, Apr. 2010.
[26] H. Zhu, G. Leus, and G.B. Giannakis. Sparsity-Cognizant Total Least-Squares for Perturbed Compressive Sampling. *IEEE Trans. Signal Proc.*, 59(5):2002–2016, May 2011.
[27] Qi Yu, Wei Dai, Zoran Cvetković, and Jubo Zhu. Dictionary learning with blotless update. *IEEE Transactions on Signal Processing*, 68:1635–1645, 2020.
[28] Bob L. Sturm and Mads Græsbøll Christensen. Comparison of orthogonal matching pursuit implementations. *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 220–224, 2012.
[29] A. Beck and M. Teboulle. Fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imag. Sci.*, 2(1):183–202, 2009.
[30] Y. Yuan, D. Ma, and Q. Wang. Hyperspectral anomaly detection via sparse dictionary learning method of capped norm. *IEEE Access*, 7:16132–16144, 2019.
[31] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *arXiv preprint arXiv:1901.01588*, 2019.
[32] 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.
[33] Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit. Technical report, Technion Univ., 2008.