Dictionary learning with cone atoms and application to anomaly detection (extended version)*

Andra Băltoiu, Denis C. Ilie-Ablachim, Bogdan Dumitrescu

Department of Automatic Control and Computers
University Politehnica of Bucharest, Romania
Emails: andra.baltoiu@upb.ro, denis.ilie_ablachim@upb.ro, bogdan.dumitrescu@upb.ro.

Abstract

We propose a variant of dictionary learning (DL) for sparse representations where the atoms are cones instead of simple vectors. The most convenient vector from a cone, called actual atom, is used to build the linear sparse representation of a given signal. We present a DL algorithm suited for cone atoms, which can update the dictionary without storing all the actual atoms that are used in the representations of the training signals. Also, the algorithm ensures that the cone atoms are disjoint and thus the representation problem is well posed. We use the proposed cone DL for anomaly detection. On a specific type of anomaly, called 'dependency', the DL methods involving cone atoms are better than those from a reputed benchmark. They are also better than standard DL.

Keywords: dictionary learning, sparse representations, anomaly detection, unsupervised learning, cone atoms

1. Introduction

Dictionary learning (DL) for sparse representations is now a mature field, comprising diverse problem formulations and algorithms for their efficient solution, with numerous applications in classification, denoising, inpainting, compressed sensing, and many others.

In the standard sparse representation problem, we are given a signal $\boldsymbol{y} \in \mathbb{R}^m$ and a dictionary $\boldsymbol{D} \in \mathbb{R}^{m\times n}$, and the purpose is to find a vector

*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.

$\boldsymbol{x} \in \mathbb{R}^n$ with at most s nonzero elements such that $\|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\|_2$ is minimized; s is the sparsity level. In DL, we have a collection of signals $\boldsymbol{Y} \in \mathbb{R}^{m\times N}$ and the purpose is to design the dictionary $\boldsymbol{D}$ such that $\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_2$ is minimum, where the representation matrix $\boldsymbol{X} \in \mathbb{R}^{n\times N}$ has at most s nonzeros on each column. To avoid multiplicative indeterminacy, the columns of $\boldsymbol{D}$, named atoms, are normalized. This is the most common DL problem, where the atoms are simply vectors.

Problem formulation. We propose here to extend the notion of atom from a vector to an infinite set. More precisely, we consider dictionaries made of cone atoms. A column $\boldsymbol{d}$ of $\boldsymbol{D}$ is the central vector of a cone $\mathcal{C}(\boldsymbol{d}, \rho)$ whose radius is ρ, in the sense that $\|\boldsymbol{a} - \boldsymbol{d}\| \le \rho$ for all $\boldsymbol{a} \in \mathcal{C}(\boldsymbol{d}, \rho)$; all the vectors in the cone have norm equal to 1. Figure 1 shows a 3D cone in a sphere; the blue volume is a sector of the sphere; since $\|\boldsymbol{a}\| = 1$, the set $\mathcal{C}(\boldsymbol{d}, \rho)$ is in fact a cap on the sphere. However, since we work with linear combinations of atoms, the fittest name is that of cone.

For building sparse representations, we can use any vector from a cone as an atom. Each atom $\boldsymbol{d}_j$ has a cone $\mathcal{C}(\boldsymbol{d}_j, \rho_j)$ associated with it; so, the cone radii may be different. The sparse representation problem is formulated as follows:

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}\in\mathbb{R}^n} & \|\boldsymbol{y} - \sum_{j=1}^{n} \boldsymbol{a}_j x_j\|_2 \\ \text{s.t.} & \|\boldsymbol{x}\|_0 \le s \\ & \boldsymbol{a}_j \in \mathcal{C}(\boldsymbol{d}_j, \rho_j),\ j = 1:n \end{array} \tag{1}$$

Here, we name $\boldsymbol{a}_j$ actual atoms, since they are effectively used in the representation. Figure 2 illustrates the linear combination of $s = 2$ atoms that is nearest from a vector $\boldsymbol{y}$. This optimal approximation is the projection of $\boldsymbol{y}$ on the plane generated by $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$; the actual atoms are such that the plane is tangent to both cones, thus minimizing the distance to $\boldsymbol{y}$. Note that neither $\boldsymbol{a}_1$ nor $\boldsymbol{a}_2$ are the nearest vector from $\boldsymbol{y}$ in their respective cones.

The DL problem with cone atoms will be exposed later, but it is a natural extension of (1). Essentially, given the signal matrix $\boldsymbol{Y} \in \mathbb{R}^{m\times N}$, we need to design the dictionary $\boldsymbol{D}$ of central atoms such that the representation error (where actual atoms are used!) is minimum. It is clear that this problem is more difficult than standard DL.

Contents of the paper. The remaining part of the introduction will discuss previous work and the relation of our contribution to it. Section 2 presents several considerations on the geometrical properties of the cones: the volume occupied by a cone or by a linear combination of cones in the (hyper)sphere, formulas for computing the distance between two cones and an algorithm for the projection of a vector on a cone, which is essential

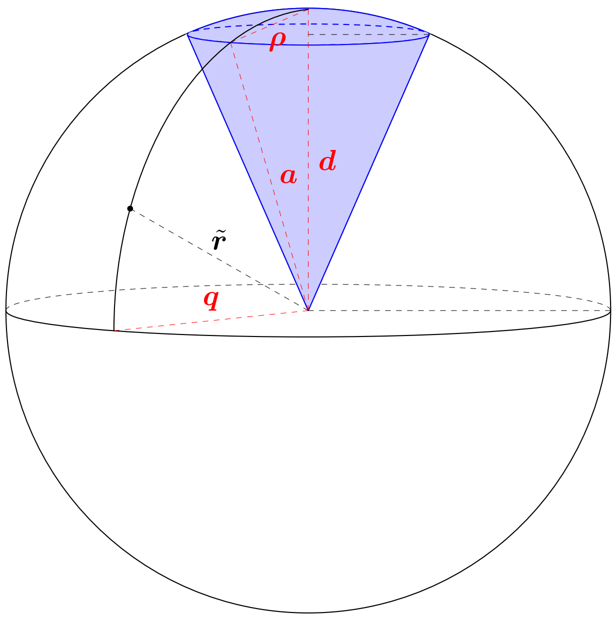

Figure 1: A spherical sector (cone).

in building the other algorithms. Section 3 contains our DL algorithm for dictionaries with cone atoms; it is based on a parallel atom update that eliminates the necessity of storing the actual atoms; it also ensures that cone atoms are disjoint. Since our application is anomaly detection, Section 4 is dedicated to it; we present the problem and the datasets that we use. Section 5 presents the numerical results and shows the advantages of cone atoms for the specific case of dependency anomalies; we compare our methods with those from a recent benchmark [1]. More results can be found in the Appendices. The idea of sparse representations with cone atoms has been presented in [2]; here, we extend it to DL, we discuss in more detail the properties of cone dictionaries and we present a different set of experiments.

Previous work. Our proposal of dictionaries with cone atoms appears to be the first where the atoms are definite infinite sets, although there are several related approaches. In shift invariant dictionaries [3, 4], shifted versions of an atom are also automatically taken into account. Extension towards continuous shift can be made by grid refinement [5] or in the context of group invariant dictionaries [6]. Convolutional DL [7] offers another view to a similar problem, but with more representation freedom than in the standard DL problem. Structured dictionaries offer the possibility to build

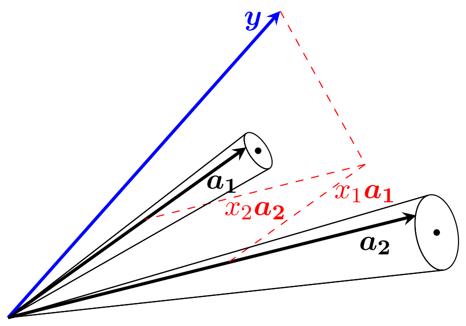

Figure 2: Optimal approximation of a signal $\boldsymbol{y}$ with the linear combination $x_1\boldsymbol{a}_1 + x_2\boldsymbol{a}_2$ of two cone atoms.

combinations of atoms of size smaller than m in order to obtain a large number of possible atoms of size m that are not represented explicitly. Examples are multi layer [8, 9], multi scale [10], and separable [11] dictionaries; in the latter case, suited for image representation, the atoms are Kronecker products of smaller atoms and can also be seen as rank-1 matrices; in multi layer dictionaries, $\boldsymbol{D}$ is a product of several dictionaries with simple structure; multi scale dictionaries are obtained using smaller building blocks.

The closest to our approach is the problem of sparse total least squares (STLS) [12, 13]; there, the representation error can be not only the vector $\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}$, but also in the dictionary; so, the representation is not $\boldsymbol{D}\boldsymbol{x}$, but $\tilde{\boldsymbol{D}}\boldsymbol{x}$, where $\tilde{\boldsymbol{D}}$ is a matrix near $\boldsymbol{D}$. However, this matrix has no structure in STLS and the error associated with an atom is not constrained. By using cones, we naturally bound the difference between the actual and the central atoms. We also hope that a clever radii allocation, possibly helped by an appropriate DL algorithm, will be well suited for applications like anomaly detection, where ideally the atoms with larger cones are used mostly for normal signals, thus reducing their representation error, while anomalies are represented with atoms with smaller cone radius, thus keeping large errors. In such a situation, cone dictionaries will simply give better anomaly detection than standard DL.

A somewhat related work [14] proposes a fuzzy approach to DL. The atoms are not fixed, but associated with a membership function. However, this association is used only for learning the dictionary; once $\boldsymbol{D}$ is found, it is used for sparse representation in the standard way.

There are also dictionaries with parametric form [15], in which the atoms are not free in $\mathbb{R}^m$, but depend on a small number of parameters. A notable example comes from direction of arrival (DOA) estimation; the atoms depend on a single parameter, which takes continuous values; no DL is involved and one can find the relevant atoms by discretization or enforcing sparsity via the atomic norm [16].

Finally, from the viewpoint of nonlinearity, there are several other approaches where the dictionary itself has a nonlinear structure, but the representation is finally linear, like in our case. Besides some of the works enumerated above, like those implying multi layer dictionaries, we could single out dictionaries involving or combined with neural networks [17, 18].

Anomaly detection is one of the important problems in machine learning and there are many algorithms dedicated to it. Since in many applications anomalies are not only different from normal samples but are also scarce, the most appropriate methods for their identification are unsupervised. For a general bibliography we point out to the review paper [19] or the more application oriented surveys like [20] for financial fraud or [21] for image data. We focus here on the use of DL in anomaly detection. While the supervised setting is relatively straight-forward for DL (as it can be cast as a binary classification problem, for which several solutions exist - see [22] for a review), unsupervised variants have only fairly recently been proposed. They do, however, cover a wide range of applications, such as abnormal electrocardiogram patterns [23, 24], abnormal network traffic [25], hyperspectral images [26, 27], telemetry [28], graph anomalies [29]. The underlying idea of most of the unsupervised DL approaches is that since normal samples outnumber the anomalies, they are better modeled via sparse representations; hence, their representation errors are lower.

2. Dictionaries with cone atoms

We examine first some geometric aspects of the cone dictionaries.

2.1. How big is a cone?

To evaluate how big is a cone, we evaluate the volume of its intersection with the unit ball, which is a hypersector of a hypersphere. In 3D, a cone seems to occupy a tangible part of the ball, as suggested by Figure 1. Is this still true when the number of dimensions grows?

The volume of a hypersector is [30]

$$v_m^s = \frac{1}{n} v_m^b I_{\sin^2\varphi}\left(\frac{n-1}{2}, \frac{1}{2}\right), \tag{2}$$

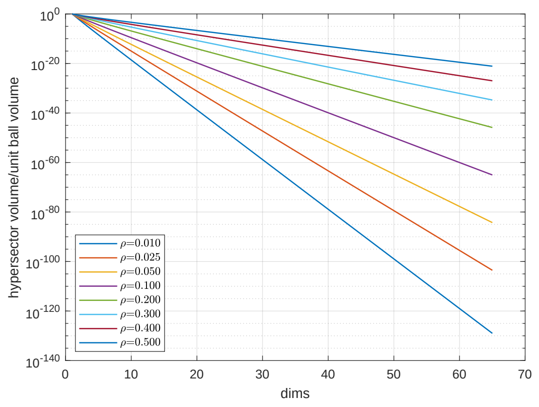

Figure 3: Ratio between the volume of the hypersector and the volume of the unit ball.

where

$$v_m^b = \frac{\pi^{m/2}}{\Gamma(\frac{m}{2}+1)} \tag{3}$$

is the volume of the unit ball in $\mathbb{R}^m$, I represents the regularized incomplete beta function, which is evaluated with respect to the sine of the half apex (or colatitude) angle of the cone φ (the angle between the cone axis $\boldsymbol{d}$ and its generatrix $\boldsymbol{a}$). Since $\sin(\varphi/2) = \rho/2$, it follows that

$$\sin\varphi = \frac{\rho\sqrt{4-\rho^2}}{2}. \tag{4}$$

Figure 3 shows the ratio v_m^s/v_m^b between the volume of the hypersector and the volume of the unit ball (which is equal to the ratio between the cap defined by the cone and the surface of the unit ball), as a function of ρ and m, the number of dimensions. We see that even for radii that are large, like $\rho = 0.5$, the volume of the cone is extremely small with respect to that of the unit ball, even for moderate number of dimensions.

Linear combinations of atoms from several cones cover a larger fraction of the space. However, the fraction is still small. For example, considering the simple case of s orthogonal central atoms, all having cones with radius ρ, the volume that can be covered with linear combinations of atoms from these cones (that lie within the unit ball) is $O((2\rho\sqrt{s})^{m-s})$, which is much smaller than the volume of the unit ball. See Appendix A for details. Even when considering the number of combinations of s atoms from a dictionary with n atoms, if s is small, then the volume of the points exactly representable as linear combinations of cone atoms is still small.

So, it is highly unlikely that, even with a trained dictionary, the representation error is zero for all signals, which would be undesirable in general and particularly in an anomaly detection application.

2.2. Distance between cones

It looks natural to have disjoint cones in a dictionary, for parsimony reasons. Moreover, intersecting cone atoms make the representation problem ill-posed.

Remark 1. Consider the worst case, that of two identical cone atoms, both equal to $\mathcal{C}(\boldsymbol{d}, \rho)$. Then, all vectors $\boldsymbol{y} \in \mathbb{R}^m$ can be represented as a linear combination of two atoms from $\mathcal{C}(\boldsymbol{d}, \rho)$. Indeed, any two linearly independent vectors from the intersection of the cone with the hyper-plane generated by $\boldsymbol{y}$ and $\boldsymbol{d}$ can form a linear combination equal to $\boldsymbol{y}$. If the cones are not identical, but have an intersection, there is still a hyper-plane containing $\boldsymbol{y}$ and two independent vectors from that intersection.

So, we must explicitly forbid cone superposition. Anyway, having distanced atoms is beneficial for the representation: a low mutual coherence is known to improve the properties of the sparsest representation and the good functioning of sparse representation algorithms. ■

We derive a formula for the minimum distance between two disjoint cones.

Proposition 2. *Let $\mathcal{C}(\boldsymbol{d}_1, \rho_1)$ and $\mathcal{C}(\boldsymbol{d}_2, \rho_2)$ be two disjoint cones. (We assume that $\boldsymbol{d}_1^T \boldsymbol{d_2} \geq 0$.) Then, the distance between their central atoms obeys to*

$$\|\boldsymbol{d}_1 - \boldsymbol{d_2}\| > \rho_1 \sqrt{1 - \frac{\rho_2^2}{4}} + \rho_2 \sqrt{1 - \frac{\rho_1^2}{4}}. \tag{5}$$

Proof. The situation where the cones are tangent is illustrated in Figure 4, where the section of the plane generated by the central atoms is shown. The half apex angles of the cones are denoted α_1 and α_2. The distance between the central atoms is $\|\boldsymbol{d}_1 - \boldsymbol{d}_2\| = \rho$. We denote $\boldsymbol{a}$ the unique atom belonging to both cones, satisfying $\|\boldsymbol{a} - \boldsymbol{d}_1\| = \rho_1$, $\|\boldsymbol{a} - \boldsymbol{d}_2\| = \rho_2$. From the isosceles triangles whose equal sides are $(\boldsymbol{d}_1, \boldsymbol{a})$, $(\boldsymbol{d}_2, \boldsymbol{a})$, and $(\boldsymbol{d}_1, \boldsymbol{d}_2)$, respectively, we derive the relations:

$$\sin\frac{\alpha_1}{2} = \frac{\rho_1}{2}, \quad \sin\frac{\alpha_2}{2} = \frac{\rho_2}{2}, \quad \sin\frac{\alpha_1 + \alpha_2}{2} = \frac{\rho}{2}.$$

By using

$$\sin\frac{\alpha_1 + \alpha_2}{2} = \sin\frac{\alpha_1}{2}\cos\frac{\alpha_2}{2} + \cos\frac{\alpha_1}{2}\sin\frac{\alpha_2}{2}$$

it results immediately that (5) holds. ■

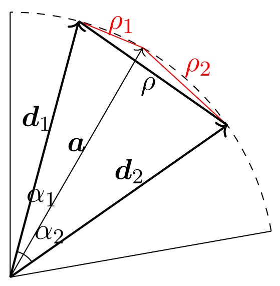

Figure 4: Tangent cones (section).

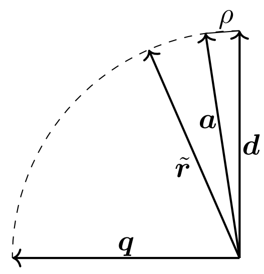

Figure 5: Nearest atom $\boldsymbol{a}$ in cone centered in $\boldsymbol{d}$ with radius ρ, with respect to a vector $\tilde{\boldsymbol{r}}$.

2.3. Projection on a cone

A typical problem that appears in sparse representation is to find the nearest atom from a given vector $\boldsymbol{r}$, typically a residual. For cone atoms, the problem is equivalent to finding the projection of $\boldsymbol{r}$ on the cone.

Proposition 3. Given cone $\mathcal{C}(\boldsymbol{d},\rho)$ and a vector $\boldsymbol{r}$, the nearest atom $\boldsymbol{a} \in \mathcal{C}(\boldsymbol{d},\rho)$ from $\boldsymbol{r}$, i.e. the solution of

$$\begin{array}{cl} \min\limits_{\boldsymbol{a}\in\mathcal{C}(\boldsymbol{d},\rho)} & \|\boldsymbol{a}-\boldsymbol{r}\|_2 \\ \text{s.t.} & \|\boldsymbol{a}\| = 1 \end{array} \tag{6}$$

is given by Algorithm 1. ■

The proof is given in [2]. Here, we only stress that the problem can be reduced to 2D, like in Figure 5, and solved with elementary geometry in the plane generated by $\boldsymbol{d}$ and $\tilde{\boldsymbol{r}} = \boldsymbol{r}/\|\boldsymbol{r}\|$. Note that Figure 5 is a section of Figure 1 made by the above plane. The figure illustrates the case where $\boldsymbol{r}$ is outside the cone; if $\boldsymbol{r}$ is inside, then the projection is trivially the

Algorithm 1: Nearest_atom: compute nearest atom in a cone from a given vector.

```
Data: vector $\boldsymbol{d} \in \mathbb{R}^m$ and $\rho > 0$ defining cone $\mathcal{C}(\boldsymbol{d}, \rho)$
      vector $\boldsymbol{r} \in \mathbb{R}^m$
Result: atom $\boldsymbol{a} \in \mathcal{C}(\boldsymbol{d}, \rho)$ nearest from $\boldsymbol{r}$
1  Normalize vector: $\tilde{\boldsymbol{r}} = \boldsymbol{r}/\|\boldsymbol{r}\|$
2  Compute $p = \boldsymbol{d}^T \tilde{\boldsymbol{r}}$
3  if $p < 0$ then
4      Change orientation: $\tilde{\boldsymbol{r}} \leftarrow -\tilde{\boldsymbol{r}}$
5  if $|p| + \rho^2/2 \geq 1$ then
6      The vector is in the cone: $\boldsymbol{a} = \tilde{\boldsymbol{r}}$
7  else
8      Set $\beta = \sqrt{\frac{1}{1-p^2}}$, $\alpha = -\beta|p|$
9      Set $\lambda = 1 - \rho^2/2$, $\mu = \sqrt{1 - \lambda^2}$
10     The desired atom is $\boldsymbol{a} = (\lambda + \mu\alpha)\boldsymbol{d} + \mu\beta\tilde{\boldsymbol{r}}$
```

normalized $\boldsymbol{r}$. We stress that Algorithm 1 uses only vector operations and hence has complexity $O(m)$.

3. Dictionary learning with cone atoms

The natural extension of the dictionary learning problem to cone atoms is formulated as follows. Given N signals $\boldsymbol{y}_\ell$, $\ell = 1 : N$, and assuming that cone radii ρ_j, $j = 1 : n$, are fixed, we want the cone centers $\boldsymbol{d}_j$, $j = 1 : n$, that solve the problem

$$\begin{array}{ll} \min\limits_{\boldsymbol{d}_i \in \mathbb{R}^m, \boldsymbol{x}_\ell \in \mathbb{R}^n} & \sum\limits_{\ell=1}^{N} \left\| \boldsymbol{y}_\ell - \sum\limits_{i=1}^{n} \boldsymbol{a}_{i\ell} x_{i\ell} \right\|_2^2 \\ \text{s.t.} & \|\boldsymbol{x}_\ell\|_0 \leq s,\ \ell = 1 : N \\ & \boldsymbol{a}_{i\ell} \in \mathcal{C}(\boldsymbol{d}_i, \rho_i),\ i = 1 : n, \ell = 1 : N \end{array} \tag{7}$$

Note that only the central atoms are optimized; cone radii are given. With an error objective and no constraints, radii optimization would lead to large values that make all errors equal to zero; such a solution is irrelevant.

To solve (7), we adopt the standard iterative approach in which the representations and the dictionary are updated successively. The Cone-OMP algorithm proposed in [2] is used for computing the actual atoms and their coefficients, with fixed central atoms and radii. It will be shortly reviewed in Section 3.1. The algorithm to update the central atoms of the dictionary will be presented in Section 3.2.

3.1. Cone-OMP

We proposed in [2] a sparse representation algorithm for dictionaries with cone atoms, named Cone Optimal Matching Pursuit (Cone-OMP). We give here only a short description.

The input-output arguments are

$$[\boldsymbol{x}, \mathcal{S}, \boldsymbol{A}] = \text{Cone-OMP}(\boldsymbol{D}, \boldsymbol{y}, s),$$

where $\boldsymbol{y}$ is the input signal, $\boldsymbol{D}$ the dictionary of central atoms and s the sparsity level; the algorithm computes the representation vector $\boldsymbol{x}$, its support $\mathcal{S}$ and the actual atoms stored in the matrix $\boldsymbol{A} \in \mathbb{R}^{m\times s}$.

Like standard OMP [31], Cone-OMP is a greedy algorithm that builds the support by adding one index at a time. To select the next atom, the cone atom that is nearest to the current residual is found using Algorithm 1. To find the optimal approximation for the current support, the least-squares solution used in OMP, which is impossible for cones, since the actual atoms are not known, is replaced by coordinate descent. All actual atoms are considered fixed with the exception of single one, say $\boldsymbol{a}_j$. The residual

$$\boldsymbol{r} = \boldsymbol{y} - \sum_{i\in\mathcal{S}, i\neq j} x_j \boldsymbol{a}_j$$

is computed and then $\boldsymbol{a}_j$ is updated as the nearest atom from $\boldsymbol{r}$ in $\mathcal{C}(\boldsymbol{d}_j, \rho_j)$. This is again done with Algorithm 1. The associated optimal coefficient x_j is readily available. A few coordinate descent rounds over the whole support are usually sufficient for providing a nearly optimal solution. Note that this procedure can also be seen as successive projections on convex sets; each projection (or coordinate descent step) improves the representation error. The complexity of Cone-OMP is only a few times that of OMP. However, unlike for OMP, batch algorithms to be used when many signals are represented with the same dictionary are not possible, since the actual atoms are each time different.

3.2. Central atoms update

The main difficulty in optimizing the central atoms is that they appear only indirectly in (7). The Cone-OMP algorithm produces the actual atoms and their coefficients. We must decide if we keep the information produced by Cone-OMP, namely the actual atoms and their coefficients, or we process it immediately. Storing the sparse matrix $\boldsymbol{X}$ is usual in standard DL algorithms. However, to store all actual atoms, we need s times more memory than for the matrix $\boldsymbol{D}$ of central atoms. This may be a significant memory demand, which we try to avoid.

We search inspiration in AK-SVD [32], which is probably the simplest among efficient DL algorithms, so it is clearly a candidate for adaptation to cone atoms. Denoting $\boldsymbol{d}_j$ the central atom to be updated while everything else is fixed, the error matrix without the contribution of this atom is $\boldsymbol{F}$, whose column ℓ is

$$\boldsymbol{f}_\ell = \boldsymbol{y}_\ell - \sum_{i \neq j} \boldsymbol{a}_{i\ell} x_{i\ell}. \tag{8}$$

The squared error has the expression

$$\sum_\ell \|\boldsymbol{f}_\ell - \boldsymbol{a}_{j\ell} x_{j\ell}\|^2. \tag{9}$$

Note that this sum involves only the signals whose representation uses an atom from $\mathcal{C}(\boldsymbol{d}_j, \rho_j)$ (for the others, $x_{j\ell} = 0$). The AK-SVD idea is to minimize the error as a function of the current atom $\boldsymbol{d}_j$ only, all other variables being fixed. If the representations are optimal, then $\boldsymbol{a}_{j\ell} = \boldsymbol{d}_j + \boldsymbol{g}_{j\ell}$, where $\boldsymbol{g}_{j\ell}$ is a vector with $\|\boldsymbol{g}_{j\ell}\| \leq \rho$, situated in the subspace generated by $\boldsymbol{d}_j$ and $\boldsymbol{f}_\ell$. In other words, $\boldsymbol{g}_{j\ell}$ is the difference between the normalized projection of $\boldsymbol{f}_\ell$ on the cone (which gives the nearest actual atom from $\boldsymbol{f}_\ell$, see section 2.3) and the cone center. After replacing $\boldsymbol{a}_{j\ell}$ with its above expression, the derivative of the error (9) with respect to $\boldsymbol{d}_j$ is

$$2\sum_\ell x_{j\ell}^2 \boldsymbol{d}_j - 2\sum_\ell x_{j\ell}(\boldsymbol{f}_\ell - x_{j\ell}\boldsymbol{g}_{j\ell}). \tag{10}$$

Setting the derivative to zero gives the update rule

$$\boldsymbol{d}_j \leftarrow \sum_\ell x_{j\ell}(\boldsymbol{f}_\ell - x_{j\ell}(\boldsymbol{a}_{j\ell} - \boldsymbol{d}_j)), \tag{11}$$

followed by normalization.

Remark 4. For insight, note that if there is a single signal, then the new $\boldsymbol{d}_j$ is the center of a cone with radius ρ that is tangent to $\boldsymbol{f}_\ell$; if $\boldsymbol{f}_\ell$ was in the cone, then $\boldsymbol{d}_j$ remains unchanged; this operation is somewhat similar to soft thresholding; the cone is moved with the minimum necessary amount to make it optimal.

To prove the above affirmations, we use the relation $\boldsymbol{f}_\ell = \boldsymbol{a}_{j\ell} x_{j\ell} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is the representation error. From (11), it results that $\boldsymbol{d}_j \leftarrow x_{j\ell}\boldsymbol{d}_j + \boldsymbol{\epsilon}$. When $\boldsymbol{f}_\ell$ was in the cone, then $\boldsymbol{\epsilon} = 0$. Since $\boldsymbol{f}_\ell - \boldsymbol{\epsilon}$ was on (or inside) the cone, now it is on the boundary (or inside) the new cone. ■

In standard AK-SVD, the update of the atoms is sequential. In the cone atom context, once $\boldsymbol{d}_j$ is updated with (11), this would imply not only updating the coefficients $x_{j\ell}$, but also the actual atoms $\boldsymbol{a}_{j\ell}$ for all signals using an atom from $\mathcal{C}(\boldsymbol{d}_j, \rho_j)$. This would imply a large cost.

However, we can easily adapt the Parallel AK-SVD (PAK-SVD) algorithm [22], in which the update of all atoms is made in parallel. (It was shown that this algorithm gives good results, although it usually needs more iterations than the standard version and convergence is more erratic.) Indeed, the sum from (11) can be computed by adding terms as they become available. Once a signal is represented with Cone-OMP, the error (8) can be computed for each atom involved in representation and added to a vector that will become (11) when all signals are represented. This can be done without any extra computation compared with PAK-SVD. An iteration of the resulting Cone-PAK-SVD is shown as Algorithm 2; since this is the only algorithm that we propose, we will simply name it Cone-DL. We note that the iterations of the for loop 2 are independent, only the results need to be summed in a common variable $\tilde{\boldsymbol{D}}$. So, there is indeed considerable parallelism in Cone-PAK-SVD.

Remark 5. A possible variation of the algorithm is to update only part of the atoms in an iteration. Denote $0 < p \leq 1$ the fraction of atoms to be updated. In each iteration of Cone-PAK-SVD, a set of $\lfloor pn \rfloor$ atoms is chosen. The only modification of Algorithm 2 is to execute the for loop 5 only for atoms that belong to the current set. Obviously, for $p = 1$ we obtain Cone-PAK-SVD. For $p = 1/n$ we would obtain an inefficient cone version of AK-SVD. ■

3.3. Ensuring disjoint cones

Algorithm 2 ignores the requirement that cones must be disjoint, see Remark 1. As a result, especially when cone radii are large, the designed dictionary has often intersecting atoms.

A simple way to obtain disjoint cones is to correct the result of each Cone-DL iteration such that all superposing pairs of cones are separated. An approach in this style was used in [33] for standard DL, but with a decorrelation technique applied on the whole dictionary with the purpose

Algorithm 2: An iteration of Cone-PAK-SVD.

```
   Data: signal matrix $\boldsymbol{Y} \in \mathbb{R}^{m\times N}$
         dictionary $\boldsymbol{D} \in \mathbb{R}^{m\times n}$ and vector of radii $\boldsymbol{\rho} \in \mathbb{R}^n$
         ($\boldsymbol{D}$ and $\boldsymbol{\rho}$ define cones $\mathcal{C}_j(\boldsymbol{d}_j, \rho_j)$, $j = 1:n$)
         sparsity level $s$
   Result: updated dictionary $\boldsymbol{D}$
 1 Initialize new dictionary: $\tilde{\boldsymbol{D}} = \boldsymbol{0}_{m\times N}$
 2 for $\ell = 1$ to $N$ do
 3     Compute representation: $[\boldsymbol{x}, \mathcal{S}, \boldsymbol{A}] = \text{Cone-OMP}(\boldsymbol{D}, \boldsymbol{y}_\ell, s)$
 4     Compute residual: $\boldsymbol{r} = \boldsymbol{y}_\ell - \boldsymbol{A}\boldsymbol{x}$
 5     for $i = 1$ to $s$ do
 6         Index of current atom is: $j = \mathcal{S}_i$
 7         Residual without this atom: $\boldsymbol{f} = \boldsymbol{r} + \boldsymbol{a}_j x_j$
 8         Update sum (11): $\tilde{\boldsymbol{d}}_j \leftarrow \tilde{\boldsymbol{d}}_j + x_i(\boldsymbol{f} - x_i(\boldsymbol{a}_i - \boldsymbol{d}_j))$
 9 Update dictionary $\boldsymbol{D} \leftarrow \tilde{\boldsymbol{D}}$
10 Normalize columns of $\boldsymbol{D}$
```

of reducing mutual coherence. Here, we target only the atoms that are too close to each other. A technique in this vein is INK-SVD [34], where atoms that are too close are simply pushed away from each other in the plane generated by them. We propose to take also into account their previous values, which are the result of the optimization process; in this way we expect better representation error than by using an artificial or arbitrary direction.

Let $\boldsymbol{D}$ be the dictionary at the beginning of the iteration and $\hat{\boldsymbol{D}}$ the updated dictionary. For any two updated central atoms $\hat{\boldsymbol{d}}_1$ and $\hat{\boldsymbol{d}}_2$, we want to ensure that the distance between them is at least

$$\delta_{\min} = \rho + \delta_0, \tag{12}$$

where ρ is the distance (5) when the cones are tangent and $\delta_0 > 0$ is the minimum desired distance between any pair of atoms of the two cones. Of course, we assume that the cones of $\boldsymbol{D}$ satisfy (12), while those of $\hat{\boldsymbol{D}}$ might not.

We propose the following algorithm. If the atoms $\hat{\boldsymbol{d}}_1$ and $\hat{\boldsymbol{d}}_2$ do not satisfy (12), we move them back towards $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$, respectively, until they satisfy (12). A possibility is by bisection: we move $\hat{\boldsymbol{d}}_1$ to the middle of the distance between $\boldsymbol{d}_1$ and $\hat{\boldsymbol{d}}_1$ and move $\hat{\boldsymbol{d}}_2$ similarly; then we continue halving the interval that contains the solution.

Another possibility is to move $\hat{\boldsymbol{d}}_1$ and $\hat{\boldsymbol{d}}_2$ proportionally with the distance ratio

$$\lambda = \frac{\|\boldsymbol{d}_1 - \boldsymbol{d}_2\| - \delta_{\min}}{\|\boldsymbol{d}_1 - \boldsymbol{d}_2\| - \|\hat{\boldsymbol{d}}_1 - \hat{\boldsymbol{d}}_2\|}.$$

The new atoms are obtained via

$$\hat{\boldsymbol{d}}_1 \leftarrow (1-\lambda)\boldsymbol{d}_1 + \lambda\hat{\boldsymbol{d}}_1,$$

$$\hat{\boldsymbol{d}}_2 \leftarrow (1-\lambda)\boldsymbol{d}_2 + \lambda\hat{\boldsymbol{d}}_2.$$

followed by normalization. Since the atoms move on a hypersphere and not linearly, the above computation gives only an approximation and must be repeated a few times. This successive approximation method is faster than bisection, but can give atoms whose distance is slightly smaller than $\delta_{\min}$. So, since there are much more costly operations involved in the DL algorithm, we prefer bisection.

4. Anomaly detection. Benchmarking

We will present anomaly detection results in the context of a recent benchmark library. This section presents the benchmark and the datasets and the next one will give the numerical results.

We use Cone-DL and other DL methods for anomaly detection in the usual way. The representation error is the anomaly score. So, large errors are associated with anomalies and small errors with normal signals; since normal signals are many and more alike, DL produces dictionaries that help their good representation. In principle, the few outliers should have worse representations.

ADBench [1] is a benchmark for anomaly detection problems, comprising a collection of 57 datasets of various types and from various domains, including satellite imagery, financial data, medical imaginary and healthcare tabular data, text, etc.

The benchmark also consists in 30 anomaly detection methods suited for tabular, time series and graph data, mostly inherited from PyOD [35], a popular anomaly detection library. Of interest to our problem are 14 unsupervised methods for tabular data (including images). Their list can be consulted in Table C.7.

Besides the original datasets, ADBench includes methods for generating synthetic datasets with specific anomaly types: global, local, cluster and dependency anomalies [1]. Local anomalies are the most similar to the normal samples, their deviation from the local normal neighbourhood being controlled by scaling the covariance matrix of the normal samples. Global anomalies are uniformly generated in the feature space and thus differ significantly from normal data. Cluster anomalies, as the name suggests, imply grouped anomalies separated from the the normal cluster(s). Dependency anomalies differ from normal samples with respect to the features dependency structure; their probability density function is obtained by removing the dependency modeled for the normal data.

The authors of [1] conclude that none of the unsupervised methods they test achieves good results on all the anomaly types and suggest the need for algorithms that are specialized for a particular type of anomaly.

Global anomalies are relatively easy to find, as for most datasets several benchmark algorithms yield perfect anomaly identification scores. In one particular case (the *cardio* dataset), 6 out of the 14 unsupervised methods detect 100% of the anomalies. Preliminary tests on four of the datasets show that Cone-DL also identifies all of the anomalies, so we do not further investigate this type of anomalies.

Cluster anomalies are not particularly suited for using the representation error as an indicator of anomalies with DL. This is because, in general, at least an atom is learned to represent the samples in the anomalies cluster, leading to good representation for those samples and, in turn, to unsatisfactory anomaly identification. DL (including our cone variant) could still be used in this case, however with other criteria for distinguishing between normal samples and anomalies.

We exclude local anomalies for a similar reason: if anomalies are too similar to the normal signals, they are likely to share atoms and have similar representation errors.

We therefore test our method on dependency anomalies. Figure 6 presents a t-SNE visualization of four of the synthetic dependency datasets and illustrates how normal samples and anomalies span the underlying (albeit reduced here by t-SNE) feature space.

In generating the dependency datasets, we use the default settings in ADBench. Specifically, the synthetic datasets are created by taking the normal samples of each dataset and synthetically generating the anomalies of a particular type. Large datasets are pruned to 10000 samples and small datasets are expanded to 1000 samples by duplicating samples. As with the standard ADBench setup, we use 70% of the signals for training and 30% for testing. We do not test in noisy or corrupted anomalies conditions (i.e. no irrelevant features, duplicated anomalies or annotation errors).

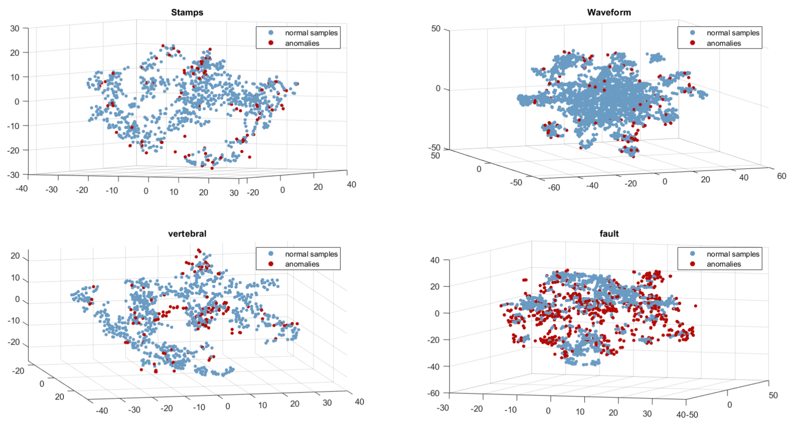

Figure 6: t-SNE illustration of four dependency datasets

The only addition to the original setting is a z-score normalization of the data that we perform prior to the train-test split.

Out of the 57 available datasets, we perform our tests on 30 datasets, chosen as so to include several distinct types of data, with different dataset dimensions and anomaly ratios. In order to suit the sparse representation framework, in all tested datasets the number of features of the signals is larger than 3. The used datasets can be found in Table C.7, sorted by decreasing value of m (the number of features).

5. Numerical results

This section is dedicated to numerical results. First of all, we present an algorithm variation that arised from practical considerations. Then, we will illustrate the general convergence behavior of Cone-DL and associated methods. Finally, the anomaly detection results will be given and commented.

5.1. Radii swap

There are many ways of choosing the cone radii values and we explore two of them in these experiments. The first choice is that of a constant radius for all cones. The other is to generate the radii according to a random uniform distribution in the interval $[\rho_{\min}, \rho_{\max}]$.

In the second case, a situation can arise that small radii atoms are learned to represent normal data, leading to insufficient coverage of the feature space that corresponds to the normal samples. We propose a method for ensuring that atoms get adequate radii according to their usage. After performing dictionary learning, we sort the atoms in descending order by the number of signals that are represented using the atoms. We then attribute the radii in descending order to the sorted atoms and recompute the sparse representation.

After radii redistribution, it is possible that some cones intersect. So, for each pair of superposing cones, we reduce proportionally their radii such that they respect the minimum distance δ_0.

The above addition to a DL algorithm will be indicated by the word 'swap' in its name.

5.2. Convergence behavior

We illustrate here the evolution of the representation error in the Cone-DL algorithm. Naturally, since cones are used, the error is always better than that obtained by a standard DL algorithm like AK-SVD. A more interesting comparison is with the combination of AK-SVD and Cone-OMP; after DL with AK-SVD (which uses standard OMP), the representations are computed with Cone-OMP. We use the same radii distribution for Cone-DL and AK-SVD + Cone-OMP.

Both Cone-DL and AK-SVD are initialized with a dictionary $\boldsymbol{D}_0$, generated as a random matrix with normally distributed entries (zero mean and unit variance), whose columns are normalized. The cone radii are given, either constant or uniformly distributed, as discussed in Section 5.1. If there are superpositions between cones, then a new $\boldsymbol{D}_0$ is generated in the same way. If ten such attempts are unsuccessful, we multiply the radii with a factor of 0.95 and repeat the procedure. Usually, the first attempt is successful, since we used rather small radii; the probability of superposing cones grows with the overcompleteness factor $c = n/m$ and decreases with m.

Figure 7 shows the evolution of the Cone-DL error (per element), averaged over 10 runs with different $\boldsymbol{D}_0$, with $s = 3$, $c = 3$, $\delta_0 = 0.01$, $\rho_{\min} = 0.01$ and $\rho_{\max} = 0.1$. The final error of AK-SVD + Cone-OMP is shown with a horizontal line. Similarly, the errors after the final swap are shown with horizontal lines. The titles of the plots are the names of the datasets. For data with medium and large m, like those in the figure, the typical situation is that Cone-DL reaches a better error than AK-SVD + Cone-OMP. Swapping the radii according to atom use (see Section 5.1), usually improves AK-SVD + Cone-OMP, which is expectable since the initial radii allocation is random. On the contrary, the swap usually produces worse error for Cone-DL; this is again expectable, since Cone-DL optimizes the atoms implicitly taking radii into account; a redistribution of the radii should not improve error if

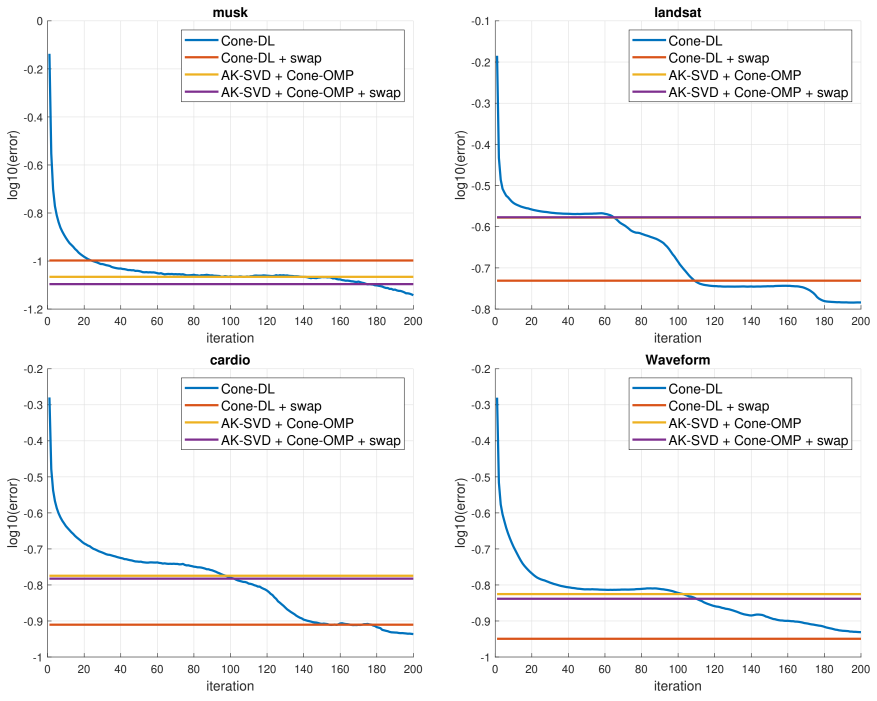

Figure 7: Examples of error evolution, $\rho_{\max} = 0.1$.

Cone-DL works well. The right bottom plot (*Waveform* dataset) shows that when Cone-DL has not reached a minimum, swapping can improve error; the same may happen if Cone-DL reaches a poor local minimum, a situation that can arise when m is small and the overcompleteness factor c is large.

Figure 8 shows similar plots for larger radii, with $\rho_{\max} = 0.2$; the other parameters are unchanged. Now, the differences between Cone-DL and AK-SVD + Cone-OMP are neater. The situations where swapping does not improve AK-SVD + Cone-OMP are also present; however, their occurence is rather scarce.

Tests with artificial data, generated with an underlying sparse representation, show an even neater advantage of Cone-DL. We omit them here.

The overall empirical conclusion is that Cone-DL manages to decrease the representation error, albeit not at each iteration, but consistently, and reaches small values of the error (with no guarantee of a local minimum). Its behavior is similar with that of other DL algorithms.

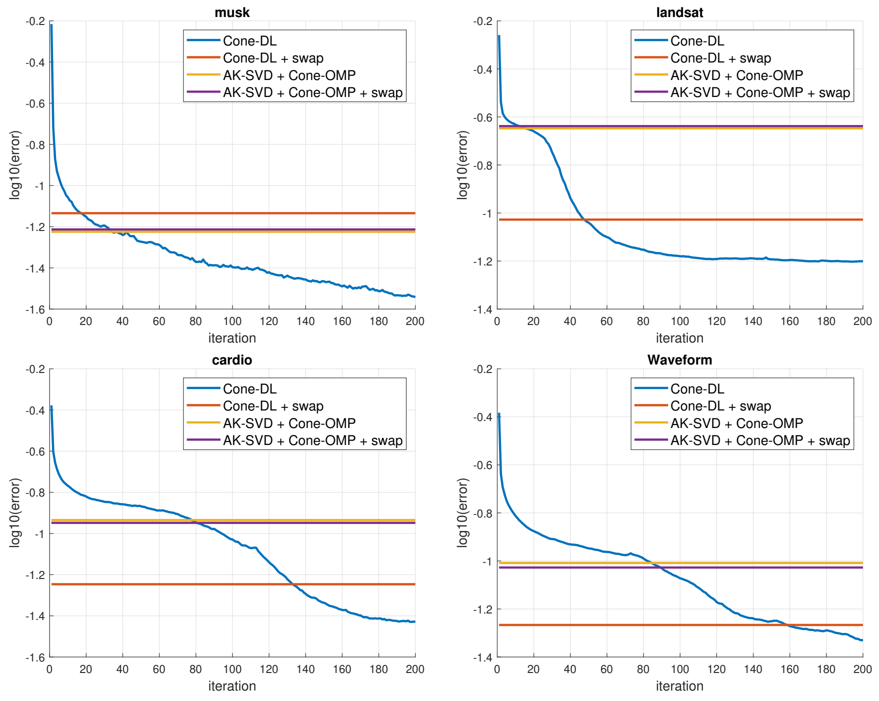

Figure 8: Examples of error evolution, $\rho_{\max} = 0.2$.

5.3. Anomaly detection results

We give now anomaly detection results for Cone-DL and other DL methods and compare them with those of the AD methods from ADBench [1]. Besides Cone-DL, AK-SVD + Cone-OMP and their swap variants, we also use the standard AK-SVD (with OMP as sparse representation algorithm). In all DL methods the dictionary is learned in unsupervised manner using the training set (whose labels are not used). The representation error is then computed on the testing set using the trained dictionary and the appropriate sparse representation method (OMP or Cone-OMP). As discussed in Section 4, the anomaly score of the DL methods is the representation error; larger errors are associated with anomalies.

The figure of merit is ROC AUC (Receiver Operating Characteristic Area Under Curve). We report the results on the test set as means of 10 trials, with different randomly initialized $\boldsymbol{D}_0$ as described in Section 5.2. We took $s \in \{2, 3\}$ and $c \in \{2, 3, 4\}$ (without the combination $s = 3$, $c = 4$). The minimum distance between cones is $\delta_0 = 0.01$. The best raw results for each method and dataset are given in Table B.6; the datasets

Algorithm	Radii	$s = 2$			$s = 3$	
		$c = 2$	$c = 3$	$c = 4$	$c = 2$	$c = 3$
AK-SVD + OMP	-	2.13	1.93	2.07	2.27	2.43
Cone-DL	$\rho = 0.05$	2.03	1.87	1.93	3.23	3.13
AK-SVD + Cone-OMP		1.87	1.80	1.80	2.17	2.47
Cone-DL		2.07	1.93	1.97	4.30	3.60
AK-SVD + Cone-OMP	uniform	1.97	1.87	1.83	2.50	2.53
Cone-DL + swap	$\rho \in [0.01, 0.1]$	1.90	1.73	1.80	3.37	3.13
AK-SVD + Cone-OMP + swap		1.73	1.67	1.67	2.10	2.23

Table 1: Ranking of DL methods in the 14 methods from ADBench, on 30 datasets.

Algorithm	Radii	$s = 2$			$s = 3$	
		$c = 2$	$c = 3$	$c = 4$	$c = 2$	$c = 3$
AK-SVD + OMP	-	2.08	1.92	1.83	1.75	1.75
Cone-DL	$\rho = 0.05$	1.83	1.67	1.67	2.00	1.50
AK-SVD + Cone-OMP		1.92	1.67	1.67	1.58	1.58
Cone-DL		1.83	1.67	1.75	4.00	2.33
AK-SVD + Cone-OMP	uniform	1.83	1.50	1.67	1.58	1.58
Cone-DL + swap	$\rho \in [0.01, 0.1]$	1.75	1.58	1.75	3.08	2.08
AK-SVD + Cone-OMP + swap		1.67	1.58	1.75	1.50	1.50

Table 2: Ranking of DL methods in the 14 methods from ADBench, on the 12 datasets with $m > 20$.

are sorted in descending order of the number of features, m. The best performing overcompleteness and sparsity level combinations are reported for each method. More comments are given in Appendix B. We used the two types of radii described in Section 5.2. When the radius is constant, we took $\rho = 0.05$; of course, swapping radii makes no sense in this case. When the radii are randomly uniform distributed, we took $\rho_{\min} = 0.01$ and $\rho_{\max} = 0.1$; the radii and the initial dictionaries are the same for all methods involving cones (separately for constant and random radii). AK-SVD + OMP is initialized with the same dictionaries as the methods with constant radii.

For comparison, Table C.7 shows the ROC AUC results of the 14 benchmark algorithms from [1] on the 30 synthetic datasets, as well as the mean scores of each method on all selected datasets. Since there are many results, we give below some synthetic performance indicators.

The first is the average ranking of each of our algorithms when individually added to the 14 algorithms tested in ADBench [1]. This is done for each pair s, c separately, such that the comparison is made for each set of parameters. On each dataset, the 15 algorithms are ranked from 1 to 15 based on the ROC AUC performance. Then, the average rank is computed for each algorithm; the best possible value is 1. Table 1 gives the rank averaged for all 30 datasets considered. The DL methods are better than all ADBench method whenever the average rank is less than 2.5; the rank of the best method among the 14 from ADBench, namely LOF (although COF has a better mean ROC AUC, see again Table C.7, LOF has better average rank), varies from 2.6 to 2.93. So, with the exception of Cone-DL for $s = 3$, the DL methods are placed first. We note also that the methods involving cones are better than AK-SVD + OMP when $s = 2$, the case where the best results are obtained.

Table 2 gives the average rank for the 12 datasets with largest number of features ($m > 20$). Here, Cone-DL and AK-SVD + Cone-OMP are tied for the best result, a ranking of 1.5. It can be seen that the best results are now obtained for $s = 3$. A quick inspection of Table B.6 shows that indeed, for large m, the best choice is usually $s = 3$, while for small m it is rather $s = 2$. From both Tables 1 and 2 we notice that swapping improves the results. The improvement is significant when Cone-DL has the weakest results, a possible sign that convergence to a good minimum is not achieved; otherwise, it is marginal. Note however that there is no direct correlation between error and anomaly detection performance, although low error is usually desirable. The second best method is again LOF, whose rank is between 2.33 and 2.75; the gap between DL methods and LOF is usually significant.

Algorithm	Radii	$s = 2$			$s = 3$	
		$c = 2$	$c = 3$	$c = 4$	$c = 2$	$c = 3$
AK-SVD + OMP	-	0.9326	0.9329	0.9328	0.9264	0.9196
Cone-DL	$\rho = 0.05$	0.9347	0.9370	0.9365	0.9023	0.8963
AK-SVD + Cone-OMP		0.9390	0.9395	0.9389	0.9270	0.9182
Cone-DL		0.9332	0.9346	0.9331	0.8868	0.8870
AK-SVD + Cone-OMP	uniform	0.9375	0.9367	0.9369	0.9212	0.9146
Cone-DL + swap	$\rho \in [0.01, 0.1]$	0.9370	0.9400	0.9396	0.9030	0.9006
AK-SVD + Cone-OMP + swap		0.9425	0.9427	0.9443	0.9287	0.9229

Table 3: ROC AUC of DL methods, averaged on 30 datasets.

Tables 3 and 4 show the average ROC AUC values on all 30 and 12 with largest m, respectively. The values confirm the ranking results. First of all,

Algorithm	Radii	$s=2$ $c=2$	$s=2$ $c=3$	$s=2$ $c=4$	$s=3$ $c=2$	$s=3$ $c=3$
AK-SVD + OMP	-	0.9867	0.9875	0.9874	0.9883	0.9886
Cone-DL	$\rho = 0.05$	0.9881	0.9890	0.9888	0.9877	0.9900
AK-SVD + Cone-OMP		0.9883	0.9890	0.9889	0.9897	0.9901
Cone-DL		0.9878	0.9893	0.9891	0.9740	0.9891
AK-SVD + Cone-OMP	uniform	0.9877	0.9887	0.9889	0.9892	0.9902
Cone-DL + swap	$\rho \in [0.01, 0.1]$	0.9889	0.9898	0.9898	0.9825	0.9887
AK-SVD + Cone-OMP + swap		0.9891	0.9897	0.9900	0.9904	0.9908

Table 4: ROC AUC of DL methods, averaged on the 12 datasets with $m > 20$.

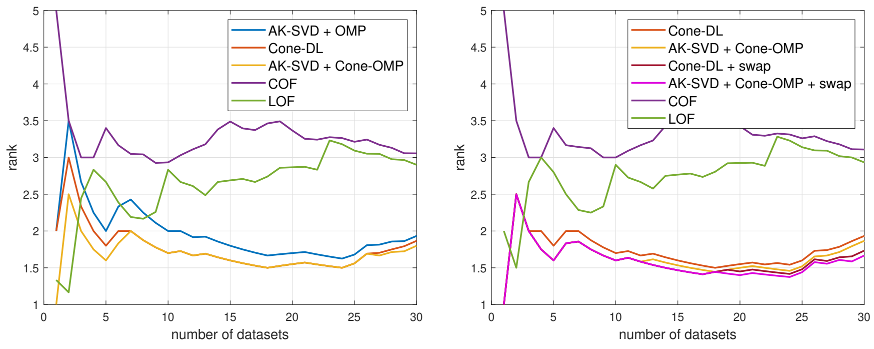

Figure 9: Average rank for the algorithms with constant (left) and random (right) radius, with $s = 2$, $c = 3$. Horizontal axis: number of datasets on which the rank is computed; the datasets are sorted in order of decreasing m.

we note that indeed most of them are cleary larger than the average ROC AUC of COF (0.9274) or LOF (0.9256), the best of the 14 methods from ADBench. Also, we see that the cone methods are better than standard AK-SVD in most cases, although the differences are not very large.

A more general view on the ranking is given by Figure 9. For each $k = 1 : 30$, we compute the average rank of the DL methods on the first k datasets (sorted by decreasing m); we show results only for $s = 2$, $c = 3$. The figure also contains the ranks of LOF and COF, the best competitors from ADBench. With the notable exception of *SpamBase*, DL methods tend to be better than LOF and COF, especially for large m. The plot also confirms the advantage of cone methods over standard AK-SVD.

Table 5 gives the average execution times of the basic DL methods, for the 30 datasets, for training and testing. The MATLAB implementation was run on a Lenovo computer with Intel i7 processor and 32 GB of RAM.

	$s=2$			$s=3$	
Algorithm	$c=2$	$c=3$	$c=4$	$c=2$	$c=3$
AK-SVD + OMP	59.45	76.84	89.70	70.16	92.84
Cone-DL	156.66	166.18	167.16	310.78	314.34

Table 5: Average execution time of DL methods over the 30 datasets.

AK-SVD + Cone-OMP is only slightly slower than AK-SVD + OMP. Similarly, swapping radii adds only negligible extra time. As expected, Cone-DL is clearly slower than AK-SVD. Much of the AK-SVD advantage comes from the efficient batch OMP implementation from [32], while Cone-OMP computes the representations one by one; the lower efficiency of Cone-OMP is especially visible from the difference of the computation times for $s = 2$ and $s = 3$. Also, we use a precompiled OMP. For comparison, COF algorithm runs in 62.65 seconds on all datasets, while LOF in 3.75 seconds. Our programs and the datasets can be found at `http://asydil.upb.ro`.

6. Conclusion

We have presented algorithms for sparse representation and dictionary learning with cone atoms. This dictionary structure offers more versatility in representation and favors the signals that are more alike over those that are different from the rest. So, anomaly detection is a natural application. We have shown that for dependency anomalies, our cone methods are clearly better than those from a popular benchmark. They are also better than standard DL. Although it gives smaller representation errors, the proposed Cone-DL algorithm is slightly inferior in anomaly detection, compared with the simpler variant of using standard DL for training but Cone-OMP for representation. Further work will be dedicated to narrowing down the situations where Cone-DL can be improved, especially by a more adequate radii allocation.

References

[1] S. Han, X. Hu, H. Huang, M. Jiang, Y. Zhao, ADBench: Anomaly Detection Benchmark, in: Neural Information Processing Systems (NeurIPS), 2022.

[2] D. C. Ilie-Ablachim, A. Băltoiu, B. Dumitrescu, Sparse representations with cone atoms, in: IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2023, pp. 1–5.

[3] G. Pope, C. Aubel, C. Studer, Learning phase-invariant dictionaries, in: Int. Conf. Acoustics Speech Signal Proc (ICASSP), Vancouver, Canada, 2013, pp. 5979–5983.

[4] C. Rusu, B. Dumitrescu, S. Tsaftaris, Explicit shift-invariant dictionary learning, IEEE Signal Proc. Letters 24 (2014) 6–9.

[5] A. Song, F. Flores, D. Ba, Convolutional dictionary learning with grid refinement, IEEE Transactions on Signal Processing 68 (2020) 2558–2573.

[6] Y. S. Soh, Group invariant dictionary learning, IEEE Transactions on Signal Processing 69 (2021) 3612–3626.

[7] C. Garcia-Cardona, B. Wohlberg, Convolutional dictionary learning: A comparative review and new algorithms, IEEE Transactions on Computational Imaging 4 (2018) 366–381.

[8] L. Le Magoarou, R. Gribonval, Chasing Butterflies: in Search of Efficient Dictionaries, in: Int. Conf. Acoustics Speech Signal Proc (ICASSP), Brisbane, Australia, 2015, pp. 3287–3291.

[9] J. Song, X. Xie, G. Shi, W. Dong, Multi-layer discriminative dictionary learning with locality constraint for image classification, Pattern Recognition 91 (2019) 135–146.

[10] J. Mairal, M. Elad, G. Sapiro, Sparse representation for color image restoration, IEEE Trans. Image Proc. 17 (2008) 53–69.

[11] S. Hawe, M. Seibert, M. Kleinsteuber, Separable dictionary learning, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 438–445.

[12] M. Herman, T. Strohmer, General Deviants: An Analysis of Perturbations in Compressed Sensing, IEEE J. Sel. Topics Signal Proc. 4 (2010) 342–349.

[13] H. Zhu, G. Leus, G. Giannakis, Sparsity-Cognizant Total Least-Squares for Perturbed Compressive Sampling, IEEE Trans. Signal Proc. 59 (2011) 2002–2016.

[14] M. Ghasemi, M. Kelarestaghi, F. Eshghi, A. Sharifi, T2-FDL: a robust sparse representation method using adaptive type-2 fuzzy dictionary learning for medical image classification, Expert Systems with Applications 158 (2020) 113500.

[15] G. Tang, B. Bhaskar, B. Recht, Sparse recovery over continuous dictionaries-just discretize, in: 2013 Asilomar Conference on Signals, Systems and Computers, IEEE, 2013, pp. 1043–1047.

[16] E. Candes, C. Fernandez-Granda, Towards a Mathematical Theory of Super-resolution, Comm. Pure Appl. Math. 67 (2014) 906–956.

[17] J. Hu, Y.-P. Tan, Nonlinear dictionary learning with application to image classification, Pattern Recognition 75 (2018) 282–291.

[18] S. Mahdizadehaghdam, A. Panahi, H. Krim, L. Dai, Deep dictionary learning: A parametric network approach, IEEE Trans. Image Proc. 28 (2019) 4790–4802.

[19] L. Ruff, J. Kauffmann, R. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. Dietterich, K. Müller, A unifying review of deep and shallow anomaly detection, Proceedings of the IEEE 109 (2021) 756–795.

[20] W. Hilal, S. Gadsden, J. Yawney, Financial fraud: a review of anomaly detection techniques and recent advances, Expert systems With applications 193 (2022) 116429.

[21] K. A. da Costa, J. P. Papa, L. A. Passos, D. Colombo, J. Del Ser, K. Muhammad, V. H. C. de Albuquerque, A critical literature survey and prospects on tampering and anomaly detection in image data, Applied Soft Computing 97 (2020) 106727.

[22] B. Dumitrescu, P. Irofti, Dictionary Learning Algorithms and Applications, Springer, 2018.

[23] A. Adler, M. Elad, Y. Hel-Or, E. Rivlin, Sparse coding with anomaly detection, Journal of Signal Processing Systems 79 (2015) 179–188.

[24] T. Andrysiak, Sparse representation and overcomplete dictionary learning for anomaly detection in electrocardiograms, volume 32, 2020, pp. 1269–1285.

[25] P. Irofti, A. Pătraşcu, A. I. Hîji, Unsupervised abnormal traffic detection through topological flow analysis, in: 2022 14th International Conference on Communications (COMM), 2022, pp. 1–6.

[26] Y. Yuan, D. Ma, Q. Wang, Hyperspectral anomaly detection via sparse dictionary learning method of capped norm, IEEE Access 7 (2019) 16132–16144.

[27] X. Han, H. Zhang, W. Sun, Spectral anomaly detection based on dictionary learning for sea surfaces, IEEE Geoscience and Remote Sensing Letters 19 (2021) 1–5.

[28] B. Pilastre, L. Boussouf, S. d'Escrivan, J. Tourneret, Anomaly detection in mixed telemetry data using a sparse representation and dictionary learning, Signal Processing 168 (2020) 107320.

[29] A. Băltoiu, A. Pătrașcu, P. Irofti, Graph anomaly detection using dictionary learning, IFAC-PapersOnLine 53 (2020) 3551–3558.

[30] S. Li, Concise formulas for the area and volume of a hyperspherical cap, Asian Journal of Mathematics and Statistics 4 (2011) 66–70. URL: `http://docsdrive.com/pdfs/ansinet/ajms/2011/66-70.pdf`.

[31] Y. Pati, R. Rezaiifar, P. Krishnaprasad, Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition, in: 27th Asilomar Conf. Signals Systems Computers, volume 1, 1993, pp. 40–44.

[32] R. Rubinstein, M. Zibulevsky, M. Elad, Efficient Implementation of the K-SVD Algorithm Using Batch Orthogonal Matching Pursuit, Technical Report CS-2008-08, Technion Univ., Haifa, Israel, 2008.

[33] D. Barchiesi, M. Plumbley, Learning Incoherent Dictionaries for Sparse Approximation Using Iterative Projections and Rotations, IEEE Trans. Signal Proc. 61 (2013) 2055–2065.

[34] B. Mailhé, D. Barchiesi, M. Plumbley, INK-SVD: Learning incoherent dictionaries for sparse representations, in: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, Kyoto, Japan, 2012, pp. 3573–3576.

[35] Y. Zhao, Z. Nasrullah, Z. Li, PyOD: A Python Toolbox for Scalable Outlier Detection, Journal of Machine Learning Research 20 (2019) 1–7. URL: `http://jmlr.org/papers/v20/19-011.html`.

Appendix A. The volume of linear combinations of s cones

We consider the simplifying case of cones with equal radii and orthogonal central atoms. Let $\boldsymbol{d}_i$, $i = 1 : s$, be orthogonal vectors with unit norm. Without loss of generality, we can assume that $\boldsymbol{d}_i$ is the unit vector of index i. Consider linear combinations

$$\boldsymbol{y} = \sum_{i=1}^{s} \alpha_i \boldsymbol{a}_i, \tag{A.1}$$

with $\boldsymbol{a}_i \in \mathcal{C}(\boldsymbol{d}_i, \rho)$. Since we are interested in the volume occupied by vectors $\boldsymbol{y}$ in the unit hypersphere, we have $\|\boldsymbol{y}\| \leq 1$. It is obvious that the first s elements of $\boldsymbol{y}$ can have all values that are possible in the unit hypersphere. So, we focus on the last $m - s$ elements.

We can write

$$\boldsymbol{a}_i = \beta_i \boldsymbol{d}_i + \gamma_i \boldsymbol{v}_i,$$

with $\boldsymbol{v}_i \perp \boldsymbol{d}_i$, $\|\boldsymbol{v}_i\| = 1$ and $\beta_i^2 + \gamma_i^2 = 1$. Since

$$\|\boldsymbol{a}_i - \boldsymbol{d}_i\| \leq \rho \Rightarrow \|(\beta_i - 1)\boldsymbol{d}_i + \gamma_i \boldsymbol{v}_i\|^2 \leq \rho^2 \Rightarrow (\beta_i - 1)^2 + \gamma_i^2 \leq \rho^2 \Rightarrow \beta_i \geq 1 - \rho^2/2$$

it follows that

$$|\gamma_i| = \sqrt{1 - \beta_i^2} \leq \rho\sqrt{1 - \rho^2/4}.$$

The component of (A.1) that lies in the space formed by the last $m - s$ dimensions of $\mathbb{R}^m$ is at most

$$\boldsymbol{\nu} = \sum_{i=1}^{s} \alpha_i \gamma_i \boldsymbol{v}_i.$$

For small ρ, the vectors $\boldsymbol{a}_i$ are nearly orthogonal, so we have

$$\sum_{i=1}^{s} \alpha_i^2 \leq \mu \Rightarrow \sum_{i=1}^{s} |\alpha|_i \leq \sqrt{\mu s},$$

where μ is a constant (depending on ρ) not much larger than 1. It results that each element $k \in s + 1 : m$ of $\boldsymbol{\nu}$ satisfies

$$|\nu_k| \leq \sum_{i=1}^{s} |\alpha_i \gamma_i| \leq \sqrt{\mu s} \cdot \rho\sqrt{1 - \rho^2/4} \leq \rho\sqrt{\mu s}. \tag{A.2}$$

Finally, the volume of the vectors (A.1) is less than

$$v_s^b (2\rho\sqrt{\mu s})^{m-s}, \tag{A.3}$$

where v_s^b is the volume of the unit ball in $\mathbb{R}^s$. This is in fact the volume of the hypercylinder having as "basis" the unit ball in $\mathbb{R}^s$ (accounting for the first s elements of $\boldsymbol{y}$) and as "height" the other $m - s$ elements, each bounded by (A.2). The part of the hypersphere in which $\boldsymbol{y}$ lies is inside this hypercylinder.

It is clear that the volume (A.3) is much smaller than the volume of the unit hypersphere in $\mathbb{R}^m$ whenever s and ρ are small, which is the practical case when using sparse representations with cone atoms.

Appendix B. Detailed DL dependency results

Table B.6 presents the best ROC AUC result of each DL method. The first two columns are the name of the dataset and the number of features, m. A group of three columns is associated with each method and radii distribution: the values of the overcompleteness factor c and sparsity level s for which the best ROC AUC value is obtained and the ROC AUC value itself. Here are a few comments on the brute results.

Out of the two parameters c and s, the algorithms are more robust with respect to sparsity: in 83% of the datasets all constant radii cone methods yield best results for the same sparsity level ($s = 2$), while in the case of randomly uniform radii algorithms the percentage is 60%. With overcompleteness, consensus is reached beetween methods in 50% of the datasets regardless of the type of radii used.

Radii swapping doesn't affect greatly the best performing parameters: Cone-DL and its reordered radii variant show the best results with the same sparsity level in all but two datasets and with the same overcompleteness in 77% of the times.

Appendix C. ADBench dependency results

Table C.7 gives the ROC AUC values for the 14 methods from ADBench [1]. They have been obtained by running the corresponding PyOD functions, according to the ADBench methodology, with default PyOD paramaters for the algorithms (when applicable). Nondeterministic methods are ran once. The dependency datasets have been generated as described in Section 4, again using the ADBench framework.

The results show that there are significant differences between the methods, confirming the comments from [1], where the results are only synthetically presented. Some of the methods, like COF, LOF, KNN and SOD

Table B.6: DL ROC AUC results on dependency datasets

Dataset	m	AK-SVD + OMP			Radii $\rho = 0.05$						Radii uniform $\rho \in [0.01, 0.1]$											
					Cone-DL			AK-SVD + Cone-OMP			Cone-DL			AK-SVD + Cone-OMP			Cone-DL + swap			AK-SVD + Cone-OMP + swap		
		c	s	ROC AUC	c	s	ROC AUC	c	s	ROC AUC	c	s	ROC AUC	c	s	ROC AUC	c	s	ROC AUC	c	s	ROC AUC
musk	166	3	3	0.9994	3	3	0.9994	3	3	0.9997	3	2	0.9995	3	3	0.9996	3	2	0.9995	3	3	0.9997
SpamBase	57	3	3	0.8955	3	3	0.9040	3	3	0.9042	3	3	0.9095	3	3	0.9043	3	3	0.9093	3	3	0.9092
landsat	36	4	2	0.9998	4	2	0.9998	4	2	0.9999	3	2	0.9997	3	3	0.9999	3	2	0.9997	3	3	0.9999
satellite	36	3	3	0.9999	3	2	1	3	3	1	2	2	1	4	2	1	2	2	1	4	2	1
Satimage-2	36	2	3	1	3	3	1	2	3	1	3	3	1	3	3	1	3	3	1	3	3	1
WPBC	33	4	2	0.9966	4	2	0.9972	4	2	0.9970	3	3	0.9976	4	2	0.9976	3	3	0.9979	2	3	0.9980
ionosphere	33	3	3	0.9909	3	3	0.9931	3	3	0.9930	3	3	0.9925	3	3	0.9927	3	2	0.9929	3	3	0.9939
letter	32	3	3	0.9996	3	3	0.9997	3	3	0.9997	3	3	0.9998	3	3	0.9996	3	3	0.9997	3	3	0.9996
WDBC	30	3	3	1	3	2	1	2	3	1	2	3	1	2	2	1	2	3	1	2	2	1
fault	27	2	3	0.9959	3	3	0.9960	2	3	0.9962	4	2	0.9965	2	2	0.9965	3	2	0.9970	2	2	0.9971
cardio	21	2	3	0.9888	3	3	0.9920	2	3	0.9942	3	2	0.9898	3	3	0.9946	3	2	0.9897	3	3	0.9943
Waveform	21	3	3	0.9999	4	2	0.9997	3	3	0.9999	3	2	0.9998	3	3	1	3	2	0.9998	3	3	1
Hepatitis	19	2	3	0.9096	3	3	0.9143	2	3	0.9231	3	3	0.9245	2	3	0.9247	2	2	0.9219	2	2	0.9247
Lymphography	18	4	2	0.9682	4	2	0.9683	4	2	0.9753	3	3	0.9730	3	3	0.9820	3	3	0.9710	3	3	0.9794
pendigits	16	4	2	0.9993	4	2	0.9995	4	2	0.9995	4	2	0.9994	3	2	0.9993	4	2	0.9992	4	2	0.9992
wine	13	2	3	0.9866	2	3	0.9888	3	3	0.9911	4	2	0.9879	4	2	0.9898	4	2	0.9890	3	3	0.9912
vowels	12	2	3	0.9959	3	2	0.9967	2	3	0.9985	4	2	0.9954	2	3	0.9983	2	2	0.9955	2	3	0.9979
cover	10	3	3	0.9967	4	2	0.9975	3	2	0.9976	4	2	0.9978	4	2	0.9978	4	2	0.9969	4	2	0.9971
magic.gamma	10	2	3	0.9623	4	2	0.9608	2	3	0.9716	2	2	0.9610	2	3	0.9670	4	2	0.9671	4	2	0.9741
breastw	9	3	2	0.8255	3	2	0.8324	3	2	0.8338	4	2	0.8432	4	2	0.8430	4	2	0.8549	4	2	0.8600
Stamps	9	4	2	0.8985	4	2	0.8987	4	2	0.9077	4	2	0.8878	4	2	0.8957	4	2	0.9011	4	2	0.9136
WBC	9	3	3	0.9013	4	2	0.9194	4	2	0.9223	3	2	0.9022	2	3	0.9160	4	2	0.9032	4	2	0.9136
Pima	8	2	2	0.8308	2	2	0.8440	2	2	0.8614	2	2	0.8468	4	2	0.8614	4	2	0.8649	4	2	0.8841
yeast	8	2	2	0.9291	2	2	0.9302	2	2	0.9362	4	2	0.9127	2	2	0.9309	4	2	0.9280	4	2	0.9457
glass	7	3	2	0.8738	4	2	0.8921	3	2	0.8853	4	2	0.8658	2	2	0.8743	4	2	0.8767	4	2	0.8757
annthyroid	6	4	2	0.8765	3	2	0.8783	2	2	0.8869	2	2	0.8770	3	2	0.8810	3	2	0.8813	4	2	0.8907
mammography	6	4	2	0.8687	4	2	0.8680	4	2	0.8797	3	2	0.8609	3	2	0.8692	3	2	0.8693	3	2	0.8758
thyroid	6	2	2	0.8763	3	2	0.8798	2	2	0.8894	3	2	0.8815	2	2	0.8985	3	2	0.8850	2	2	0.9024
vertebral	6	2	2	0.8688	2	2	0.8720	3	2	0.8783	2	2	0.8645	2	2	0.8660	2	2	0.8852	2	2	0.8948
Wilt	5	2	2	0.6922	4	2	0.6722	2	2	0.7035	3	2	0.6736	2	2	0.6905	3	2	0.6895	2	2	0.7129

Table C.7: ADBench ROC AUC results on dependency datasets

Dataset	IForest	OCSVM	CBLOF	COF	COPOD	ECOD	HBOS	KNN	LODA	LOF	PCA	SOD	DeepSVDD	DAGMM
musk	0.9518	0.9614	0.9709	0.9925	0.6444	0.6084	0.6508	0.9982	0.7639	0.9994	0.6514	0.9976	0.8402	0.6616
SpamBase	0.7260	0.7855	0.8590	0.9618	0.6022	0.5306	0.6628	0.8900	0.5054	0.9645	0.7558	0.9190	0.6125	0.6296
landsat	0.9276	0.9897	0.9753	0.9988	0.6589	0.6512	0.6328	0.9985	0.7734	0.9952	0.6832	0.9955	0.6707	0.7337
satellite	0.9395	0.9784	0.9852	0.9982	0.5885	0.5731	0.6253	0.9999	0.7206	0.9977	0.5435	0.9969	0.5059	0.6813
Satimage-2	0.9783	0.9912	0.9883	0.9957	0.5675	0.5480	0.6267	0.9991	0.7938	0.9994	0.5370	0.9981	0.9275	0.6552
WPBC	0.8440	0.9005	0.9251	0.9969	0.6131	0.5522	0.5646	0.9927	0.6712	0.9994	0.6146	0.9803	0.8018	0.5157
Ionosphere	0.8580	0.8803	0.9479	0.9951	0.6856	0.6748	0.6450	0.9912	0.7055	0.9951	0.7284	0.9826	0.7081	0.4513
letter	0.9220	0.9716	0.9696	0.9936	0.6801	0.6751	0.6471	0.9933	0.7652	0.9973	0.7008	0.9935	0.8974	0.5327
WDBC	0.9452	0.9786	0.9726	0.9991	0.7539	0.7350	0.6931	0.9966	0.9225	0.9983	0.8476	0.9949	0.9799	0.8399
fault	0.9627	0.9797	0.9856	0.9912	0.8876	0.8943	0.8577	0.9948	0.8944	0.9525	0.9232	0.9812	0.5222	0.7300
cardio	0.8255	0.8615	0.9287	0.9543	0.6305	0.6227	0.6173	0.9579	0.6909	0.9915	0.6537	0.9471	0.7194	0.6444
Waveform	0.8617	0.9251	0.9484	0.9895	0.6505	0.6562	0.6342	0.9921	0.7319	0.9982	0.6613	0.9856	0.7513	0.7652
Hepatitis	0.7048	0.7276	0.8034	0.8383	0.5589	0.5564	0.5475	0.8332	0.5496	0.9194	0.5796	0.8424	0.3691	0.4884
Lymphography	0.8100	0.8309	0.8789	0.8419	0.5607	0.5419	0.4792	0.8901	0.5835	0.8783	0.6065	0.8901	0.5937	0.5127
pendigits	0.9385	0.9466	0.9579	0.9808	0.6457	0.6407	0.5188	0.9964	0.7744	0.9949	0.6229	0.9925	0.7406	0.6542
wine	0.8029	0.8379	0.8532	0.9746	0.5950	0.5939	0.5713	0.9173	0.6806	0.9637	0.6046	0.9520	0.6898	0.4343
vowels	0.8009	0.8543	0.9030	0.9387	0.5532	0.5872	0.5777	0.9370	0.6796	0.9622	0.5888	0.9186	0.7667	0.7065
cover	0.8341	0.8331	0.8639	0.9707	0.5227	0.5247	0.5685	0.9929	0.5694	0.9916	0.5464	0.9958	0.7605	0.5354
magic.gamma	0.7431	0.7578	0.8240	0.9346	0.5783	0.5803	0.5368	0.9725	0.6131	0.8632	0.6127	0.9403	0.3926	0.3555
breastw	0.6960	0.6863	0.7413	0.8378	0.6206	0.6172	0.5994	0.8006	0.5833	0.8129	0.6303	0.8043	0.4059	0.5128
Stamps	0.7304	0.7582	0.8118	0.9068	0.6334	0.6386	0.6227	0.8335	0.7023	0.8858	0.6786	0.8832	0.6001	0.4534
WBC	0.6490	0.6307	0.6848	0.8438	0.5083	0.5007	0.4983	0.7610	0.4455	0.8534	0.4886	0.7838	0.3728	0.4669
Pima	0.5728	0.5757	0.6714	0.8505	0.4682	0.4444	0.4659	0.7755	0.4916	0.8214	0.4654	0.7933	0.4720	0.4555
yeast	0.8112	0.7932	0.8629	0.8702	0.6916	0.6998	0.6688	0.9146	0.6430	0.6681	0.7168	0.8797	0.4117	0.3731
glass	0.6457	0.6752	0.7290	0.8185	0.4902	0.5334	0.4514	0.7711	0.5366	0.8507	0.5789	0.7593	0.5433	0.5637
annthyroid	0.6980	0.7181	0.7806	0.9336	0.5982	0.5875	0.5475	0.8909	0.6028	0.9368	0.6206	0.8882	0.4824	0.5441
mammography	0.7771	0.7338	0.8058	0.9272	0.5565	0.5466	0.5323	0.9460	0.5388	0.9525	0.5619	0.9615	0.6428	0.4858
thyroid	0.6605	0.7000	0.7366	0.8843	0.5706	0.5948	0.5639	0.8396	0.5581	0.9144	0.6255	0.8248	0.4385	0.5070
vertebral	0.7421	0.7653	0.8002	0.8795	0.5672	0.5996	0.5796	0.8588	0.6270	0.8611	0.5932	0.8177	0.4288	0.4473
Wilt	0.5671	0.5912	0.6172	0.7242	0.5257	0.5396	0.5214	0.7266	0.5091	0.7479	0.5502	0.6690	0.4390	0.4826
Mean	**0.7976**	**0.8206**	**0.8594**	**0.9274**	**0.6069**	**0.6016**	**0.5903**	**0.9154**	**0.6542**	**0.9256**	**0.6324**	**0.9123**	**0.6162**	**0.5607**

behave very well, while many of the others have rather poor results. The results are consistent with those from [1, Fig.5c], where the same four methods occupy the first four positions (on 57 datasets, without details on individual datasets results).

As in the case of the DL algorithms, the ADBench results depend on the signal dimension m, in the sense that better detection is obtained, in general, for the datasets with larger m.