Graph Representation and Feature Selection using Sparse Coding and Dictionary Learning. Application to clustering in Cybersecurity.

Barbara Pilastre, Tristan Bitard-Feildel

***Abstract*—In recent year, graph signal processing have received an increasing interest in many fields and cybersecurity has not escaped to this trend as complex cybersecurity data can often be depicted as graph. This paper introduces GABAIN (Graph leArning Based on spArse codINg), a new graph representation method based on sparse coding. The method takes advantage of sparse coding to perform similarity search and infers relationships between sample data preserving the local structure of the data. Additionally, the proposed method includes a dictionary learning step with feature selection constraint via $\ell_{2,1}$ norm regularization to identify informative features and offer some explainability and data knowledge. We evaluate the proposed method for clustering application in cybersecurity use cases and demonstrate the competitiveness of the proposed method and the efficiency of its feature selection option.**

***Index Terms*—Similarity Graph Leaning, Sparse Coding, Dictionary Learning, Clustering, Explainability.**

I. Introduction

GRAPH signal processing is an active area of research for many applications such as anomaly detection, classification or pattern recognition [1], [2]. Indeed, representing data as graph can be very interesting to highlight some useful structure for processing, analysis and visualization of the data. Formally, the problem of data representation as graph is the following : transform a dataset classically represented as matrix data $\boldsymbol{Y} \in \mathbb{R}^{N \times P}$ with N observations of P variables or features into a graph composed of nodes connected by edges.

Graph representation appears straightforward in various application domains where relational data is available [3]. As example, it is quite classical to represent social media data as graph in which users are represented as nodes and edges describe relationships or interactions between them [4], [5]. In the same way in cybersecurity for intrusion detection [6], system activity data is regularly represented as graph in which system objects (*Process*, *Thread* or *Files*) are represented as nodes and edges describe actions between them (creation, reading, etc.). In other cases, the representation as graph is not obvious. To address this drawback, numerous approaches, referred as graph learning (GL), have been designed to infer relationships between features (i.e., variables of the original dataset are represented as nodes in the learned graph). They are commonly based on statistical or physical motivated models and are useful for applications such as image encoding and compression applications or brain signal analysis [7]. In contrast, clustering or outlier detection applications often infer relationships between observations by exploiting data similarity. Resulting graph, called similarity graph, is a directed and weighted graph in which each node corresponds to a sample or observation of the original dataset, and weighted edges connect similar samples. For clustering task, for instance, the similarity graph learning (SGL) strategy as data preproccessing step can be interesting since the clustering can be performed on the learned graph using efficient graph clustering or community detection methods that not require the number of cluster to be provided a priory. This paper introduces a new SGL method based on sparse coding. The proposed method, inspired by the works conducted in [8], [9], takes advantage of sparse coding to detect similar data and capture the local structure of the data.

Furthermore, explanability and data knowledge are important issues in many field, especially in cybersecurity. Indeed, cybersecurity analyst require to be provided with explanations about system alerts or events that occurred to make decisions fixing problems and preserving system performance and functionality. Thus, although automatic processing can be very helpful, especially to deal with large amounts of data, it is important to make available model explanability to understand how processing end result is reached and improve data knowledge. To this end, a good strategy is to identify informative features using feature selection techniques based on sparse coding and dictionary learning [10], [11]. Aware of this, we augmented the GABAIN method with a selection feature option using dictionary learning step via a $\ell_{2,1}$ norm regularization. Finally, the interest of the GABAIN method is twofold : the similarity graph learning to represent original matrix data as graph on one hand, and feature selection on the other.

To sum-up, the main contributions of this paper are:

- A similarity graph learning preserving the local structure of the data based on sparse coding.
- A feature selection using dictionary learning.
- An evaluation of the proposed method on clustering problems in cybersecurity applications and a comparison to the state-of-the-art approaches.

The paper is organized as follows. Similarity graph learning methods and feature selection approaches are reviewed in section 2. Section 3 details the proposed method and its optimization. Experimental results from clustering problems

Barbara Pilastre and Tristan Birtard Feildel are with Agence Ministerielle de l'Intelligence Artificielle de Defense, French Ministry of Defense (MoD), France.

in cybersecurity domain are compared to the state-of-the art in section 4. Finally, conclusion and future works are given in section 5.

II. Related Works

A. graph learning from data

Graph data representation is often very interesting since the obtained data structure can improve processing, analysis and visualization of the data. A generalized graph representation method can represent a real challenge given the variety of application domains. We distinguish two main strategies to infer graph from data : graph learning (GL) [7], which infer relationships between features, and similarity graph learning (SGL) [12], which infer relationships between data samples. This paper focus on SGL in order to investigate solutions for clustering applications. As a results of SGL, each rows of the original data matrix $\boldsymbol{Y} \in \mathbb{R}^{N \times P}$, is defined in the node set of the estimated graph. Node connections by weighted edges are given using a similarity function or distance metric. Simple SGL methods build fully connected graph. The idea is to use similarity function to compute similarity matrix and infer a fully connected graph with weighted edges. Weights correspond to the similarity measure of each pair of nodes. The Gaussian kernel is one of the most used similarity function [13] since it allows complex and non-linear relationships to be captured and local structure of the data to be preserved. As demonstrated in [14] for clustering application, the Laplacian kernel can be interesting. Popular SGL methods are based on nearest neighbors approaches. A nearest-neighbor graph (NNG) [15] is a sparse graph in which nodes are connected to its nearest neighbor, i.e., to the data sample of $\boldsymbol{Y}$ whose similarity from it is maximum among all the given points other than itself. An extension to the NNG is the k-NNG [16] in which each node is connected to its k nearest neighbors, meaning the k samples whose similarity is in the top-k highest similarity from it. In the recent years, many variants of the K-NNG have been proposed. Some of them are devoted to adjust the k value [17], [18], [19], [20] and other defend the use of other similarity function, especially the Gaussian similarity function whose the parameter σ can be better tuned [21], [22], [23]. Other well used SGL methods are sparse subspace similarity graph building methods. These methods assume that the data points close to each other can be expressed in the data space through the linear combination of them [24]. These approach based on sparse coding define an objective function to exploit self-expressiveness of the data and develop strategies to preserve local and global structure of the data [25] or to make the sparse representation more discriminative [26].

B. Feature selection

Feature selection is one of the two main dimensionality reduction strategies used to reduce the number of features in a dataset. There are multiple benefit attached to reduction dimension since it can solve problems of data sparsity in high dimension (curse of dimensionality) [27], scalability or explainability. As mentioned, dimensionality reduction methods can be divided into two categories: feature extraction (FE) and feature selection (FS) [10], [11]. FE is to find a new lower-dimensional space than the original space and project the original data in such new space. The new data are described by features of the new space estimated by linear or non-linear transformation. These techniques obtained promising results on real-world data, especially the non-linear transformation strategies[28], but they raise difficult questions about results interpretation since features of the new space, computed from a mathematical optimization problem, are no longer interpretable. Conversely, FS methods aims to identify and select relevant features in the original space and eliminate less informative ones. As original data have commonly be extracted from operational process or physical measurements, FS ensures an easier understanding and interpretation of the results. FS approaches, which are, for the main part, based on sparse coding are studied in this paper.

Traditional unsupervised FS algorithms aim to remove redundant or irrelevant features from the dataset. The selected features should preserve the overall data structure and properties and are more discriminatory with respect to a given criteria than the removed features. Most FS algorithms can be categorized into sparse coding or dictionary learning approaches. Each method formulates an objective function as an optimization problem including a $\ell_{2,1}$-norm regularization on which the FS is based. The approaches based on sparse coding select features by estimating a FS matrix which is sparse in rows. It is the case of UDF [29] which proposes a linear classifier based on sparse coding to exploit discriminatory information of the data. NDFS [30] extends UDF [29] by adding spectral clustering to exploit better these information. Regularized models are also proposed to preserve structure of the data. JELSR [31] takes advantage of spectral regression and Laplacian matrix to preserve manifold of the data. SRFS [32] preserves the local structure of features by sparse regression and the global structure among samples and among features using a low-rank constraint. LDSSL [33] adopts a twofold strategy and proposes a sparse representation as linear model that preserves both the local informative structure and local geometric structure of the data.

The methods based on dictionary learning approaches try to learn a basis matrix with feature selection and provide a new data representation along with the elimination of uninformative or redundant features. The DLUFS [34] method uses a dictionary learning approach with a low-rank constraint to obtain a sparse representation of the data and eliminate uninformative and redundant features. In CDLFS [35], dictionary learning aims to select the features that can well preserve the data distribution. According to recent results, the DLUFS [34] solution is FS state-of-the-art. Unfortunately, the actual implementation of the DLUFS method is not scalable and could not be applied to the data of these experiments that can count hundreds samples.

III. The GABAIN Algorithm

A. Notations

We summarize notations used in this paper in Table I.

TABLE I: Used notations.

λ	scalars
$\boldsymbol{v}$	vector v
$\boldsymbol{v}_i$	the ith element of $\boldsymbol{v}$
$\boldsymbol{M}$	matrix M
$\boldsymbol{M}^T$	the transpose of M
$\boldsymbol{M}^{-1}$	the inverse of M
$\boldsymbol{M}(i,j)$	the element in the ith row and jth column of M
$\|\boldsymbol{v}\|_1$	the l_1-norm of $\boldsymbol{v}$, i.e., $\|\boldsymbol{v}\|_1 = \sum_i \|\boldsymbol{v}_i\|$
$\|\boldsymbol{M}\|_{1,1}$	the $l_{1,1}$-norm of M, i.e., $\|\boldsymbol{M}\|_{1,1} = \sum_j \|\boldsymbol{M}(:j)\|_1$
$\|\boldsymbol{M}\|_{2,1}$	the $l_{2,1}$-norm of M, i.e., $\|\boldsymbol{M}\|_{2,1} = \sum_i \sqrt{\sum_j \boldsymbol{M}(i,j)^2}$
$\|\boldsymbol{M}\|_F$	the Frobenius norm of M, i.e., $\|\boldsymbol{M}\|_F = \sqrt{\sum_{i,j} \boldsymbol{M}(i,j)^2}$

B. Sparse Representation & Feature Selection

The main idea of the proposed sparse coding based graph data representation is to express each of the N data samples of the data matrix $\boldsymbol{Y} \in \mathbb{R}^{N\times P}$ as a sparse linear combination of the rest of the dataset. More precisely, we consider the transpose matrix of $\boldsymbol{Y}$, denoted as $\boldsymbol{Y}^T$ and a data-driven dictionary $\boldsymbol{D} \in \mathbb{R}^{P\times L}$ ($L < N$) corresponding to a subset of $\boldsymbol{Y}^T$ since $\boldsymbol{D}$ is composed of L atoms (i.e., columns) drawn randomly from the data $\boldsymbol{Y}^T$. This dictionary exploits the self-expressiveness of the data to represent data samples by the others. Using a sparsity constraint via a ℓ_1 norm regularization, the number of samples used in the representation is limited in order to promote similarity search. The highest coefficients of the resulting sparse representation allow pairs of similar data to be detected and edges of the graph to be identified. Furthermore, in order to identify the more relevant features, the GABAIN algorithm includes an optional dictionary learning step. The idea is to divide the dictionary $\boldsymbol{D}$ into two dictionaries, $\boldsymbol{D}_1 \in \mathbb{R}^{P\times L}$ and $\boldsymbol{D}_2 \in \mathbb{R}^{P\times L}$, sparse in rows such as $\boldsymbol{D} = \boldsymbol{D}_1 + \boldsymbol{D}_2$. This dictionary step aims to identify and distinguish informative features which will compose dictionary $\boldsymbol{D}_1$ from uninformative features which will compose dictionary $\boldsymbol{D}_2$. This dictionary learning step is performed via a suitable $\ell_{2,1}$ norm regularization. Finally, The proposed strategy decomposed the data as follows:

$$\boldsymbol{Y}^T = \boldsymbol{D}_1\boldsymbol{X}_1 + \boldsymbol{D}_2\boldsymbol{X}_2 + \boldsymbol{B} \tag{1}$$

where $\boldsymbol{X}_1 \in \mathbb{R}^{L\times N}$ is the sparse coefficient matrix corresponding to the representation of the data matrix $\boldsymbol{Y}$ in the space of the dictionary $\boldsymbol{D}_1$, $\boldsymbol{X}_2 \in \mathbb{R}^{L\times N}$ is the coefficient matrix corresponding to the representation of the data matrix $\boldsymbol{Y}$ in the space of the dictionary $\boldsymbol{D}_2$ and $\boldsymbol{B} \in \mathbb{R}^{P\times N}$ is an additive noise.

Thus, the GABAIN algorithm is to solve the following problem:

$$\begin{aligned}\widehat{\boldsymbol{X}_1}, \widehat{\boldsymbol{X}_2}, \widehat{\boldsymbol{D}_1}, \widehat{\boldsymbol{D}_2} = \arg\min_{\boldsymbol{X}_1,\boldsymbol{X}_2,\boldsymbol{D}_1,\boldsymbol{D}_2} &\|\boldsymbol{Y}^T - \boldsymbol{D}_1\boldsymbol{X}_1 - \boldsymbol{D}_2\boldsymbol{X}_2\|_F^2 \\ &+ \lambda\|\boldsymbol{X}_1\|_{1,1} + \beta\|\boldsymbol{D}_2\boldsymbol{X}_2\|_{2,1} \\ \text{s.t. } &\boldsymbol{D} = \boldsymbol{D}_1 + \boldsymbol{D}_2.\end{aligned} \tag{2}$$

where λ and β are regularization parameters that control respectively the level of sparsity of $\boldsymbol{X}_1$ and $\boldsymbol{D}_2\boldsymbol{X}_2$. The greater the value of λ the less the number of non-zero value in $\boldsymbol{X}_1$ and the greater the value of β the less the number of non-zero rows in $\boldsymbol{D}_2\boldsymbol{X}_2$ (i.e., the more the number of selected features).

This strategy is inspired by similar successful works conducted on anomaly detection [8], [9]. Note that the model integrates two constraints via ℓ_1 and $\ell_{2,1}$ norm regularizations : one for each objective. The ℓ_1 regularization limits the number of dictionary atoms (i.e., the number of samples). The selected samples in the sparse representation can be used for the graph construction by creating edges connecting the similar samples. The $\ell_{2,1}$ norm regularization limits the number of available rows in $\boldsymbol{D}_2$, i.e., the number of features of the dataset that can be used to estimate $\boldsymbol{D}_2\boldsymbol{X}_2$ which can be seen as the residual of the sparse representation. Through this game of constraints, the features distribution between the two dictionaries $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ is mechanically conducted. Indeed, since the ℓ_1 norm constraints the $\boldsymbol{X}_1$ estimation and no constraint affect the estimation of $\boldsymbol{X}_2$, the informative features will be automatically selected in $\boldsymbol{D}_1$ whereas uninformative features will be selected in $\boldsymbol{D}_2$. Details about problem 2 iterative optimization and update equations of the different variables at the kth iteration are provided in the next section.

C. Optimization

By adding the variable $\boldsymbol{E}$, the problem (2) can reformulated as follow :

$$\begin{aligned}\widehat{\boldsymbol{X}_1}, \widehat{\boldsymbol{X}_2}, \widehat{\boldsymbol{D}_1}, \widehat{\boldsymbol{D}_2}, \widehat{\boldsymbol{E}} = \arg\min_{\boldsymbol{X}_1,\boldsymbol{X}_2,\boldsymbol{D}_1,\boldsymbol{D}_2} &\|\boldsymbol{Y}^T - \boldsymbol{D}_1\boldsymbol{X}_1 - \boldsymbol{E}\|_F^2 \\ &+ \lambda\|\boldsymbol{X}_1\|_{1,1} + \beta\|\boldsymbol{E}\|_{2,1} \\ \text{s.t. } &\boldsymbol{D} = \boldsymbol{D}_1 + \boldsymbol{D}_2, \;\; \boldsymbol{E} = \boldsymbol{D}_2\boldsymbol{X}_2.\end{aligned} \tag{3}$$

and corresponds to an extension of [36]. the problem (3) can be solved by the Alternative Direction Method of Multipliers (ADMM) [37] by adding the auxiliary variable $\boldsymbol{Z}$:

$$\begin{aligned}\widehat{\boldsymbol{X}_1}, \widehat{\boldsymbol{X}_2}, \widehat{\boldsymbol{D}_1}, \widehat{\boldsymbol{D}_2}, \widehat{\boldsymbol{E}}, \widehat{\boldsymbol{Z}} = \arg\min_{\boldsymbol{X}_1,\boldsymbol{X}_2,\boldsymbol{D}_1,\boldsymbol{D}_2} &\|\boldsymbol{Y}^T - \boldsymbol{D}_1\boldsymbol{X}_1 - \boldsymbol{E}\|_F^2 \\ &+ \lambda\|\boldsymbol{Z}\|_{1,1} + \beta\|\boldsymbol{E}\|_{2,1} \\ \text{s.t. } &\boldsymbol{D} = \boldsymbol{D}_1 + \boldsymbol{D}_2, \;\; \boldsymbol{E} = \boldsymbol{D}_2\boldsymbol{X}_2, \;\; \boldsymbol{Z} = \boldsymbol{X}_1.\end{aligned} \tag{4}$$

(5)

and the constraint $\boldsymbol{Z} = \boldsymbol{X}_1$. Note that, contrary to Problem (3), the first and second terms of (4) are decoupled, which allows an easier estimation of the matrix $\boldsymbol{X}_1$. However, this problem is not convex in all variables to estimate, but it is convex in each variable by fixing the others. It can be solved using the ADMM algorithm by minimizing the following augmented Lagrangian:

$$\begin{aligned}\mathcal{L}_A(\boldsymbol{X}_1, \boldsymbol{E}, \boldsymbol{Z}, \boldsymbol{M}, \mu) = &\frac{1}{2}\|\boldsymbol{Y}^T - \boldsymbol{D}_1\boldsymbol{X}_1 - \boldsymbol{E}\|_F^2 \\ &+ \lambda\|\boldsymbol{Z}\|_{1,1} + \beta\|\boldsymbol{E}\|_{2,1} + \boldsymbol{M}(\boldsymbol{Z} - \boldsymbol{X}_1) + \frac{\mu}{2}\|\boldsymbol{Z} - \boldsymbol{X}_1\|_F^2\end{aligned} \tag{6}$$

The update of each variable at the kth iteration is detailed below.

Updating of $\boldsymbol{X}_1$. $\boldsymbol{X}_1$ is update by minimizing the following problem:

$$\boldsymbol{X}_1^{k+1} = \arg\min_{\boldsymbol{X}_1} \|\boldsymbol{Y}^T - \boldsymbol{D}_1^k\boldsymbol{X}_1 - \boldsymbol{E}^k\|_F^2 + \boldsymbol{M}^k(\boldsymbol{Z}^k - \boldsymbol{X}_1) + \frac{\mu^k}{2}\|\boldsymbol{Z}^k - \boldsymbol{X}_1\|_F^2 \tag{7}$$

Simple algebra leads to:

$$\boldsymbol{X}_1^{k+1} = ({\boldsymbol{D}_1^k}^T\boldsymbol{D}_1^k + \mu^k I)^{-1}({\boldsymbol{D}_1^k}^T\boldsymbol{R}^k + \boldsymbol{M}^k + \mu^k\boldsymbol{Z}^k) \tag{8}$$

where $\boldsymbol{R}^k = \boldsymbol{Y} - \boldsymbol{E}^k$.

Updating of $\boldsymbol{Z}$. The update equation of $\boldsymbol{Z}$ is obtained from:

$$\boldsymbol{Z}^{k+1} = \arg\min_{\boldsymbol{Z}} \lambda\|\boldsymbol{Z}\|_{1,1} + \boldsymbol{M}^k(\boldsymbol{Z} - \boldsymbol{X}_1^{k+1}) + \frac{\mu^k}{2}\|\boldsymbol{Z} - \boldsymbol{X}_1^{k+1}\|_F^2 \tag{9}$$

Which can be simplified to:

$$\boldsymbol{Z}^{k+1} = \arg\min_{\boldsymbol{Z}} \frac{1}{2}\|\boldsymbol{P} - \boldsymbol{Z}\|_F^2 + \gamma\|\boldsymbol{Z}\|_{1,1} \tag{10}$$

where $\boldsymbol{P} = \boldsymbol{X}_1^{k+1} - \frac{1}{\mu^k}\boldsymbol{M}^k$ and $\gamma = \frac{\lambda}{\mu^k}$. The solution to this problem is given by the soft-threshodling operator:

$$\boldsymbol{Z}^{k+1} = \mathcal{S}_\lambda(\boldsymbol{P}) \tag{11}$$

$$\text{with } \mathcal{S}_\lambda(a) = \begin{cases} a-\lambda & if \ \ a > \lambda \\ 0 & if \ \ |a| \leq \lambda \\ a+\lambda & if \ \ a < \lambda \end{cases}$$

Updating of $\boldsymbol{E}$. $\boldsymbol{E}$ is updated by solving the following problem:

$$\boldsymbol{E}^{k+1} = \arg\min_{\boldsymbol{E}} \frac{1}{2}\|\boldsymbol{Y}^T - \boldsymbol{D}_1^k\boldsymbol{X}_1^{k+1} - \boldsymbol{E}\|_F^2 + \beta\|\boldsymbol{E}\|_{2,1} \tag{12}$$

which results in:

$$\boldsymbol{E}^{k+1}(i,:) = \begin{cases} \frac{\|\boldsymbol{Q}(i,:)\|_2 - \beta}{\|\boldsymbol{Q}(i,:)\|_2} & if \ \ \|\boldsymbol{Q}(i,:)\|_2 > \beta \\ 0 & otherwise \end{cases} \tag{13}$$

where $\boldsymbol{Q} = \boldsymbol{Y}^T - \boldsymbol{D}_1^k\boldsymbol{X}_1^{k+1}$ and $\boldsymbol{Q}(i,:)$ is the ith row of $\boldsymbol{Q}$.

Simple algebra allows the variables $\boldsymbol{X}_2, \boldsymbol{D}_2$ to be inferred from $\boldsymbol{X}_1, \boldsymbol{Z}$ and $\boldsymbol{E}$ to respect constraints (5).

Updating of $\boldsymbol{D}_2$. The update of $\boldsymbol{D}_2$, using the updated matrix $\boldsymbol{E}$, is:

$$\boldsymbol{D}_2^{k+1}(i,j) = \boldsymbol{D}_E^{k+1} = \begin{cases} \boldsymbol{D}(i,j) & if \ \boldsymbol{E}^{k+1}(i,j) > 0 \\ 0 & otherwise \end{cases} \tag{14}$$

Updating of $\boldsymbol{D}_1$. According to the constraint $\boldsymbol{D} = \boldsymbol{D}_1 + \boldsymbol{D}_2$, the update of $\boldsymbol{D}_2$ is given by:

$$\boldsymbol{D}_1^{k+1} = \boldsymbol{D} - \boldsymbol{D}_2^{k+1} \tag{15}$$

Updating of $\boldsymbol{X}_2$. According to the constraint $\boldsymbol{E} = \boldsymbol{D}_2\boldsymbol{X}_2$, the update of $\boldsymbol{X}_2$ is given by:

$$\boldsymbol{X}_2^{k+1} = \boldsymbol{D}_2^{k+1}[\boldsymbol{E}^{k+1}]^{-1} \tag{16}$$

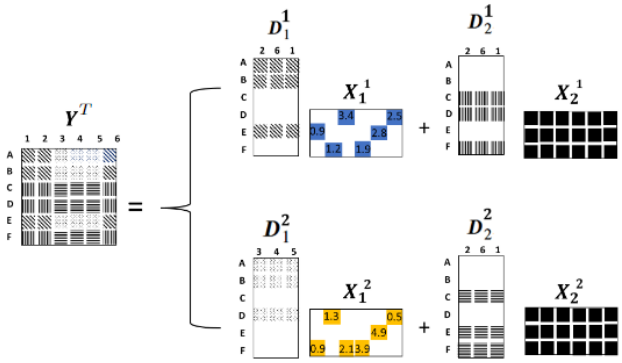
The main stages of the ADMM-based algorithm are described above to solve (4)(5) and summarized in Algorithm 1. Theoretical convergence properties of the ADMM algorithm are details in [38].

Algorithm 1 GABAIN ADDM-Based algorithm

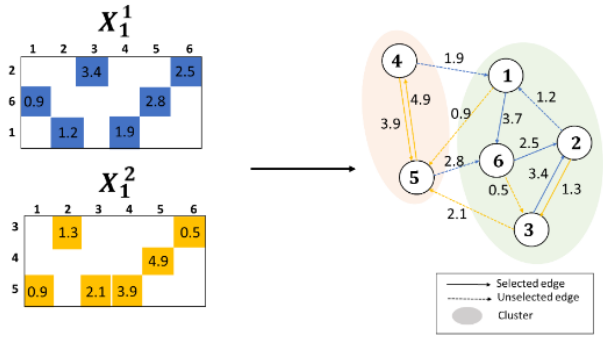
Input: the transpose of the matrix of data $\boldsymbol{Y}^T \in \mathbb{R}^{P\times N}$, the dictionary $\boldsymbol{D} \in \mathbb{R}^{P\times L}$, the scalars β and ϵ, $\rho = 1.1$.
Initialization: $k = 0$, $\boldsymbol{Z}^0$, $\boldsymbol{E}^0$, $\boldsymbol{M}^0$, $\boldsymbol{D}_1^0$, $\boldsymbol{D}_2^0$, $\boldsymbol{X}_2^0$, μ^0.
Repeat:
1) $\boldsymbol{X}_1^{k+1} = \arg\min_{\boldsymbol{X}_1} \mathcal{L}_A(\boldsymbol{X}_1, \boldsymbol{Z}^k, \boldsymbol{E}^k, \boldsymbol{M}^k, \mu^k)$
2) $\boldsymbol{Z}^{k+1} = \arg\min_{\boldsymbol{Z}} \mathcal{L}_A(\boldsymbol{X}_1^{k+1}, \boldsymbol{Z}, \boldsymbol{E}^k, \boldsymbol{M}^k, \mu^k)$
3) $\boldsymbol{E}^{k+1} = \arg\min_{\boldsymbol{E}} \mathcal{L}_A(\boldsymbol{X}_1^{k+1}, \boldsymbol{Z}^{k+1}, \boldsymbol{E}, \boldsymbol{M}^k, \mu^k)$
4) $\boldsymbol{M}^{k+1} = \boldsymbol{M}^k + \mu^k(\boldsymbol{Z}^{k+1} - \boldsymbol{X}_1^{k+1})$
5) $\boldsymbol{\mu}^{k+1} = \rho\mu^k$
6) $\boldsymbol{D}_2^{k+1} = \boldsymbol{D}_E^{k+1}$
7) $\boldsymbol{D}_1^{k+1} = \boldsymbol{D} - \boldsymbol{D}_2^{k+1}$
8) $k = k + 1$
Until: $\frac{\|\boldsymbol{Z}^k - \boldsymbol{X}_1^k\|_F^2}{\|\boldsymbol{X}_1^k\|_F^2} < \epsilon$
Output: $\boldsymbol{X}_1^k$

D. Illustrative example

This section details the GABAIN algorithm for clustering application through a simple example shown in Figure 2. As a reminder, The GABAIN algorithm aims to represent any data matrix as graph data preserving local structure of the data. In addition it includes a FS option to offer data knowledge and explainability. In the example of Figure 2, we consider a the transpose of a data matrix noted $\boldsymbol{Y}^T \in \mathbb{R}^{6\times 6}$. The first step of the GABAIN algorithm is to draw dictionaries by splitting the matrix $\boldsymbol{Y}^T$ into smaller dictionaries. In figure 2, two dictionaries are composed of three atoms each : samples 2, 6 and 1 compose columns of the first dictionary $\boldsymbol{D}_1^1$, and samples 3, 4, 5 compose columns of the second dictionary $\boldsymbol{D}_1^2$. The second step of the GABAIN algorithm is to apply the ADMM-based algorithm to each dictionary. In our example of figure 2, results reveal that features A,B and E have been selected when the ADMM-based algorithm have been performed on the dictionary $\boldsymbol{D}_1^1$, and features A,B and D have been selected when the ADMM-based algorithm has been performed on the dictionary $\boldsymbol{D}_1^2$. In the third step, the graph connections or edges are deducted from the estimation of $\boldsymbol{X}_1^1$ and $\boldsymbol{X}_1^2$ obtained in step 2. Indeed, we assume that the highest values



(a) GABAIN Step 1-2



(b) GABAIN Step 3-5

Fig. 2: GABAIN Algorithm

of sparse representation of each sample (i.e., highest values of each column of $\boldsymbol{X}_1^1$ and $\boldsymbol{X}_1^2$) highlight similarities which are extracted to connect corresponding nodes by weighted edges and build the graph as illustrated in figure 2. Note that self-loop, i.e., edges that connect a node to itself, are not included in the graph since they are not informative. Note that for each dictionary, N edges are identified, one per node. The dictionary split allows more edges to be identified. Furthermore, since the ADMM-based implies matrix inversion, the use of smaller dictionaries allows an easier estimation. The number of edges per node to select is an hyperparameter fix by user. In our example, two dictionaries are drawn but only one edge per node is selected : edges with the highest weight. In addition, to promote stochasticity, the GABAIN algorithm can be apply several times, denoted as epoch. Thus, more dictionaries are drawn and more interesting similarities can be identified.

IV. Experiments

A. Datasets

In these experiments, two cybersecurity use cases are studied. The first one is about facial expression recognition in the images of the popular Yale face database. The second use case is about android malware categorization. The corresponding dataset, called APK, has been built from open-source VirusTotal analysis reports of hundreds Android applications. The main specifications of the two datasets are reported in Table II.

Algorithm 2 GABAIN Algorithm for clustering applications

Input: the transpose of the matrix data $\boldsymbol{Y}^T \in \mathbb{R}^{P\times N}$, the scalars β, number of edges C and size of dictionaries L

1) Draw dictionaries by splitting $\boldsymbol{Y}^T$ into $\frac{N}{L}$ random dictionaries composed of L atoms each.
2) Repeat Algorithm 1 for each dictionaries and construct graphs.
3) Merge graphs returned by ADMM executions.
4) Select C edges per nodes.
5) Apply graph clustering algorithm.

Output: Labels of clusters, selected features

TABLE II: Characteristics of datasets.

Dataset	Samples	Features	Classes	Category
Yale	165	1024	15	Image
APK	934	47	5	Android Application

B. Compared methods

The GABAIN algorithm has been evaluated for clustering in two cybersecurity use-cases. Combined to Louvain community detection method [39], the GABAIN approach has been compared to three state-of-the-art similarity graph learning methods : k-NNG, RBF similarity graph, Laplacian similarity graph. In the same way as for GABAIN, this similarity graph learning have been combined to Louvain community detection to perform clustering. In addition, popular clustering methods have been applied, such as the K-means algorithm since its the most used clustering approach, and two clustering techniques which have the benefit of not requiring the number of clusters provided a priory, namely HDBSCAN [40] and OPTICS [41]. As mentioned before, the DLUFS [34] feature selection state-of-the-art solution is not evaluated in this work since the actual implementation is not scalable. The different methods are all described in the following:

- **K-NNG** : Graph similarity method which connect each node to its k nearest neighbors according to the euclidean distance.
- **AKNNG**[20] : A variant of K-NNG method that build the adaptive k-nearest neighbors similarity graph (AKNNG) by automatically adjusting k for different data points.
- **MAKNNG**[20] : A robust version of the AKNNG solution.
- **RBF SGL** : Graph similarity learning based on RBF similarity matrix.
- **Laplacian SGL** Graph similarity learning based on Laplacian similarity matrix.
- **K-means**: Centroid-based clustering method that minimizes the average distance between data samples within clusters.
- **HDBSCAN**: Hierarchical density-based clustering that finds core samples of high density and expands clusters from them.
- **OPTICS**: Density based clustering method that keeps cluster hierarchy for a variable neighborhood radius.

Methods	Yale	APK
K-means	0.90 ± 0.01	**0.90 ± 0.01**
HDBSCAN	0.28 ± 0.00	0.84 ± 0.00
OPTICS	0.64 ± 0.00	0.84 ± 0.00
k-NNG + Louvain	0.81 ± 0.01	0.88 ± 0.00
RBF-SGL + Louvain	0.85 ± 0.01	**0.90 ± 0.00**
Laplacian-SGL + Louvain	0.83 ± 0.00	0.86 ± 0.01
AKNNG + Louvain	0.87 ± 0.00	0.22 ± 0.00
MAKNNG + Louvain	0.90 ± 0.00	0.22 ± 0.00
GABAIN + Louvain	**0.93 ± 0.00**	**0.90 ± 0.00**

TABLE III: Clustering Results (Adjusted Rand Score ± std)

Methods	Yale	APK
K-means	0.49 ± 0.2	**0.80 ± 0.01**
HDBSCAN	0.12 ± 0.00	0.62 ± 0.00
OPTICS	0.45 ± 0.00	0.61 ± 0.00
k-NNG + Louvain	0.36 ± 0.02	0.75 ± 0.03
RBF-SGL + Louvain	0.46 ± 0.04	0.79 ± 0.00
Laplacian-SGL + Louvain	0.43 ± 0.02	0.74 ± 0.01
AKNNG + Louvain	0.48 ± 0.01	0.00 ± 0.00
MAKNNG + Louvain	0.57 ± 0.01	0.00 ± 0.00
GABAIN + Louvain	**0.58 ± 0.02**	0.75 ± 0.00

TABLE IV: Clustering Results (NMI ± std)

C. Evaluation measures

Two common measures used to evaluate clustering methods are the Adjusted Rand Score (ARS) and the Normalized Mutual Information (NMI), which can be computed by considering the clustering results, noted $\widehat{\boldsymbol{y}}$, and the ground-truth, noted $\boldsymbol{y}$. These two metrics are particularly useful since label correspondence between the clustering prediction and the ground-truth is not required.

The Adjusted Rand Score (ARS) is a corrected version of the rand index [42] which measures degree of overlapping between two partitions. It was introduced to determine whether two clustering results are similar to each other. The ARS can be defined as follow:

$$ARS = \frac{RI - ExpectedRI}{max(RI) - ExpectedRI}$$

In the formula, the RI stands for the Rand Index. The ARS is equal to 0 when samples are assigned into different clusters and it equals to 1 when the two clusters results are the same.

The second evaluation metric is the NMI which is defined as:

$$NMI(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = \frac{2I(\boldsymbol{y}, \widehat{\boldsymbol{y}})}{H(\boldsymbol{y}) + H(\widehat{\boldsymbol{y}})}$$

where H(.) is the entropy measure and $I(\boldsymbol{y}, \widehat{\boldsymbol{y}})$ [43] is the mutual information of $\boldsymbol{y}$ and $\widehat{\boldsymbol{y}}$. The more the NMI is close to 1, the better the clustering is.

D. Experimental setting

This section gives details about methods implementation and use. Graph similarity based on k-NNG or GABAIN have 20 or fewer edges per nodes since it represent a good setting for these methods applied on these two datasets. The HDBSCAN and OPTICS clustering algorithms have been applied with default values. Regarding the K-means algorithm, the number k of clusters has been set to the true number of classes given by the ground-truth (see in Table II). Finally, the GABAIN algorithm have been applied with the same value of its hyperparameters for the two datasets : $\beta = 12$, $L = \frac{N}{2}$, $C = 20$, $\alpha = 0.3$. The NMI and ARS measures are computed on 30 times running of each algorithms in two descriptive statistical quantities, mean and standard deviation (std).

E. Experiment results

This section evaluates the proposed solution applied for clustering in cybersecurity domain and compares it to state-of-the-art methods. Comparison results based on ARS and NMI are respectively reported in Table III and Table IV. The best results are marked by bold numbers and second-best results are marked by underlined numbers. Quantitative results demonstrate the interest of the SGL strategy combined with graph clustering, especially the GABAIN solution. Indeed, SGL approaches combined to Louvain clustering outperforms the HDBSCAN and OPTICS clustering and are competitive with the famous K-means clustering bu returning the best or second-best clustering for the two use-cases. In addition, note that the K-means method depends on the number of clusters provided a priory which is not the case of the Louvain method. Thus, since the SGL allows the use of efficient graph clustering methods, it seems more suitable in many real-world applications where the number of cluster is unknown. Beyond its great clustering performances, the GABAIN algorithm includes an efficient feature selection option which can represent an important advantage compared to the state-of-the-art. The obtained feature selection on the Yale dataset is displayed in figure 3. In this use-case about face expression recognition, the GABAIN method selects features (red pixels) corresponding to relevant face areas such as forehead face or cheekbones on which some specific wrinkles allow emotions to be detected. Regarding the APK dataset, the feature selection is not so visual but is also significant. Indeed, the GABAIN method applied on the APK dataset returns 22 features identified as informative, which is less than the half of the number of features in this dataset. Among the selected features the ones

(a) YALE

Fig. 3: Example of GABAIN feature selection. Yale sample image with selected features marked by red pixels.

relative to the permissions required by the android application can be found. Note that unexpected permissions required by android applications often represent an evidence of the presence of malware. Conversely, features such as the number of called libraries or the number of files in the application code source were not selected. This feature selection appeared as logical results to cybersecurity experts, confirming the efficiency of the GABAIN feature selection.

V. Conclusion

This paper investigates a new similarity graph learning method based on sparse coding. The proposed GABAIN algorithm identifies similarities between sample data and represent any matrix of data as a graph preserving the local structure of the data. In addition, the GABAIN method includes a selection feature option based on dictionary learning that can offer some explainability and data knowledge. The approach was combined to the Louvain community detection and evaluated for clustering in cybersecurity applications. The experiments results demonstrate the interest of the GABAIN strategy combined with community detection for clustering task, especially if limited knowledge on the data is available and the number of cluster is unknown. Then, the results showed the efficiency of its feature selection strategy which returns relevant features and improves user introspective insight into its data and facilitate clustering explainability. For future work, the interest of the GABAIN strategy for other application such as anomaly detection deserves to be investigated. Then, it could be interesting to quantify features importance of the selected features.

References

[1] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, IEEE Signal Process. Mag. 30 (3) (2013) 83–98.

[2] A. Sandryhaila, J. M. F. Moura, Discrete signal processing on graphs, IEEE Trans. Signal Process 61 (7) (2013) 1644–1656.

[3] A. Ortega, P. Frossard, J. K. J. M. F. Moura, P. Vandergheynst, Graph signal processing: Overview, challenges, and applications, Proceedings of the IEEE 106 (5) (2018) 808–828.

[4] I. Pitas, Graph-Based Social Media Analysis, Chapman and Hall/CRC, Minneapolis, Minnesota, U.S.A, 2016.

[5] M. Newman, Networks: An Introduction., Oxford Univ. Press, Oxford, U.K., 2010.

[6] F. Yang, J. Xu, C. Xiong, Z. Li, K. Zhang, Prographer: An anomaly detection system based on provenance graph embedding, in: Proc. USENIX'23, 2023.

[7] X. Dong, D. Thanou, M. Rabbat, P. Frossard, Learning graphs from data: A signal representation perspective, IEEE Signal Process. Mag. 36 (3) (2019) 44–63.

[8] A. Adler, M. Elad, Y. Hel-Or, E. Rivlin, Sparse coding with anomaly detection, in: IEEE International Workshop on Machine Learning for Signal Processing (MLSP'13), 2013.

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

[10] P. Dhal, C. Azad, A comprehensive survey on feature selection in the various fields of machine learning, Applied Intelligence 52 (4) (2022) 4543–4581.

[11] J. Miao, L. Niu, A survey on feature selection, Procedia Computer Science 91 (2016) 919–926.

[12] U. V. Luxburg, A tutorial on spectral clusterin, Stat. Comput. 17 (2007) 395–416.

[13] Y. N. . M. S. Yang, Powered gaussian kernel spectral clustering, Neural Computing and Applications 31 (2019) 557–572.

[14] Z. Kang, C. Peng, Q. Cheng, Kernel-driven similarity learning, Neurocomputing 267 (2017) 210–219.

[15] F. P. Preparata, M. I. Shamos, Computational Geometry - An introduction, Springer-Verlag, New-York. USA, 1985.

[16] D. Eppstein, M. S. Paterson, F. F. Yao, On nearest-neighbor graphs, Discrete Computational Geometry 17 (1997) 263–282.

[17] X. Ye, T. Sakurai, pectral clustering using robust similarity measure based on closeness of shared nearest neighbors, in: Proc. International joint conference on neural networks (IJCNN'2015), 2015.

[18] M. Alshammari, J. Stavrakakis, M. Takatsuka, Refining a k-nearest neighbor graph for a computationally efficient spectral clustering, Pattern Recogn. 114 (23) (2021) 107869.

[19] K. Sharma, A. Seal, Spectral embedded generalized mean based k-nearest neighbors clustering with s-distance, Expert Syst. Appl. 169 (2021) 557–572.

[20] Y. Cai, J. Z. Huang, J. Yin, A new method to build the adaptive k-nearest neighbors similarity graph matrix for spectral clustering, Neurocomputing. 493 (2022) 191–209.

[21] L. Zelnik-Manor, P. Perona, Self-tuning spectral clustering, in: In Proc. Neural Information Processing Systems 17 (NIPS'2004), 2014.

[22] P. Favati, G. Lotti, O. Menchi, F. Romani, Construction of the similarity matrix for the spectral clustering method: Numerical experiments, J. Comput. Appl. Math (375) (2020) 112795.

[23] Y. Nataliani, M.-S. Yang, Powered gaussian kernel spectral clustering, Neural Comput. Appl. 31 (1) (2020) 557–572.

[24] E. Elhamifar, R. Vidal, sparse subspace clustering: Algorithm, theory, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 35 (11) (2013) 2765–2781.

[25] X. Zhu, S. Zhang, R. Hu, Y. Zhu, J. Song, Local and global structure preservation for robust unsupervised spectral feature selection, IEEE Trans. Knowl. Data Eng. 3 (30) (2017) 517–529.

[26] J. Xu, M. Yu, L. Shao, W. Zuo, D. Meng, L. Zhang, D. Zhang, Scaled simplex representation for subspace clustering, IEEE Trans. Cybern. 51 (3) (2021) 1493–1505.

[27] D. Dohono, High-dimensional data analysis: The curses and blessings of dimensionality, AMS Math Challenges Lecture (Aug. 2000).

[28] P. D. V. Bhasha, Dimension reduction techniques: Current status and perspectives, Materials Today: Proceedings 62 (13) (2022) 7024–7027.

[29] Y. Yang, H. T. Shen, Z. Ma, Z. Huang, X. Zhou, L2,1-norm regularized discriminative feature selection for unsupervised learning, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI'11), 2011.

[30] Z. Li, Y. Yang, J. Liu, X. Zhou, H. Lu, Unsupervised feature selection using nonnegative spectral analysis, in: Proc. AAAI Conference on Artificial Intelligence, 2012.

[31] C. Hou, F. Nie, X. Li, D. Yi, Y. Wu, Joint embedding learning and sparse regression: A framework for unsupervised feature selection, IEEE Trans. Cybern. 44 (6) (2014) 793–804.

[32] X. Zhu, S. Zhang, R. Hu, Y. Zhu, J. Song, Local and global structure preservation for robust unsupervised spectral feature selection, IEEE Trans. Knowl. Data. Eng. 30 (3) (2018) 517–529.

[33] R. Shang, Y. Meng, W. Wang, F. Shang, L. Jiao, Local discriminative based sparse subspace learning for feature selection, Pattern Recognition 92 (2019) 219–230.

[34] M. G. Parsa, H. Zare, M. Ghatee, Low-rank dictionary learning for unsupervised feature selection, Expert Systems with Applications 202 (Sept. 2022).

[35] Z. Pengfei, H. Qinghua, Z. Changqing, Z. Wangmeng, Coupled dictionary learning for unsupervised feature selection, in: Proc. AAAI Conference on Artificial Intelligence, 2016.

[36] A. Adler, M. Elad, Y. Hel-Or, E. Rivlin, Sparse coding with anomaly detection, J. Signal Process. Syst. 79 (2) (2015) 179–188.

[37] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via alternating direction method of multipliers, Foundations and Trends in Machine Learning 3 (1) (2010) 1–222.

[38] J. Eckstein, D. Bertsekas, On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators, Mathematical programming (Series A and B) 55 (3) (1992) 293–318.

[39] V. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics Theory and Experiment 2008 (Apr. 2008).

[40] R. Campello, D. Moulavi, J. Sander, Density-based clustering based on hierarchical density estimates 7819 (2013) 160–172.

[41] M. Ankerst, M. M. Breunig, H.-P. Kriegel, S. Jörg, Optics: Ordering points to identify the clustering structure, in: In Proc. ACM SIGMOD International Conference on Management of Data, 1999.
[42] L. J. Hubert, P. Arabie, Comparing partitions, Pattern Recognition 2 (2-3) (1985) 193–218.
[43] N. X. Vinh, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance, Journal of Machine Learning Research 11 (95) (2010) 2837–2854.