JYVÄSKYLÄ STUDIES IN COMPUTING
223

Aviv Rotbart

# High-Dimensional Big Data Processing with Dictionary Learning and Diffusion Maps

JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 223

Aviv Rotbart

# High-Dimensional Big Data Processing with Dictionary Learning and Diffusion Maps

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Agora-rakennuksen Lea Pulkkisen salissa
joulukuun 4. päivänä 2015 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in building Agora, Lea Pulkkinen hall, on December 4, 2015 at 12 o'clock noon.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2015

# High-Dimensional Big Data Processing with Dictionary Learning and Diffusion Maps

JYVÄSKYLÄ STUDIES IN COMPUTING 223

Aviv Rotbart

# High-Dimensional Big Data Processing with Dictionary Learning and Diffusion Maps

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2015

Editors
Timo Männikkö
Department of Mathematical Information Technology, University of Jyväskylä
Pekka Olsbo, Ville Korkiakangas
Publishing Unit, University Library of Jyväskylä

URN:ISBN:978-951-39-6402-3
ISBN 978-951-39-6402-3 (PDF)

ISBN 978-951-39-6401-6 (nid.)
ISSN 1456-5390



Jyväskylä University Printing House, Jyväskylä 2015

## ABSTRACT

Rotbart, Aviv
High-Dimensional Big Data Processing with Dictionary Learning and Diffusion Maps
Jyväskylä: University of Jyväskylä, 2015, 22 p.(+included articles)
(Jyväskylä Studies in Computing
ISSN 1456-5390; 223)
ISBN 978-951-39-6401-6 (nid.)
ISBN 978-951-39-6402-3 (PDF)
Finnish summary
Diss.

Algorithms for modern Big Data analysis deal with both massive amount of samples and a large number of features (high-dimension). One way to cope with these challenges is to assume and discover the existence of localization in the data by uncovering its intrinsic geometry. This approach suggests that different data segments can be analyzed separately and then unified in order to gain an understanding of the whole phenomenon. Methods that utilize efficiently localized data are attractive for high-dimensional big data analysis, because they can be parallelized, and thus the computational resources, which are needed for their utilization, are realistic and affordable. These methods can explore local properties such as intrinsic dimension that vary among different pieces of data.

This thesis presents two different methods to locally analyze large datasets for classification, clustering and anomaly detection. The first method localizes dictionary learning based on matrix factorization techniques. We utilize randomized LU decomposition and QR-decomposition algorithms to build dictionaries that describe different types of data. Then, these dictionaries are used to assign new samples to their respective class. One application in cyber security deals with learning of computer files and detecting executable code hidden in PDF files. In a different application, a dictionary learned from a normally behaving computer network data is used to detect anomalies in test data which may imply a cyber threat.

The second method is localized diffusion process (LDP), which constitutes a coarse-graining of the classic Diffusion Maps algorithm. In LDP, a Markov walk is calculated on small data point clouds instead of the original data points. This work establishes a theoretical foundation for the Localized Diffusion Folders for hierarchical data analysis.

Keywords: Localized Diffusion; Dictionary Learning; Randomized LU; QR Factorization

**Author**

Aviv Rotbart
Department of Mathematical Information Technology
University of Jyväskylä
Finland

**Supervisors**

Professor Amir Averbuch
School of Computer Science
Tel Aviv University
Israel

Professor Pekka Neittaanmäki
Department of Mathematical Information Technology
University of Jyväskylä
Finland

**Reviewers**

Dr. Dan Kushnir
Bell Labs
Murray Hill, NJ, USA

Dr. Neta Rabin
Afeka Tel Aviv Academic College
College of Engineering
Tel Aviv, Israel

**Opponent**

Prof. Keijo Ruotsalainen
Faculty of Information and Electrical Engineering
Mathematics Division
University of Oulu

## ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my thesis advisers, Prof. Amir Averbuch and Prof. Pekka Neittaanmäki. I wish to thank Amir and Pekka for their invaluable advice, guidance, support and help throughout my journey of PhD studies. I thank Amir and Pekka for introducing me to new and fascinating scientific domains, for fruitful collaborations and for doing everything possible to make my studies succeed. This thesis has been completed in the PHD program of the IT faculty, University of Jyväskylä. I thank the IT faculty for financial support of this work. Finally, I thank my beloved wife Na'ama for her support, encouragement and endless love. Thank you for walking this path with me. I thank my son Roee who was born during the work of this thesis. I thank my parents Ahuva and Shlomo and my brothers Amit and Lior for their inspiration and dedicated love for many years, and for providing the foundation for this work.

## LIST OF FIGURES

FIGURE 1 (a) A 3D Swiss roll comprised of 3,000 points. (b)-(d): the two most significant coordinates of the ICPQR embedding of the Swiss roll with (b) $\mu = 0.1$, $s = 1,246$, (c) $\mu = 1$, $s = 752$, and (d) $\mu = 5$, $s = 382$, where $s$ denotes the dictionary size. Data coloring is consistent with Figure. 1(a)..... 15
FIGURE 2 Construction of the pruned-kernel $\widehat{K}_{ij}$ between the clusters $C_i$ and $C_j$..... 17
FIGURE 3 Illustration of the difference between localized and non-localized paths ..... 17
FIGURE 4 Illustration of non-trivial localized paths..... 18

## LIST OF TABLES

TABLE 1 Confusion matrix for file type detection using Markov-Walk-based features. 100 files of each type were examined. ..... 14
TABLE 2 Confusion matrix for malicious PDF detection experiment. 100 clean PDF files and 10 malicious PDF files were examined. ..... 14

## CONTENTS

ABSTRACT
ACKNOWLEDGEMENTS
LIST OF FIGURES AND TABLES
CONTENTS
LIST OF INCLUDED ARTICLES

1 INTRODUCTION ..... 9

2 CONTRIBUTION OF THE THESIS ..... 12

YHTEENVETO (FINNISH SUMMARY) ..... 19

REFERENCES ..... 20

INCLUDED ARTICLES

## LIST OF INCLUDED ARTICLES

PI Guy Wolf, Aviv Rotbart, Gil David, Amir Averbuch. Coarse-grained localized diffusion. *Applied and Computational Harmonic Analysis*(3):388-400, 2012.

PII Guy Wolf, Aviv Rotbart, Gil David, Amir Averbuch. Hierarchical data organization, clustering and denoising via Coarse-grained localized diffusion. *CJR conference, Yale*, 2012.

PIII Aviv Rotbart, Gil Shabat, Yaniv Shmueli, Amir Averbuch. Randomized LU decomposition: An algorithm for dictionaries construction. *Submitted to IEEE transaction on Information Forensics and Security*, 2014.

PIV Amit Bermanis, Aviv Rotbart, Moshe Salhov, Amir Averbuch. Incomplete Pivoted QR-based Dimensionality Reduction. *Submitted*, 2015.

In [PI], the author's contributed in the development the algorithm, the theory and writing of the paper. The author presented the main results and applications of this paper in a poster presentation in [PII]. The author developed the theory, the algorithms and the experiments in [PIII]. The author also analyzed the results and wrote the paper. In [PIV], the author developed all the experimental part and was an integral part in the development of the algorithms. The author also wrote that paper.

# 1 INTRODUCTION

Big Data has given rise to industrial revolution in machine learning. New technologies change the way we handle data the same way heavy machinery and factories changed the way manufacturing is done. The amount of data captured these days is exploding, and grows rapidly each year [14, 8]. Cloud services for storage and computing are replacing PC's and servers which can not handle these tasks for big datasets. New methods for information retrieval and data analysis are being developed with emphasis on big, dynamic, and high-dimensional datasets.

Two trends can be identified in the Big Data domain. The first trend relates to the generation, collection and digital storage of data, while the second deals with turning it into useful knowledge. The two are intertwined, each being meaningless without the other. While the tools for collection and storage of data are becoming increasingly simpler and cheaper, the methods for analyzing it are not as scalable or do not automatically fit new sizes and types of data. Therefore, much effort in both industry and academy is put into developing algorithms that deal with modern demands of data analysis. Such algorithms aim to find meaning in oceans of bits representing numbers, text, images, entities and relations between them, computer code, and many other data structures. The insights from data analysis algorithms are used to find clusters in the data, to classify observations to their cluster, or detect suspicious activity or anomalous data points. They can be used to identify trends and provide alerts when appropriate triggers are defined.

Mathematical and statistical tools underlie many data analysis algorithms. Methods such as subsampling, spectral and kernel analysis, manifold learning, dictionary learning, dimensionality reduction, sparsification, random processing, to name some, are common. In this thesis, the concept of *localization* is used to develop machine learning algorithms that separately analyze different pieces of data and then gain insights and draw conclusions about the entire dataset. Generally speaking, localization can be done in the observations domain by subsampling data points, in the features domain by analyzing subsets of the data variables (features) and in combination of the two. In supervised or semi-supervised

data analysis scenarios, localization can also be done in the labeling domain where each cluster of the training data is treated locally to train a classifier. We use a data localization framework in order to provide additional tools for well-known problems in analyzing high-dimensional big data.

Dictionary construction:: Dictionaries are small and compact structures used to describe large and complex datasets. They can be comprised of a subset of the original data or constitute the result of an algorithm that manipulates the data. Dictionaries can be used as a dimensionality reduction tool where each data point is approximated by a linear combination of the dictionary elements. When the dictionary size is smaller than the original data dimension, a low dimensional representation of the data is achieved. Dictionary base dimension reduction faithfully represents the original dataset by preserving important properties such as pairwise distances. Dictionaries are used extensively in anomaly detection, clustering and classification, noise reduction and signal reconstruction as described in [PIV] and [PIII].

Pruning and inter-cluster similarity analysis:: Many data analysis algorithms use pairwise distances between data points to perform machine learning tasks. For very large datasets, the tasks of distance matrix construction and storage have become increasingly difficult. Moreover, manipulations of these matrices, such as SVD, are computationally expensive. A possible remedy for this problem is achieved by pruning the original data into clusters prior to the application of computationally expensive algorithms. This approach is described in [PI].

## Related Work

Localized data analysis algorithms are used to solve various types of machine learning problems. Dictionary-based classification models, for example, have recently led to results in face recognition [26, 23, 24, 25, 15, 10], digit recognition [25], and object categorization [15, 10]. In these problems, local dictionary is learned separately for each class of the data or, alternatively, one large dictionary is constructed for all the classes. In the latter case, localization is achieved inside the dictionary, where each element is labeled as belonging to one data class. Dictionaries can be constructed by utilizing matrix factorization methods such as SVD, LU or QR. The output of each method constitutes a dictionary-matrix and a representation-matrix. The dictionary serves as a spanning set of the original data, and the representation contains the exact coefficients that generate the original data by linear combinations of the dictionary elements. Dictionaries can also be used for dimensionality reduction. Johnson-Lindenstrauss Lemma [11] constitutes a basis for many random projection based dimensionality reduction methods [13, 9, 17, 1, 3, 6, 4, 21]. When the number of dictionary elements is smaller than the original dimension of the data, the matrix factorization provides a low-dimensional representation of the data.

Different approaches for Big Data analysis seek localization by analyzing small data patches and then combining the results into a unified view of the entire dataset. Such approaches include coarse-grained Diffusion Maps (DM) [12], where DM is applied to pruned clusters of data points. In a different set of works, patches of the underlying manifold of the dataset are considered [16, 22, 19, 20]. This approach defines non-scalar affinities between patches and uses them to embed the data into a low-dimensional space.

These methods and their related applications show the usefulness of local substructures analysis for machine learning problems. In this thesis, we expand these ideas to new theoretical and experimental domains and show that deep insights in Big Data can be learned by separately analyzing small pieces of it.

# 2 CONTRIBUTION OF THE THESIS

The thesis explores several mathematical frameworks for local pattern analysis and their application to machine learning tasks such as anomaly detection, classification and noise reduction. The first part describes a localized compact representation of data by two dictionary learning algorithms which are applied to several classification and anomaly detection tasks. In the second part, the problems of massive and high-dimensional data are simultaneously treated by coarse-graining of the Diffusion Maps framework to local clusters of the data. The rest of this section presents a brief overview of each part.

**Localized Dictionaries**

High-dimensional data appears in many research and application fields. Often, the exploration of such data by classical algorithms encounters difficulties due to the “curse of dimensionality" [5] phenomenon. Therefore, dimensionality reduction methods are applied to data prior to its analysis. Many of these methods are based on principal components analysis (PCA), which is statistically driven: namely, the data is mapped into a low-dimension subspace that preserves significant statistical properties of the high-dimensional data. As a consequence, these methods do not directly identify the geometry of the data reflected by mutual distances between data points. Thus, classification, anomaly detection or other machine learning tasks are affected. The work in [PIII] and [PIV] provides a dictionary-based framework for data analysis algorithms. It is demonstrated in applications such as dimensionality reduction, out-of-sample extension, classification and anomaly detection.

The work in [PIII] deals with a distinctive dictionary construction. This method is needed in data sampling. Usually, one or more dictionaries are constructed from a training data. Then they are used to classify signals that did not participate in the training process. A new dictionary construction algorithm is introduced. It is based on a low-rank matrix factorization achieved by the application of the randomized LU decomposition [18] to a training data. This method is fast, scalable, parallelizable, consumes low memory, outperforms SVD in these

categories and works well on large matrices. Given an input matrix $A$, the randomized LU decomposition technique provides a factorization

$$PAQ \approx LU$$

of $A$, such that $L$ and $U$ are the lower and upper triangular matrices, respectively, and $P$ and $Q$ are orthogonal permutation matrices. In this construction, $D = P^T L$ is the dictionary and $UQ^T$ is the coefficient matrix representing $A$ by the dictionary $D$.

In contrast to existing methods [2], the randomized LU decomposition constructs an under-complete dictionary, which simplifies both the construction and the classification processes of newly arrived signals. An overview of Randomized LU-based dictionaries training is described in Algorithm 2.0.1. For each class of the training data, a distinct dictionary is constructed. Members of a specific class are spanned more accurately by the dictionary of their class than by other dictionaries. Formally, we define a distance function between a signal and a dictionary in the following way:

**Definition 2.0.1** *Let $x$ be a signal and $D$ be a dictionary. The distance between $x$ and the dictionary $D$ is defined by*

$$dist(x, D) \triangleq ||DD^{\dagger}x - x||,$$

*where $D^{\dagger}$ is the pseudo-inverse of the matrix $D$.*

**Algorithm 2.0.1:** Dictionaries Training using Randomized LU

**Input:** $X = \{X_1, X_2, \ldots, X_r\}$ training datasets for $r$ sets;
$K = \{k_1, k_2, \ldots, k_r\}$ dictionary size of each set.
**Output:** $D = \{D_1, D_2, \ldots, D_r\}$ set of dictionaries.

1: **for** $t \in \{1, 2, \ldots, r\}$ **do**
   $P_t, Q_t, L_t, U_t \leftarrow Randomized\ LU\ Decomposition(X_t)$, $D_t \leftarrow P_t^T L_t$
2: $D \leftarrow \{D_1, D_2, \ldots, D_r\}$

The distance function from Definition 2.0.1 is used for classification by assigning an unknown signal $x$ to the class $D_i$ that minimizes $dist(x, D_j), \quad j = 1, \ldots, r$. The dictionary construction is generic and fits into different applications. We demonstrate the capabilities of this algorithm for file type identification, which is a fundamental task in digital security, performed nowadays for example by a sandboxing mechanism, deep packet inspection, firewalls and anti-virus systems. We propose a content-based method for file type detection that neither depends on file extension nor on metadata. Such approach is harder to deceive. In addition, we show that only a few file fragments from the whole file are needed for a successful classification. The method was tested on 6 popular file types and showed success in identifying the correct type of each file by examining only 10 randomly selected 2 KB fragments from each file. Table 1 presents the results

of the method for one of the feature sets described in [PIII]. Based on the constructed dictionaries, we demonstrate that the proposed method can effectively identify execution code fragments in PDF files, as described in Table 2.

TABLE 1 Confusion matrix for file type detection using Markov-Walk-based features. 100 files of each type were examined.

| | | **Correct File Type** | | | | | |
|---|---|---|---|---|---|---|---|
| | | PDF | DOC | EXE | GIF | JPG | HTM |
| **Classified File Type** | PDF | **93** | 1 | 0 | 0 | 9 | 0 |
| | DOC | 0 | **98** | 0 | 0 | 0 | 0 |
| | EXE | 2 | 0 | **98** | 1 | 0 | 0 |
| | GIF | 3 | 1 | 1 | **99** | 0 | 0 |
| | JPG | 1 | 0 | 0 | 0 | **91** | 0 |
| | HTM | 1 | 0 | 1 | 0 | 0 | **100** |

TABLE 2 Confusion matrix for malicious PDF detection experiment. 100 clean PDF files and 10 malicious PDF files were examined.

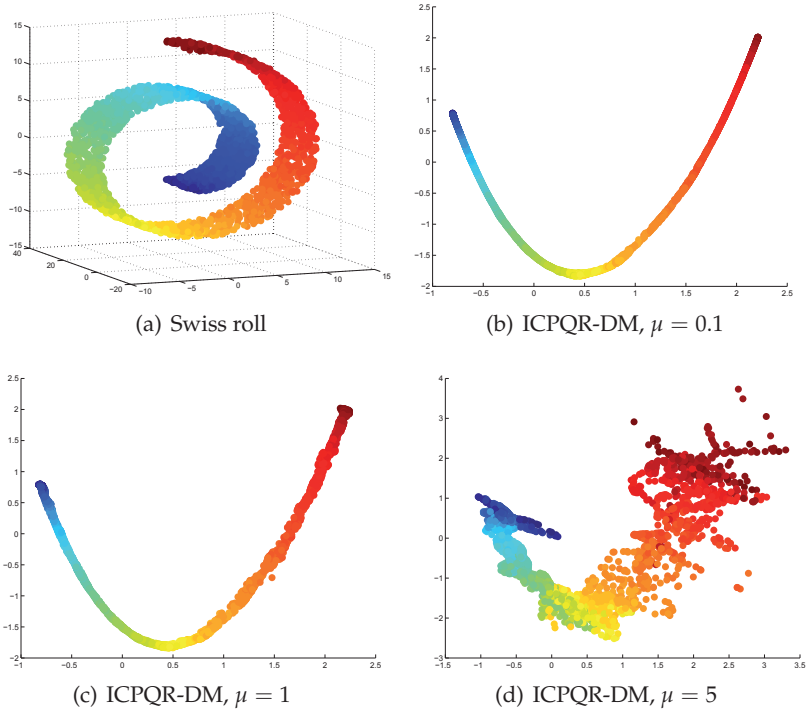
| | **Correct File Type** | | |
|---|---|---|---|
| | | PDF | Malicious PDF |
| **Classified File Type** | Safe PDF | **92** | 0 |
| | Malicious PDF | 8 | **10** |

The work in [PIV] presents a framework for geometrically-driven data analysis that includes dimensionality reduction, out-of-sample extension and anomaly detection. The method is designated to preserve a high-dimensional Euclidean geometry of parametric data, up to a user-specified distortion rate, according to the following definition:

**Definition 2.0.2 ($\mu$-distortion)** *Let $(\mathcal{H}, \mathbf{m}_{\mathcal{H}})$ and $(\mathcal{L}, \mathbf{m}_{\mathcal{L}})$ be metric spaces, let $\mathcal{A} \subset \mathcal{H}$ and let $\mu \geq 0$. A map $\mathbf{F} : \mathcal{A} \to \mathcal{L}$ is called a $\mu$-distortion of $\mathcal{A}$ if* $\sup_{x,y \in \mathcal{A}} |\mathbf{m}_{\mathcal{H}}(x, y) - \mathbf{m}_{\mathcal{L}}(\mathbf{F}(x), \mathbf{F}(y))| \leq \mu$. *The space $\mathcal{L}$ is referred to as a $\mu$-embedding space of $\mathcal{A}$.*

The proposed method is dictionary-based, where the dictionary is chosen from the analyzed dataset $\mathcal{A}$. The method identifies a Euclidean embedding space, spanned by the dictionary elements, on which the orthogonal projection of the data provides a user-defined distortion of the original high-dimensional dataset. In that sense, our method is geometrically-driven, as opposed to PCA and randomized LU decomposition [18]. Our framework preserves global patterns of the data and trades local geometry for low-dimensional representation, as dense regions in the original data (such as clusters) are more sensitive to distortions than sparse regions, such as gaps between clusters. The dictionary of the proposed method is obtained by applying a customized QR factorization algorithm, called ICPQR (InComplete Pivoted QR) on the input matrix. This algorithm iteratively selects rows (corresponding to high-dimensional data points) from the

input matrix and adds them to the dictionary. The process is finished when all the original data points lose no more than $\mu$ of their energy when projected into the space spanned by the dictionary. The result of ICPQR is demonstrated in Figure 1, where the three-dimensional Swiss roll (Figure 1(a)) is analyzed. Figures 1(b) – 1(d) present three DM dimensionality reduction versions of the Swiss roll, with different distortion levels. The larger $\mu$ is, the more distorted the two-dimensional approximation and the smaller the resulting dictionary. Figure 1(b) presents an embedding of a very low distortion, while Figures 1(c) and 1(d) show larger distortions of the embedded space.



(a) Swiss roll (b) ICPQR-DM, $\mu = 0.1$

(c) ICPQR-DM, $\mu = 1$ (d) ICPQR-DM, $\mu = 5$

FIGURE 1 (a) A 3D Swiss roll comprised of 3,000 points. (b)-(d): the two most significant coordinates of the ICPQR embedding of the Swiss roll with (b) $\mu = 0.1$, $s = 1,246$, (c) $\mu = 1$, $s = 752$, and (d) $\mu = 5$, $s = 382$, where $s$ denotes the dictionary size. Data coloring is consistent with Figure. 1(a).

Additionally to dimensionality reduction, we present two strongly related schemes for out-of-sample extension and anomaly detection. The schemes naturally stem from the proposed method for dimensionality reduction. Thus, the dimensionality reduction phase, followed by out-of-sample extension and anomaly detection, constitutes a complete framework for semi-supervised learning, where the original (in-sample) dataset $\mathcal{A}$ functions as a training set. In this context, the learning phase is reflected in the extraction of a $\mu$-embedding subspace of $\mathcal{A}$, as defined in Definition 2.0.2. Out-of-sample data points, whose projection on the representative subspace is of low distortion are classified as normal, while the

rest are classified as abnormal. Therefore, the original dataset $\mathcal{A}$ is considered as normal by definition.

To conclude, the contribution of this work is threefold. First, the suggested method identifies landmark data points (dictionary) that represent the data, as opposed to PCA that lacks this feature, and therefore in some sense is less informative. Second, the presented method requires very low storage compared to PCA. Additionally, in the worst case, its computational complexity is identical to that of PCA computation. Lastly, the proposed out-of-sample extension and anomaly detection constitute natural consequences from the dimensionality reduction phase. The suggested method is demonstrated on synthetic and real-world datasets and achieves good results in a benchmark classification task.

**Localized Diffusion Process**

The work in [PI] proposes to efficiently analyze local neighborhoods from a big data. The localization is done by deriving clusters from the data and then applying a coarse-grained version of the DM dimensionality reduction framework on clusters instead of data points. This way, the size of the analyzed dataset is decreased by only referring to clusters instead of individual data points. Figure 2 presents the pruning process which takes place after the clustering of the data and before applying the DM spectral decomposition on the clusters affinity matrix. The affinity between two clusters $C_i, C_j$ is calculated based only on the transition probabilities between points in these clusters, while ignoring diffusion paths that traverse through other clusters. We denote these direct paths as "localized diffusion paths", formally defined as

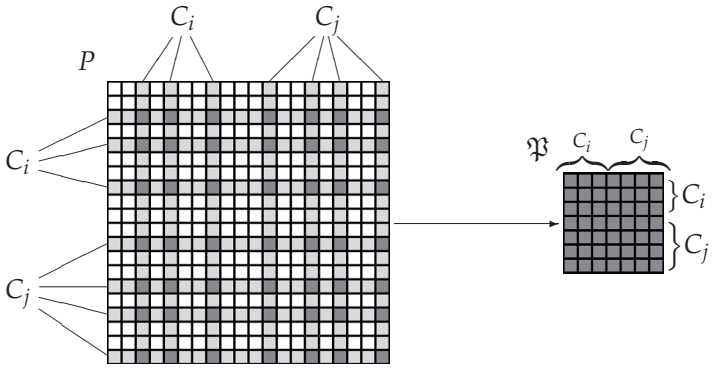
**Definition 2.0.3 (localized $\ell$-path)** *A localized $\ell$-path in a diffusion process $\mathcal{P}$ is the path $\mathcal{p} \in \mathcal{P}^\ell$ of length $\ell$ that traverses solely through data points in its source and destination clusters, i.e., $\mathcal{p}_0, \mathcal{p}_1, \ldots, \mathcal{p}_\ell \in C(\mathcal{p}_0) \cup C(\mathcal{p}_\ell)$, where $C(\mathcal{p}_i)$ denotes the cluster to which the point $\mathcal{p}_i$ in the path belongs.*

The presented coarse-graining process in this chapter copes with the rapid convergence toward a stationary distribution by only preserving localized paths between clusters while ignoring paths that are "global" from a cluster point-of-view. The difference between local and global diffusion paths is illustrated in Figures 3 and 4. While it is desirable that the clusters be sufficiently coherent to become a continuous partitioning of the dataset and its underlying manifold, the properties of the presented coarse-graining process neither depend on such assumptions nor on the exact clustering method used. We show that the essential properties, such as ergodicity, of the underlying DM-based diffusion process are preserved by the coarse-graining. The affinity, which is generated by the coarse-grained process, is called *Localized Diffusion Process* (LDP) and defined as

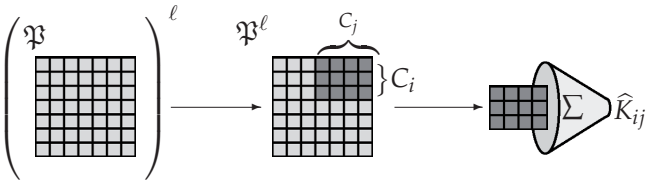
**Definition 2.0.4 ($\ell$-path localized diffusion process)** *Let $\mathcal{P}$ be a diffusion (random walk) process defined on the data points of the dataset $X$. An $\ell$-path localized diffusion process $\widehat{\mathcal{P}}$ is a random walk on the clusters $C_1, C_2, \ldots, C_{\widehat{n}}$ where a transition from $C_i$ to*

*$C_j$, $i, j = 1, \ldots, \widehat{n}$, represents all the localized $\ell$-paths in the diffusion process $\mathcal{P}$ from data points in $C_i$ to data points in $C_j$. The probability of such a transition, according to $\mathcal{P}$, is the probability to reach the destination cluster $C_j$ when starting at the source cluster $C_i$ and traveling solely via localized $\ell$-paths.*

This process is related to the hierarchical clustering algorithm *Localized Diffusion Folders* (LDF) in [7]. The theoretical work presented in this chapter proves that the LDP coarse-graining in [7] is in fact equivalent to the affinity-pruning that is achieved at each folder-level in the LDF hierarchy.

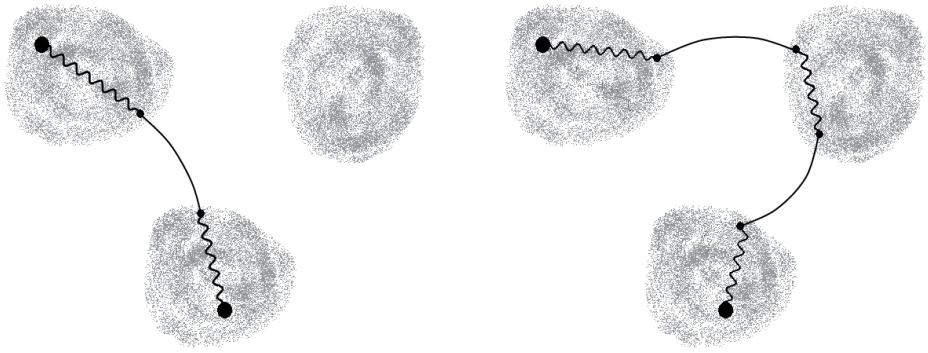



(a) Construction of the submatrix $\mathfrak{P}$ from the matrix $P$




(b) Computation of $\widehat{K}_{ij}$ as a weighted sum of cells in $\mathfrak{P}^\ell$

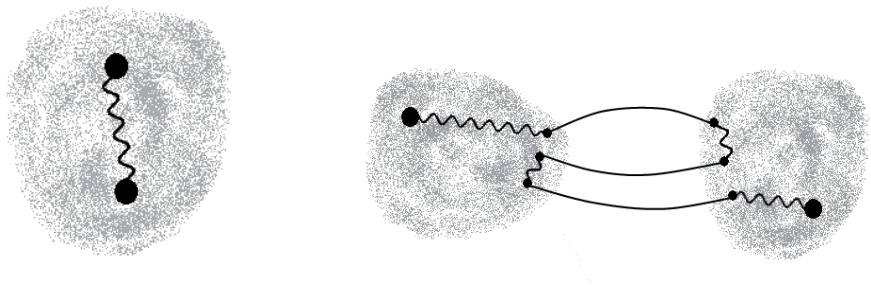
FIGURE 2 Construction of the pruned-kernel $\widehat{K}_{ij}$ between the clusters $C_i$ and $C_j$



(a) A localized path.

(b) A non-localized path

FIGURE 3 Illustration of the difference between localized and non-localized paths



(a) Localized path with identical source and destination clusters

(b) Localized path with multiple transitions between the source and destination clusters

FIGURE 4 Illustration of non-trivial localized paths

## Applications

The methods developed in this work have practical applications in real-world machine learning problems. The dictionary construction algorithm presented in [PIV] can be used for semi-supervised multi-class classification of high-dimensional data, and for anomaly detection. The classification is demonstrated on the ISOLET dataset which contains records of English letter pronunciation. The goal was to assign an unknown sample to one of 26 classes. For this task we constructed 26 dictionaries that model each of the letters, and can predict with good accuracy the class of a new letter. The anomaly detection is demonstrated on the DARPA dataset, which contains network traffic samples. Each sample can be either normal or abnormal, the latter kind pertaining to an attack and intrusion to the network. In this case only one dictionary is needed to decide whether the new sample is normal or abnormal. This dictionary is learned from the normal network traffic samples. The work presented in [PIII] has applications in computer security. This work proposes algorithms for construction of dictionaries that describe different types of computer files. By using the constructed dictionaries, the algorithm classifies the content of a file and can deduct its type by examining a few file fragments. The algorithm can also detect anomalies in PDF files (or any other rich content formats) which can be malicious. This approach can be applied to detect suspicious files that can potentially contain malicious payload. Anti-virus systems and firewalls can therefore analyze and classify PDF files by using the described method and block suspicious files.

## YHTEENVETO (FINNISH SUMMARY)

Suuria datajoukkoja (Big data) analysoivat algoritmit joutuvat käsittelemään nykyisin valtavia määriä datapisteitä, jotka sijaitsevat korkeaulotteisissa piirreavaruuksissa. Tämän kaltaisten datajoukkojen käsittelyä voi helpottaa olettamalla ja paljastamalla datan luontaisesta geometriasta lokaaleja rakenteita. Datan erilliset lokaalit rakenteet voidaan nyt analysoida erillään ja lopulta yhdistää tuloksia varten, jotta datan esittämä ilmiö voidaan ymmärtää. Käytännön Big data sovellukset edellyttävät nopeaa ja kustannustehokasta analysointia. Lokalisaatiota hyödyntävät menetelmät ovat laskennallisesti rinnakkaistettavissa ja ovat täten mielenkiintoisia Big datan käsittelyssä. Nämä menetelmät voivat tutkia datan lokaaleja ominaisuuksia, jotka vaihtelevat datan eri osissa.

Tämä väitöskirjatutkimus koostuu kahdesta eri menetelmästä, joiden avulla voidaan luokitella, klusteroida tai havaita anomalioita lokaalisti suurista datajoukkoista. Ensimmäinen menetelmä lokalisoi kirjastopohjaisen oppimisen matriisien osittamismenetelmillä. Tässä hyödynnetään satunnaistettuja LU- ja QR-hajotelma algoritmeja dataa kuvaavien kirjastojen rakentamiseen. Kirjastoja käytetään uusien datapisteiden kiinnittämiseen oikeisiin luokkiin. Yhtenä kyberturvallisuussovelluksena esitetään tietokoneen tiedostoihin pohjautuva kirjasto, joka pystyy havaitsemaan PDF-tiedostoihin piilotetut ohjelmakoodit. Toisessa sovelluksessa kirjasto on opetettu tietokoneen verkkoliikenteen normaalilla käyttöllä ja sitten testattu havaitsemaan anomaloita, jotka voivat viitata kyberuhkaan.

Toinen väitöskirjassa esitettävä menetelmä on lokalisoitu diffuusioprosessi (LDP), joka perustuu karkeasti klassiseen diffuusiokuvausalgoritmiin. Diffuusiokuvauksessa siirtymätodennäköisyysmatriisi on laskettu koko datajoukolle, kun taas LPD:ssä se on laskettu lokaaleissa pienissä datapisteiden pilvissä. Tässä työssä esitetään lokalisoidun diffuusioprosessin teoria hierarkkiselle data-analyysille.

## REFERENCES


[1] D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003.

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

[3] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 557–563. ACM, 2006.

[4] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, 2008.

[5] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[6] K. L. Clarkson. Tighter bounds for random projections of manifolds. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 39–48. ACM, 2008.

[7] G. David and A. Averbuch. Hierarchical data organization, clustering and denoising via localized diffusion folders. *Applied and Computational Harmonic Analysis*, 33(1):1–23, 2012.

[8] J. Gantz and D. Reinsel. The digital universe decade-are you ready. *IDC White Paper*, 2010.

[9] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[10] Z. Jiang, Z. Lin, and L.S. Davis. Learning a discriminative dictionary for sparse coding via label consistent k-svd. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1697–1704. IEEE, 2011.

[11] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[12] S. Lafon and A.B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1393–1403, 2006.

[13] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[14] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, Angela H. Byers, and McKinsey Global Institute. Big data: The next frontier for innovation, competition, and productivity. 2011.

[15] D.S. Pham and S. Venkatesh. Joint learning and dictionary construction for pattern recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[16] M. Salhov, G. Wolf, and A. Averbuch. Patch-to-tensor embedding. *Applied and Computational Harmonic Analysis*, 33(2):182–203, 2012.

[17] L. J. Schulman. Clustering for edge-cost minimization. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 547–555. ACM, 2000.

[18] G. Shabat, Y. Shmueli, and A. Averbuch. Randomized LU decomposition. *arXiv preprint arXiv:1310.7202*, 2015.

[19] A. Singer and H-T. Wu. Orientability and diffusion maps. *Applied and computational harmonic analysis*, 31(1):44–58, 2011.

[20] A. Singer and H-T. Wu. Vector diffusion maps and the connection laplacian. *Communications on pure and applied mathematics*, 65(8):1067–1144, 2012.

[21] S. Vempala. *The random projection method*, volume 65. American Mathematical Soc., 2005.

[22] G. Wolf and A. Averbuch. Linear-projection diffusion on smooth euclidean submanifolds. *Applied and Computational Harmonic Analysis*, 34(1):1–14, 2013.

[23] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):210–227, 2009.

[24] A.Y. Yang, J. Wright, Y. Ma, and S.S. Sastry. Feature selection in face recognition: A sparse representation perspective. *submitted to IEEE Transactions Pattern Analysis and Machine Intelligence*, 2007.

[25] M. Yang, D. Zhang, and X. Feng. Fisher discrimination dictionary learning for sparse representation. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 543–550. IEEE, 2011.

[26] Q. Zhang and B. Li. Discriminative k-svd for dictionary learning in face recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2691–2698. IEEE, 2010.

# ORIGINAL PAPERS

# PI

## COARSE-GRAINED LOCALIZED DIFFUSION

by

Guy Wolf, Aviv Rotbart, Gil David, Amir Averbuch 2012

Applied and Computational Harmonic Analysis(3):388-400

# Coarse-Grained Localized Diffusion

Guy Wolf, Aviv Rotbart, Gil David, Amir Averbuch*

**Abstract**

Data-analysis methods nowadays are expected to deal with increasingly large amounts of data. Such massive datasets often contain many redundancies. One effect from these redundancies is the high-dimensionality of datasets, which is handled by dimensionality reduction techniques. Another effect is the duplicity of very similar observations (or data-points) that can be analyzed together as a cluster. We propose an approach for dealing with both effects by coarse-graining the popular *Diffusion Maps* (DM) dimensionality reduction framework from the data-point level to the cluster level. This way, the size of the analyzed dataset is decreased by only referring to clusters instead of individual data-points. Then, the dimensionality of the dataset can be decreased by the DM embedding. We show that the essential properties (e.g., ergodicity) of the underlying diffusion process of DM are preserved by the coarse-graining. The affinity that is generated by the coarse-grained process, which we call *Localized Diffusion Process* (LDP), is strongly related to the recently introduced *Localized Diffusion Folders* (LDF)[1] hierarchical clustering algorithm. We show that the LDP coarse-graining is in fact equivalent to the affinity-pruning that is achieved at each folder-level in the LDF hierarchy.

*Keywords:* Diffusion Maps, Localized Diffusion Folders, coarse-graining, dimensionality reduction

## 1. Introduction

Massive high-dimensional datasets have become an increasingly common input for data-analysis tasks. When dealing with such datasets, one requires a method that reduces the complexity of the data while preserving the essential information for the analysis. One approach for obtaining this goal is to analyze sets of closely-related data-points, instead of directly analyzing the raw data-points. A recent approach for obtaining such an analysis is the Localized

*Amir Averbuch, Tel: +972-54-5694455, Fax: +972-3-6422020
*Email address:* amir@math.tau.ac.il (Amir Averbuch)

[1] G. David, A. Averbuch, Hierarchical data organization, clustering and denoising via localized diffusion folders, Applied and Computational Harmonic Analysis. In press, to appear.

Diffusion Folders (LDF) method [1]. This method recursively prunes closely-related clusters, while preserving the information about local relations between the pruned clusters.

The Diffusion Maps (DM) framework [2, 3] provides an essential foundation for LDF to succeed. This framework is based on defining similarities between data-points by using an ergodic Markovian diffusion process on the dataset. The ergodicity of this process ensures it has a stationary distribution and numerically-stable spectral properties. The transition probability matrix of this process can be used to define diffusion affinities between data-points. The first few eigenvectors of this diffusion affinity kernel represent the long-term behavior of the process and they can be used to obtain a low dimensional representation of the dataset, in which the Euclidean distances between data-points correspond to diffusion distances between their original (high-dimensional) counterparts. We present a coarse-graining of this diffusion process, while preserving its essential properties (e.g., ergodicity). We show that this coarse-graining is equivalent to the pruning method that appeared in the LDF.

The LDF method performs an iterative process that obtains a folder hierarchy that represents the points in the dataset. Each level in the hierarchy is constructed by pruning clusters of folders (or data-points) from the previous level. The iterative process has two main phases in each iteration:

1. **Clustering phase:** the "shake & bake" method is used to cluster the folders (or data-points) of the current level in the hierarchy by using a diffusion affinity matrix.
2. **Pruning phase:** the clusters of the current level are pruned and given as folders of the next level in the hierarchy. The diffusion affinity is also pruned to represent affinities between pruned clusters (i.e., folders of the next level in the hierarchy) instead of folders in the current hierarchial level.

In this paper, we focus on exploring the pruning that is performed in the second phase of this process, while considering the clustering of the data, which may be performed by "shake & bake" process [1] or by another clustering algorithm, as prior knowledge.

Essentially, LDF provides an hierarchical data clustering with additional affinity information for each level in the hierarchy. Other examples of hierarchical clustering methods can be found in [4, 5]. However, these methods are not related to DM and to its underlying diffusion process. Since we are mainly concerned with the pruning phase of the algorithm, the clustering aspect of LDF and its relation with these methods is beyond the scope of this paper. A detailed survey of clustering algorithms and their relation to LDF is provided in [1, Section 2].

While there are many empirical justifications for the merits of LDF and its utilization in various fields (e.g., unsupervised learning and image processing), it lacked theoretical justifications. In this paper, we introduce a coarse-graining of the underlying diffusion process of DM. The resulting coarse-grained process, which we call Localized Diffusion Process (LDP), preserves essential properties

of the original process, which enable its utilization for dimensionality reduction tasks. We relate this process, or rather the diffusion affinity generated by it, to the one achieved by the LDF pruning phase. This relation adds the needed complimentary foundations for the LDF framework by providing theoretical justifications for its already-obtained empirical support. Additionally, the presented relation shows that the applications presented in [1] in fact demonstrate the utilization of the LDP for data-analysis tasks and the results presented there provide empirical support of its benefits.

A similar coarse-graining approach was presented in [6]. The approach there is based on a graph representation of the diffusion random-walk process. The clustering of data-points was performed by graph partitioning. Then, transition probabilities between partitions were achieved by averaging transition probabilities between their vertices. The resulting random-walk process maintains most of the spectral properties of the original diffusion process and its eigendecomposition can be approximated by the original spectral decomposition. However, the approximation error strongly depends on the exact partitioning used. In addition, since all the random-walk paths are considered in the averaging process, there is a limited number of viable time-scales (in the diffusion process) that can be used by this process before it converges to the averaging of the stationary distribution.

The presented coarse-graining process in this paper copes with the rapid convergence toward the stationary distribution by only preserving localized paths between clusters while ignoring paths that are "global" from the cluster point-of-view. While it is desirable that the clusters will be sufficiently coherent to consist of a continuous partitioning of the dataset and its underlying manifold, the properties of the presented coarse-graining process are neither depend on such assumptions nor on the exact clustering method used.

An alternative approach for local sets considerations of data-points is to analyze them as patches on the underlying manifold of the dataset [7, 8, 9, 10]. The relations between patches are represented by non-scalar affinities that combine the information about both geodesic proximity of the patches and the alignment of their tangent spaces. This approach was used in [9] to modify DM to preserve the orientation of the manifold through the embedding process. A more comprehensive utilization of this approach was presented in [7] and [10], where affinities between patches were defined as matrices that transform vectors between tangent spaces. Parallel transport operators were used in [10], with the resulting affinity block matrix being related to the connection-Laplacian. Linear-projections were used in [7] and further explored in [8], where the resulting diffusion process was shown to propagate tangent vectors on the manifold.

Both discussed methods in [7, 10] lead to an embedded tensor space instead of a vector space. Also, the resulting diffusion process is not necessarily ergodic and may not have a stationary distribution. Therefore, they do not preserve one of the crucial properties of the diffusion process used in DM. The approach used in this paper produces a scalar-affinity matrix between closely-related clusters of data-points. It neither depends explicitly on the existence nor on the knowledge

of the (usually unknown) underlying manifold of the dataset. The resulting diffusion process is similar to the one used in DM (for data-points), and it preserves the essential properties of that diffusion process. Finally, the same spectral analysis, which is performed in DM, can be used to obtain an embedding that is based on the coarse-grained process presented here, which results with an embedding of clusters to vectors (and not tensors).

The paper has the following structure. The problem setup is described in Section 2. Specifically, the DM method is discussed in Section 2.1 and the LDF method is discussed in Section 2.2. Section 3 introduces the localized diffusion process (LDP), which is the main construction in this paper. The pruning algorithm for constructing the LDP is presented in Section 3.1. Finally, the strong relation between LDF and LDP is presented in Section 3.2.

## 2. Problem Setup

Let $X \subset \mathbb{R}^d$ be a dataset of $n$ data-points that are sampled from a low dimensional manifold that lies in a high dimensional Euclidean ambient space. Assume the data consists of $\widehat{n}$ coherent disjoint clusters, which correspond to dense local neighborhoods that were generated by an affinity kernel. Assume that $C_1, C_2, \ldots, C_{\widehat{n}}$ are these clusters in the underlying manifold, where $X = \cup_{i=1}^{\widehat{n}} C_i$ and $C_i \cap C_j = \emptyset$ for $i \neq j \in \{1, 2, \ldots, \widehat{n}\}$. Assume that

$$C : X \to \{C_1, C_2, \ldots, C_{\widehat{n}}\} \tag{2.1}$$

maps each data-point $x \in X$ to its cluster $C(x)$.

*Remark about matrix notation:.* In this paper, we will deal with several matrices that represent relations between data-points or clusters of data-points. Let $M$ be such a matrix where every row and column of $M$ corresponds to a data-point in the dataset $X$ or a subset of this dataset. It is convenient in this case to use the lowercase notation $m(x, y)$ to denote the cell in the $x$'s row and the $y$'s column in $M$. For $t \in \mathbb{Z}$, the notation $m^t(x, y)$ denotes cells in $M^t$, which is the $t$-th power of $M$. Similar notation will also be used for matrices with rows and columns that correspond to clusters of data-points.

### *2.1. Diffusion maps*

The diffusion maps (DM)[2] methodology is based on constructing a Markovian diffusion process $\mathcal{P}$ over a dataset. This process essentially defines random walks over data-points in the dataset. It consists of paths between these data-points, where each path $\mathcal{p} \in \mathcal{P}$ is a series of transitions (steps on the data-points), denoted by $\mathcal{p}_0 \to \mathcal{p}_1 \to \ldots \to \mathcal{p}_\ell$, where $\mathcal{p}_0, \mathcal{p}_1, \ldots, \mathcal{p}_\ell \in X$, $\ell \geq 1$. Each path has a probability, which is defined by the probabilities of its transitions and will be discussed later. The length of the path $\mathcal{p}$, denoted by $\mathrm{len}(\mathcal{p}) = \ell$, is its number of transitions. The source (i.e., the starting data-point) of the path is denoted by $\mathrm{s}(\mathcal{p}) = \mathcal{p}_0$ and its destination is denoted by $\mathrm{t}(\mathcal{p}) = \mathcal{p}_{\mathrm{len}(\mathcal{p})} = \mathcal{p}_\ell$. When only paths of specific length $\ell = 1, 2, \ldots$, are considered, the notation

$\mathcal{P} \in \mathcal{P}^\ell$ will be used to denote that $\mathcal{P} \in \mathcal{P}$ and $\text{len}(\mathcal{P}) = \ell$. For example, a single transition in $\mathcal{P}$ is a path of unit length $\mathcal{P} \in \mathcal{P}^1$.

In order to assign transition probabilities between data-points, an $n \times n$ affinity kernel $K$ is defined on the dataset. Each cell $k(x, y), x, y \in X$, in this kernel represents similarity, or proximity, between data-points. The kernel $K$ is interpreted as both an affinity measure between data-points and a weighted adjacency matrix of a graph whose vertices are the data-points. It is assumed to satisfy the following properties:

- Each data-point has positive self-affinity: $k(x, x) > 0$, $x \in X$;
- Affinities are non-negative: $k(x, y) \geq 0$, $x, y \in X$;
- Affinities are symmetric: $k(x, y) = k(y, x)$, $x, y \in X$;
- The graph defined by the weighted adjacencies $K$ is connected.

A popular affinity kernel is the isotropic Gaussian diffusion kernel $k(x, y) = \exp(-\|x - y\| / \varepsilon)$ with a suitable $\varepsilon > 0$. An alternative kernel, which is based on clustering patterns in the dataset, is the "shake-and-bake" kernel [1] that will be discussed in more details in Section 2.2.

Each data-point $x \in X$ corresponds to a vertex in the graph that is defined by $K$. The degree of a vertex in this graph is $q(x) \triangleq \sum_{y \in X} k(x, y)$, $x \in X$. The degree matrix $Q$ is a diagonal matrix whose main diagonal holds these degrees (i.e., $q(x, x) = q(x)$ and $q(x, y) = 0$ for $x \neq y \in X$). Normalization by these degrees yields a row-stochastic matrix $P \triangleq Q^{-1}K$ that defines the transition probabilities

$$p(x, y) = \frac{k(x, y)}{q(x)} \qquad x, y \in X,$$

between data-points. These transition probabilities define the Markovian diffusion process $\mathcal{P}$ over the dataset.

The diffusion process $\mathcal{P}$ specifies the probability of "moving" from one data-point to another via paths of any given integer length $\ell \geq 1$. We denote this probability by

$$\Pr[x \xrightarrow{\mathcal{P}^\ell} y] \triangleq \Pr[\text{t}(\mathcal{P}) = y | \, \text{s}(\mathcal{P}) = x \wedge \mathcal{P} \in \mathcal{P}^\ell] \qquad x, y \in X. \tag{2.2}$$

Since the diffusion process is a Markovian process with single-transition probabilities defined by $P$, Eq. (2.2) becomes

$$\Pr[x \xrightarrow{\mathcal{P}^\ell} y] = p^\ell(x, y) \qquad x, y \in X, \ell = 1, 2, \ldots,$$

where in particular $\Pr[x \xrightarrow{\mathcal{P}^1} y] = p(x, y)$.

The diffusion process $\mathcal{P}$ is an ergodic Markov process. This means that $\mathcal{P}$ has a stationary distribution in the limit $\ell \to \infty$ of the path lengths. Spectral analysis of this kernel yields a decaying spectrum $1 = \lambda_0 \geq |\lambda_1| \geq |\lambda_2| \geq \ldots \geq 0$, where $\lambda_i$, $i = 0, 1, 2, \ldots$, are the eigenvalues of $P$. When an isotropic

Gaussian kernel is used, the decay of the spectrum can be used to approximate the intrinsic dimension of the dataset's underlying manifold [2]. Dimensionality reduction can be achieved by spectral analysis of $P$ [11] or, more conveniently, its symmetric conjugate $A = Q^{1/2}PQ^{-1/2}$ that is referred to as the diffusion-affinity matrix. Let $\phi_0, \phi_1, \phi_2, \ldots$ be the eigenvectors of $A$ that correspond to the eigenvalues $\lambda_0, \lambda_1, \lambda_2, \ldots$ (conjugation maintains the same eigenvalues of $A$), then DM is defined by the embedding

$$x \mapsto \Phi(x) \triangleq (|\lambda_0|\,\phi_0(x), |\lambda_1|\,\phi_1(x), |\lambda_2|\,\phi_2(x), \ldots)^T \qquad x \in X.$$

A subset of these coordinates can be used by ignoring the eigenvectors with sufficiently small eigenvalues, which will anyway result with approximately-zero embedded coordinates.

A simple coarse-graining of the original diffusion process can be done by cluster pruning while defining transition probability between two clusters by considering all the paths between them. However, due to the decay of the diffusion kernel's spectrum, this method will converge fast (especially when applied several times) to the stationary distribution of the diffusion process. An alternative coarse graining method, which excludes paths that are considered "global" from clusters point-of-view, will be presented in Section 3.

One aspect of any diffusion process coarse-graining is to translate a data-point terminology to a cluster terminology. This aspect must be addressed regardless of the paths that are considered when computing the transitional probabilities between clusters, since any path in the diffusion process is defined in terms of data-points. The probability of reaching every data-point on a path is determined by its starting data-point and by suitable powers of $P$. Since the clusters are disjoint, these probabilities can be easily interpreted as the probability of reaching a destination cluster. Specifically, it can be done by using the function $C$ (Eq. (2.1)) and by summing the appropriate probabilities. Paths that start in a source cluster, denoted by $\mathrm{s}(\mathcal{P}) \in C_i$, $\mathcal{P} \in \mathcal{P}$, $i = 1, \ldots, \widehat{n}$, require a nontrivial interpretation in terms of a source data-point $\mathrm{s}(\mathcal{P}) = x \in C_i$. This interpretation should be defined by probability terms.

We will use the same intuition that was used to construct the transitional probability matrix $P$ in order to define the probability $\Pr[\mathrm{s}(\mathcal{P}) = x \in C_i \,|\, \mathrm{s}(\mathcal{P}) \in C_i]$, $i = 1, \ldots, \widehat{n}$, $\mathcal{P} \in \mathcal{P}$. The kernel $K$ was interpreted as weighted adjacencies of a graph whose vertices are the data-points in $X$. According to this interpretation, the degree of each data-point $x \in X$ is a sum of the edges weights $k(x, y), y \in X$ that begin at $x$. To measure the occurrence probability of the transition $x \to y$ when starting at $x$, the weight of the edge $(x, y)$ is divided by the total weight of the edges starting at $x$, which gives the probability measure $p(x, y) = k(x, y)/q(x)$. Assume the volume of the cluster $C_i$, $i = 1, \ldots, \widehat{n}$, is defined as $\mathrm{vol}(C_i) \triangleq \sum_{x \in C_i} q(x)$. Therefore, the volume of a cluster is the total sum of the degrees of the data-points in this cluster, which is the sum of the weights of all the edges that start in $C_i$. According to the same reasoning as before, the occurrence probability of the transition $x \to y, x \in C_i, y \in X$, which started at the cluster $C_i$, is $\frac{k(x,y)}{\mathrm{vol}(C_i)}$. Therefore, the transition probability, which

starts at $C_i$ to actually starts at a specific data-point $x \in C_i$, is

$$\Pr[\mathrm{s}(\mathcal{P}) = x \in C_i | \, \mathrm{s}(\mathcal{P}) \in C_i] = \sum_{y \in X} \frac{k(x,y)}{\mathrm{vol}(C_i)} = \frac{q(x)}{\mathrm{vol}(C_i)}, \tag{2.3}$$

because the transitions to different designated data-points are independent events. Notice that the choice of the first transition in a path is independent of its length. Thus, the presented probability is independent of the length of the path $\mathcal{P}$ and the assumption that $\mathcal{P} \in \mathcal{P}^\ell$ for some $\ell \geq 1$ does not affect it.

### *2.2. Localized diffusion folders (LDF)*

As described in the DM brief overview in Section 2.1, $P$ is the affinity matrix of the dataset and it is used to find the diffusion distances between data-points. This distance metric can be used to cluster data-points according to the diffusion distances propagation that is controlled by the time parameter $t$. In addition, it can be used to construct a bottom-up hierarchical data clustering. For $t = 1$, the affinity matrix reflects direct connections between data-points. These connections can be interpreted as local adjacencies between data-points. The resulting clusters preserve the local neighborhood of each data-point. These clusters are the bottom level in the hierarchy. By raising $t$, which means time advancement, the affinity matrix is changed accordingly and it reflects indirect rare connections between data-points in the graph. The diffusion distance between data-points in the graph accounts for all possible paths of length $t$ between these data-points at a given time step. The more we advance in time the more we increase indirect and global connections. Therefore, by raising $t$ we can construct the upper levels of the clustering hierarchy. In each time step, it is possible to merge more and more low-level clusters since there are more and more new paths between them. The resulting clusters reflect global neighborhood of each data-point that is highly affected by the advances of the parameter $t$.

The major risk in this global approach is that increasing $t$ will also increase noise, which is classified as connections between data-points that are not closely related in the affinity matrix. Moreover, clustering errors in the lower levels of the hierarchy will diffuse to the upper levels of the hierarchy and hence will significantly affect the correctness of the upper levels clustering. As a result, some areas in the graph, which are assumed to be separated, will be connected by the new noise-result and error-result paths. Thus, erroneous clusters will be generated (a detailed description of this situation is given in [1]). This type of noise significantly affects the diffusion process and eventually the resulting clusters will not reflect the correct relations among the data-points. Although these clusters consist of data-points that are adjacent according to their diffusion distances, the connections among these data-points in each cluster can be classified as too global and too loose that generate inaccurate clusters.

A hierarchical clustering method of high-dimensional data via the *localized diffusion folders* (LDF) methodology is introduced in [1]. This methodology overcomes the problems that were described above. It is based on the key

idea that clustering of data-points should be achieved by utilizing the local geometry of the data and the by local neighborhood of each data-point and by constructing a new local geometry every advance in time. The new geometry is constructed according to local connections and according to diffusion distances in previous time steps. This way, as we advance in time, the geometry from the induced affinity reflects better the data locality while the "affinity noise" in the new localized matrix decreases and the accuracy of the resulting clusters is improved.

LDF is introduced to achieve the described local geometry and to preserve it along the hierarchical construction. The LDF framework provides a multi-level partitioning (similar to Voronoi diagrams in diffusion metric) of the data into local neighborhoods that are initiated by several random selections of data-points or folders of data-points in the diffusion graph and by defining local diffusion distances between them. Since every different selection of initial data-points yields a different set of *diffusion folders* (DF), it is crucial to repeat this selection process several times. The multiple system of folders, which we get at the end of this random selection process, defines a new affinity and this reveals a new geometry in the graph. This localized affinity is a result of what is called the "shake & bake" process in [1]. First, we "shake" the multiple Voronoi diagrams together in order to get rid of the noise in the original affinity. Then, we "bake" a new cleaner affinity that is based on the actual geometry of the data while eliminating rare connections between data-points. This affinity is more accurate than the original affinity since instead of defining a general affinity on the graph, we let the data define its localized affinity on the graph.

In every time step, this multi-level partitioning defines a new localized geometry of the data and a new localized affinity matrix that is used in the next time step. In every time step, we use the localized geometry and the LDF that were generated in the previous time step to define the localized affinity between DF. The affinity between two DF is defined by the localized diffusion distance metric between data-points in the two DF. In order to define this distance between these DF, we construct a local sub-matrix that contains only the affinities between data-points (or between DF) of the two DF. This sub-matrix is raised to the power of the current time step (according to the current level in the hierarchy) and then it is used to find the localized diffusion distance between the two DF.

The result of this clustering method is a bottom-up hierarchical data clustering where each level in the hierarchy contains DF of DF from lower levels. Each level in the hierarchy defines a new localized affinity (geometry) that is dynamically constructed and it is used by the upper level. This methodology preserves the local neighborhood of each data-point while eliminating the noisy connections between distinct points and areas in the graph.

In summary, [1] deals with new methodologies to denoise empirical graphs. Usually, in applications data is connected through spurious connections. One of the goals of [1] is to introduce a notion of consistency of connections in order to repair a noisy network. This consistency is achieved through the construction of a forest of partition trees, which redefine the connectivity in the network. This

opens the door to robust processing of data clouds in which group consistency is exploited.

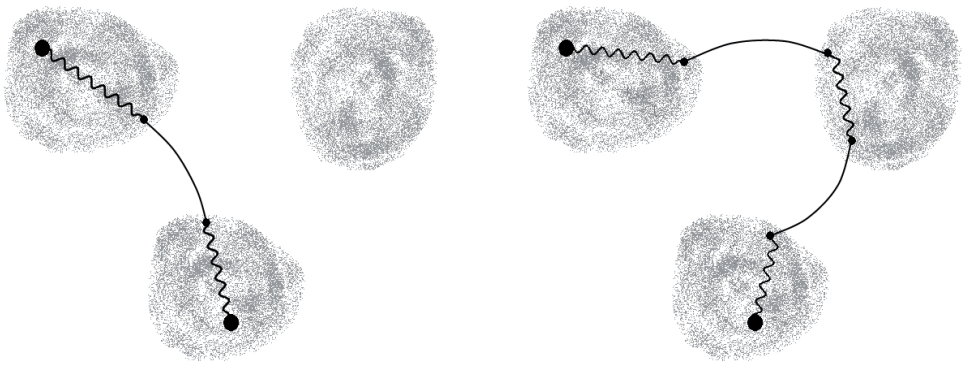
## 3. Localized Diffusion Process

In this section, we present a coarse-graining diffusion process between clusters in a dataset. The transitions between clusters, which are considered as vertices in this process, will be defined by certain paths in the original diffusion process. Definition 3.1 introduces the notion of a localized path, which will be used to define these transitions. Then, in Definition 3.3, these localized paths will be used to define the localized diffusion process between clusters.

**Definition 3.1** (localized $\ell$-path)**.** *A localized $\ell$-path in a diffusion process $\mathcal{P}$ is the path $\mathcal{p} \in \mathcal{P}^\ell$ of length $\ell$ that traverses solely through data-points in its source and destination clusters, i.e., $\mathcal{p}_0, \mathcal{p}_1, \ldots, \mathcal{p}_\ell \in C(\mathrm{s}(\mathcal{p})) \cup C(\mathrm{t}(\mathcal{p}))$.*

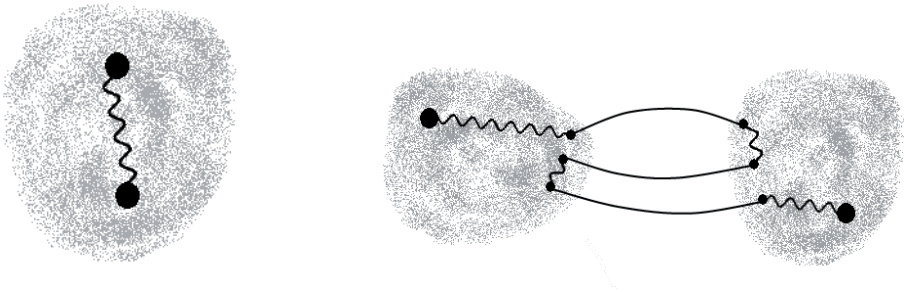
The difference between localized and non-localized paths is demonstrated in Fig. 3.1. The path in Fig. 3.1(a) traverses through data-points in its source cluster, then passes via a single transition to its destination cluster and then traverses in it to its destination data-point. Therefore, it does not pass through any cluster other than its source and destination clusters and thus it is localized. On the other hand, the non-localized path in Fig 3.1(b), traverses through a third intermediary cluster, thus it is not localized.

Notice that a localized path does not necessarily contain a single transition between its source and destination clusters. Figure 3.2 illustrates two such non-trivial paths. A path that traverses solely in a single cluster (see Fig. 3.2(a)) is in fact a localized path from the cluster to itself. A localized path can also alternate between its source and destination clusters a few times before reaching its final destination, as shown in Fig. 3.2(b). As long as the cluster involves only its source and destination cluster/s (whether they are identical or not) without passing through any intermediary cluster, then it is considered to be localized.



(a) A localized path. (b) A non-localized path

Figure 3.1: Illustration of the difference between localized and non-localized paths



(a) Localized path with identical source and destination clusters

(b) Localized path with multiple transitions between the source and destination clusters

Figure 3.2: Illustration of non-trivial localized paths

We denote the set of all localized $\ell$-paths in the diffusion process $\mathcal{P}$ by $\mathfrak{L}(\mathcal{P}^\ell) \subseteq \mathcal{P}^\ell$. The usual diffusion transition probabilities between data-points in a dataset via paths of a given length $\ell \geq 1$ were described in Section 2.1. These probabilities consider all the paths of length $\ell$ between two data-points. The construction presented in this paper only considers localized paths and ignores other paths, which are considered "global" from a cluster point-of-view. Therefore, we define the localized transition probabilities, which describe the probabilities of a transition from $x \in C_i$ to $y \in C_j$, $i, j = 1, \ldots, \widehat{n}$, via localized $\ell$-paths, as

$$\Pr[x \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} y] \triangleq \Pr[\mathrm{t}(\mathcal{P}) = y \wedge \mathcal{P} \in \mathfrak{L}(\mathcal{P}^\ell) | \, \mathrm{s}(\mathcal{P}) = x \wedge \mathcal{P} \in \mathcal{P}^\ell]. \tag{3.1}$$

Similarly, the localized transition probability from $x \in C_i$ to the cluster $C_j$ is defined as

$$\Pr[x \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} C_j] \triangleq \Pr[\mathrm{t}(\mathcal{P}) \in C_j \wedge \mathcal{P} \in \mathfrak{L}(\mathcal{P}^\ell) | \, \mathrm{s}(\mathcal{P}) = x \wedge \mathcal{P} \in \mathcal{P}^\ell]. \tag{3.2}$$

Finally, the localized transition probability from the cluster $C_i$ to the cluster $C_j$ is defined as

$$\Pr[C_i \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} C_j] \triangleq \Pr[\mathrm{t}(\mathcal{P}) \in C_j \wedge \mathcal{P} \in \mathfrak{L}(\mathcal{P}^\ell) | \, \mathrm{s}(\mathcal{P}) \in C_i \wedge \mathcal{P} \in \mathcal{P}^\ell]. \tag{3.3}$$

The original transition probabilities are clearly related to the diffusion operator $\mathcal{P}$ via Eq. (2.2). The defined localized transition probabilities do not have such a direct relation with the diffusion operator. They will be further explored in Section 3.1.

The localized transition probabilities in Eq. (3.3) consider localized paths from a source cluster to a destination cluster. Not all the paths from the source cluster are localized. Therefore, only a portion of the paths from a given cluster (to any other cluster) are actually viable for consideration with these probabilities. Definition 3.2 provides a measure for the portion of viable paths going out from a cluster from all the paths starting in it.

**Definition 3.2** ($\ell$-path localization probability)**.** *The $\ell$-path localization probability (lpr) of a cluster $C_i$, $i = 1, \ldots, \widehat{n}$, is*

$$\mathrm{lpr}^{\ell}(C_i) \triangleq \Pr[p \in \mathfrak{L}(\mathcal{P}^{\ell}) | \, \mathrm{s}(\mathscr{P}) \in C_i \wedge \mathscr{P} \in \mathcal{P}^{\ell}].$$

*It is the probability that a path of length $\ell$, which starts at this cluster, is a localized path.*

Definition 3.3 uses the defined localized paths in the original diffusion process to define a localized diffusion process between clusters.

**Definition 3.3** ($\ell$-path localized diffusion process)**.** *Let $\mathcal{P}$ be a diffusion (random walk) process defined on the data-points of the dataset $X$. An $\ell$-path localized diffusion process $\widehat{\mathcal{P}}$ is a random walk on the clusters $C_1, C_2, \ldots, C_{\widehat{n}}$ where a transition from $C_i$ to $C_j$, $i, j = 1, \ldots, \widehat{n}$, represents all the localized $\ell$-paths in the diffusion process $\mathcal{P}$ from data-points in $C_i$ to data-points in $C_j$. The probability of such a transition, according to $\mathcal{P}$, is the probability to reach the destination cluster $C_j$ when starting at the source cluster $C_i$ and traveling solely via localized $\ell$-paths.*

The $\ell$-path localized diffusion process is a Markovian random-walk process. Thus, its transition probabilities are completely governed by its single-step transition probabilities. These probabilities can be computed, by definition, according to the transition probabilities of the original diffusion process. Using notations similar to the ones used for the original diffusion process, we get the single-step transition probabilities

$$\Pr[C_i \xrightarrow{\widehat{\mathcal{P}}^1} C_j] \triangleq \Pr[\mathrm{t}(\mathscr{P}) \in C_j | \, \mathrm{s}(\mathscr{P}) \in C_i \wedge \mathscr{P} \in \mathfrak{L}(\mathcal{P}^{\ell})], \; i, j = 1, \ldots, \widehat{n}, \quad (3.4)$$

for the $\ell$-path localized diffusion process $\widehat{\mathcal{P}}$. Notice that these differ from the probabilities in Eq. (3.3). The former considers the term $\mathscr{P} \in \mathfrak{L}(\mathcal{P}^{\ell})$ in the hypothesis part, since it considers only the localized paths of the original diffusion process. The latter considers this term in the condition part, since it computes the probability over all the paths in the original diffusion.

Ergodicity is one of the main properties in the diffusion process that is used by DM [2]. Ergodicity means that the eigenvalues of $P$ have a magnitude of at most one, and therefore its spectrum decays with time as a function of the numerical rank of the transitional kernel. As we advance the diffusion process in time, it converges to a stationary distribution and therefore its long term state can be represented by a low-dimensional space. Proposition 3.1 shows that the coarse graining suggested here preserves this property, i.e., the $\ell$-path localized diffusion process is ergodic and its transition matrix has a decaying spectrum.

**Proposition 3.1.** *The localized diffusion process $\widehat{\mathcal{P}}$, which is defined by Definition 3.3, is an ergodic Markov process.*

*Proof.* According to [2], the original diffusion process $\mathcal{P}$ is aperiodic and irreducible. We will show that $\widehat{\mathcal{P}}$ is also aperiodic and irreducible process. The Ergodicity follows from these properties.

From the aperiodicity of $\mathcal{P}$ we have $p(x,x) > 0$ for every $x \in X$. Let $\mathcal{p} \in \mathcal{P}^\ell$ be a path with $\mathcal{p}_i = x$, $i = 0, \ldots, \ell$. Obviously, this path is a localized $\ell$-path from $C(x)$ to itself. The probability of this path is $(p(x,x))^\ell > 0$. Therefore, the transition probability $\Pr[C(x) \xrightarrow{\widehat{\mathcal{P}}^\ell} C(x)]$, which sums the probabilities of all the localized $\ell$-paths from $C(x)$ to itself, must be nonzero. This argument holds for every $x \in C_i \subseteq X$, $i = 1, \ldots, \widehat{n}$, and thus holds for every cluster $C_i = C(x)$. Therefore, the process $\widehat{\mathcal{P}}$ is aperiodic.

Due to the irreducibility of the original diffusion process $\mathcal{P}$, there exists a path $\mathcal{p} \in \mathcal{P}$ with nonzero probability between every pair of data-points $x \neq y \in X$. For each transition $\mathcal{p}_i \to \mathcal{p}_{i+1}$, $i = 0, \ldots, \mathrm{len}(\mathcal{p}) - 1$ in this path, the following localized $\ell$-path

$$\mathcal{p}' = \underbrace{\mathcal{p}_i \to \mathcal{p}_i \to \ldots \to \mathcal{p}_i}_{\ell-1 \text{ transitions}} \to \mathcal{p}_{i+1}$$

between $C(\mathcal{p}_i)$ and $C(\mathcal{p}_{i+1})$ is constructed. Due to the aperiodicity of $\mathcal{P}$, the first $\ell - 1$ transitions have nonzero probability. The last transition of $\mathcal{p}'$ is the same transition $\mathcal{p}_i \to \mathcal{p}_{i+1}$ from $\mathcal{p}$, which also has nonzero probability. Therefore, the path $\mathcal{p}'$ is a localized $\ell$-path between $C(\mathcal{p}_i)$ and $C(\mathcal{p}_{i+1})$ with nonzero probability. This holds for every transition in $\mathcal{p}$ and thus the transition probability from $C(\mathcal{p}_i)$ to $C(\mathcal{p}_{i+1})$ via the localized $\ell$-paths is nonzero for each $i = 0, \ldots, \mathrm{len}(\mathcal{p}) - 1$. Thus, the path

$$C(x) = C(\mathcal{p}_0) \to C(\mathcal{p}_1) \to \ldots \to C(\mathcal{p}_{\mathrm{len}(\mathcal{p})}) = C(y)$$

has nonzero probability in $\widehat{\mathcal{P}}$. Since $x, y$ were chosen arbitrarily, this holds for every pair of clusters and thus $\widehat{\mathcal{P}}$ is irreducible. Together with the aperiodicity that was shown above, $\widehat{\mathcal{P}}$ is ergodic. □

In this section, we introduced a coarse-grained diffusion process (i.e., the $\ell$-path localized diffusion process) that preserves the crucial properties of the DM. A coarse-graining algorithm, which constructs this process, is presented in Section 3.1. In Section 3.2, we will show that this process is directly related to the construction presented in [1]. Specifically, the presented coarse graining is related to the pruning done at the transition between levels in the LDF hierarchy.

### *3.1. Pruning algorithm*

The $\ell$-path localized diffusion process described in Section 3 is a coarse-grained version of the original diffusion process. As a Markovian process, it defines a transition probability matrix between clusters. Algorithm 3.1 shows how to construct this transition probability matrix, denoted by $\widehat{P}$, based on the transition probability matrix of the original diffusion process. In addition to $\widehat{P}$, the algorithm outputs the degree matrix $\widehat{Q}$ that holds the degrees of the clusters on its diagonal. Theorem 3.2 shows that the resulting matrix $\widehat{P}$ defines a localized diffusion process.

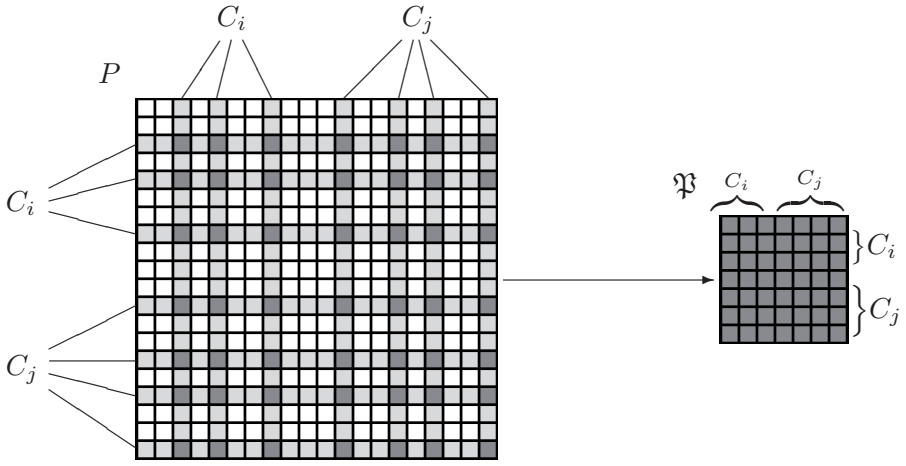
Algorithm 3.1 performs a coarse-graining of the original diffusion process by pruning the clusters into vertices of a Markovian random-walk process. The transition probabilities of this process are determined by the row-stochastic transition matrix $\widehat{P}$. For each pair of clusters, $C_i$ and $C_j$, the algorithm considers the sub-matrix $\mathfrak{P}$ of $P$, which contains only rows and columns of data-points in $C_i \cup C_j$ (see Fig. 3.3(a)). The algorithm then calculates the affinity, denoted by $K_{ij}$, between the clusters $C_i$ and $C_j$. First, It raises the sub-matrix $\mathfrak{P}$ to the $\ell$-th power in order to generate the $\ell$-path localized transition probabilities between points in $C_i$ and $C_j$. Then, the affinity between these clusters is a weighted sum of the elements in $\mathfrak{P}^\ell$, where the weight of the element $\mathfrak{p}^\ell(x,y)$ is $q(x)$ (see Fig. 3.3(b)). Finally, the degree of each cluster is calculated by summing its affinities $\widehat{Q}_{ii} = \sum_{j=1}^{\widehat{n}} \widehat{K}_{ij}$ with all the clusters.

Notice that Algorithm 3.1 is similar to the pruning algorithm described in [1, Section 3.3]. Both algorithms get an input matrix of relations between data-points and a clustering function that assigns each point to its cluster. Then, for each pair of clusters, these algorithms consider a sub-matrix that contains the relations between the data-points in the two considered clusters. In order to achieve a scalar representation of the relation between the considered clusters,

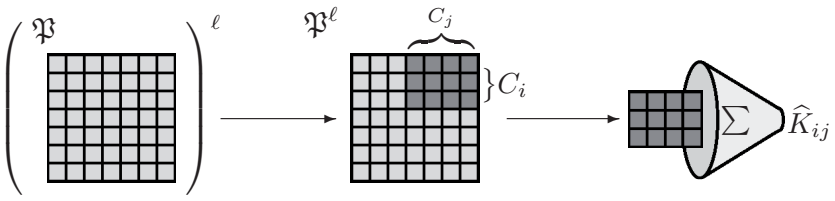
**Algorithm 3.1:** transition matrix pruning

**Input**: Dataset $X$ of $n$ data-points;
Clustering function $C : X \to \{C_1, C_2, \ldots, C_{\widehat{n}}\}$ of the data into $\widehat{n}$ clusters;
Transition probability matrix $P$ between data-points in $X$;
Parameter $\ell$ of the path length.
**Output**: A row-stochastic $\widehat{n} \times \widehat{n}$ matrix $\widehat{P}$ that represents transitions between clusters;
A diagonal matrix $\widehat{Q}$ that contains the degrees of the clusters on its diagonal.

```
foreach i, j = 1, ..., n̂ do
    𝔓 ⟵ |C_i ∪ C_j| × |C_i ∪ C_j| square matrix ;
    // Denote by 𝔭(x, y) the cell of 𝔓 in the row of x
    // and the column of y (x, y ∈ C_i ∪ C_j).
    // Denote by 𝔭^ℓ(x, y) the same cell in 𝔓^ℓ.
    foreach x, y ∈ C_i ∪ C_j do
        𝔭(x, y) ⟵ p(x, y) ;
    end
    K̂_ij ⟵ Σ_{x∈C_i, y∈C_j} q(x) 𝔭^ℓ(x, y) ;
end
foreach i = 1, ..., n̂ do
    Q̂_ii ⟵ Σ_{j=1}^{n̂} K̂_ij ;
end
P̂ ⟵ Q̂^{-1} K̂ ;
```




(a) Construction of the submatrix $\mathfrak{P}$ from the matrix $P$




(b) Computation of $\widehat{K}_{ij}$ as a weighted sum of cells in $\mathfrak{P}^\ell$

Figure 3.3: Construction of the pruned-kernel $\widehat{K}_{ij}$ between the clusters $C_i$ and $C_j$

both algorithms aggregate the elements of a suitable power of the considered sub-matrix.

However, these algorithms differ in the input matrix itself and in the aggregation function that is being used. Algorithm 3.1 gets a transition probability matrix as an input and uses a weighted sum for the aggregation, while the algorithm in [1, Section 3.3] gets an affinity matrix as an input and suggests three different aggregations of the sub-matrix elements.

Theorem 3.2 shows that the resulting Markov process in Algorithm 3.1 is in fact a transition probability matrix of the $\ell$-path localized diffusion process that was defined in Definition 3.3.

**Theorem 3.2.** *The output matrix $\widehat{P}$ from Algorithm 3.1 is a transition probability matrix of an $\ell$-path localized diffusion process.*

In order to prove Theorem 3.2, we need Lemmas 3.3 and 3.4 that relate the matrices used in Algorithm 3.1 to the original diffusion process.

**Lemma 3.3.** *Let $\mathfrak{P}$ be the sub-matrix of $P$ defined in a single iteration of Algorithm 3.1 for specific $i, j = 1, \ldots, \widehat{n}$. $\mathfrak{P}$ is related to the localized transition probabilities of $\mathcal{P}$ in the following ways:*

1. $\mathfrak{p}^\ell(x,y) = \Pr[x \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} y]$;
2. $\sum_{y\in C_j} \mathfrak{p}^\ell(x,y) = \Pr[x \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} C_j]$;
3. $\sum_{x\in C_i}\sum_{y\in C_j} \frac{q(x)}{\mathrm{vol}(C_i)}\mathfrak{p}^\ell(x,y) = \Pr[C_i \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} C_j]$.

The proof of Lemma 3.3 is given in Appendix A.

**Lemma 3.4.** *Let $\widehat{q}(C_i) \triangleq \widehat{Q}_{ii} \triangleq \sum_{j=1}^{\widehat{n}} \widehat{K}_{ij}$, $i = 1,\ldots,\widehat{n}$, be the degree (i.e., row sum) defined in Algorithm 3.1. Then, $\widehat{q}(C_i) = \mathrm{vol}(C_i)\cdot \mathrm{lpr}^\ell(C_i)$.*

*Proof.* According to Definition 3.2

$$\begin{aligned}\mathrm{lpr}^\ell(C_i) &= \Pr[p \in \mathfrak{L}(\mathcal{P}^\ell) |\, \mathrm{s}(\mathcal{P}) \in C_i \wedge \mathcal{P} \in \mathcal{P}^\ell] \\ &= \sum_{j=1}^{\widehat{n}} \Pr[C_i \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} C_j], \qquad i = 1,\ldots,\widehat{n}.\end{aligned}$$

Combining with property (3) of Lemma 3.3 yields

$$\mathrm{lpr}^\ell(C_i) = \sum_{j=1}^{\widehat{n}} \sum_{x\in C_i}\sum_{y\in C_j} \frac{q(x)}{\mathrm{vol}(C_i)}\mathfrak{p}^\ell(x,y) \qquad i = 1,\ldots,\widehat{n},$$

where $\mathfrak{P}$ depends on the choice of $i$ and $j$. By using the matrix $\widehat{K}$ from Algorithm 3.1 we get

$$\mathrm{lpr}^\ell(C_i) = \sum_{j=1}^{\widehat{n}} \frac{K_{ij}}{\mathrm{vol}(C_i)} = \frac{\widehat{q}(C_i)}{\mathrm{vol}(C_i)} \qquad i = 1,\ldots,\widehat{n},$$

and multiplying by $\mathrm{vol}(C_i)$ yields the desired result. □

Lemmas 3.3 and 3.4 relate the localized transition probabilities from the diffusion process to the original diffusion transition probabilities via the matrices constructed in Algorithm 3.1. These relations can now be used to prove Theorem 3.2.

*Proof of Theorem 3.2.* Consider two clusters $C_i$ and $C_j$, $i,j = 1,\ldots,\widehat{n}$. According to Algorithm 3.1, $\widehat{P}_{ij} \triangleq \frac{\widehat{K}_{ij}}{\widehat{Q}_{ii}}$ and $\widehat{K}_{ij} \triangleq \sum_{x\in C_i, y\in C_j} q(x)\mathfrak{p}^\ell(x,y)$. Using Lemmas 3.3 and 3.4 we obtain

$$\widehat{P}_{ij} = \frac{\mathrm{vol}(C_i)\cdot \Pr[C_i \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} C_j]}{\mathrm{vol}(C_i)\cdot \mathrm{lpr}^\ell(C_i)} = \frac{\Pr[C_i \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} C_j]}{\mathrm{lpr}^\ell(C_i)}.$$

By Definition 3.2 and Eq. (3.3),

$$\widehat{P}_{ij} = \frac{\Pr[\mathrm{t}(\mathcal{P}) \in C_j \wedge \mathcal{P} \in \mathfrak{L}(\mathcal{P}^\ell) |\, \mathrm{s}(\mathcal{P}) \in C_i \wedge \mathcal{P} \in \mathcal{P}^\ell]}{\Pr[p \in \mathfrak{L}(\mathcal{P}^\ell) |\, \mathrm{s}(\mathcal{P}) \in C_i \wedge \mathcal{P} \in \mathcal{P}^\ell]},$$

and by conditional probability considerations we get

$$\widehat{P}_{ij} = \Pr[\mathrm{t}(\mathscr{P}) \in C_j | p \in \mathfrak{L}(\mathcal{P}^\ell) \wedge \mathrm{s}(\mathscr{P}) \in C_i \wedge \mathscr{P} \in \mathcal{P}^\ell]. \tag{3.5}$$

The term $\mathscr{P} \in \mathcal{P}^\ell$ in the hypothesis of Eq. (3.5) is redundant by the localized $\ell$-path. Thus, by combining with Eq. (3.4) we get $\widehat{P}_{ij} = \Pr[C_i \xrightarrow{\widehat{\mathcal{P}}^1} C_j]$ and the theorem is proved. □

*3.2. Relation to LDF*

In the original diffusion, it is assured that the magnitude of the eigenvalues of $P$ is between zero and one. Another important property of $P$ is the existence of a symmetric conjugate $A$. Being a symmetric matrix, the eigenvalues of $A$ are all real and its left and right eigenvectors are identical. The matrix $A$ has the same eigenvalues as $P$ and its eigenvectors are related to those of $P$ by the same conjugation that relates $A$ to $P$. The additional information provided by the symmetric conjugate $A$ allows for a simple spectral analysis to be used to achieve dimensionality reduction as described in [2]. Theorem 3.5 shows that these properties also apply to the ergodic $\ell$-path localized diffusion process $\widehat{\mathcal{P}}$.

**Theorem 3.5.** *Let $\widehat{P}$ be the transition probability matrix of a localized $\ell$-path diffusion process, which resulted from Algorithm 3.1. Let $\widehat{Q}$ be the corresponding degree matrix. Then the conjugate matrix $\widehat{A} = \widehat{Q}^{1/2}\widehat{P}\widehat{Q}^{-1/2}$ is symmetric. Furthermore, $\widehat{A}$ is equivalent to the result from the weighted-sum LDF runner in [1, Section 3.3].*

*Proof.* Consider two clusters $C_i$ and $C_j$, $i, j = 1, \ldots, \widehat{n}$. Let $\mathfrak{P}$ be the matrix defined for them in the corresponding iteration of Algorithm 3.1. Let $\mathfrak{Q}$ be a diagonal $|C_i \cup C_j| \times |C_i \cup C_j|$ matrix where each cell on its diagonal corresponds to a data-point $x \in C_i \cup C_j$ and holds its degree $\mathfrak{q}(x) = q(x)$. As discussed in Section 2.1, the diffusion affinity matrix $A = Q^{1/2}PQ^{-1/2}$ is a symmetric conjugate of the diffusion operator $P$. Its cells are

$$a(x, y) = \sqrt{\frac{q(x)}{q(y)}} \cdot p(x, y) \qquad x, y \in X.$$

Let $\mathfrak{A} = \mathfrak{Q}^{1/2}\mathfrak{P}\mathfrak{Q}^{-1/2}$ be a $|C_i \cup C_j| \times |C_i \cup C_j|$ conjugate of $\mathfrak{P}$, then its cells are

$$\mathfrak{a}(x, y) = \sqrt{\frac{\mathfrak{q}(x)}{\mathfrak{q}(y)}} \cdot \mathfrak{p}(x, y) = \sqrt{\frac{q(x)}{q(y)}} \cdot p(x, y) = a(x, y) \qquad x, y \in C_i \cup C_j. \tag{3.6}$$

From the symmetry of $A$, we get $\mathfrak{a}(x, y) = \mathfrak{a}(y, x)$, $x, y \in C_i \cup C_j$ and $\mathfrak{A}$ is symmetric. The powers of a symmetric matrix are also symmetric, thus $\mathfrak{A}^\ell$, $\ell \geq 1$, which was given as a parameter to Algorithm 3.1, is also symmetric and since the terms $\mathfrak{Q}^{-1/2}\mathfrak{Q}^{1/2} = I$ are canceled, then

$$\mathfrak{A}^\ell = \underbrace{\mathfrak{Q}^{1/2}\mathfrak{P}\mathfrak{Q}^{-1/2} \cdot \mathfrak{Q}^{1/2}\mathfrak{P}\mathfrak{Q}^{-1/2} \cdots \mathfrak{Q}^{1/2}\mathfrak{P}\mathfrak{Q}^{-1/2}}_{\ell \text{ times}} = \mathfrak{Q}^{1/2}\mathfrak{P}^\ell\mathfrak{Q}^{-1/2}. \tag{3.7}$$

The symmetry is maintained by multiplying the symmetric matrix $\mathfrak{A}^\ell$ from left and right by the diagonal matrix $\mathfrak{Q}^{1/2}$. Thus the resulting matrix is

$$\mathfrak{Q}^{1/2}\mathfrak{A}^\ell\mathfrak{Q}^{1/2} = \mathfrak{Q}^{1/2}\mathfrak{Q}^{1/2}\mathfrak{P}^\ell\mathfrak{Q}^{-1/2}\mathfrak{Q}^{1/2} = \mathfrak{Q}\mathfrak{P}^\ell.$$

According to Algorithm 3.1 and since $\mathfrak{q}(x) = q(x)$ for $x \in C_i \cup C_j$, then

$$\widehat{K}_{ij} = \sum_{x\in C_i, y\in C_j} \mathfrak{q}(x)\mathfrak{p}^\ell(x,y).$$

According to the symmetry of $\mathfrak{Q}\mathfrak{P}^\ell$, we obtain

$$\widehat{K}_{ij} = \sum_{x\in C_i}\sum_{y\in C_j} \mathfrak{q}(y)\mathfrak{p}^\ell(y,x).$$

The same matrices $\mathfrak{P}$ and $\mathfrak{Q}$ are also used in the iteration that computes $\widehat{K}_{ji}$ in Algorithm 3.1, thus

$$\widehat{K}_{ji} = \sum_{y\in C_j}\sum_{x\in C_i} \mathfrak{q}(y)\mathfrak{p}^\ell(y,x) = \widehat{K}_{ij}$$

holds and $\widehat{K}$ is symmetric. Since $\widehat{P} \triangleq \widehat{Q}^{-1}\widehat{K}$ by Algorithm 3.1, then

$$\widehat{A} = \widehat{Q}^{1/2}\widehat{P}\widehat{Q}^{-1/2} = \widehat{Q}^{-1/2}\widehat{K}\widehat{Q}^{-1/2}. \tag{3.8}$$

Multiplication by the diagonal matrix $\widehat{Q}^{-1/2}$ from both sides maintains the symmetry of $\widehat{K}$, thus $\widehat{A}$ is also symmetric.

Combining Eq. (3.8) and the definition of $\widehat{K}$ in Algorithm 3.1, yields

$$\widehat{A}_{ij} = \frac{\widehat{K}_{ij}}{\sqrt{\widehat{Q}_{ii}}\sqrt{\widehat{Q}_{ii}}} = \sum_{x\in C_i}\sum_{y\in C_j} \frac{q(x)\mathfrak{p}^\ell(x,y)}{\sqrt{\widehat{q}(C(x))\widehat{q}(C(y))}},$$

Together with Eq. (3.7), we receive

$$\widehat{A}_{ij} = \sum_{x\in C_i}\sum_{y\in C_j} \frac{\sqrt{q(x)}\sqrt{q(y)}\mathfrak{a}^\ell(x,y)}{\sqrt{\widehat{q}(C(x))\widehat{q}(C(y))}}.$$

Let

$$w_{xy} \triangleq \sqrt{\frac{q(x)q(y)}{\widehat{q}(C(x))\widehat{q}(C(y))}} \qquad x,y \in X,$$

then the following weighted sum is obtained:

$$\widehat{A}_{ij} = \sum_{x\in C_i}\sum_{y\in C_j} w_{xy}\mathfrak{a}^\ell(x,y).$$

Finally, according to Eq. (3.6), the matrix $\mathfrak{A}$ is a sub-matrix of $A$, which contains cells in rows and columns that correspond to data-points in $C_i \cup C_j$. This is

exactly the sub-matrix used in the corresponding iteration (for $C_i$ and $C_j$) in the LDF algorithm [1, Section 3.3], and thus $\widehat{A}_{ij}$ contains a weighted sum of the cells that are combined by the LDF runners [1, Section 3.3]. Therefore, the matrix $\widehat{A}$, which is a symmetric conjugate of $\widehat{P}$, can be directly obtained by a weighted-sum LDF runner with the defined weights $w_{xy}$, $x, y \in X$. □

From Theorem 3.5, the symmetric matrix $\widehat{A}$ can be used for spectral analysis of the localized diffusion since it has the same spectrum as $\widehat{P}$ and its eigenvectors are related to the eigenvectors of $\widehat{P}$ by the same conjugation that relates $\widehat{A}$ to $\widehat{P}$. In fact, $\widehat{A}$ is a result of the LDF runner, which is used to prune a level in the LDF hierarchy to the next (higher) level, and thus, it can be constructed directly from $A$ without using $P$, $\widehat{P}$ and the conjugations (see Fig. 3.4).

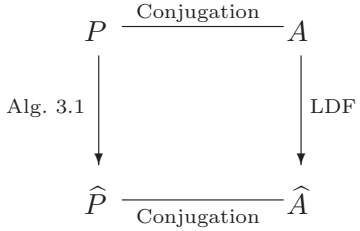



Figure 3.4: The relation between Algorithm 3.1 and the LDF pruning algorithm

If we denote the eigenvalues of $\widehat{A}$ by $1 = \widehat{\lambda}_0 \geq |\widehat{\lambda}_1| \geq |\widehat{\lambda}_2| \geq \ldots$ and the corresponding eigenvectors by $\widehat{\phi}_0, \widehat{\phi}_1, \widehat{\phi}_2, \ldots$, we can define a coarse-grained DM, which we call the $\ell$-path localized diffusion map (LDM). This map embeds each cluster $C_i$, $i = 1, \ldots, \widehat{n}$, to a point

$$\widehat{\Phi}(C_i) = (|\widehat{\lambda}_0|\widehat{\phi}_0(C_i), |\widehat{\lambda}_1|\widehat{\phi}_1(C_i), |\widehat{\lambda}_2|\widehat{\phi}_2(C_i), \ldots)^T.$$

According to the above discussion, this embedding has the same properties as the DM embedding, which was presented in [2].

By combining the original DM, which embeds data-points, and the presented LDM, which embeds clusters, we obtain a two-level embedding (i.e., a data-point level and a cluster level). Moreover, the LDM is defined by spectral analysis of the affinity constructed by LDF. Therefore, it can be defined for each level of the LDF hierarchy. Thus, we get a multilevel embedding of the data where, in each level, the corresponding DF are embedded. Furthermore, each coarse-graining iteration (between LDF levels) prunes longer paths to single transitions, and thus, a wider time scale of the diffusion is considered. Therefore, the multilevel embedding, which results from the LDM and from the LDF hierarchy, provides a multiscale coarse-grained DM.

## 4. Conclusion

The presented $\ell$-path localized diffusion process introduces a coarse-grained version of the diffusion process that is used in DM for high-dimensional data

analysis and dimensionality reduction. This coarse-grained process preserves the locality of the data by pruning previously detected clusters while considering only localized paths between them. A simple pruning algorithm can be used to perform the described coarse-graining while maintaining the essential algebraic and spectral properties of the DM process as was introduced in [2]. Furthermore, this pruning is equivalent (via conjugation) to the one performed by the LDF algorithm when it computes the LDF hierarchy. By combining the results of this paper with the ones in [1], the LDF hierarchy is shown to provide the foundations for a multi-scale coarse-grained DM-based embedding of data-points and clusters/folders to a low-dimensional space.

## Appendix A. Proof of Lemma 3.3

*Proof.* Since $\mathcal{P}$ is a Markovian random walk process with a transition probability matrix $P$, then the probability of a path $\mathcal{P} \in \mathcal{P}^\ell$ is $\prod_{\xi=1}^{\ell} p(\mathcal{P}_{\xi-1}, \mathcal{P}_\xi)$. The probability $\Pr[x \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} y]$, which is defined in Eq. (3.1), considers only paths with $\mathcal{P}_0 = \mathrm{s}(\mathcal{P}) = x$, $\mathcal{P}_\ell = \mathrm{t}(\mathcal{P}) = y$ and $\mathcal{P}_\xi \in C_i \cup C_j, \xi = 1, \ldots, \ell - 1$, thus

$$\Pr[x \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} y] = \sum_{\mathcal{P}_1,\ldots,\mathcal{P}_{\ell-1} \in C_i \cup C_j} [p(x, \mathcal{P}_1) \cdot \prod_{\xi=2}^{\ell-1} p(\mathcal{P}_{\xi-1}, \mathcal{P}_\xi) \cdot p(\mathcal{P}_{\ell-1}, y)] \qquad x \in C_i, y \in C_j.$$

By Algorithm 3.1, $\mathfrak{p}(\mathcal{P}_{\xi-1}, \mathcal{P}_\xi) = p(\mathcal{P}_{\xi-1}, \mathcal{P}_\xi), \xi = 1, \ldots, \ell$ when $\mathcal{P}_1, \ldots, \mathcal{P}_{\ell-1} \in C_i \cup C_j$, $\mathcal{P}_0 = x \in C_i$ and $\mathcal{P}_\ell = y \in C_j$. Therefore,

$$\begin{aligned} \Pr[x \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} y] &= \sum_{\mathcal{P}_1,\ldots,\mathcal{P}_{\ell-1} \in C_i \cup C_j} [\mathfrak{p}(x, \mathcal{P}_1) \cdot \prod_{\xi=2}^{\ell-1} \mathfrak{p}(\mathcal{P}_{\xi-1}, \mathcal{P}_\xi) \cdot \mathfrak{p}(\mathcal{P}_{\ell-1}, y)] \\ &= \mathfrak{p}^\ell(x, y) \qquad x \in C_i, y \in C_j, \end{aligned}$$

and the first part of the lemma is proved.

The probability $\Pr[x \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} C_j]$, which was defined in Eq. (3.2), combines all the probabilities in Eq. (3.1) with $y \in C_j$. Different paths are considered independent events, thus

$$\Pr[x \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} C_j] = \sum_{y \in C_j} \Pr[x \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} y] = \sum_{y \in C_j} \mathfrak{p}^\ell(x, y) \qquad x \in C_i,$$

and the second part of the lemma is proved. The probability $\Pr[C_i \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} C_j]$, which was defined in Eq. (3.3), combines all the probabilities in Eq. (3.2) with $x \in C_i$. Since $x$ is part of the condition in these probabilities, we get

$$\begin{aligned} \Pr[C_i \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} C_j] &= \sum_{x \in C_i} \Pr[\mathrm{s}(\mathcal{P}) = x | \, \mathrm{s}(\mathcal{P}) \in C_i] \cdot \Pr[x \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} C_j] \\ &= \sum_{x \in C_i} \sum_{y \in C_j} \Pr[\mathrm{s}(\mathcal{P}) = x | \, \mathrm{s}(\mathcal{P}) \in C_i] \cdot \mathfrak{p}^\ell(x, y). \end{aligned}$$

Using Eq. (2.3) we get

$$\Pr[C_i \xrightarrow{\mathfrak{L}(\mathcal{P}^\ell)} C_j] = \sum_{x\in C_i}\sum_{y\in C_j} \frac{q(x)}{\mathrm{vol}(C_i)}\mathfrak{p}^\ell(x,y),$$

and the final part of the lemma is proved. □

**Acknowledgments**

This research was partially supported by the Israel Science Foundation (Grant No. 1041/10). The first author was also supported by the Eshkol Fellowship from the Israeli Ministry of Science & Technology.

**References**

[1] G. David, A. Averbuch, Hierarchical data organization, clustering and denoising via localized diffusion folders, Applied and Computational Harmonic Analysis. In press, to appear.

[2] R. Coifman, S. Lafon, Diffusion maps, Applied and Computational Harmonic Analysis 21 (1) (2006) 5–30.

[3] R. Coifman, S. Lafon, A. Lee, M. Maggioni, B. Nadler, F. Warner, S. Zucker, Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps, Proceedings of the National Academy of Sciences of the United States of America 102 (21) (2005) 7426–7431.

[4] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: An efficient data clustering method for very large databases, in: SIGMOD '96: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, 1996, pp. 103–114.

[5] L. Kaufman, P. Rousseeuw, Finding groups in Data - An introduction to Cluster Analysis, Wiley. New York: John Wiely & Sons, Inc., 1990.

[6] S. Lafon, A. Lee, Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization, IEEE Transactions on Pattern Analysis and Machine Intelligence (2006) 1393–1403.

[7] G. Wolf, M. Salhov, A. Averbuch, Patch-to-tensor embedding, Applied and Computational Harmonic Analysis. In press, to appear.

[8] G. Wolf, A. Averbuch, Linear-projection diffusion on smooth Euclidean submanifolds, Applied and Computational Harmonic Analysis.

[9] A. Singer, H. Wu, Orientability and diffusion maps, Applied and Computational Harmonic Analysis. In press, to appear.

[10] A. Singer, H. Wu, Vector diffusion maps and the connection Laplacian, Arxiv preprint arXiv:1102.0075.

[11] F. Chung, Spectral Graph Theory, Vol. 92, CBMS-AMS, 1997.

# PII

## HIERARCHICAL DATA ORGANIZATION, CLUSTERING AND DENOISING VIA COARSE-GRAINED LOCALIZED DIFFUSION

by

Guy Wolf, Aviv Rotbart, Gil David, Amir Averbuch 2012

CJR conference, Yale

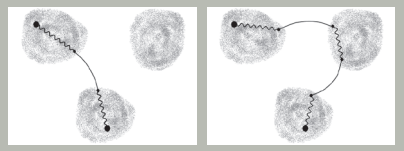
# Hierarchical Data Organization, Clustering and Denoising via Coarse-Grained Localized Diffusion

Guy Wolf  Aviv Rotbart  Gil David  Amir Averbuch

## Introduction

- Massive high-dimensional datasets are very popular nowadays, but they pose two main challenges for data-analysis methods:
  1. The "curse of dimensionality" due to the redundancy of measured parameters.
  2. Most data-points are redundant due to close relations (or even duplicities) between them.
- We propose an approach for dealing with both effects by coarse-graining the popular *Diffusion Maps* (DM) framework from the data-point level to the cluster level.
  - The analyzed dataset becomes smaller by considering coarse-grained clusters instead of individual data-points.
  - Essential properties of the diffusion process in DM are preserved by coarse-graining localized paths between clusters. As a result, the dimensionality-reduction properties of DM can be used to solve the "curse of dimensionality" with this Localized-Diffusion construction.
  - Hierarchical clustering is achieved by iterating the coarse-graining several time, thus getting a Localized Diffusion Folder (LDF) hierarchy over the dataset.

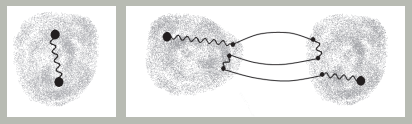
## Localized vs. Non-localized Paths



## Definition (localized $\ell$-path)

A localized $\ell$-path in a diffusion process $\mathcal{P}$ is the path $\mathcal{P} = \mathcal{P}_0 \to \mathcal{P}_1 \to \ldots \to \mathcal{P}_\ell$ of length $\ell$ that traverses solely through data-points in its source and destination clusters, i.e., $\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_\ell \in C(\mathsf{s}(\mathcal{P})) \cup C(\mathsf{t}(\mathcal{P}))$.

## Localized Paths



## Transition Probabilities

The transition probability in $\ell$ time steps between the points $x, y$ in the original diffusion process can be described as

$$\Pr[x \xrightarrow{P^\ell} y] \triangleq \Pr[\mathsf{t}(\mathcal{P}) = y \,|\, \mathsf{s}(\mathcal{P}) = x \wedge \mathcal{P} \in \mathcal{P}^\ell] \qquad x, y \in X.$$

The localized transition probability between these two points is

$$\Pr[x \xrightarrow{\mathfrak{L}(P^\ell)} y] \triangleq \Pr[\mathsf{t}(\mathcal{P}) = y \wedge \mathcal{P} \in \mathfrak{L}(\mathcal{P}^\ell) \,|\, \mathsf{s}(\mathcal{P}) = x \wedge \mathcal{P} \in \mathcal{P}^\ell].$$

Similarly, the localized transition probability from $x \in C_i$ to the cluster $C_j$ and from the cluster $C_i$ to $C_j$ are defined as

$$\Pr[x \xrightarrow{\mathfrak{L}(P^\ell)} C_j] \triangleq \Pr[\mathsf{t}(\mathcal{P}) \in C_j \wedge \mathcal{P} \in \mathfrak{L}(\mathcal{P}^\ell) \,|\, \mathsf{s}(\mathcal{P}) = x \wedge \mathcal{P} \in \mathcal{P}^\ell],$$

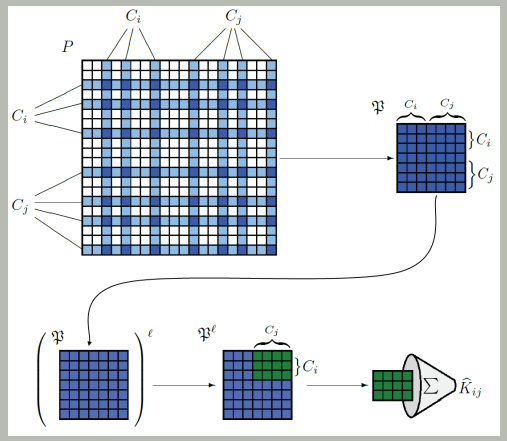
and

$$\Pr[C_i \xrightarrow{\mathfrak{L}(P^\ell)} C_j] \triangleq \Pr[\mathsf{t}(\mathcal{P}) \in C_j \wedge \mathcal{P} \in \mathfrak{L}(\mathcal{P}^\ell) \,|\, \mathsf{s}(\mathcal{P}) \in C_i \wedge \mathcal{P} \in \mathcal{P}^\ell].$$

## Localized Diffusion Process

The diffusion process described in DM considers transition probabilities at different scales, between data points in a given data set. At scale $\ell > 0$, all paths of length $\ell$ are used to calculate the transition probabilities. We suggest to generalize this approach into transition probabilities between clusters on the data. In order to reduce noisy connections between clusters, we only consider localized paths, namely paths that traverse through their source and target cluster solely.

## Pruning Algorithm




## Pruning Algorithm: From the original diffusion to the localized one

- The illustration describes a coarse-graining of the original diffusion process.
- Calculating cluster affinities from data-point affinities:
  1. Affinity between the clusters $C_i$ and $C_j$ is based on the sub-matrix containing points from both clusters.
  2. Localized Diffusion Process is run forward on this matrix (i.e., the matrix is raised to some power).
  3. Aggregation method is applied on the matrix, generating a single number - the affinity between the clusters $C_i$ and $C_j$.
- The result of the algorithm is a localized diffusion process.
  - This is a Markovian process that defines a transition probability matrix between clusters.
  - Applying Localized Diffusion Process to the data yields both denoising effect (by removing noisy and indirect connections between clusters) and Hierarchical view of the data (by clustering each level of the kernel pruning process).

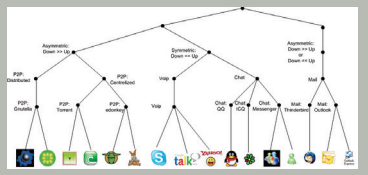
## Definition ($\ell$-path localized diffusion process)

Let $\mathcal{P}$ be a diffusion (random walk) process defined on the data-points of the dataset $X$. An $\ell$-path localized diffusion process $\widehat{\mathcal{P}}$ is a random walk on the clusters $C_1, C_2, \ldots, C_{\widehat{n}}$ where a transition from $C_i$ to $C_j$, $i, j = 1, \ldots, \widehat{n}$, represents all the localized $\ell$-paths in the diffusion process $\mathcal{P}$ from data-points in $C_i$ to data-points in $C_j$. The probability of such a transition, according to $\widehat{\mathcal{P}}$, is the probability to reach the destination cluster $C_j$ when starting at the source cluster $C_i$ and traveling solely via localized $\ell$-paths.

## Proposition

*The localized diffusion process $\widehat{\mathcal{P}}$ is an ergodic Markov process (i.e., the random walk it defines is irreducible and aperiodic).*

## Experimental Results: Hierarchical clustering of communication-oriented applications




## Hierarchical clustering explained

- Illustration of hierarchical clustering of applications
  - The clustering of software applications is based on their communication properties.
  - Bottom level of the tree shows a fine-graining of the data, and distinguishes between similar applications (e.g., ICQ and Messenger).
  - Higher levels of the tree show further coarse graining of the data and group similar applications, applications types, communication methods and communication behaviors into one cluster.
- Localized Diffusion Folders (LDF) Algorithm
  - Performs clustering of the data by selecting the centers of the clusters at random (shake & bake).
  - Construct a localized affinity, which defines a "purified geometry" of the data by "shaking" multiple Diffusion Folders to get rid of the noise and "baking" a cleaner affinity that eliminates rare connections.

## Experimental Results: Denoising images using Localized Diffusion Folders

## Denoising images explained

- Image was distorted with 30% salt & pepper noise.
- Using LDF to restore the image
  1. Each pixel in the image is represented by a window of $5 \times 5$ neighbors around it, creating a 25-dimensional vector (mega-pixel).
  2. A sliding window of size $9 \times 9$ mega-pixels is considered.
  3. For each window, the LDF algorithm was applied and the value of the center pixel in the window was set to the average value of the pixels of the largest folder (cluster) in the process.
- The result: denoised image, LDF shows better results than the diffusion regularization method.

## References


- Guy Wolf, Aviv Rotbart, Gil David, Amir Averbuch, Coarse-grained localized diffusion, Applied and Computational Harmonic Analysis, Available online 1 March 2012, ISSN 1063-5203, 10.1016/j.acha.2012.02.004.
- Gil David, Amir Averbuch, Hierarchical data organization, clustering and denoising via localized diffusion folders, Applied and Computational Harmonic Analysis, Volume 33, Issue 1, July 2012, Pages 1-23.

# PIII

## RANDOMIZED LU DECOMPOSITION: AN ALGORITHM FOR DICTIONARIES CONSTRUCTION

by

Aviv Rotbart, Gil Shabat, Yaniv Shmueli, Amir Averbuch 2014

Submitted to IEEE transaction on Information Forensics and Security

# Randomized LU decomposition: An Algorithm for Dictionaries Construction

Aviv Rotbart[1,2] Gil Shabat[3] Yaniv Shmueli[1] Amir Averbuch[1]

[1]School of Computer Science, Tel Aviv University, Israel
[2]Department of Mathematical Information Technology, University of Jyväskylä, Finland
[3]School of Electrical Engineering, Tel Aviv University, Israel

**Abstract**

In recent years, distinctive-dictionary construction has gained importance due to his usefulness in data processing. Usually, one or more dictionaries are constructed from a training data and then they are used to classify signals that did not participate in the training process. A new dictionary construction algorithm is introduced. It is based on a low-rank matrix factorization being achieved by the application of the randomized LU decomposition to a training data. This method is fast, scalable, parallelizable, consumes low memory, outperforms SVD in these categories and works also extremely well on large sparse matrices. In contrast to existing methods, the randomized LU decomposition constructs an under-complete dictionary, which simplifies both the construction and the classification processes of newly arrived signals. The dictionary construction is generic and general that fits different applications. We demonstrate the capabilities of this algorithm for file type identification, which is a fundamental task in digital security arena, performed nowadays for example by sandboxing mechanism, deep packet inspection, firewalls and anti-virus systems. We propose a content-based method that detects file types that neither depend on file extension nor on metadata. Such approach is harder to deceive and we show that only a few file fragments from a whole file are needed for a successful classification. Based on the constructed dictionaries, we show that the proposed method can effectively identify execution code fragments in PDF files.

**Keywords.** Dictionary construction, classification, LU decomposition, randomized LU decomposition, content-based file detection, computer security.

## I. Introduction

Recent years have shown a growing interest in dictionary learning. Dictionaries were found to be useful for applications such as signal reconstruction, denoising, image impainting, compression, sparse representation, classification and more. Given a data matrix $A$, a dictionary learning algorithm produces two matrices $D$ and $X$ such that $||A-DX||$ is small where $D$ is called dictionary and $X$ is a coefficients matrix also called representation matrix. Sparsity of $X$, means that each signal from $A$ is described with only a few signals (also called atoms) from the dictionary $D$. It is a major property being pursued by many dictionary learning algorithms. The algorithms, which learn dictionaries for sparse representations, optimize a goal function $\min_{D,X} ||A-DX||+\lambda||X||_0$, which considers both the accuracy and the sparsity of the solution, by optimizing alternately these two properties ($\lambda$ is a regularization term). This construction is computationally expensive and does not scale well to big data. It becomes even worse when dictionary learning is used for classification since another distinctive term in addition to the two aforementioned is being introduced in the objective function. This term provides the learned dictionary a discriminative ability. This can be seen for example in the optimization problem $\min_{D,X,W} ||A-DX|| + \lambda||X||_0 + \xi||H-WX||$ where $W$ is a classifier and $H$ is a vector of labels. $||H-WX||$ is the penalty term for achieving a wrong classification. In order to achieve the described properties, dictionaries are usually over-complete, namely, they contain more atoms than the signal dimension. As a consequence, dictionaries are redundant such that there are linear dependencies between atoms. Therefore, a given signal can be represented in more than one way

Corresponding author: aviv.rotbart@cs.tau.ac.il

using dictionary atoms. This enables us on one hand to get sparse representations, but on the other hand it complicates the representation process because it is NP-hard to find the sparsest representation for a signal by an over-complete dictionary [13].

In this work, we provide a generic way to construct an under-complete dictionary. Its capabilities will be demonstrated for signal classification task. Since we do not look for sparse signal representation, we remove the alternating optimization process from the construction of over-complete dictionaries. Our dictionary construction is based on matrix factorization. We use the randomized LU matrix factorization algorithm [16] for a dictionary construction. This algorithm, which is applied to a given data matrix $A \in \mathbb{R}^{m \times n}$ of $m$ features and $n$ data-points, decomposes $A$ into two matrices $L$ and $U$, where $L$ is the dictionary and $U$ is the coefficient matrix. The size of $L$ is determined by the decaying spectrum of the singular values of the matrix $A$, and bounded by $\min\{n, m\}$. Both $L$ and $U$ are linearly independent. The proposed dictionary construction has couple of advantages: it is fast, scalable, parallelizable and thus can run on GPU and multicore-based systems, consumes low memory, outperforms SVD in these categories and works extremely well on large sparse matrices. Under this construction, the classification of a newly arrived signal is done by a fast projection method that represents this signal by the columns of the matrix $L$. The computational cost of this method is linear in the input size, while in the under-complete case finding the optimal solution is NP-hard [13]. Approximation algorithms for sparse signal reconstruction, like Orthogonal Matching Pursuit [17] or Basis Pursuit [6], have no guarantees for general dictionaries.

In order to evaluate the performance of the dictionaries, which are constructed by the application of the randomized LU algorithm to a training set, we use them to classify file types. The experiments were conducted on a dataset that contains files of various types. The goal is to classify each file or portion of a file to the class describing its type. To the best of our knowledge, this work is the first to use dictionary learning method for file type classification. This work considers three different scenarios that represent real security tasks: examining the full content of the tested files, classifying a file type using a small number of fragments from the file and detecting malicious code hidden inside innocent looking files. While the first two scenarios were examined by other works, none of the papers described in this work dealt with the latter scenario. It is difficult to compare our results to other algorithms since the used datasets are not publicly available. For similar testing conditions, we improve the state-of-the-art results. The datasets we used will be made publicly available.

The paper has the following structure: Section II reviews related work on dictionary construction and on file content recognition algorithms. Section III presents the dictionary construction algorithm. Section IV shows how to utilize it to develop our classification algorithms for file content detection. Section V addresses the problem of computing the correct dictionaries sizes needed by the classifiers. Experimental results are presented in Section VI and compared with other content classification methods.

## II. Related Work

Dictionary-based classification models have been the focus of much recent research leading to results in face recognition [9], [15], [19]–[22], digit recognition [21], object categorization [9], [15] and more. Many of these works [9], [15], [22] utilize the K-SVD [1] for their training, or in other words, for their dictionary learning step. Others define different objective functions such as the Fisher Discriminative Dictionary Learning [21]. Majority of these methods use an alternating optimization process in order to construct their dictionary. This optimization procedure seeks a dictionary which is re-constructive, enables sparse representation and sometimes also has a discriminative property. In some works (see for example [9], [22]) the dictionary learning algorithm requires meta parameters to regulate these properties of the learned dictionary. Finding the optimal values for these parameters is a challenging task that adds complexity to the proposed solutions. A dictionary construction, which uses a multivariate optimization process, is computationally expensive task (as described in [15], for example). The proposed approach in this paper suggests to avoid these complexities by using the randomized LU Algorithm [16]. The dictionary it creates is under-complete where the number of atoms is smaller than the signal dimension.

The outcome is that the dictionary construction is fast that does not compromise its abilities to achieve high classification accuracy. We improve upon the state-of-the-art results in file type classification [4] as demonstrated by the experimental results.

The testing phase in many dictionary learning schemes is simple. Usually, linear classifier is used to assign test signals to one of the learned classes [9], [22]. However, classifier learning combined with dictionary learning adds additional overhead to the process [9], [22]. The proposed method in this paper does not require to allocate special attention to a classifier learning. We utilize the output from the randomized LU algorithm to create a projection matrix. This matrix is used to measure the distance between a test signal and the dictionary. The signal is then classified as belonging to the class that approximates it best. The classification process is fast and simple. The results described in Section VI show high accuracy in the content-based file type classification task.

We used this classification task to test the randomized LU dictionary construction and to measure its discriminative power. This task is useful in computer security applications like anti-virus systems and firewalls that need to detect files transmitted through network and response quickly to threats. Previous works in this field use mainly deep packet inspection (DPI) and byte distribution frequency features (1-gram statistics) in order to analyze a file [2]–[5], [7], [10]–[12], [18]. In some works, other features were tested like consecutive byte differences [4], [5] and statistical properties of the content [5]. The randomized LU decomposition [16] construction is capable of dealing with a large number of features. This enables us to test our method on high dimensional feature sets like double-byte frequency distributions (2-grams statistics) where each measurement has 65536 Markov-walk based features. We refer the reader to [4] and references within for an exhaustive comparison of the existing methods for content-based file type classification.

Throughout this work, when $A$ is a matrix, the norm $\|A\|$ indicates the spectral norm (the largest singular value of $A$) and when $A$ is a vector it indicates the standard $l_2$ norm (Euclidean norm).

## III. Randomized LU

In this section, we present the randomized LU decomposition algorithm for computing the rank $k$ LU approximation of a full matrix (Algorithm III.1). The main building blocks of the algorithm are random projections and Rank Revealing LU (RRLU) [14] to obtain a stable low-rank approximation for an input matrix $A$.

The RRLU algorithm, used in the randomized LU algorithm, reveals the connection between LU decomposition of a matrix and its singular values. This property is very important since it connects between the size of the decomposition to the actual numerical rank of the data. Similar algorithms exist for rank revealing QR decompositions (see, for example [8]).

**Theorem III.1** ( [14]). *Let $A$ be an $m \times n$ matrix ($m \geq n$). Given an integer $1 \leq k < n$, then the following factorization*

$$PAQ = \begin{pmatrix} L_{11} & 0 \\ L_{21} & I_{n-k} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}, \tag{III.1}$$

*holds where $L_{11}$ is a lower triangular with ones on the diagonal, $U_{11}$ is an upper triangular, $P$ and $Q$ are orthogonal permutation matrices. Let $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$ be the singular values of $A$, then:*

$$\sigma_k \geq \sigma_{min}(L_{11}U_{11}) \geq \frac{\sigma_k}{k(n-k)+1},$$

*and*

$$\sigma_{k+1} \leq \|U_{22}\| \leq (k(n-k)+1)\sigma_{k+1}.$$

Based on Theorem III.1, we have the following definition:

**Definition III.1** (RRLU Rank $k$ Approximation denoted RRLU$_k$)**.** *Given a RRLU decomposition (Theorem III.1) of a matrix $A$ with an integer $k$ (as in Eq. III.1) such that $PAQ = LU$, then the RRLU rank $k$ approximation is defined by taking $k$ columns from $L$ and $k$ rows from $U$ such that*

$$\textit{RRLU}_k(PAQ) = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \begin{pmatrix} U_{11} U_{12} \end{pmatrix},$$

*where $L_{11}, L_{21}, U_{11}, U_{12}, P$ and $Q$ are defined in Theorem III.1.*

**Lemma III.2** ( [16] RRLU Approximation Error)**.** *The error of the RRLU$_k$ approximation of $A$ is*

$$\|PAQ - \textit{RRLU}_k(PAQ)\| \leq (k(n-k)+1)\sigma_{k+1}.$$

Algorithm III.1 describes the flow of the RLU decomposition algorithm.

**Algorithm III.1:** Randomized LU Decomposition

**Input:** $A$ matrix of size $m \times n$ to decompose; $k$ rank of $A$; $l$ number of columns to use (for example, $l = k + 5$).

**Output:** Matrices $P, Q, L, U$ such that $\|PAQ - LU\| \leq \mathcal{O}(\sigma_{k+1}(A))$ where $P$ and $Q$ are orthogonal permutation matrices, $L$ and $U$ are the lower and upper triangular matrices, respectively, and $\sigma_{k+1}(A)$ is the $(k+1)$th singular value of $A$.

1: Create a matrix $G$ of size $n \times l$ whose entries are i.i.d. Gaussian random variables with zero mean and unit standard deviation.
2: $Y \leftarrow AG$.
3: Apply RRLU decomposition (See [14]) to $Y$ such that $PYQ_y = L_y U_y$.
4: Truncate $L_y$ and $U_y$ by choosing the first $k$ columns and $k$ rows, respectively: $L_y \leftarrow L_y(:, 1:k)$ and $U_y \leftarrow U_y(1:k, :)$.
5: $B \leftarrow L_y^{\dagger} PA$. ($L_y^{\dagger}$ is the pseudo inverse of $L_y$).
6: Apply LU decomposition to $B$ with column pivoting $BQ = L_b U_b$.
7: $L \leftarrow L_y L_b$.
8: $U \leftarrow U_b$.

**Remark III.3.** *In most cases, it is sufficient to compute the regular LU decomposition in Step 3 instead of computing the RRLU decomposition.*

The running time complexity of Algorithm III.1 is $\mathcal{O}(mn(l+k) + l^2 m + k^3 + k^2 n)$ (see Section 4.1 and [16] for a detailed analysis). It is shown in Section 4.2 in [16] that the error bound of Algorithm III.1 is given by the following theorem:

**Theorem III.4** ( [16])**.** *Given a matrix $A$ of size $m \times n$. Then, its randomized LU decomposition produced by Algorithm III.1 with integers $k$ and $l$ ($l \geq k$) satisfies*

$$\|LU - PAQ\| \leq \left(2\sqrt{2nl\beta^2\gamma^2 + 1} + 2\sqrt{2nl}\beta\gamma\,(k(n-k)+1)\right)\sigma_{k+1}(A),$$

*with probability not less than*

$$1 - \frac{1}{\sqrt{2\pi(l-k+1)}}\left(\frac{e}{(l-k+1)\beta}\right)^{l-k+1} - \frac{1}{4(\gamma^2-1)\sqrt{\pi n \gamma^2}}\left(\frac{2\gamma^2}{e^{\gamma^2-1}}\right)^{n},$$

*for all* $\beta > 0$ *and* $\gamma > 1$.

## IV. Randomized LU Based Classification Algorithm

This section describes the application of the randomized LU Algorithm III.1 to a classification task. The training phase includes dictionary construction for each learned class from a given dataset. The classification phase assigns a newly arrived signal to one of the classes based on its similarity to the learned dictionaries. Let $X \in \mathbb{R}^{m\times n}$ be the matrix whose $n$ columns are the training signals (samples). Each column is defined by $m$ features. Based on Section III, we apply the randomized LU decomposition (Algorithm III.1) to $X$, yielding $PXQ \approx LU$. The outputs $P$ and $Q$ are orthogonal permutation matrices. Theorem IV.1 shows that $P^T L$ forms (up to a certain accuracy) a basis to $A$. This is the key property of the classification algorithm.

**Theorem IV.1.** *Given a matrix $A$. Its randomized LU decomposition is $PAQ \approx LU$. Then, the error of representing $A$ by $P^T L$ satisfies:*

$$\|(P^T L)(P^T L)^\dagger A - A\| \leq \\ \left(2\sqrt{2nl\beta^2\gamma^2+1} + 2\sqrt{2nl}\beta\gamma\left(k(n-k)+1\right)\right)\sigma_{k+1}(A)$$

*with the same probability as in Theorem III.4.*

*Proof.* By combining Theorem III.4 with the fact that $BQ = L_b U_b = L_y^\dagger PAQ$ we get

$$\|LU - PAQ\| = \|L_y L_b U_b - PAQ\| = \|L_y L_y^\dagger PAQ - PAQ\|.$$

Then, by using the fact that $L_b$ is square and invertible we get

$$\begin{aligned}\|L_y L_y^\dagger PAQ - PAQ\| &= \|L_y L_b L_b^{-1} L_y^\dagger PAQ - PAQ\| \\ &= \|LL^\dagger PAQ - PAQ\|.\end{aligned}$$

By using the fact that the spectral norm is invariant to orthogonal projections, we get

$$\begin{aligned}&\|LL^\dagger PAQ - PAQ\| = \|LL^\dagger PA - PA\| \\ &= \|P^T LL^\dagger PA - A\| = \|(P^T L)(P^T L)^\dagger A - A\| \\ &\leq \left(2\sqrt{2nl\beta^2\gamma^2+1} + 2\sqrt{2nl}\beta\gamma\left(k(n-k)+1\right)\right)\sigma_{k+1}(A),\end{aligned}$$

with the same probability as in Theorm III.4. □

Assume that our dataset is composed of the sets $X_1, X_2, \ldots, X_l$. We denote by $D_i = P_i^T L_i$ the dictionary learned from the set $X_i$ by Algorithm III.1. $U_i Q_i^T$ is the corresponding coefficient matrix. It is used to reconstruct signals from $X_i$ as a linear combination of atoms from $D_i$. The training phase of the algorithm is done by the application of Algorithm III.1 to different training datasets that correspond to different classes. For each class, a different dictionary is learned. The size of $D_i$, namely its number of atoms, is determined by the parameter $k_i$ that is related to the decaying spectrum of the matrix $X_i$. The dictionaries do not have to be of equal sizes. A discussion about the dictionary sizes appears later in this section and in Section V. The third parameter, which Algorithm III.1 needs, is the number of projections $l$ on the random matrix columns. $l$ is related to the error bound in Theorem III.4 and it is used to ensure high success probability for Algorithm III.1. Taking $l$ to be a little bigger than $k$ is sufficient. The training process of our algorithm is described in Algorithm IV.1.

For the test phase of the algorithm, we need a similarity measure that provides a distance between a given test signal and a dictionary.

**Definition IV.1.** *Let $x$ be a signal and $D$ be a dictionary. The distance between $x$ and the dictionary $D$ is defined by*

**Algorithm IV.1:** Dictionaries Training using Randomized LU

**Input:** $X = \{X_1, X_2, \ldots, X_r\}$ training datasets for $r$ sets; $K = \{k_1, k_2, \ldots, k_r\}$ dictionary size of each set.

**Output:** $D = \{D_1, D_2, \ldots, D_r\}$ set of dictionaries.

1: **for** $t \in \{1, 2, \ldots, r\}$ **do**
   $P_t, Q_t, L_t, U_t \leftarrow Randomized\ LU\ Decomposition(X_t, k_t, l)$, $(l \triangleq k_t + 5)$; (Algorithm III.1)
   $D_t \leftarrow P_t^T L_t$
2: $D \leftarrow \{D_1, D_2, \ldots, D_r\}$

$$dist(x, D) \triangleq ||DD^{\dagger}x - x||,$$

*where $D^{\dagger}$ is the pseudo-inverse of the matrix $D$.*

The geometric meaning of $dist(x, D_i)$ is related to the projection of $x$ onto the column space of $D_i$, where $D_i$ is the dictionary learned for class $i$ of the problem. $dist(x, D_i)$ denotes the distance between $x$ and $D_i D_i^{\dagger} x$ which is the vector $x$ built with the dictionary $D_i$. If $x \in \text{column-span}\{D_i\}$ then Theorem IV.1 guarantees that $dist(x, D_i) < \epsilon$. For $x \notin \text{column-span}\{D_i\}$, $dist(x, D_i)$ is large. Thus, $dist$ is used for classification as described in Algorithm IV.2.

**Algorithm IV.2:** Dictionary based Classification

**Input:** $x$ input test signal; $D = \{D_1, D_2, \ldots, D_r\}$ set of dictionaries.

**Output:** $t_X$ the classified class label for $x$.

1: **for** $t \in \{1, 2, \ldots, r\}$ **do**
   $ERR_t \leftarrow dist(x, D_t)$
2: $t_X \leftarrow \arg\min_t \{ERR_t\}$

The core of Algorithm IV.2 is the $dist$ function from Definition IV.1. This is done by examining portion of the signal that is spanned by the dictionary atoms. If the signal can be expressed with high accuracy as a linear combination of the dictionary atoms then their $dist$ will be small. The best accuracy is achieved when the examined signal belongs to the span of $D_i$. In this case, $dist$ is small and bounded by Theorem III.4. On the other hand, if the dictionary atoms cannot express well a signal then their $dist$ will be large. The largest distance is achieved when a signal is orthogonal to the dictionary atoms. In this case, $dist$ will be equal to the norm of the signal. Signal classification is accomplished by finding a dictionary with a minimal distance to it. This is where the dictionary size comes into play. The more atoms a dictionary has, the larger is the space of signals that have low $dist$ to it and vice versa. By adding or removing atoms from a dictionary, the distances between this dictionary and the test signals are changed. This affects the classification results of Algorithm IV.2. The practical meaning of this observation is that dictionary sizes need to be chosen carefully. Ideally, we wish that each dictionary will be of $dist$ zero to test signals of its type, and of large $dist$ values for signals of other types. However, in reality, some test signals are represented more accurately by a dictionary of the wrong type than by a dictionary of their class type. For example, we encountered several cases where GIF files were represented more accurately by a PDF dictionary than by a GIF dictionary. An incorrect selection of the dictionary size, $k$, will result in either a dictionary that cannot represent well signals of its own class (causes misdetections), or in a dictionary that represents too accurately signals from other classes (causes false alarms). The first problem occurs when the dictionary is too small whereas the second occurs when the dictionary is too large. In Section V, we discuss the problem of finding the optimal dictionaries sizes and how they relate it to the spectrum of the training data matrices.

## V. Determining the Dictionaries Sizes

One possible way to find the dictionaries sizes is to observe the spectrum decay of the training data matrix. In this approach, the number of atoms in each dictionary is selected as the number of singular values that capture most of the energy of the training matrix. This method is based on estimating the numerical rank of the matrix, namely on the dimension of its column space. Such a dictionary approximates well the column space of the data and represents accurately signals of its own class. Nevertheless, it is possible in this construction that dictionary of a certain class will have high rate of false alarms. In other words, this dictionary might approximate signals from other classes with a low error rate.

Two different actions can be taken to prevent this situation. The first option is to reduce the size of this dictionary so that it approximates mainly signals of its class and not from other classes. This should be done carefully so that this dictionary still identifies well signals of its class better than other dictionaries. The second option is to increase the sizes of other dictionaries in order to overcome their misdetections. This should also be done with caution since we might represent well signals from other classes using these enlarged dictionaries. Therefore, relying only on the spectrum analysis of the training data is insufficient, because this method finds the size of each dictionary independently from the other dictionaries. It ignores the interrelations between dictionaries, while the classification algorithm is based on those relations. Finding the optimal $k$ values can be described by the following optimization problem:

$$\arg \min_{k_1,k_2,\ldots,k_r} \sum_{\substack{1\leq i\leq r\\ 1\leq j\leq r\\ i\neq j}} C_{i,j,X}(k_i,k_j), \tag{V.1}$$

where $C_{i,j,X}(k_i,k_j)$ is the number of signals from class $i$ in the dataset $X$ classified as belonging to class $j$ for the respective dictionary sizes $k_i$ and $k_j$. The term, which we wish to minimize in Eq. V.1, is therefore the total number of wrong classifications in our dataset $X$ when using a set of dictionaries $D_1, D_2, \ldots, D_r$ with sizes $k_1, k_2, \ldots, k_r$, respectively.

We propose an algorithm for finding the dictionary sizes by examining each specific pair of dictionaries separately, and thus identifying the optimized dictionary sizes for this pair. Then, the global $k$ values for all dictionaries will be determined by finding an agreement between all the local results. This process is described in Algorithm V.1.

**Algorithm V.1:** Dictionary Sizes Detection

**Input:** $X = \{X_1, X_2, \ldots, X_r\}$ training datasets for the $r$ classes; $K_{range}$ set of possible values of $k$ to search in.

**Output:** $K = \{k_1, k_2, \ldots, k_r\}$ dictionaries sizes.

1: **for** $i, j \in \{1, 2, \ldots, r\}$, $i < j$ **do**
  **for** $k_i, k_j \in K_{range}$ **do**
    $ERROR_{i,j}(k_i, k_j) \leftarrow C_{i,j,X}(k_i, k_j) + C_{j,i,X}(k_j, k_i)$
2: $K \leftarrow \mathbf{find\_optimal\_agreement}(\{ERROR_{i,j}\}_{1\leq i<j\leq r})$

Algorithm V.1 examines each pair of classes $i$ and $j$ for different $k$ values and produces the matrix $ERROR_{i,j}$, such that the element $ERROR_{i,j}(s,t)$ is the number of classification errors for those two classes, when the dictionary size of class $i$ is $s$ and the dictionary size of class $j$ is $t$. This number is the sum of signals from each class that were classified as belonging to the other class. The matrix $ERROR_{i,j}$ reveals the ranges of $k$ values for which the number of classification errors is minimal. These are the ranges that fit when dealing with a problem that contains only two classes of signals. However, many classification problems need to deal with a large number of classes. For this case, we create the $ERROR$ matrix for all possible pairs, find the $k$ ranges for each pair and then find the optimal agreement between all pairs. The step **find_optimal_agreement** describes this idea in Algorithm V.1. Finding this agreement

can be done by making a list of constraints for each pair and then finding $k$ values that satisfy all the constraint and bring the minimal solution to the problem described in Eq. V.1. The constraints can bound from below or above the size of a specific dictionary, or the relation between sizes of two dictionaries (for example, the dictionary of the first class should have 10 more elements than the dictionary of the second class). The step **find_optimal_agreement** is not described here formally but demonstrated in details as part of Algorithm V.1 in Section VI-B.

## VI. Experimental Results

In order to evaluate the performance of the dictionary construction and classification algorithms in Section IV, Algorithm IV.2 was applied to a dataset that contains six different file types. The goal is to classify each file or portion of a file to the class that describes its type. This dataset consists of 1200 files that were collected in the wild using automated Web crawlers. The files were equally divided into six types: PDF, EXE, JPG, GIF, HTM and DOC. 100 files of each type were chosen randomly as training datasets and the other 100 files served for algorithms testing. In order to get results that reflect the true nature of the problem, no restrictions were imposed on the file collection process. Thus, some files contain only a few kilobytes while others are of several megabytes in size. In addition, some of the PDF files contain pictures, which make it hard for a content-based algorithm to classify the correct file type. Similarly, DOC files may contain pictures and the executables may contain text and pictures. Clearly, these phenomena have negative effect on the accuracy of the results in this section. However, we chose to leave the dataset in its original form.

Throughout this work, we came across several similar works [2]–[5], [7], [10]–[12], [18] that classify unknown files to their type based on their content. None of these works made their datasets publicly available for analysis and comparison with other methods. We decided to publicize the dataset of files that we collected to enable future comparisons. The details about downloading and using the dataset can be obtained by contacting one of the authors.

Three different scenarios were tested with the common goal of classifying files or portions of files to their class type, namely, assigning them to one of the six file types described above. In each scenario, six dictionaries were learned that correspond to the six file types. Then, the classification algorithm (Algorithm IV.1) was applied to classify the type of a test fragment or a file. The learning phase, which is common to all scenarios, was done by applying Algorithm V.1 to find the dictionary sizes and Algorithm IV.1 to construct the dictionaries. The testing phase varies according to the specific goal of each scenario. Sections VI-A, VI-B and VI-C provide a detailed description for each scenario and its classification results.

### *A. Scenario A: Entire File is Analyzed*

In this scenario, we process a whole file and the extracted features are taken from its entire content. The features are byte frequency distribution (BFD) that contains 256 features followed by consecutive differences distribution (CDD) that adds another 256 features. Total of 512 features are measured for each training and testing files. CDD is used in addition to BFD because the latter fails to capture any information about bytes ordering in the file. CDD turned out to be very discriminative and improved the classification results. The features extracted from each file were normalized by its size since there are files of various sizes in the dataset. Example for BFD construction is described in Fig. VI.1 and example for CDD construction is given in Fig. VI.2.

This scenario can be useful when the entire tested file is available for inspection. The training was done by the application of Algorithms V.1 and IV.1 to the training data. The $K_{range}$ parameter to Algorithm V.1 was determined by the numerical rank of the training matrix. The possible dictionary sizes need to be close to this rank in order to represent well their datasets. The dictionary sizes were 60 atoms per dictionary. The set of dictionaries $D = \{D_{PDF}, D_{DOC}, D_{EXE}, D_{GIF}, D_{JPG}, D_{HTM}\}$ is the output of Algorithm IV.1, which is later used for classification of test files. Each test file was analyzed using Algorithm VI.1 and

AABCCCDR $\Longrightarrow$

| Byte | Probability (BFD) |
|---|---|
| A | 0.25 |
| B | 0.125 |
| C | 0.375 |
| D | 0.125 |
| R | 0.125 |
| ⋮ | ⋮ |
| | 0 |

Fig. VI.1. Byte Frequency Distribution (BFD) features extracted from the file fragment "AABCCCDR".

AABCCCDFG $\Longrightarrow$

| Difference | Probability (CDD) |
|---|---|
| 0 | 0.375 |
| 1 | 0.5 |
| 2 | 0.125 |
| ⋮ | ⋮ |
| | 0 |

Fig. VI.2. Consecutive Differences Distribution (CDD) features extracted from the file fragment "AABCCCDFG". There are three consecutive-pairs of bytes with difference 0, four with difference 1 and one with difference 2. These distributions are normalized to produce the shown probabilities. The normalization factor is the length of the string minus one (8 in this example).

classified to one of the six classes. The classification results are presented as a confusion matrix in Table VI.1. Each column corresponds to an actual file type and the rows correspond to the classified file type by Algorithm VI.1. A perfect classification produces a table with score 100 on the diagonal and zero elsewhere. Our results are similar to those achieved in [4] (Table II) that use different methods. However, we did not have the dataset that [4] used and there is no way to perform a fair comparison.

TABLE VI.1
CONFUSION MATRIX FOR SCENARIO A. 100 FILES OF EACH TYPE WERE CLASSIFIED BY ALGORITHM VI.1.

| | | Correct File Type | | | | | |
|---|---|---|---|---|---|---|---|
| | | PDF | DOC | EXE | GIF | JPG | HTM |
| Classified File Type | PDF | **98** | 0 | 1 | 1 | 0 | 0 |
| | DOC | 0 | **97** | 1 | 0 | 0 | 0 |
| | EXE | 0 | 3 | **98** | 2 | 1 | 0 |
| | GIF | 0 | 0 | 0 | **97** | 1 | 0 |
| | JPG | 2 | 0 | 0 | 0 | **98** | 0 |
| | HTM | 0 | 0 | 0 | 0 | 0 | **100** |

### *B. Scenario B: Fragments of a File*

In this scenario we describe a situation in which the entire file is unavailable for the analysis but only some fragments that were taken from random locations are available. The goal is to classify the file type based on this partial information. This serves a real application such as a firewall that examines packets transmitted through a network or a file being downloaded from a network server. This scenario contains three experiments where different features were used in each. The training phase, which is common to all three experiments, includes extracted features from a 10 kilobytes fragments that belong to the training data. These features serve as an input to Algorithm IV.1, which produces the dictionaries for the classification phase. The second parameter in Algorithm IV.1 is a set of dictionary sizes, which were determined by Algorithm V.1. We use the first set of features in this scenario (described hereafter) to demonstrate more deeply how Algorithm V.1 works. The sizes of six dictionaries need to be determined based on the agreement between the pairwise error matrices. Fig. VI.3 shows the matrices $ERROR_{PDF\text{-}JPG}$ and $ERROR_{PDF\text{-}EXE}$.

Fig. VI.3(a) describes the number of classification-errors for the PDF and JPG types, as a function of the respective dictionary sizes. It can be observed that there is a large number of errors for many size pairs, suggesting that the PDF and JPG dictionaries exhibit a high measure of similarity. This property makes the distinction between these two types a hard task. A closer look on Fig. VI.3(a) enables us to find the optimal sizes for those dictionaries, by making the following observations. Only a few values in

**Algorithm VI.1:** File Content Dictionary Classification

**Input:** $x$ input file; $D = \{D_{PDF}, D_{DOC}, D_{EXE}, D_{GIF}, D_{JPG}, D_{HTM}\}$ set of dictionaries.
**Output:** $t_x$ file type predicted for $x$.

1: **for** $t \in \{PDF, DOC, EXE, GIF, JPG, HTM\}$ **do**
   $ERR_t \leftarrow dist(x, D_t)$
2: $t_x \leftarrow \arg\min_t \{ERR_t\}$

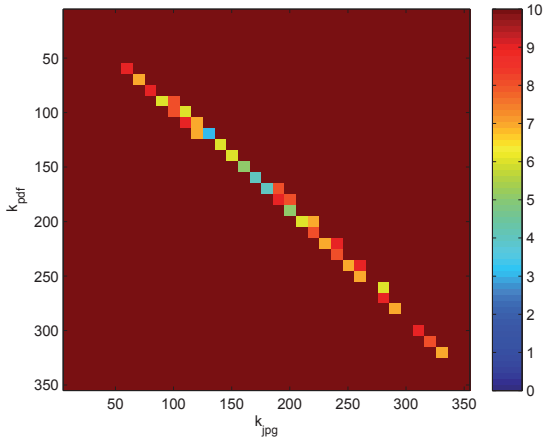
the cells above the main diagonal provide good results for this pair. Additionally, JPG dictionary should have 10 atoms more than the PDF dictionary. It cab be also learned that both dictionaries sizes should be greater than 50 atoms.

The PDF and EXE error values in Fig. VI.3(b) indicate that these dictionaries are well separated. There is a large set of dictionary sizes near the diagonal for which the classification error is low. The following intuition helps to understand why a large range of low errors will achieve better classification results. The error matrices are built based on training data $X$ and represent the classification error $C_{PDF,JPG,X} + C_{JPG,PDF,X}$ of the algorithm when it applies to this data (See Eq. V.1 when using 2 sets). The best $k$ values from the $ERROR$ matrix fit the training data, in the sense that a PDF training signal will be represented more accurately by a PDF dictionary of size $k_{pdf}$ than by a JPG dictionary of size $k_{jpg}$. However, this is not necessarily the case for a PDF *test* signal, which may need a larger PDF dictionary or smaller JPG dictionary in order to be classified correctly. This might happen because many PDF-dictionary atoms are irrelevant for reconstructing this signal while too many JPG-dictionary atoms are relevant for it. This means that from this signal's perspective, the PDF dictionary size is smaller than $k_{pdf}$ and the JPG dictionary is larger than $k_{jpg}$. In terms of Fig. VI.3, which shows the classification errors for the two discussed pairs, this means moving away from the diagonal (which has the best dictionary sizes for the training set). In the JPG-PDF case, this shift will increase the classification error because all the off-diagonal entries in Fig. VI.3(a) have higher errors numbers. On the other hand, there is a low probability to get a classification error in Fig. VI.3(b), because there are many off-diagonal options for dictionary sizes that will generate a low error. The pair JPG-PDF is more sensitive to noise than the pair EXE-PDF. This observation is supported by the confusion matrix of the first experiment, as shown in Table VI.2.

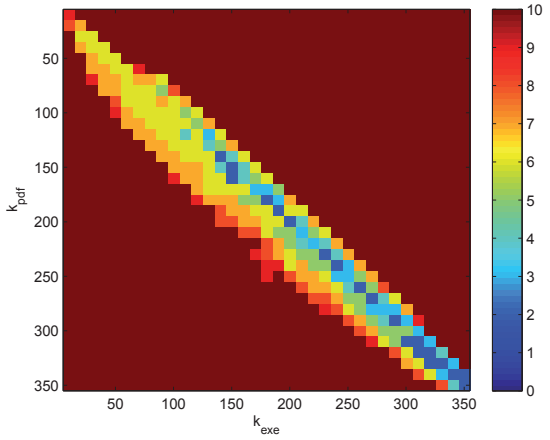
In the first experiment, the dictionary sizes, which were determined by Algorithm V.1, are 150 atoms per PDF, DOC, EXE, GIF, and HTM dictionaries and 160 atoms per JPG dictionary. 10 fragments of 1500 bytes each were sampled randomly from each examined file. BFD and CDD based features were extracted from each fragment and then normalized by the fragment size (similarly to the normalization by file size conducted in Scenario A in Section VI-A). Then, the distance between each fragment and each of the six dictionaries was calculated. The mean value of the distances was computed for each dictionary. Eventually, the examined file was classified to the class that has the minimal mean value. This procedure is described in Algorithm VI.2. The classification results are presented in Table VI.2.

TABLE VI.2
CONFUSION MATRIX FOR SCENARIO B WHERE BFD+CDD BASED FEATURES WERE CHOSEN. 100 FILES OF EACH TYPE WERE CLASSIFIED BY ALGORITHM VI.2.

| | | **Correct File Type** | | | | | |
|---|---|---|---|---|---|---|---|
| | | PDF | DOC | EXE | GIF | JPG | HTM |
| **Classified File Type** | PDF | **93** | 0 | 2 | 0 | 14 | 0 |
| | DOC | 0 | **96** | 2 | 0 | 0 | 0 |
| | EXE | 0 | 4 | **95** | 0 | 0 | 0 |
| | GIF | 0 | 0 | 0 | **100** | 2 | 0 |
| | JPG | 6 | 0 | 0 | 0 | **82** | 0 |
| | HTM | 1 | 0 | 1 | 0 | 2 | **100** |




(a) Error matrix for the pair PDF-JPG




(b) Error matrix for the pair PDF-EXE

Fig. VI.3. Error matrices produced by Algorithm V.1. The matrix is presented in cold to hot colormap to show ranges of low (blue) and high (red) errors.

The second experiment used a double-byte frequency distribution (DBFD), which contains 65536 features. Figure VI.4 demonstrates the DBFD feature extraction from a small file fragment.

AABCCC $\Longrightarrow$

| Double-Byte | Probability (DBFD) |
|---|---|
| AA | 0.2 |
| AB | 0.2 |
| BC | 0.2 |
| CC | 0.4 |
| ⋮ | 0 |

Fig. VI.4. Features extracted from the file fragment "AABCCC" using Double Byte Frequency Distribution (DBFD). The normalization factor is the length of the string minus one.

Similarly to the first experiment, 10 fragments were sampled from random locations at each examined file. However, this time we used 2000 bytes per fragment since smaller fragment sizes do not capture suf-

**Algorithm VI.2:** File fragment classification using dictionary learning

**Input:** $X = \{x_1, x_2, \cdots, x_r\}$ input fragments; $D = \{D_{PDF}, D_{DOC}, D_{EXE}, D_{GIF}, D_{JPG}, D_{HTM}\}$ set of dictionaries.

**Output:** $t_X$ file type predicted to $X$.

1: **for** $i \leftarrow 1, \ldots, r$ **do**
  **for** $t \in \{PDF, DOC, EXE, GIF, JPG, HTM\}$ **do**
    $ERR_{i,t} \leftarrow dist(x_i, D_t)$
2: **for** $t \in \{PDF, DOC, EXE, GIF, JPG, HTM\}$ **do**
  $MEAN_t \leftarrow \text{mean}\{ERR_{i,t}\}_{i=1}^{r}$
3: $t_X \leftarrow \arg\min_t \{MEAN_t\}$

ficient information when DBFD features are used. The feature vectors were normalized by the fragment's size as before. Algorithm VI.2 was applied to classify the type of each examined file. The dictionaries sizes in this experiment are 80 atoms per PDF, DOC and JPG and 60 atoms per EXE, GIF and HTM. The classification results of this experiment are presented in Table VI.3. We see that DBFD based features reveal patterns in the data that were not revealed by using BFD and CDD based features. In particular, it captures very well GIF files that BFD and CDD based features fail to capture.

TABLE VI.3
CONFUSION MATRIX FOR SCENARIO B THAT IS BASED ON DBFD BASED FEATURES. 100 FILES OF EACH TYPE WERE CLASSIFIED BY ALGORITHM VI.2.

| | | **Correct File Type** | | | | | |
|---|---|---|---|---|---|---|---|
| | | PDF | DOC | EXE | GIF | JPG | HTM |
| **Classified File Type** | PDF | **92** | 0 | 2 | 0 | 5 | 1 |
| | DOC | 2 | **97** | 2 | 0 | 5 | 0 |
| | EXE | 3 | 1 | **88** | 2 | 0 | 0 |
| | GIF | 1 | 1 | 5 | **98** | 0 | 0 |
| | JPG | 1 | 1 | 2 | 0 | **90** | 0 |
| | HTM | 1 | 0 | 1 | 0 | 0 | **99** |

The third experiment defines a Markov-walk (MW) like set of 65536 features extracted from the dataset for each signal. The transition probability between each pair of bytes is calculated. Figure VI.5 demonstrates how to extract MW type features from a file fragment.

AABCCCF $\Longrightarrow$

| Transition | Probability (MW) |
|---|---|
| A $\rightarrow$ A | 0.5 |
| A $\rightarrow$ B | 0.5 |
| B $\rightarrow$ C | 1 |
| C $\rightarrow$ C | 0.66 |
| C $\rightarrow$ F | 0.33 |
| $\vdots$ | 0 |

Fig. VI.5. Markov Walk (MW) based features extracted from the file fragment "AABCCCF".

Both MW based features and DBFD based features are calculated using the double byte frequencies, but they capture different information from the data. DBFD based features are focused on finding pairs of bytes that are most prevalent and those who have low chances of appearing in a file. On the other hand, MW based features represent the probability that a specific byte will appear in the file given the appearance of a previous byte. This is well suited to file types such as EXE where similar addresses and opcodes are used repeatedly. Each memory address or opcode is comprised of two or more bytes, therefore, it can be described by the transition probability between these bytes. Text files also constitute a good example for the applicability of MW based features because it is well known that natural language can be described

by patterns of transition probabilities between words or letters. Our study shows that MW based features capture also the structure of media files like GIF and HTM files. The relatively unsatisfactory performance on JPG files is because our PDF dictionary was trained on PDF files that contain pictures. Therefore, it detected some of the JPG files. The prediction accuracy is described in Table VI.4. Those results (97% avg. accuracy) outperform the results obtained by the BFD+CDD and DBFD features. It also improve over all the surveyed methods in [4] (Table VI), including the algorithm proposed in [4], that has 85.5% average accuracy. However, it should be noted that we used 10 fragments for the classification of each file whereas in [4] a single fragment is used. In Scenario B, the dictionary sizes are 500 atoms per PDF, DOC and EXE files, 600 per GIF files, 800 per JPG files and 220 per HTM files. The HTM dictionary is smaller than the other dictionaries due to the fact that the HTM training set contains only 230 samples, and the LU dictionary size is bounded by the dimensions of the training matrix (see Algorithm III.1).

TABLE VI.4
CONFUSION MATRIX FOR SCENARIO B USING MW BASED FEATURES. 100 FILES OF EACH TYPE WERE CLASSIFIED BY ALGORITHM VI.2.

| | | **Correct File Type** | | | | | |
|---|---|---|---|---|---|---|---|
| | | PDF | DOC | EXE | GIF | JPG | HTM |
| **Classified File Type** | PDF | **93** | 1 | 0 | 0 | 9 | 0 |
| | DOC | 0 | **98** | 0 | 0 | 0 | 0 |
| | EXE | 2 | 0 | **98** | 1 | 0 | 0 |
| | GIF | 3 | 1 | 1 | **99** | 0 | 0 |
| | JPG | 1 | 0 | 0 | 0 | **91** | 0 |
| | HTM | 1 | 0 | 1 | 0 | 0 | **100** |

### *C. Scenario C: Detecting Execution Code in PDF Files*

PDF is a common file format that can contain different media elements such as text, fonts, images, vector graphics and more. This format is widely used in the Web due to the fact that it is self contained and platform independent. While PDF format is considered to be safe, it can contain any file format including executables such as EXE files and various script files. Detecting malicious PDF files can be challenging as it requires a deep inspection into every file fragment that can potentially hide executable code segments. The embedded code is not automatically executed when the PDF is being viewed using a PDF reader since it first requires to exploit a vulnerability in the viewer code or in the PDF format. Still, detecting such a potential threat can lead to a preventive action by the inspecting system.

To evaluate how effective our method can be in detecting executable code embedded in PDF files, we generated several PDF files which contain text, images and executable code. We used four datasets of PDF files as our training data:

$X_{PDF}$: 100 PDF files containing mostly text.
$X_{GIFinPDF}$: 100 PDF files containing GIF images.
$X_{JPGinPDF}$: 100 PDF files containing JPG images.
$X_{EXEinPDF}$: 100 PDF files containing EXE files.

All the GIF, JPG and EXE files were taken from previous experiments and were embedded into the PDF files. We generated 4 dictionaries for each dataset using Algorithm IV.1. The input for the algorithm was

$$X = \{X_{PDF}, X_{GIFinPDF}, X_{JPGinPDF}, X_{EXEinPDF}\}.$$

We then created a test dataset which consisted of: 100 regular PDF files and 10 PDF files that contain executable code. Algorithm VI.2 classified the 110 files. The input fragments $X$ were the PDF file fragments. The input set of dictionaries

$$D = \{D_{PDF}, D_{GIFinPDF}, D_{JPGinPDF}, D_{EXEinPDF}\}$$

were the output from Algorithm IV.1. A file is classified as malicious (contains an executable code) if we find more than $T_{EXE}$ fragments of type EXE inside, otherwise it is classified as a safe PDF file. We used $T_{EXE} = 10$ as our threshold since it minimized the total number of miss-classifications. The training step was applied to 10 kilobytes fragments and the classification step was applied to five kilobytes fragments. We used the MW based features (65,536 extracted features). By using Algorithm VI.2, we managed to detect all the 10 malicious PDF files with $8\%$ of false alarm rate (8 PDF files that were classified as malicious PDF files). The results are summarized in Table VI.5.

TABLE VI.5
CONFUSION MATRIX FOR MALICIOUS PDF DETECTION EXPERIMENT. 110 FILES WERE CLASSIFIED BY ALGORITHM VI.2.

| | | **Correct File Type** | |
|---|---|---|---|
| | | PDF | Malicious PDF |
| **Classified File Type** | Safe PDF | **92** | 0 |
| | Malicious PDF | 8 | **10** |

Other file formats, which contain embedded data (DOC files for example), can be classified in the same way.

### *D. Time Measurements*

Computer security software face frequent situations that were described in sections VI-A–VI-C. Therefore, any solution to a file type classification must provide a quick response to queries. We measured the time required for both the training phase and the classification phase of our method that classifies a file or a fragment of a file. Since the training phase operates offline it does not need to be fast. On the other hand, classification query should be fast for real-time considerations and for high-volume applications. Tables VI.6 and VI.7 describe the execution time in Scenarios A (Section VI-A) and B (Section VI-B), respectively. The times are divided into a preprocessing step and into the actual analysis step. The preprocessing includes feature extraction from files (data preparation) and loading this data into Matlab. The feature extraction was done in Python and the output files were loaded to Matlab. Obviously, this is not an optimal configuration as it involves intensive slow disk I/O. We did not optimize these steps. We note that the computation time of the dictionary size is not included in the table, because this is a meta-parameter to Algorithm IV.1 which can be computed in different ways, based on the application. The analysis time refers to the time needed by Algorithm IV.1 to build six dictionaries (left column in each table) and to classify a single file to one of the six classes (right column). The classification was performed by Algorithm VI.1 in Scenario A (Table VI.6), and by Algorithm VI.2 in Scenario B (Table VI.7). All training and classification times are normalized by the data size, which allows evaluation of the algorithm performance regardless of actual file sizes (which vary largely). Classification time of Scenario B is not normalized because Algorithm VI.2 is not dependent on the input file size (it samples the same amount of data from each file, ignoring its size). Our classification process is fast. The preprocessing step can be further optimized for real-time applications. All the experiments were conducted on Windows 64-bit, Intel i7, 2.93 GHz CPU machine with 8 GB of RAM.

TABLE VI.6
RUNNING TIMES FOR SCENARIO A.

| Features | | Training time (sec) per 1 MB of data | Classification time (sec) per 1 MB of data |
|---|---|---|---|
| BFD+CDD | Preprocessing | 1.8 | 1.88 |
| | Analysis | **0.004** | **0.0005** |
| | Total | 1.804 | 1.8805 |

TABLE VI.7
RUNNING TIMES FOR SCENARIO B.

| Features | | Training time (sec) per 1 MB of data | Classification time (sec) |
|---|---|---|---|
| BFD+CDD | Preprocessing | 1.93 | 0.1 (per 1 MB) |
| | Analysis | **0.008** | **0.01** (per file) |
| | Total | 1.938 | |
| DBFD | Preprocessing | 13.78 | 1.6 (per 1 MB) |
| | Analysis | **0.54** | **0.26** (per file) |
| | Total | 14.32 | |
| MW | Preprocessing | 18.42 | 2.41 (per 1 MB) |
| | Analysis | **0.65** | **0.27** (per file) |
| | Total | 19.07 | |

## VII. CONCLUSION

In this work, we presented a novel algorithm for dictionary construction, which is based on a randomized LU decomposition. By using the constructed dictionary, the algorithm classifies the content of a file and can deduct its type by examining a few file fragments. The algorithm can also detect anomalies in PDF files (or any other rich content formats) which can be malicious. This approach can be applied to detect suspicious files that can potentially contain malicious payload. Anti-virus systems and firewalls can therefore analyze and classify PDF files using the described method and block suspicious files. The usage of dictionary construction and classification in our algorithm is different from other classical methods for file content detection, which use statistical methods and pattern matching in the file header for classification via deep packet inspection. The fast dictionary construction allows to rebuild the dictionary from scratch when it is out-of-date which is important when building evolving systems that classify continuously changing data.

## ACKNOWLEDGMENT

This research was partially supported by the Israel Science Foundation (Grant No. 1041/10), by the Israeli Ministry of Science & Technology (Grants No. 3-9096, 3-10898), by US - Israel Binational Science Foundation (BSF 2012282) and by a Fellowship from Jyväskylä University.

## REFERENCES

[1] M. AHARON, M. ELAD, AND A. BRUCKSTEIN, *K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation*, IEEE Trans. on Signal Processing, 54 (2006), pp. 4311–4322.
[2] I. AHMED, K.-S. LHEE, H. SHIN, AND M. HONG, *On improving the accuracy and performance of content-based file type identification*, in Information Security and Privacy, Springer, 2009, pp. 44–59.
[3] ———, *Fast file-type identification*, in Proceedings of the 2010 ACM Symposium on Applied Computing, ACM, 2010, pp. 1601–1602.
[4] M. C. AMIRANI, M. TOORANI, AND S. MIHANDOOST, *Feature-based type identification of file fragments*, Security and Communication Networks, 6 (2013), pp. 115–128.
[5] W. C. CALHOUN AND D. COLES, *Predicting the types of file fragments*, Digital Investigation, 5 (2008), pp. S14–S20.
[6] S. S. CHEN, D. L. DONOHO, AND M. A. SAUNDERS, *Atomic decomposition by basis pursuit*, SIAM journal on scientific computing, 20 (1998), pp. 33–61.
[7] R. F. ERBACHER AND J. MULHOLLAND, *Identification and localization of data types within large-scale file systems*, in Systematic Approaches to Digital Forensic Engineering, 2007. SADFE 2007. Second International Workshop on, IEEE, 2007, pp. 55–70.
[8] M. GU AND S. C. EISENSTAT, *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM Journal on Scientific Computing, 17 (1996), pp. 848–869.
[9] Z. JIANG, Z. LIN, AND L. S. DAVIS, *Learning a discriminative dictionary for sparse coding via label consistent k-svd*, in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, 2011, pp. 1697–1704.
[10] M. KARRESAND AND N. SHAHMEHRI, *File type identification of data fragments by their binary structure*, in Information Assurance Workshop, 2006 IEEE, IEEE, 2006, pp. 140–147.
[11] W.-J. LI, K. WANG, S. J. STOLFO, AND B. HERZOG, *Fileprints: Identifying file types by n-gram analysis*, in Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC, IEEE, 2005, pp. 64–71.

[12] M. McDaniel and M. H. Heydari, *Content based file type detection algorithms*, in System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on, IEEE, 2003, pp. 10–pp.
[13] B. K. Natarajan, *Sparse approximate solutions to linear systems*, SIAM journal on computing, 24 (1995), pp. 227–234.
[14] C.-T. Pan, *On the existence and computation of rank-revealing LU factorizations*, Linear Algebra and its Applications, 316 (2000), pp. 199–222.
[15] D.-S. Pham and S. Venkatesh, *Joint learning and dictionary construction for pattern recognition*, in Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–8.
[16] G. Shabat, Y. Shmueli, and A. Averbuch, *Randomized LU decomposition*, arXiv preprint arXiv:1310.7202, (2013).
[17] J. Tropp, *Greed is good: algorithmic results for sparse approximation*, Information Theory, IEEE Transactions on, 50 (2004), pp. 2231–2242.
[18] C. J. Veenman, *Statistical disk cluster classification for file carving*, in Information Assurance and Security, 2007. IAS 2007. Third International Symposium on, IEEE, 2007, pp. 393–398.
[19] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, *Robust face recognition via sparse representation*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 31 (2009), pp. 210–227.
[20] A. Y. Yang, J. Wright, Y. Ma, and S. S. Sastry, *Feature selection in face recognition: A sparse representation perspective*, submitted to IEEE Transactions Pattern Analysis and Machine Intelligence, (2007).
[21] M. Yang, D. Zhang, and X. Feng, *Fisher discrimination dictionary learning for sparse representation*, in Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 543–550.
[22] Q. Zhang and B. Li, *Discriminative k-svd for dictionary learning in face recognition*, in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 2691–2698.

# PIV

## INCOMPLETE PIVOTED QR-BASED DIMENSIONALITY REDUCTION

by

Amit Bermanis, Aviv Rotbart, Moshe Salhov, Amir Averbuch 2015

Submitted

# Deterministic Controllable-Distortion Dictionary-Based Dimensionality Reduction

Aviv Rotbart[1,2], Amit Bermanis[1], Moshe Salhov[1] and Amir Averbuch[1]

[1]School of Computer Science, Tel Aviv University, Israel
[2]Department of Mathematical Information Technology,
University of Jyväskylä, Finland

September 27, 2015

**Abstract**

High-dimensional data appears in many research fields, such as image recognition, biology and collaborative filtering. Often, the exploration of such data by classic algorithms is encountered with difficulties due to the 'curse of dimensionality' phenomenon. Therefore, dimensionality reduction methods are applied to the data prior to its analysis. Many of those methods are based on principal components analysis, which is statistically driven, namely they map the data into a low-dimension subspace that preserves significant statistical properties of the high-dimensional data. As a consequence, such methods do not directly address the geometry of the data, reflected by mutual distances between data point. Thus, classification, anomaly detection or other machine learning tasks may be affected.

This work provides a dictionary-based framework for geometrically driven data analysis that includes dimensionality reduction, out-of-sample extension and anomaly detection. It embeds high-dimensional data in a low-dimensional subspace. This embedding preserves the original high-dimensional geometry of the data, up to a user-defined distortion rate. In addition, it identifies a subset of landmark data points that constitute a dictionary for the analyzed dataset. The dictionary enables a natural extension of the low-dimensional embedding to out-of-sample points, which gives rise to a distortion-based criterion for anomaly detection. The suggested method is demonstrated on synthetic and real-world datasets and achieves good results in a benchmark classification task.

## 1 Introduction

Data collection and storage processes have been changed significantly throughout the history of statistics and data analysis. In the past, those tasks were manually accomplished, which restricted the size of datasets to several hundreds. Technological progress has removed the barriers from the storage of massive data collections (Big Data), and nowadays automated sensor systems continuously sample real-world processes and generate ever-growing datasets with millions of new samples per day. Analysis of data collections is a fundamental task in many scientific and industrial fields, whose goal is inference of significant information out of a collection of observations, which may illuminate the underlying phenomenon that generates the observed data, and assist in tasks such as efficient storage (compressing), classification, clustering, forecasting and anomaly detection.

High-dimensional data analysis is of special interest, as in high dimensions data points occupy a very small fraction of the entire space, compared to the same amount of data in lower dimensional space. Therefore, the number of points required to represent a certain phenomenon in the data is becoming larger. This phenomenon is referred to as the 'curse of dimensionality' [7]. For example, data clustering requires regions of high density, which constitute clusters. In high dimensions, a huge amount of data is required to create high density regions, and this amount grows exponentially with dimension. High-dimensional data appear in two main forms: parametric and non parametric. In the former case, every observed data point consists of many parameters, each one corresponds to a single dimension. Typically, parametric data analysis concerns the geometry of such data. Non parametric data is typically originated in artificial geometry ascription to the analyzed data, usually encapsulated in a *kernel* matrix. Analysis of this geometry, through analysis of the associated kernel, might uncover latent features of the data, alas such geometries may be high-dimensional. Either way, for efficient analysis of high-dimensional data, a dimensionality reduction that preserves the original high-dimensional geometry is needed.

Principal Component Analysis (PCA) [30] is a statistically driven linear dimensionality reduction method, originally designates for the analysis of parametric data. It acts on the Gram matrix of the data, and embeds the data in a low-dimensional space, whose coordinates are directions of high variances of the data, also known as principal components. PCA can be implemented by application of singular value decomposition (SVD) [27] to the data matrix. However, this implementation is computationally expensive. Section 1.4 details the essentials of SVD.

Induced by PCA for linear analysis of parametric data, the kernel PCA approach for non linear analysis of non parametric data was introduces in [43]. Essentially, if the kernel matrix is semi-positive definite, then it can be treated as a Gram matrix of the data in the high-dimensional space (which is referred to as *feature space*.) Then, PCA is applied to the kernel matrix, in order to embed the data in a low-dimensional space. Some examples for kernel methods, among many others, are local linear embedding (LLE) [41], Isomap [45] and Diffusion Maps (DM) [20]. LLE embeds the data in a low-dimensional space whose geometry represents local linear geometries of the high-dimensional data. The low-dimensional geometry produced by Isomap preserves geodesic distances in the original data. DM provides a low-dimensional representation of the diffusion geometry of the data, as defined in [15, 20].

In this work an incomplete pivoted QR-based deterministic method for dimensionality reduction is presented. The method is designated to preserve a high-dimensional Euclidean geometry of parametric data, up to a user-specified distortion rate, according to the following definition:

**Definition 1.1** ($\mu$-distortion)**.** *Let $(\mathcal{H}, \mathbf{m}_{\mathcal{H}})$ and $(\mathcal{L}, \mathbf{m}_{\mathcal{L}})$ be metric spaces, let $\mathcal{A} \subset \mathcal{H}$ and let $\mu \geq 0$. A map $\mathbf{F} : \mathcal{A} \to \mathcal{L}$ is called a $\mu$-distortion of $\mathcal{A}$ if $\sup_{x,y \in \mathcal{A}} |\mathbf{m}_{\mathcal{H}}(x, y) - \mathbf{m}_{\mathcal{L}}(\mathbf{F}(x), \mathbf{F}(y))| \leq \mu$. The space $\mathcal{L}$ is referred to as a $\mu$-embedding space of $\mathcal{A}$.*

The proposed method is dictionary-based, where the dictionary is chosen from the analyzed dataset $\mathcal{A}$. The method identifies a Euclidean embedding space, spanned by the dictionary elements, on which the orthogonal projection of the data provides a user-defined distortion of the original high-dimensional dataset. In that sense, our method is geometrically driven, as opposed to PCA. Clearly, there is an interplay between the distortion rate and the dimension of the resulted embedding subspace: the smaller $\mu$, the higher the embedding's dimensional, and vice-versa. Our methods preserves global patterns of the data and trades local geometry for low-dimensional representation, as dense regions in the original data (such as clusters) are more sensitive to distortions than sparse regions, such as gaps between clusters.

Additionally to dimensionality reduction, we present two strongly related schemes for out-of-sample extension and anomaly detection, which are naturally stem from the proposed method

for dimensionality reduction. Thus, the dimensionality reduction phase, followed by out-of-sample extension and anomaly detection, constitutes a complete framework for semi-supervised learning, where the original (in-sample) dataset $\mathcal{A}$ functions as a training set. In this context, the learning phase is reflected in the extraction of a $\mu$-embedding subspace of $\mathcal{A}$, as defined in Definition 1.1. Out-of-sample data points, whose projection on the representative subspace is of low distortion are classified as normal, while the rest are classified as abnormal. Therefore, the original dataset $\mathcal{A}$ is considered as normal by definition.

To conclude, the contribution of this work is threefold: first, the suggested method identifies landmark data points (dictionary) that represent the data, as opposed to PCA that lacks this feature, and therefore in some sense is less informative. In matrix decomposition terminology, this is known as the Columns Subset Selection (CSS) problem. Secondly, the presented method requires very low storage budget, relatively to PCA. In addition, in the worst case, its computational complexity is identical to that of PCA computation. Lastly, the proposed out-of-sample extension and anomaly detection constitute natural consequences from the dimensionality reduction phase.

The rest of this paper is organized as follows: in the rest of this section we review related works (Section 1.1), describe the used notation and our general approach (Sections 1.2 and 1.3), and discuss the essentials of PCA-based dimensionality reduction (Section 1.4). Section 2 establishes the theory and the technical tools on which our method is based. It presents the robustness of our method to noise and its relation to matrix approximation, as well. Although the proposed method is designated for parametric data analysis, section 3 describes its utilization for the Diffusion Maps (DM), which is a non-parametric analysis method. Section 4 presents experimental results. Finally, Section 5 concludes the paper and discusses future works.

## 1.1 Related works

Johnson-Lindenstrauss Lemma [34] constitutes a basis for many random projection based dimensionality reduction methods [35, 32, 44, 1, 2, 19, 6] and [46] to name some. In contrast to the absolute bound from Definition 1.1, the lemma guarantees a relative distortion bound of the form $(1-\varepsilon)\cdot\mathbf{m}^2_{\mathcal{H}}(x,y) \leq \mathbf{m}^2_{\mathcal{L}}(\mathbf{F}(x),\mathbf{F}(y)) \leq (1+\varepsilon)\cdot\mathbf{m}^2_{\mathcal{H}}(x,y)$ with high probability. From data analysis perspective, the absolute bound may be more useful when the data is comprised of dense clusters separated by sparse regions. In such scenario, the absolute bound guarantees embedding such that intra-cluster distances may be distorted but inter-cluster distances are preserved, so that the global high-dimensional geometry is preserved in the $\mu$-embedding space. The relative bound, on the other hand, may produce a low-dimensional space in which clusters are becoming too close to each other.

Another significant branch of dimensionality reduction methods that is strongly related to the presented work deals with the CSS problem [25, 23, 12, 11, 10, 36] to name a few. Interpolative decomposition (ID) of a matrix was first introduces for operator compression [18]. It is designated to spectrally approximate linear integral operators. Randomized version of ID is presented in [38]. The class of randomized matrix decomposition algorithms is out of the scope of this paper. CUR decomposition [13, 37] of a given matrix $A$ generalizes ID in the sense that both subsets of columns and rows of the original matrix are selected to form the matrices $C$, $R$ respectively such that $A \approx CUR$ for a low rank matrix $U$. In data analysis terms, this interprets to both data points and features sampling and allows different perspective on the data. Similar to ID and unlike this work, CUR decompositions are designated to spectrally approximate the original matrix $A$. Randomized versions of CUR also exist, see [48, 24] and references therein.

## 1.2 Notation

In the rest of the paper the following notations are used: for $k \in \mathbb{N}$, $[k] = \{1, \ldots, k\}$. $I_n$ is the $n \times n$ unit matrix. The $i$-th coordinate of a vector $\mathbf{v} \in \mathbb{R}^n$ is denoted by $\mathbf{v}_{(i)}$. The $(i,j)$-th entry of a matrix $A$ is $A_{(i,j)}$ and its $i$-th row and $j$-th column are $A_{(i,:)}$ and $A_{(:,j)}$, respectively. If $A$ is of size $m \times n$ and $\mathcal{I} = \{i_1, \ldots, i_k\} \subset [m]$ and $\mathcal{J} = \{j_1, \ldots, j_\ell\} \subset [n]$ are two ordered sets, then $A_{(\mathcal{I},:)}$ is the $k \times n$ matrix $B$, for which $B_{(r,:)} = A_{(i_r,:)}$, $r \in [k]$, $A_{(:,\mathcal{J})}$ is the $m \times \ell$ matrix $C$, for which $C_{(:,r)} = A_{(:,j_r)}$, $r \in [\ell]$ and $A_{(\mathcal{I},\mathcal{J})}$ is the $k \times \ell$ matrix $D$, whose $(p,q)$-th entry is $D_{(p,q)} = A_{(i_p,j_q)}$, $p \in [k], q \in [\ell]$. The transposed matrix of $A$ is denoted by $A^*$ and $A^\dagger$ is the Moore-Penrose pseudo-inverse of $A$ . $\boldsymbol{\pi} : [n] \to [n]$ denotes permutation, and $\Pi$ is the associated $n \times n$ permutation matrix, i.e. $\Pi_{(i,j)} = 1$ if $\boldsymbol{\pi}(j) = i$, otherwise $\Pi_{(i,j)} = 0$. For a subspace $\mathcal{S} \subset \mathbb{R}^m$, $\mathcal{S}^\perp$ is the complementary perpendicular subspace of $\mathcal{S}$ in $\mathbb{R}^m$, and $\mathbf{W}_\mathcal{S}, \mathbf{W}_\mathcal{S}^\perp : \mathbb{R}^m \to \mathbb{R}^m$ are the corresponding orthogonal projections on these subspaces, respectively. Finally, $\|\mathbf{v}\|$ is the standard Euclidean norm of $\mathbf{v}$, and where $\|\mathbf{v}\|_1 = \sum_{i=1}^n |\mathbf{v}_{(i)}|$ is its $\ell_1$ norm.

The explored dataset is $\mathcal{A} = \{\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(n)}\} \subset \mathbb{R}^m$ and the associated data matrix is the $m \times n$ matrix $A$, whose $j$-th column is $A_{(:,j)} = \mathbf{a}^{(j)}$, $j \in [n]$.

## 1.3 General approach

Our approach for a low-rate distortion embedding consists of two main steps:

1. Given a nonnegative distortion parameter $\mu$, an $s$-dimensional $2\mu$-embedding subspace $\mathcal{S} \subset \mathbb{R}^m$ is identified, for which the orthogonal projection of any $\mathbf{a} \in \mathcal{A}$ results in an energy loss of mostly $\mu$, i.e.

$$\left\|\mathbf{W}_\mathcal{S}^\perp(\mathbf{a}^{(i)})\right\| \leq \mu, \quad i \in [n]. \tag{1.1}$$

   Here, the nonnegative distortion parameter is a user defined input. Clearly, $s$ is a non-increasing function of $\mu$.

2. The subspace $\mathcal{S}$ is orthogonally aligned with $\mathbb{R}^s$ to achieve an $s$-dimensional representation of $\mathcal{A}$, i.e. $\mathbf{O}_\mathcal{S} : \mathbb{R}^m \to \mathbb{R}^s$ is an orthogonal transformation that satisfies

$$\|\mathbf{O}_\mathcal{S}\mathbf{v}\| = \|\mathbf{v}\|, \quad \mathbf{v} \in \mathcal{S}. \tag{1.2}$$

   Obviously, such an alignment (which is not unique) does not affect the geometry of the projected set $\mathcal{A}$ on $\mathcal{S}$.

Application of the above two-stage scheme to $\mathcal{A}$ results in a $2\mu$-distortion, as Lemma 1.1 shows.

**Lemma 1.1.** *Let $\mathcal{S} \subset \mathbb{R}^m$ be an $s$-dimensional subspace of $\mathbb{R}^m$ that satisfies condition 1.1 and let $\mathbf{O}_\mathcal{S} : \mathbb{R}^m \to \mathbb{R}^s$ be an orthogonal transformation that satisfies condition. 1.2. Then, the $s$-dimensional map $\mathbf{F}_s : \mathbb{R}^m \to \mathbb{R}^s$,*

$$\mathbf{F}_s \triangleq \mathbf{O}_\mathcal{S} \circ \mathbf{W}_\mathcal{S} \tag{1.3}$$

*is a $2\mu$-distortion of $\mathcal{A}$.*

*Proof.* First, due to Eqs. 1.2 and 1.3, we get $\|\mathbf{F}_s(\mathbf{v})\| = \|\mathbf{O}_\mathcal{S} \circ \mathbf{W}_\mathcal{S}(\mathbf{v})\| = \|\mathbf{W}_\mathcal{S}(\mathbf{v})\|$ note that since $\mathbf{W}_\mathcal{S}$ is orthogonal projection, $\|\mathbf{W}_\mathcal{S}(\mathbf{v})\| \leq \|\mathbf{v}\|$. Thus, $0 \leq \|\mathbf{v}\| - \|\mathbf{F}_s(\mathbf{v})\| = \|\mathbf{v}\| - \|\mathbf{W}_\mathcal{S}(\mathbf{v})\| \leq \|\mathbf{v} - \mathbf{W}_\mathcal{S}(\mathbf{v})\| = \|\mathbf{W}_\mathcal{S}^\perp(\mathbf{v})\|$, $\mathbf{v} \in \mathbb{R}^m$. Substituting $\mathbf{v} = \mathbf{a}^{(i)} - \mathbf{a}^{(j)}$, $i, j \in [n]$ yields $0 \leq \|\mathbf{a}^{(i)} - \mathbf{a}^{(j)}\| - \|\mathbf{F}_s(\mathbf{a}^{(i)}) - \mathbf{F}_s(\mathbf{a}^{(j)})\| \leq \|\mathbf{a}^{(i)} - \mathbf{a}^{(j)} - \mathbf{F}_s(\mathbf{a}^{(i)}) + \mathbf{F}_s(\mathbf{a}^{(j)})\| \leq \|\mathbf{W}_\mathcal{S}^\perp(\mathbf{a}^{(i)})\| + \|\mathbf{W}_\mathcal{S}^\perp(\mathbf{a}^{(j)})\| \leq 2\mu$. The last inequality is due to Eq. 1.1. □

We stress the fact that our aim is to approximate the *geometry* of the dataset $\mathcal{A}$ rather than its elements, therefore we use $\mathbf{F}_s$ rather than $\mathbf{W}_\mathcal{S}$.

## 1.4 PCA-based dimensionality reduction

A common practice for dimensionality reduction is based on PCA [30] of the (centered) $m \times n$ data matrix $A$. This method uses a singular value decomposition (SVD) [27] of the data matrix, to detect a set of maximum variance orthogonal directions (singular vectors) in $\mathbb{R}^m$. Projection of the data onto the $\rho$ most significant directions yields the best $\rho$-dimensional embedding of the data in the sense of mean square error. The computational and storage complexities of SVD are $O(\min\{m,n\} \cdot mn)$ and $O(\min\{m,n\}^2)$, respectively.

Numerical methods for SVD approximation attract a growing interest. Out of many methods, we mention here some central ones. In the recent years randomized algorithms for SVD approximation of large matrices became popular. We refer the reader to [29] and references therein for a review of such methods. An efficient incremental algorithms for computing a thin SVD that considers $\rho$ components has been suggested in [14]. The computational complexity of this method is $O(\rho nm)$ for $\rho \leq \min\{m,n\}$. The incremental nature of the algorithm makes it suitable for analysis of dynamic data, where rows/columns are dynamically added and subtracted from the data matrix. Another interesting approach to reduce the SVD computational cost is matrix sparsification by zeroing out small values in the data matrix. This widely used approach utilizes a sparse eigensolver such as Lanczos to compute the relevant $\rho$ eigen components [21]. If $\rho$ is small in comparison to the matrix size, then the computational complexity of Lanczos is $O\left(\max\{m,n\}^2 \cdot \rho\right)$ [27]. The storage requirements for Lanczos is $O(m)$. Additionally, Lanczos method can be modified to terminate when the smallest estimated eigenvalue is well approximated and its value is lower than a given threshold. More sparsification approaches are given in [47]. Finally, the Nyström extension method [5] provides an additional technique to reduce the SVD computation cost, using a low rank sketch of the data matrix.

Mathematically, suppose that the rank of $A$ is $\rho$, and let $A = USV^*$ be the (thin) SVD of $A$, where $U$ and $V$ are $m \times \rho$ and $n \times \rho$ matrices, respectively, whose columns are orthonormal, and $S$ is a diagonal $\rho \times \rho$ matrix, whose diagonal elements are ordered decreasingly $s_1 \geq \ldots \geq s_\rho \geq 0$. The columns of $U$ and $V$ are referred to as the left and right singular vectors of $A$ respectively, and the diagonal elements of $S$ as its singular values. Then, for any $k \in [\rho]$ we have $\|A - A_k\| \leq \|A - B\|$ for any orthogonally invariant matrix norm and any $m \times n$ matrix $B$ of rank $k$ or less, where $A_k$ is the $k$-SVD of $A$, i.e. $A_k = U_{(:,[k])} S_{([k],[k])} (V_{(:,[k])})^*$. Let $\mathcal{U}$ be the subspace spanned by $U_{(:,[k])}$'s columns, then the $k$-dimensional embedding $\mathbf{F}_k : \mathbb{R}^m \to \mathbb{R}^k$, defined by $\mathbf{F}_k(\mathbf{v}) \triangleq (U_{(:,[k])})^* \mathbf{v}$, is a decomposition of the orthogonal map $\mathbf{O}_\mathcal{S} : \mathbb{R}^m \to \mathbb{R}^m$, $\mathbf{O}_\mathcal{S}(\mathbf{v}) \triangleq (U_{(:,[k])})^* \mathbf{v}$ and the orthogonal projection $\mathbf{W}_\mathcal{U}$. The following lemma quantifies the distortion rate of $\mathbf{F}_k$, applied to $\mathcal{A}$, with respect to the spectrum of $A$, which is encapsulated in $S$.

**Lemma 1.2.** *The $k$-dimensional embedding $\mathbf{F}_k$ is a $2s_{k+1}$-distortion of $\mathcal{A}$.*

*Proof.* From the triangular inequality we have $\|A_{(:,i)} - A_{(:,j)}\| \leq \|\mathbf{W}_\mathcal{U}^\perp(A_{(:,i)})\| + \|\mathbf{W}_\mathcal{U}^\perp(A_{(:,j)})\| + \|\mathbf{W}_\mathcal{U}(A_{(:,i)}) - \mathbf{W}_\mathcal{U}(A_{(:,j)})\|$. Since $\mathbf{W}_\mathcal{U}^\perp = I - \mathbf{W}_\mathcal{U}$, and due to $A$'s SVD, we have $\|\mathbf{W}_\mathcal{U}(A_{(:,i)})\|$, $\|\mathbf{W}_\mathcal{U}(A_{(:,j)})\| < s_{k+1}$. Moreover, since $\mathbf{F}_k$ is an orthogonal map, we have $\|\mathbf{F}_k(A_{(:,i)} - A_{(:,j)})\| \leq \|A_{(:,i)} - A_{(:,j)}\|$. Thus, $|\|A_{(:,i)} - A_{(:,j)}\| - \|\mathbf{F}_k(A_{(:,i)} - A_{(:,j)})\|| \leq 2s_{k+1}$ □

Notice that a particular case of Lemma 1.2 is when $k = \rho$. Then, $\mathbf{F}_k$ embeds $\mathcal{A}$ accurately in $\mathbb{R}^\rho$. The computational and storage complexities of the thin SVD are $O(\rho mn)$ and $O(\max\{m,n\}^2)$, respectively. In addition, the principal subspace $\mathcal{U}$, on which the data is projected, is a mixture of the entire columns set of $A$ which, in terms of data analysis, may be less informative than a dictionary-based subspace.

# 2 Incomplete Pivoted QR-based Data Analysis

In this section, a QR-based data sampling and dimensionality reduction method is suggested, as well as consequent out-of-sample and anomaly detection schemes. The method is geometrically driven, in the sense that the low-dimensional approximation is constructed to constitute a user-defined distortion of the high-dimensional dataset $\mathcal{A}$. This is done using an incomplete pivoted QR decomposition of the data matrix $A$, described in section 2.3.

The suggested method incrementally and simultaneously constructs a low-dimensional subspace and projects the data on it. The basis elements of the constructed subspace are chosen from $\mathcal{A}$. Thus, this method also identifies a subset of representative landmark data points, according to the user-defined distortion parameter. The landmarks subset $\mathcal{D} \subset \mathcal{A}$ is referred to as *dictionary*. The dictionary enables both an efficient out-of-sample extension and anomaly detection for any data point $\mathbf{x} \in \mathbb{R}^m \backslash \mathcal{A}$. In this context $\mathcal{A}$ would be referred to as a *training set*, and each of its elements is considered as normal. The out-of-sample extension is based only on the geometrical relations between $\mathbf{x}$ and the dictionary members, as described in Section 2.5.

Both the computational and the storage costs of the proposed method depend on the dimension of the embedding space. In the worst case, where the dictionary consists of the whole data, these complexities are identical to the corresponding complexities of the thin SVD of the associated data matrix $A$. Moreover, since the proposed algorithm does not use the powers of $AA^*$ (nor $A^*A$), as opposed to classical algorithms for SVD computations [27], there is no necessity to store $A$ in RAM.

There are several methods for practical computation of the QR decomposition. Householder [31], Givenes rotations [26] and Gram-Schmidt or modified Gram-Schmidt [40]. In [33] an incomplete Gram-Schmidt and incomplete Givens transform are utilized to find an incomplete QR decomposition. Another relevant approach is the Rank Revealing QR (RRQR) method [16]. The RRQR can be used for matrix approximation by proper manipulation of the QR output [17]. The proposed pivoted incomplete QR algorithm is one of many methods to compute a partial orthogonal decomposition [4, 39]. Yet, the theoretical basis for our method is valid for any other version of pivoted incomplete QR algorithm.

Robustness of the proposed method to noise is presented in Section 2.2.1 and the resulted matrix approximation is proved in Section 2.2.2.

## 2.1 Mathematical preliminaries

A QR factorization with columns pivoting [27] of an $m \times n$ matrix $A$ of rank $\rho$ is

$$A\Pi = QR, \tag{2.1}$$

where $\Pi$ is an $n \times n$ permutation matrix, $Q$ is an $m \times \rho$ matrix whose columns constitute an orthonormal basis for the columns space of $A$, and $R$ is a $\rho \times n$ upper diagonal matrix. This decomposition represents the Gram-Schmidt process, applied to $A$'s columns one-by-one, due the order determined by $\Pi$. Therefore, for any $k \in [\rho]$ we have

$$\mathcal{A}_{\boldsymbol{\pi}([k])} = \mathcal{Q}_{[k]}, \tag{2.2}$$

and

$$Q_{(:,k)} = \mathbf{W}^{\perp}_{\boldsymbol{\pi}([k-1])}(A_{(:,\boldsymbol{\pi}(k))}) / \left\| \mathbf{W}^{\perp}_{\boldsymbol{\pi}([k-1])}(A_{(:,\boldsymbol{\pi}(k))}) \right\|, \tag{2.3}$$

where $\mathcal{A}_{\mathcal{I}}$ and $\mathcal{Q}_{\mathcal{I}}$ are the subspace spanned by the columns of $A_{(:,\mathcal{I})}$ and $Q_{(:,\mathcal{I})}$, respectively, and $\mathbf{W}_{\boldsymbol{\pi}([k])} : \mathbb{R}^m \to \mathcal{A}_{\boldsymbol{\pi}([k])}$ is the orthogonal projection on $\mathcal{A}_{\boldsymbol{\pi}([k])}$. Equation 2.2 suggests that for any $k \in [\rho]$

$$\mathbf{W}_{\boldsymbol{\pi}([k])}(\mathbf{v}) = Q_{(:,[k])}(Q_{(:,[k])})^*\mathbf{v}, \;\; \mathbf{W}^{\perp}_{\boldsymbol{\pi}([k])}(\mathbf{v}) = \mathbf{v} - \mathbf{W}_{\boldsymbol{\pi}([k])}(\mathbf{v}), \quad \mathbf{v} \in \mathbb{R}^m. \tag{2.4}$$

The presented dimensionality reduction method is based on incomplete pivoted QR decomposition of the data matrix $A$. The criteria for pivoting and incompleteness are based on the following lemma:

**Lemma 2.1.** *Consider Eq. 2.1. Then, for any $k \in [\rho]$,*

$$R_{(k,k)} = \left\| \mathbf{W}^{\perp}_{\boldsymbol{\pi}([k-1])}(A_{(:,\boldsymbol{\pi}(k))}) \right\|. \tag{2.5}$$

*Proof.* Since $\mathbf{W}^{\perp}_{\boldsymbol{\pi}([k-1])}$ is orthogonal projection, then $(\mathbf{W}^{\perp}_{\boldsymbol{\pi}([k-1])})^* \mathbf{W}^{\perp}_{\boldsymbol{\pi}([k-1])} = \mathbf{W}^{\perp}_{\boldsymbol{\pi}([k-1])}$. Therefore,

$$\begin{aligned}
\left\| \mathbf{W}^{\perp}_{\boldsymbol{\pi}([k-1])}(A_{(:,\boldsymbol{\pi}(k))}) \right\|^2 &= (\mathbf{W}^{\perp}_{\boldsymbol{\pi}([k-1])}(A_{(:,\boldsymbol{\pi}(k))}))^* \mathbf{W}^{\perp}_{\boldsymbol{\pi}([k-1])}(A_{(:,\boldsymbol{\pi}(k))}) \\
&= (A_{(:,\boldsymbol{\pi}(k))}))^* \mathbf{W}^{\perp}_{\boldsymbol{\pi}([k-1])}(A_{(:,\boldsymbol{\pi}(k))}) \\
&= \left\| \mathbf{W}^{\perp}_{\boldsymbol{\pi}([k-1])}(A_{(:,\boldsymbol{\pi}(k))}) \right\| (A_{(:,\boldsymbol{\pi}(k))}))^* Q_{(:,[k])} \\
&= \left\| \mathbf{W}^{\perp}_{\boldsymbol{\pi}([k-1])}(A_{(:,\boldsymbol{\pi}(k))}) \right\| R_{(k,k)}.
\end{aligned}$$

□

Lemma 2.2 stresses the recursive relations between $Q$'s columns. This relation will be used in Section 2.5.

**Lemma 2.2.** *For any $k \in [\rho]$, $Q_{(:,k)} = (A_{(:,\boldsymbol{\pi}(k))} - \sum_{i=1}^{k-1} R_{(i,k)} Q_{(:,i)})/R_{(k,k)}$.*

*Proof.* According to Eqs. 2.3 and 2.4, and since $R_{(i,j)} = (Q_{(:,i)})^* A_{(:,\boldsymbol{\pi}(j))}$ for any $i \in [\rho], j \in [n]$, we have $\mathbf{W}_{\boldsymbol{\pi}([k-1])}(A_{(:,\boldsymbol{\pi}(k))}) = \sum_{i=1}^{k-1} R_{(i,k)} Q_{(:,i)}$. Therefore, due to Eq. 2.5, the lemma is proved. □

## 2.2 Incomplete pivoted QR-based dimensionality reduction - theoretical background

The geometry of $A$'s (permuted) columns is isomorphic to the geometry of $R$'s columns, i.e. for any $i, j \in [n]$, $(A_{(:,\boldsymbol{\pi}(i))})^* A_{(:,\boldsymbol{\pi}(j))} = (R_{(:,i)})^* R_{(:,j)}$. Thus, the upper triangularity of $R$ suggests to embed the dataset $\mathcal{A}$ by an incomplete (truncated) version of $R$'s rows. Mathematically, following Eq. 1.3, if we set $\mathcal{S} = \mathcal{A}_{\boldsymbol{\pi}([s])}$ and the orthogonal map $\mathbf{O}_{\mathcal{S}} : \mathbb{R}^m \to \mathbb{R}^s$,

$$\mathbf{O}_{\mathcal{S}}(\mathbf{v}) \triangleq (Q_{(:,[s])})^* \mathbf{v}, \tag{2.6}$$

then, due to the orthogonality of $Q$'s columns and Eqs. 2.1, 2.2 and 2.4, the $s$-dimensional embedding from Eq. 1.3 becomes

$$\mathbf{F}_s(\mathbf{v}) = (Q_{(:,[s])})^* \mathbf{v} \tag{2.7}$$

and specifically,

$$\mathbf{F}_s(A_{(:,\boldsymbol{\pi}(i))}) = R_{([s],i)}, \quad i \in [n]. \tag{2.8}$$

Notice that Eq. 1.2 is satisfied by $\mathbf{O}_{\mathcal{S}}$ from Eq. 2.6. Although this specific choice for $\mathbf{O}_{\mathcal{S}}$ yields $\mathbf{O}_{\mathcal{S}} = \mathbf{F}_s$, this is not always the case since, as aforementioned, $\mathbf{O}_{\mathcal{S}}$ is not unique. For example, in Section 2.3 a different choice of $\mathbf{O}_{\mathcal{S}}$ is presented. The incompleteness of the discussed QR decomposition is reflected in Eq. 2.8, where the $s$-dimensional embedding is defined via only a partial set of $R$'s rows. According to the triangularity of $R$, the geometry of such an embedding is exact on the basis elements of $\mathcal{S}$,

$$\mathcal{D} \triangleq \{A_{(:,\boldsymbol{\pi}(1))}, \ldots, A_{(:,\boldsymbol{\pi}(s))}\}. \tag{2.9}$$

This set is referred to as the *dictionary* of $\mathcal{A}$, and its elements are referred to as *pivots*. As Lemma 1.1 suggests, a careful choice of $\Pi$ might result in a low-dimensional distortion $\mathbf{F}_s$ of $\mathcal{A}$. An algorithm for such a choice is presented in Section 2.3.

We conclude this section with an example that demonstrates the permutation's significance. Let $A$ be the following matrix:

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 20 \end{pmatrix}$$

Then, in order to achieve a 3-distortion for $\Pi = I_7$, we must have $s = \rho = 7$. On the other hand, one can verify that for any permutation that satisfies $\boldsymbol{\pi}([3]) = \{1, 4, 7\}$ $\|\mathbf{W}^{\perp}_{\boldsymbol{\pi}([3])}(A_{(:,i)})\| < 1.5,\ i \in [7]$, which according to Lemma 1.1 is a sufficient condition for a 3-dimensional embedding of $\mathcal{A}$, with distortion rate bounded by 3. Consequently, $F_s$ is an $s$-dimensional $2\mu$-distortion of $\mathcal{A}$, with $s = 3$ and $\mu = 1.5$.

Proposition 2.3, which is a rephrased version of Lemma 1.1 in terms of pivoted incomplete QR, concludes the above discussion:

**Proposition 2.3.** *Let $\mu > 0$. If there exist a permutation $\boldsymbol{\pi}$ and $s \in [\rho]$ for which $\|\mathbf{W}^{\perp}_{\boldsymbol{\pi}([s])}(A_{(:,i)})\| < \mu$ for any $i \in [n]$, then $\mathbf{F}_s$ from Eq. 2.7 is an $s$-dimensional $2\mu$-distortion of $\mathcal{A}$.*

### 2.2.1 Stability to noise

In real-life data may be noisy, and instead of analyzing clean data, stored in $A$, an noisy version $\tilde{A} = A + N$ is analyzed. Here, the matrix $N$ represents an additive noise. Proposition 2.4 shows that a distortion of the noisy data is also a distortion of the original clean data, where the error originated by noise is additive.

**Proposition 2.4.** *Let $A$ be an $m \times n$ data matrix and let $\tilde{A} = A + N$, where $N$ is a noise matrix of the same size of $A$. Assume that $\|N\| \leq \eta$ for some $\eta \geq 0$, and let $\tilde{\mathbf{F}}_s$ be a $2\mu$-distortion of $\tilde{A}$'s columns, as defined in Eq. 2.8. Then, $\tilde{\mathbf{F}}_s$ is a $2(\mu + \eta)$-distortion of $A$'s columns.*

*Proof.* Let $\tilde{\mathbf{F}}_s$ be the map defined in Eq. 1.3, with the corresponding elements $\tilde{\mathcal{S}}$, $\mathbf{O}_{\tilde{\mathcal{S}}}$ and $\mathbf{W}_{\tilde{\mathcal{S}}}$. Then we have

$$\begin{aligned}
\left| \left\| A_{(:,i)} - A_{(:,j)} \right\| - \left\| \tilde{\mathbf{F}}_s(\tilde{A}_{(:,i)}) - \tilde{\mathbf{F}}_s(\tilde{A}_{(:,j)}) \right\| \right| &= \left| \left\| A_{(:,i)} - A_{(:,j)} \right\| - \left\| \mathbf{W}_{\tilde{\mathcal{S}}}(\tilde{A}_{(:,i)}) - \mathbf{W}_{\tilde{\mathcal{S}}}(\tilde{A}_{(:,j)}) \right\| \right| \\
&\leq \left\| A_{(:,i)} - A_{(:,j)} - \mathbf{W}_{\tilde{\mathcal{S}}}(\tilde{A}_{(:,i)}) + \mathbf{W}_{\tilde{\mathcal{S}}}(\tilde{A}_{(:,j)}) \right\| \\
&\leq \left\| A_{(:,i)} - \mathbf{W}_{\tilde{\mathcal{S}}}(\tilde{A}_{(:,i)}) \right\| + \left\| A_{(:,j)} - \mathbf{W}_{\tilde{\mathcal{S}}}(\tilde{A}_{(:,j)}) \right\| \\
&\leq \left\| A_{(:,i)} - \tilde{A}_{(:,i)} \right\| + \left\| \tilde{A}_{(:,i)} - \mathbf{W}_{\tilde{\mathcal{S}}}(\tilde{A}_{(:,i)}) \right\| \\
&+ \left\| A_{(:,j)} - \tilde{A}_{(:,j)} \right\| + \left\| \tilde{A}_{(:,j)} - \mathbf{W}_{\tilde{\mathcal{S}}}(\tilde{A}_{(:,j)}) \right\| \\
&= \left\| N_{(:,i)} \right\| + \left\| \tilde{A}_{(:,i)} - \mathbf{W}_{\tilde{\mathcal{S}}}(\tilde{A}_{(:,i)}) \right\| \\
&+ \left\| N_{(:,j)} \right\| + \left\| \tilde{A}_{(:,j)} - \mathbf{W}_{\tilde{\mathcal{S}}}(\tilde{A}_{(:,j)}) \right\| \\
&\leq 2(\mu + \eta).
\end{aligned}$$

The first equality is due to Eqs. 1.2 and 1.3, and the last inequality is due to the fact that $\|N_{(:,i)}\| \leq \|N\|$, the proposition's assumption and Eq. 1.1.

□

### 2.2.2 Matrix approximation error

The notions of low-rank matrix approximation and geometry-preserving dimensionality reduction are different, yet related. While the $s$-SVD of a data matrix enables an $s$-dimensional embedding of the associated data (as established in Lemma 1.2), it can be shown that the incomplete QR factorization $Q_{(:,[s])}R_{([s],:)}$ of the pivoted version of $A$ (see Eq. 2.1) can be used to form a low-rank approximation of $A$. The operator norm of an arbitrary matrix $M$, denoted by $\|M\|_2$, is equal to its maximal singular value. Thus, as was explained in Section 1.4, it measures the maximal linear trend of the data stored in its columns (or rows). Therefore, the operator norm of the difference of two matrices measures the strength of the maximal linear trend of the associated error. Proposition 2.5 provides a bound for the approximation error of $A\Pi$ by its incomplete QR factorization. Its proof uses the Frobenius norm of a matrix $\|M\|_F \triangleq \sqrt{\sum_{i,j} M_{(i,j)}^2}$ and the norms inequality $\|M\|_2 \leq \|M\|_F$ for any matrix $M$.

**Proposition 2.5.** *Let $\mu$, $\boldsymbol{\pi}$, $s$ and $A$ satisfy the condition of Proposition 2.3, then $\|A\Pi - Q_{(:,[s])}R_{([s],:)}\|_\eta \leq \mu\sqrt{\rho - s}$ for $\eta \in \{2, F\}$.*

*Proof.* Following Eqs. 2.1 and 2.4, and according to the orthonormality of $Q$'s columns, we have $\mathbf{W}_{\boldsymbol{\pi}([s])}(A_{(:,i)}) = Q_{(:,[s])}R_{([s],i)}$ for any $i \in [n]$. On the other hand, due to the triangularity of $R$, $A_{(:,\boldsymbol{\pi}([s]))} = Q_{(:,[s])}R_{([s],[s])}$. Thus, at least $s$ columns of $A - Q_{(:,[s])}R_{(:,[s])}$ are vanishing, and the norms of the rest (mostly) $\rho - s$ are bounded by $\mu$, according to the proposition's assumption. This leads to $\left\|A - Q_{(:,[s])}R_{(:,[s])}\right\|_2 \leq \left\|A - Q_{(:,[s])}R_{(:,[s])}\right\|_F \leq \mu\sqrt{\rho - s}$. □

## 2.3 Incomplete pivoted QR-based dimensionality reduction - implementation

Algorithm 1 iteratively constructs an incomplete pivoted $QR$ version of the data matrix $A$, to obtain a $2\mu$-distortion of $\mathcal{A}$. In its $j$-th iteration, the algorithm selects a pivot $A_{(:,\boldsymbol{\pi}(j))}$, and projects the dataset $\mathcal{A}$ on $\mathcal{A}_{\boldsymbol{\pi}([j])}$. Based on condition 1 from Section 1.3, the $j$-th pivot $A_{(:,\boldsymbol{\pi}(j))}$ is chosen to be the element in $\mathcal{A}$, whose approximation by its orthogonal projection on $\mathcal{A}_{\boldsymbol{\pi}([j-1])}$ is the worst. Thus, the permutation $\boldsymbol{\pi} : [n] \to [n]$ is determined due to the following condition:

$$\boldsymbol{\pi}(j) = \arg\max_{i \in [n] \setminus \boldsymbol{\pi}([j-1])} \left\|\mathbf{W}^{\perp}_{\boldsymbol{\pi}([j-1])}(A_{(:,i)})\right\|. \tag{2.10}$$

Then, the corresponding new column $Q_{(:,j)}$ and row $R_{(j,:)}$ are computed, according to Eqs. 2.1, 2.3 and 2.4. Since the columns permutation is updated in every iteration $j$, the columns of $R_{(:,[j-1])}$ have to be permuted correspondingly. The algorithm terminates when the quantity in Eq. 2.10 is less than $\mu$. Thus $s$, which is the number of iterations required to provide a $2\mu$-distortion, is a non-increasing function of $\mu$, bounded from above by $\rho$, and not known a-priori. When the algorithm ends, the correspondence rule, defined by Eq. 2.8, provides an $s$-dimensional $2\mu$-distortion of $\mathcal{A}$, according to Proposition 2.3. Notice that for $\mu = 0$ the application of Algorithm 1 to $A$ results in a complete pivoted QR factorization.

Let us make a couple of technical remarks concerning Algorithm 1: first, in case of limited computational or storage budgets, Algorithm 1 can be easily modified to make a limited number of iterations $d$ or, equivalently, to provide a $d$-dimensional embedding. In this case the distortion parameter $\mu$ is a non-increasing function of $d$. Secondly, the dictionary $\mathcal{D}$ is chosen regardless to the data indexing order. This property ensures a relatively sparse dictionary, as demonstrated

**Algorithm 1:** Incomplete pivoted QR decomposition (ICPQR)

**Input** : An $m \times n$ data matrix $A$ and a nonnegative distortion parameter $\mu$.
**Output**: An $m \times s$ matrix $Q$ whose columns are orthonormal, an $s \times n$ upper diagonal matrix $R$ and a permutation $\boldsymbol{\pi}$, such that the correspondence rule defined by Eq. 2.8 is a $2\mu$-distortion of $A$'s columns.

Initialization: set $\boldsymbol{\pi} = identity$, $\Pi = I_n$, $\delta > \mu$, and $j = 0$
**while** $\delta > \mu$ **do**
  set $j = j + 1$
  set $i_j = \arg\max_{i \in [n] \setminus \boldsymbol{\pi}([j-1])} \|\mathbf{W}^{\perp}_{\boldsymbol{\pi}([j-1])}(A_{(:,i)})\|$ (see Eq. 2.4)
  `//consider` $\boldsymbol{\pi}([0]) = \emptyset$
  set $\boldsymbol{\Delta} = \mathbf{W}^{\perp}_{\boldsymbol{\pi}([j-1])}(A_{(:,i_j)})$ and $\delta = \|\boldsymbol{\Delta}\|$
  set $Q_{(:,j)} = \boldsymbol{\Delta}/\delta$
  switch $\boldsymbol{\pi}(j) \leftrightarrow \boldsymbol{\pi}(i_j)$, set $\Pi = \Pi\Pi_{j \leftrightarrow i_j}$
  `//`$\Pi_{j \leftrightarrow i_j}$ `is` $I_n$ `with columns` $j, i_j$ `swapped`
  set $R = R\Pi_{j \leftrightarrow i_j}$
  set $R_{(j,:)} = (Q_{(:,j)})^* A\Pi$
**end**
set $s = j$

in Section 4.1.4. Lastly, in order to achieve an optimal[1] coordinates system for the geometry represented by $R$, an SVD can be utilized. Mathematically, let $R = USV^*$ be the SVD decomposition of $R$, where $U$ and $S$ are $s \times s$ orthogonal and diagonal matrices, respectively, and $V$ is an $n \times s$ matrix, whose columns are orthonormal. Then, the map $\hat{\mathbf{O}}_{\mathcal{S}} : \mathbb{R}^m \to \mathbb{R}^s$,

$$\hat{\mathbf{O}}_{\mathcal{S}}(\mathbf{v}) \triangleq (QU)^*\mathbf{v}, \quad \mathbf{v} \in \mathbb{R}^m \tag{2.11}$$

is still isometric on $\mathcal{S}$, as condition 1.2 in Section 1.3 requires. Thus, following Eqs. 2.1, 2.8 and 2.11, the map $\hat{\mathbf{F}}_{\mathbf{s}} : \mathbb{R}^m \to \mathbb{R}^k$

$$\hat{\mathbf{F}}_{\mathbf{s}}(\mathbf{v}) \triangleq \hat{\mathbf{O}}_{\mathcal{S}} \circ \mathbf{W}_{\boldsymbol{\pi}([s])}(\mathbf{v}), \quad \mathbf{v} \in \mathbb{R}^m$$

is still an $s$-dimensional $2\mu$ distortion of $\mathcal{A}$. Moreover, $\hat{\mathbf{F}}_{\mathbf{s}}$ is optimal in the sense that the axes are aligned correspondingly to the variances directions. The computational and storage costs of such an alignment are $O(ns^2)$ and $O(ns)$, respectively. Therefore, the total complexity of Algorithm 1 is not affected by this optional step (see Table 2.4.)

## 2.4 Reduced cost ICPQR

Equation 2.8 suggests that $Q$ is not needed for the low rank embedding of $\mathcal{A}$ by $\mathbf{F}_s$. In this section we present a more efficient version of Algorithm 1, by which the results in Section 4 were obtained. The algorithm produces no $Q$ and applies no physical permutations.

Consider Eq. 2.1 with $\boldsymbol{\pi}$ defined by Eq. 2.10, then

$$A = Q\bar{R}, \quad \bar{R} \triangleq R\Pi^*, \tag{2.12}$$

where $\bar{R}$ is no longer triangular. Algorithm 2 is a translated version of Algorithm 1 to this case, where the permutation $\Pi$ is absorbed in $\bar{R}$. The low-dimensional embedding from Eq. 2.8 then becomes

$$\mathbf{F}_s(A_{(:,i)}) = \bar{R}_{([s],i)}, \quad i \in [n]. \tag{2.13}$$

[1] A coordinates system that is determined by principal components.

For the establishment of Algorithm 2, steps 4 and 9 of Algorithm 1, which are dependent on $Q$, have to be modified: according to Eq. 2.12 $(A_{(:,\boldsymbol{\pi}(i))})^* A_{(:,\boldsymbol{\pi}(j))} = (\bar{R}_{(:,\boldsymbol{\pi}(i))})^* \bar{R}_{(:,\boldsymbol{\pi}(j))} = \sum_{\ell=1}^{\min\{i,j\}} \bar{R}_{(\ell,\boldsymbol{\pi}(i))} \bar{R}_{(\ell,\boldsymbol{\pi}(j))}$, for any $i, j \in [n]$ . The upper limit in the sum is due to the upper triangularity of $R$. Thus, the following recursive relations between the entries of $\bar{R}$ hold:

$$\bar{R}_{(i,\boldsymbol{\pi}(j))} = \begin{cases} \bar{R}_{(i,\boldsymbol{\pi}(i))}^{-1} \left( u_{ij} - \sum_{\ell=1}^{i-1} \bar{R}_{(\ell,\boldsymbol{\pi}(i))} \bar{R}_{(\ell,\boldsymbol{\pi}(j))} \right) & \text{if } i < j \\ \left( u_{ij} - \sum_{\ell=1}^{i-1} (\bar{R}_{(\ell,\boldsymbol{\pi}(i))})^2 \right)^{1/2} & \text{if } i = j \\ 0 & \text{if } i > j \end{cases} , \tag{2.14}$$

where $u_{ij} \triangleq (A_{(:,\boldsymbol{\pi}(i))})^* A_{(:,\boldsymbol{\pi}(j))}$. Moreover, comparing Eq. 2.5 with Eq. 2.14 yields

$$\left\| \mathbf{W}_{\boldsymbol{\pi}([i-1])}^{\perp} (A_{(:,\boldsymbol{\pi}(i))}) \right\| = \left( u_{ii} - \sum_{\ell=1}^{i-1} (\bar{R}_{(\ell,\boldsymbol{\pi}(i))})^2 \right)^{1/2} .$$

Thus, the pivoting criterion from Eq. 2.10 becomes

$$\boldsymbol{\pi}(j) = \arg \max_{i \in [n] \setminus [j-1]} \left( u_{ii} - \sum_{\ell=1}^{i-1} (\bar{R}_{(\ell,\boldsymbol{\pi}(i))})^2 \right)^{1/2} .$$

**Algorithm 2:** Incomplete pivoted Q-less QR decomposition

**Input** : An $m \times n$ matrix $A$ and a nonnegative distortion parameter $\mu$.
**Output**: An $s \times n$ matrix $\bar{R}$ and a permutation $\boldsymbol{\pi}$, for which the embedding defined by Eq. 2.13 is a $2\mu$-distortion of $A$'s columns.

1 Initialization: set $\boldsymbol{\pi} = identity$, $j = 0$, $\delta > \mu^2$, $\mathbf{y} = \mathbf{0}_n$ the $n$-long all zeros vector, and $\mathbf{z} \in \mathbb{R}^n$, for which $\mathbf{z}_{(i)} = \|A_{(:,i)}\|^2$, $i \in [n]$
2 **while** $\delta \geq \mu^2$ **do**
3  set $j = j + 1$
4  set $i_j = \arg\max_{i \in [n] \setminus [j-1]} (\mathbf{z}_{(\boldsymbol{\pi}(i))} - \mathbf{y}_{(\boldsymbol{\pi}(i))})$
5  switch $\boldsymbol{\pi}(j) \leftrightarrow \boldsymbol{\pi}(i_j)$
6  set $\delta = \mathbf{z}_{(\boldsymbol{\pi}(j))} - \mathbf{y}_{(\boldsymbol{\pi}(j))}$
7  for every $i \in [j-1]$ set $\bar{R}_{(j,\boldsymbol{\pi}(i))} = 0$
8  set $\bar{R}_{(j,\boldsymbol{\pi}(j))} = \delta^{1/2}$
9  **for** $i \in [n] \setminus [j]$ **do**
10  set $u_{ji} = (A_{(:,\boldsymbol{\pi}(j))})^* A_{(:,(\boldsymbol{\pi}(i))}$
11  $\bar{R}_{(j,\boldsymbol{\pi}(i))} = (u_{ji} - \sum_{\ell=1}^{j-1} \bar{R}_{(\ell,\boldsymbol{\pi}(j))} \bar{R}_{(\ell,\boldsymbol{\pi}(i))}) / \bar{R}_{(j,\boldsymbol{\pi}(j))}$
12  set $\mathbf{y}_{(\boldsymbol{\pi}(i))} = \mathbf{y}_{(\boldsymbol{\pi}(i))} + \bar{R}_{(j,\boldsymbol{\pi}(i))}$
13  **end**
14 **end**
15 set $s = j$

The resulted dictionary is the set $\mathcal{D}$, as defined in Eq. 2.9.

Table 2.4 presents the computational and storage complexities of Algorithm 2. The storage of the input matrix $A$ has not been taken into account, since due to the nature of Algorithm 2, there is no necessity of its complete storage. For example, the relevant rows and columns of $A$ can be individually computed in each iteration. Therefore, the total storage complexity is smaller than the required storage complexity of the SVD, which is $O(\max\{m,n\}^2)$. The computational complexity of Algorithm 2 is dependent on $\mu$. In the worst case, when $\mu = 0$ and $s = \rho$, the complexity of Algorithm 2 equals to the complexity of the thin rank-$\rho$ SVD. Otherwise, Algorithm 2 is more efficient than SVD.

| Step | Operations | Storage (not including storage of $A$) |
|---|---|---|
| 1 | $O(mn)$ | $O(n)$ |
| 4 | $O(ns - s^2)$ | $O(1)$ |
| 7 | $O(s^2)$ | $O(s^2)$ |
| 10 | $O(m(ns - s^2))$ | $O(m)$ |
| 11 | $O(s(ns - s^2))$ | $O(ns - s^2)$ |
| 12 | $O(ns - s^2)$ | $O(1)$ |
| Total: | $O(mns)$ | $O(ns)$ |

Table 2.1: Computational and storage complexities of Algorithm 2.

## 2.5 Out-of-sample extension and anomaly detection

Two fundamental questions may be naturally raised for an out-of-sample data point $\mathbf{x} \in \mathbb{R}^m \backslash \mathcal{A}$: first, is it normal, related to the training dataset $\mathcal{A}$? and secondly, if it is, then how can the produced low rank embedding be extended to this point? This section addresses these two questions. Since $\mathbf{F}_s$ (Eq. 2.7) is defined for the entire space of $\mathbb{R}^m$, it is used to define an out-of-sample extension of the embedding from Eq. 2.13, for any $\mathbf{x} \in \mathbb{R}^m$ and, based on this, to detect anomalies.

### 2.5.1 Out-of-sample extension

As mentioned above, the out-of-sample extension of the embedding form Eq. 2.13 is simply defined to entire $\mathbb{R}^m$ by Eq. 2.7. As discussed in Section 2.4, since Algorithm 2 produces no $Q$, $\mathbf{F}_s$ cannot be directly applied to an out-of-sample point $\mathbf{x} \in \mathbb{R}^m$. Therefore, a Q-less tool for calculating the out-of-sample extension, as defined in Eq. 2.7, is provided in Algorithm 3, based on the following proposition:

**Proposition 2.6.** *Let $\bar{R}$ be the $s \times n$ matrix produced by Algorithm 2, $\mathbf{x} \in \mathbb{R}^m$, and $\mathbf{f} = \mathbf{F}_s(\mathbf{x}) \in \mathbb{R}^s$, as defined in Eq. 2.7. Then, the following recursive relation holds for the coordinates of $\mathbf{f}$:*

$$\mathbf{f}_{(j)} = (\bar{R}_{(j,\boldsymbol{\pi}(j))})^{-1}(A_{(:,\boldsymbol{\pi}(j))})^*\mathbf{x} - \sum_{i=1}^{j-1} \bar{R}_{(i,\boldsymbol{\pi}(j))}\mathbf{f}_{(i)}, \qquad j \in [s].$$

*Proof.* According to Eq. 2.7, $\mathbf{f}_{(j)} = (Q_{(:,j)})^*\mathbf{x}$. Thus, due to the recursive relations of $Q$'s columns, as presented in Lemma 2.2, we have

$$\begin{aligned}
\mathbf{f}_{(j)} &= (R_{(j,j)})^{-1}((A_{(:,\boldsymbol{\pi}(j))})^*\mathbf{x} - \sum_{i=1}^{j-1} R_{(i,j)}(Q_{(:,i)})^*\mathbf{x}) \\
&= (R_{(j,j)})^{-1}((A_{(:,\boldsymbol{\pi}(j))})^*\mathbf{x} - \sum_{i=1}^{j-1} R_{(i,j)}\mathbf{f}_{(i)}) \\
&= (\bar{R}_{(j,\boldsymbol{\pi}(j))})^{-1}((A_{(:,\boldsymbol{\pi}(j))})^*\mathbf{x} - \sum_{i=1}^{j-1} \bar{R}_{(i,\boldsymbol{\pi}(j))}\mathbf{f}_{(i)}),
\end{aligned}$$

where the last equality is due to Eq. 2.12. □

Since the out-of-sample extension of a data point $\mathbf{x} \in \mathbb{R}^m$ is its orthogonal projection on the $s$-dimensional dictionary subspace $\mathcal{S}$, the only required information are the geometric relations between $\mathbf{x}$ and the elements of the dictionary $\mathcal{D}$ (Eq. 2.9), as Proposition 2.6 shows. Algorithm 3 summarized the above.

**Algorithm 3:** Out-of-sample extension for incomplete pivoted Q-less QR decomposition

**Input** : Dictionary $\mathcal{D} = \{\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(s)}\}$ (see Eq. 2.9) and $s \times n$ matrix $\bar{R}$, which are the outputs of Algorithm 2, and vector $\mathbf{v} \in \mathbb{R}^m$.

**Output**: $\mathbf{F}_s(\mathbf{v})$, as defined by Eq. 2.7.

Initialization: set $\mathbf{f} \in \mathbb{R}^s$ to be vector of all-zeros, except of the first coordinate, $\mathbf{f}_{(1)} = (\bar{R}_{(1,\boldsymbol{\pi}(1))})^{-1}(\mathbf{b}^{(1)})^* \cdot \mathbf{v}$.

**for** $j = 2 : s$ **do**

  set the $j$-th coordinate of $\mathbf{f}$, $\mathbf{f}_{(j)} = (\bar{R}_{(j,\boldsymbol{\pi}(j))})^{-1}(\mathbf{b}^{(j)})^* \cdot \mathbf{v} - \sum_{i=1}^{j-1} \bar{R}_{(i,\boldsymbol{\pi}(j))}\mathbf{f}_{(i)}$

**end**

set $\mathbf{F}_s(\mathbf{v}) = \mathbf{f}$

### 2.5.2 Anomaly detection

The $2\mu$-embedding subspace $\mathcal{S} \subset \mathbb{R}^m$ satisfies $\boldsymbol{\mu}(\mathbf{a}) \leq \mu$ for any $\mathbf{a} \in \mathcal{A}$, where the distortion rate function $\boldsymbol{\mu} : \mathbb{R}^m \to \mathbb{R}$, is defined to be $\boldsymbol{\mu}(\mathbf{x}) \triangleq \left\|\mathbf{W}_{\mathcal{S}}^{\perp}(\mathbf{x})\right\|$ (see discussion in Section 1.3.) Once the out-of-sample extension of $\mathbf{F}_s(\mathbf{x})$ was computed by Algorithm 3, the distortion rate of $\mathbf{x} \in \mathbb{R}^m$ can be easily calculated: $\boldsymbol{\mu}^2(\mathbf{x}) = \|\mathbf{x}\|^2 - \|\mathbf{W}_{\mathcal{S}}(\mathbf{x})\|^2 = \|\mathbf{x}\|^2 - \|\mathbf{O}_{\mathcal{S}}\mathbf{W}_{\mathcal{S}}(\mathbf{x})\|^2 = \|\mathbf{x}\|^2 - \|\mathbf{F}_s(\mathbf{x})\|^2$. The first equality is due to the fact that $\mathbf{W}_{\mathcal{S}}$ is orthogonal projection. The second is due to condition 1.2 that suggests that $\|\mathbf{O}_{\mathcal{S}} \circ \mathbf{W}_{\mathcal{S}}(\mathbf{x})\| = \|\mathbf{W}_{\mathcal{S}}(\mathbf{x})\|$ for any $\mathbf{x} \in \mathbb{R}^m$ and the last equality is due to Eq. 2.7. Consequently, we rephrase the distortion rate function

$$\boldsymbol{\mu}(\mathbf{x}) \triangleq (\|\mathbf{x}\|^2 - \|\mathbf{F}_s(\mathbf{x})\|^2)^{1/2}. \tag{2.15}$$

The following general definition for normality of data points will serve us in two main forms in the rest of the paper.

**Definition 2.1** ($\kappa$-normality)**.** *Let $\mathbf{F}_s : \mathcal{A} \to \mathbb{R}^m$ be an $s$-dimensional embedding of $\mathcal{A} \subset \mathbb{R}^m$, computed by Algorithm 2, and let $\mathbf{F}_s(\mathbf{x}) \in \mathbb{R}^s$ be its extension to $\mathbf{x} \in \mathbb{R}^m$, produced by Algorithm 3. Then, $\mathbf{x}$ is considered as a $\kappa$-normal point relatively to $\mathcal{A}$ if $\boldsymbol{\mu}(\mathbf{x}) \leq \kappa$. Otherwise, it is considered $\kappa$-abnormal.*

Notice that due to Definition 2.1, all the data points in $\mathcal{A}$ are $\mu$-normal. Nevertheless, $\mu$ is not necessarily the minimal $\kappa$ for which $\mathcal{A}$ is considered as $\kappa$-normal, as the most strict $\kappa$ for which $\mathcal{A}$ is still $\kappa$-normal, is $\kappa = \mu_{strict}$, where

$$\mu_{strict} \triangleq \sup_{\mathbf{a} \in \mathcal{A}} \boldsymbol{\mu}(\mathbf{a}).$$

We conclude this section with a definition of two variants of $\kappa$-normality, as defined in Definition 2.1.

**Definition 2.2** (normality and strict normality)**.** *Let $\mathbf{F}_s : \mathcal{A} \to \mathbb{R}^m$ be an $s$-dimensional embedding of $\mathcal{A} \subset \mathbb{R}^m$, computed by Algorithm 2, with distortion parameter $\mu$, and let $\mathbf{F}_s(\mathbf{x}) \in \mathbb{R}^s$ be its extension to $\mathbf{x} \in \mathbb{R}^m$, produced by Algorithm 3. Then, $\mathbf{x}$ is considered as a normal point relatively to $\mathcal{A}$ if $\boldsymbol{\mu}(\mathbf{x}) \leq \mu$, and as a strictly normal point if $\boldsymbol{\mu}(\mathbf{x}) \leq \mu_{strict}$. Otherwise, it is considered as a (strictly) abnormal.*

Obviously, all the data points in $\mathcal{A}$ are strictly-normal and any strictly normal point is also a normal point.

# 3 QR-based Diffusion Maps

Although the QR-based dimensionality reduction method that was presented in Section 2 is designated for parametric data analysis, this section presents a utilization of our method for the Diffusion Maps (DM) [20], which is a graph Laplacian based method for analysis of non-parametric data, via exploration of a Markov chain defined on the data. It is mainly utilized for clustering and manifold learning. Typically, application of DM involves a kernel PCA, which is computationally prohibitive to large amount of data.

## 3.1 DM framework - overview

### 3.1.1 Diffusion geometry

Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a dataset and let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric point-wise positive kernel that defines a connected undirected weighted graph over $\mathcal{X}$. Then, a Markov process over $\mathcal{X}$ can be defined, using the $n \times n$ row-stochastic transition probabilities matrix

$$P = D^{-1}K, \tag{3.1}$$

where $K_{(i,j)} = k(x_i, x_j),\ i, j \in [n]$ and $D$ is a diagonal matrix with diagonal $D_{(i,i)} = \mathbf{d}_{(i)}$, where

$$\mathbf{d}_{(i)} \triangleq \sum_{j=1}^{n} K_{(i,j)}, \quad i \in [n].$$

The vector $\mathbf{d} \in \mathbb{R}^n$ is referred to as the *degrees function* or *degrees vector* of the graph. The associated time-homogeneous Markov chain is defined as follows: for any two time points $t, t_0 \in \mathbb{N}$, $\mathbb{P}(x(t + t_0) = x_j | x(t_0) = x_i) = (P^t)_{(i,j)}$. Assuming that the defined Markov chain is aperiodic (for example, if there is $x \in \mathcal{X}$, for which $k(x, x) > 0$), then it has a unique stationary distribution $\hat{\mathbf{d}} \in \mathbb{R}^n$ which is the steady state of the process, i.e. $\hat{\mathbf{d}}_{(j)} = \lim_{t\to\infty} (P^t)_{(i,j)}$, regardless to the initial point $x_i$. This steady state is the probability distribution resulted from $\ell_1$ normalization of the degrees function $\mathbf{d}$, i.e.,

$$\hat{\mathbf{d}} = \mathbf{d}/\|\mathbf{d}\|_1. \tag{3.2}$$

The *diffusion distance* in time $t \in \mathbb{N}$ is defined by the metric $\mathbf{D}^{(t)} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

$$\mathbf{D}^{(t)}(x_i, x_j) \triangleq \left\| (P^t)_{(i,:)} - (P^t)_{(j,:)} \right\|_{\ell^2(\hat{\mathbf{d}}^{-1})}, \quad i, j \in [n]. \tag{3.3}$$

By definition, $(P^t)_{(i,:)}$ is the probability distribution over $\mathcal{X}$ after $t$ time steps, where the initial state is $x_i$. Therefore, the diffusion distance from Eq. 3.3 measures the difference between two propagations along $t$ time steps, one originated in $x_i$ and the other in $x_j$. Weighing the metric by the inverse of the steady state results in ascribing high weight for similar probabilities on rare states and vice versa.

Due to the above interpretation, the diffusion distances are naturally utilized for multiscale clustering, as they uncover the connectivity properties of the graph across time. In [15, 20] it was proved that under some conditions, if $\mathcal{X}$ is sampled from a low intrinsic dimensional manifold then, as $n$ tends to infinity, the Markov chain converges to a diffusion process over that manifold.

### 3.1.2 Diffusion maps - low rank representation of the diffusion geometry

Diffusion maps [20] are family of Euclidean representations of the diffusion geometry of $\mathcal{X}$ in different time steps, where the Euclidean distances approximate the diffusion distances from Eq. 3.3. Let $G^{(t)}$ be the $n \times n$ matrix, defined by

$$G^{(t)} \triangleq \|\mathbf{d}\|_1^{1/2} D^{-1/2}(P^*)^t,\ t \in \mathbb{N}. \tag{3.4}$$

Then, due to Eqs. 3.1-3.3, the Euclidean $n$-dimensional geometry of $G^{(t)}$'s column is isomorphic to the diffusion geometry of the associated data points i.e.,

$$\mathbf{D}^{(t)}(x_i, x_j) = \left\| {G^{(t)}}_{(:,i)} - {G^{(t)}}_{(:,j)} \right\|, \quad i, j \in [n]. \tag{3.5}$$

Although embedding of the dataset $\mathcal{X}$ in $\mathbb{R}^n$ by the columns of $G^{(t)}$ preserves the diffusion geometry, it may be ineffective for large $n$ due to the 'curse of dimensionality', as was explained in Section 1. Therefore, a dimensionality reduction is required.

Since the transition probabilities matrix $P$ (Eq. 3.1) is conjugated to the symmetric matrix

$$M \triangleq D^{-1/2} K D^{-1/2}$$

via the relation $P = D^{-1/2} M D^{1/2}$, $P$ has a complete real eigen-system. Moreover, due to Gershgorin's circle theorem [27] and the fact that $P$ is stochastic, all its eigenvalues are lying in the interval $(-1, 1]$ (the exclusion of $-1$ is due to the assumption that the chain is aperiodic.) Let

$$M = USU^* \tag{3.6}$$

be the eigen-decomposition of $M$, where $U$ is an orthogonal $n \times n$ matrix and $S$ is a diagonal $n \times n$ matrix, whose diagonal elements $s_i$ are ordered decreasingly, due to their modulus $1 = s_1 > |s_2| \geq \ldots \geq |s_n| \geq 0$. The first inequality is due to the assumption that the graph is connected. Thus, following Eq. 3.4

$$G^{(t)} = (M^*)^t D^{-1/2} = US^t U^* D^{-1/2}. \tag{3.7}$$

Therefore, due to the orthogonality of $U$ and according to Eq. 3.5

$$\mathbf{D}^{(t)}(x_i, x_j) = \|\mathbf{d}\|_1^{1/2} \left\| S^t U^* D^{-1/2} (\mathbf{e}^{(i)} - \mathbf{e}^{(j)}) \right\|, \quad i, j, \in [n],$$

where $\mathbf{e}^{(i)}$ denotes the $i$-th standard unit vector in $\mathbb{R}^n$. As a consequence, the diffusion maps $\mathbf{\Psi}^{(t)} : \mathcal{X} \to \mathbb{R}^n$, are defined as follows:

$$\begin{aligned} \mathbf{\Psi}^{(t)}(x_i) &\triangleq \|\mathbf{d}\|_1^{1/2} U^* G^{(t)} \mathbf{e}^{(i)} \\ &= \|\mathbf{d}\|_1^{1/2} S^t U^* D^{-1/2} \mathbf{e}^{(i)} \\ &= \|\mathbf{d}\|_1^{1/2} \mathbf{d}_{(i)}^{-1/2} S^t (U_{(i,:)})^* \\ &= \hat{\mathbf{d}}_{(i)}^{-1/2} [s_1^t U_{(i,1)}, \ldots, s_n^t U_{(i,n)}]^*, \end{aligned} \tag{3.8}$$

Of course, the diffusion maps provide the required embedding of $\mathcal{X}$ in $\mathbb{R}^n$, since their Euclidean geometry in $\mathbb{R}^n$ is identical to the diffusion geometry of the dataset $\mathcal{X}$, i.e. $\mathbf{D}^{(t)}(x_i, x_j) = \left\| \mathbf{\Psi}^{(t)}(x_i) - \mathbf{\Psi}^{(t)}(x_j) \right\|,\ i, j \in [n]$. In order to achieve a low-dimensional embedding, the diffusion map from Eq. 3.8 is projected onto its significant principal components, according to the decay rate of the spectrum of $M^t$. Specifically, for sufficiently small $|s_{k+1}|^t$, the $k$-dimensional embedding is $\mathbf{T}_k \circ \mathbf{\Psi}_t$, where $\mathbf{T}_k : \mathbb{R}^n \to \mathbb{R}^k$ is the projection on the first $k$ coordinates. Lemma 3.1 quantifies the distortion resulted by such a projection.

**Lemma 3.1.** *Let* $c = \max_{i\in[n]} \hat{\mathbf{d}}_{(i)}^{-1/2}$ *(see Eq. 3.2). Then the* $k$*-dimensional embedding* $\mathbf{T}_k \circ \mathbf{\Psi}^{(t)}$ *is a* $\sqrt{2}c\,|s_{k+1}|^t$*-distortion of the* $n$*-dimensional diffusion map* $\mathbf{\Psi}^{(t)}$ *from Eq. 3.8.*

*Proof.* Following Eq. 3.8 we get[2]

$$\begin{aligned}\left|\left\|\mathbf{\Psi}^{(t)}(x_i) - \mathbf{\Psi}^{(t)}(x_j)\right\| - \left\|\mathbf{T}_k \circ \mathbf{\Psi}^{(t)}(x_i) - \mathbf{T}_k \circ \mathbf{\Psi}^{(t)}(x_j)\right\|\right| &\leq \left\|\mathbf{\Psi}^{(t)}(x_i) - \mathbf{T}_k \circ \mathbf{\Psi}^{(t)}(x_i)\right\| \\ &+ \left\|\mathbf{\Psi}^{(t)}(x_j) - \mathbf{T}_k \circ \mathbf{\Psi}^{(t)}(x_j)\right\| \\ &\leq \left|\mathbf{s}_{(k+1)}\right|^t (\hat{\mathbf{d}}_{(i)}^{-1} + \hat{\mathbf{d}}_{(j)}^{-1})^{1/2} \\ &\leq \sqrt{2}c\left|\mathbf{s}_{(k+1)}\right|^t.\end{aligned}$$

□

The distortion bound from Lemma 3.1 is referred to as *analytic bound*. In many cases, spectral properties of the utilized kernel are known a-priori, with no need of its explicit computation. The Gaussian kernel it just one example (see [8, 9]), but not the only. In such cases, only a partial SVD can be calculated, to produce the relevant principal components, according to the required distortion.

## 3.2 Efficient ICPQR-based DM framework for data analysis

In this section we provide a QR-based framework for low-dimensional representation of the DM for a training set $\mathcal{X}$, its out-of-sample extension, and anomaly detection. For this purpose, $\mathcal{X}$ is assumed to be a subset of $\bar{\mathcal{X}}$, on which a symmetric point-wise positive kernel $k : \bar{\mathcal{X}} \times \bar{\mathcal{X}} \to \mathbb{R}$ is defined.

### 3.2.1 QR-based low-dimensional embedding

Equation 3.5 suggests that the diffusion geometry is already embodied in the Euclidean geometry of $G^{(t)}$'s columns $\mathcal{G}^{(t)} \triangleq \{(G^{(t)})_{(:,1)}, \ldots, (G^{(t)})_{(:,n)}\}$ (see Eq. 3.4.) According to Proposition 2.3 and Eq. 3.5, application of Algorithm 2 to $G^{(t)}$ with distortion rate $\mu > 0$ produces an $s$-dimensional $2\mu$-distortion $\mathbf{F}_s : \mathcal{G}^{(t)} \to \mathbb{R}^s$, for which

$$\max_{i,j\in[n]} \left|\mathbf{D}^{(t)}(x_i, x_j) - \left\|\mathbf{h}_s^{(t)}(x_i) - \mathbf{h}_s^{(t)}(x_j)\right\|\right| \leq 2\mu,$$

where $\mathbf{h}_s^{(t)} : \mathcal{X} \to \mathbb{R}^s$ is defined by

$$\mathbf{h}_s^{(t)}(x_i) \triangleq \mathbf{F}_s((G^{(t)})_{(:,i)}), \quad i \in [n]. \tag{3.9}$$

As was discussed in Section 2.3, the embedding dimension $s$ is not known a-priori, and is a non-increasing function of $\mu$. Algorithm 4 summarizes the above.

The output parameters $\bar{R}$ and $\mathbf{d}$ of Algorithm 4 are needed for the out-of-sample phase, described in Section 3.2.2. In addition, for anomaly detection due to Definition 2.2, in the DM context $\mu_{strict}$ take the form

$$\mu_{strict}^{(t)} = \sup_{i\in[n]} \left\|(G^{(t)})_{(:,i)} - \mathbf{h}_s^{(t)}(x_i)\right\|. \tag{3.10}$$

[2]Here, for comparison purposes, we use the convention that $\mathbf{T}_k : \mathbb{R}^n \to \mathbb{R}^n$, is the operator that zeros out the last $n - k$ coordinates.

**Algorithm 4:** ICPQR-based DM

**Input** : An $n \times n$ kernel matrix $K$, time step $t \in \mathbb{N}$, and a nonnegative distortion parameter $\mu$.

**Output**: An $s$-dimensional $2\mu$-distortion $\mathbf{h}_s^{(t)} : \mathcal{X} \to \mathbb{R}^s$ of $\mathbf{\Psi}^{(t)}$, a dictionary of $s$ elements $\mathcal{D} \subset \mathcal{G}^{(t)}$, an $s \times n$ matrix $\bar{R}$ and a degrees vector $\mathbf{d} \in \mathbb{R}^n$.

1 set $\mathbf{d} \in \mathbb{R}^n$, $\mathbf{d}_{(i)} = \sum_{j=1}^n K_{(i,j)}$

2 set the $n \times n$ diagonal matrix $D$, whose $i$-th diagonal element is $\mathbf{d}_{(i)}$

3 set the $n \times n$ row stochastic transition probabilities matrix in time $t$, $P^t = (D^{-1}K)^t$

4 set $G^{(t)} = \|\mathbf{d}\|_1^{1/2} D^{-1/2}(P^t)^*$ (see Eq. 3.4)

5 apply Algorithm 2 to $G^{(t)}$ and $\mu$, to get a permutation $\boldsymbol{\pi} : [n] \to [n]$ and an $s \times n$ matrix $\bar{R}$

6 define $\mathcal{D} = \{(G^{(t)})_{(:,\boldsymbol{\pi}(1))}, \ldots, (G^{(t)})_{(:,\boldsymbol{\pi}(s))}\}$ and $\mathbf{h}_s^{(t)}(x_i) = \bar{R}_{(:,i)},\ i \in [n]$.

### 3.2.2 Out-of-sample extension and anomaly detection

Given an out-of-sample data point $x \in \bar{\mathcal{X}} \backslash \mathcal{X}$, the goal of the present section is to provide an extension of $\mathbf{h}_s^{(t)}$ from Eq. 3.9 to $x$.

For this purpose, a user-defined probabilities vector $\mathbf{p}^{(t)}(x) \in \mathbb{R}^n$ has to be defined. The $i$-th entry of $\mathbf{p}^{(t)}(x)$ defines the transition probabilities from $x$ to $x_i \in \mathcal{X}$ in $t$ time steps, i.e. $\mathbf{p}^{(t)}(x)_i = \mathbb{P}(x(t) = x_i | x(0) = x),\ i \in [n]$. Consequently, consistently with Eq. 3.4, the $n$ dimensional extension of $\mathcal{G}^{(t)}$ to $x$ is defined by $\mathbf{g}^{(t)}(x) \triangleq \|\mathbf{d}\|_1^{1/2} D^{-1/2}\mathbf{p}^{(t)}(x) \in \mathbb{R}^n$. Then, Algorithm 3 is applied to $\mathbf{g}^{(t)}(x)$, to produce an $s$-dimensional embedding $\mathbf{h}_s^{(t)}(x) = \mathbf{F}_s(\mathbf{g}^{(t)}(x))$.

This scheme is consistent with the low-dimensional embedding scheme, described in Section 3.2.1, in the sense that if the probabilities vector $\mathbf{p}^{(t)}(x)$ equals to an in-sample probabilities vector $(P^*)_{(:,i)}$ for a certain $i \in [n]$, then $\mathbf{h}_s^{(t)}(x) = \mathbf{h}_s^{(t)}(x_i)$. Algorithm 5 summarizes the above.

**Algorithm 5:** Out-of-sample extension for ICPQR-based DM

**Input** : A dictionary $\mathcal{D}$, an $s \times n$ matrix $\bar{R}$ and a degrees vector $\mathbf{d} \in \mathbb{R}^n$, which are the outputs of Algorithm 4, and a transition probabilities vector $\mathbf{p}^{(t)}(x) \in \mathbb{R}^n$.

**Output**: The extension of $\mathbf{h}_s^{(t)}$ to $x$, $\mathbf{h}_s^{(t)}(x)$ and the associated distortion rate $\boldsymbol{\mu}(x)$.

1 set $\mathbf{g}^{(t)}(x) = \|\mathbf{d}\|_1^{1/2} D^{-1/2}\mathbf{p}^{(t)}(x)$, where $D$ is the diagonal $n \times n$ matrix $diag(\mathbf{d})$

2 apply Algorithm 3 to $\mathcal{D}$, $\bar{R}$ and $\mathbf{g}^{(t)}(x)$, to get $\mathbf{h}_s^{(t)}(x) \in \mathbb{R}^s$

3 define $\boldsymbol{\mu}(x) = (\|\mathbf{g}^{(t)}(x)\|^2 - \|\mathbf{h}_s^{(t)}(x)\|^2)^{1/2}$

One possibility for the definition of $\mathbf{p}^{(1)}(x)$, which is the first time step transfer probabilities from an out-of-sample data point to $\mathcal{X}$, is via the kernel function $k$ by

$$\mathbf{p}^{(1)}(x)_{(i)} \triangleq k(x, x_i) / \sum_{j=1}^n k(x, x_j),\ i \in [n]. \tag{3.11}$$

This definition is consistent with Eq. 3.1. Then, the corresponding transition probabilities vector in time step $t$ can be heuristically defined by $\mathbf{p}^{(t)} = (P^*)^{t-1}\mathbf{p}^{(1)}(x)$. This definition represents a Markovian process for which the new data point $x$ is inaccessible from the dataset $\mathcal{X}$, and the transition probabilities from $x$ to $\mathcal{X}$ in time-step $t$ are determined by the transition probabilities from $x$ to $\mathcal{X}$ in the first time-step, represented by $\mathbf{p}^{(1)}(x)$, and the transition probabilities among the elements of $\mathcal{X}$ after $t-1$ time-steps, represented by $(P^*)^{t-1}$.

Finally, the (strict) normality of an out-of-sample data point $x$ is determined due to Definition 2.2 and Eq. 3.10, using the distortion rate function $\boldsymbol{\mu}(x)$ from step 3 of Algorithm 5.

# 4 Experimental Results

This section demonstrates analyses of three different datasets, both synthetic and real, using the proposed methodologies from Sections 2 and 3. Section 4.1 exemplifies the basic notions of geometry preservation, anomaly detection and out-of-sample extension through the application of the QR-based DM, as described in Section 3, to a synthetic dataset. A comparison with the method proposed in [42] for diffusion geometry preservation is presented in this section as well. A QR-based DM analysis of real data is demonstrated in Section 4.2. The analyses in both of the above examples are based on the corresponding first time step DM. Finally, Section 4.3 presents a multiclass classification of parametric data, using generalizations of the out-of-sample extension and anomaly detection methods, presented in Section 2.5.

## 4.1 QR-based DM analysis - toy example

In this section we present a diffusion-based analysis of a synthetic two dimensional manifold, immersed in a three dimensional Euclidean space. The analyzed dataset $\mathcal{X} \subset \mathbb{R}^3$ consists of $n = 3,000$ data points, uniformly sampled from a Swiss roll, shown in Fig. 4.1.

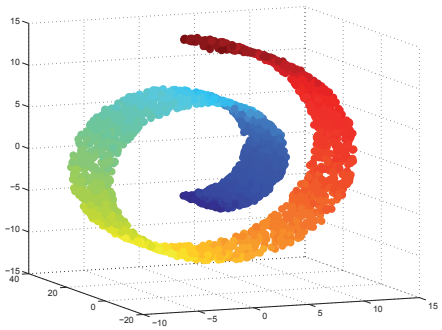



Figure 4.1: Swiss roll consists of $3,000$ uniformly distributed points. Data points are colored according to their distance from the origin.

The utilized kernel function is the commonly-used Gaussian kernel $k_\varepsilon : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$,

$$k_\varepsilon(x, y) \triangleq e^{-\|x-y\|^2/\varepsilon}, \quad \varepsilon > 0, \tag{4.1}$$

where the norm appears in the exponent is the standard three dimensional Euclidean norm. The associated $n \times n$ kernel matrix is $K_\varepsilon$, whose $(i, j)$-th entry is

$$(K_\varepsilon)_{(i,j)} \triangleq k_\varepsilon(x_i, x_j), \quad i, j \in [n].$$

The corresponding degrees function is $\mathbf{d}_\varepsilon \in \mathbb{R}^n$, whose $i$-th entry is $(\mathbf{d}_\varepsilon)_{(i)} \triangleq \sum_{j=1}^n k_\varepsilon(x_i, x_j)$, $i \in [n]$ and $D_\varepsilon$ is the $n \times n$ diagonal matrix, whose $i$-th diagonal entry is $(\mathbf{d}_\varepsilon)_{(i)}$. Based on these, according to Eq. 3.1, the $n \times n$ transition probabilities matrix is defined to be

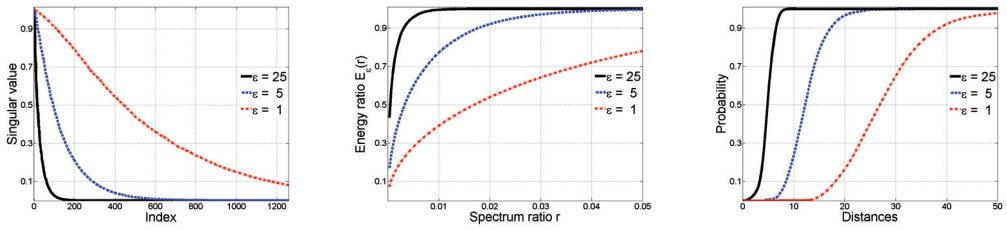
$$P_\varepsilon \triangleq D_\varepsilon^{-1} K_\varepsilon. \tag{4.2}$$

Thus, transition probabilities between close data points are high, and low for far points.

Section 4.1.1 addresses the qualitative dependency between the neighborhood parameter $\varepsilon$, and the required embedding's dimensionality. Section 4.1.2 shows a further step of dimensionality reduction, using the optimal coordinates system, as described in Section 2.3. Out-of-sample extension and anomaly detection, as described in Section 3.2.2, are demonstrated in Section 4.1.3. Finally, Section 4.1.4 presents a brief description of the $\mu$-IDM method [42] and performances comparison with our method.

### 4.1.1 The dependency between $\varepsilon$ and the embedding's dimension $s$

As was proved in [8] as $\varepsilon$ increases, the numerical rank of $P_\varepsilon$ decreases and vice versa. Mathematically, let $1 = s_1^{(\varepsilon)} \geq s_2^{(\varepsilon)} \geq \ldots \geq s_n^{(\varepsilon)} \geq 0$ be the eigenvalues[3] of $P_\varepsilon$ and define $\mathbf{E}_\varepsilon(r) : [0,1] \to [0,1]$, $\mathbf{E}_\varepsilon(r) \triangleq (\sum_{i=1}^{t} (s_i^{(\varepsilon)})^2 / \sum_{i=1}^{n} (s_i^{(\varepsilon)})^2)^{1/2}$ be the energy's portion of $P_\varepsilon$, captured by the first $t$ eigenvalues of $P_\varepsilon$, where $r = t/n$ is the corresponding spectrum ratio. Then, as $\varepsilon$ increases the number of significant eigen-components decreases, as demonstrated in Fig. 4.2. This fact, combined with Lemma 3.1, suggests that as $\varepsilon$ decreases, the number of components required to achieve a certain distortion increases, as Fig. 4.3 demonstrates.

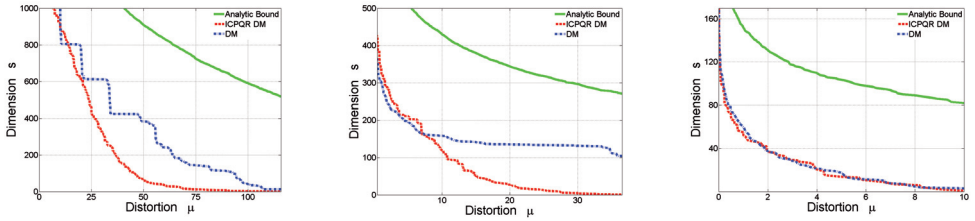



(a) Spectra of $P_\varepsilon$ (b) Spectra ratios of $P_\varepsilon$ (c) Diffusion distances distributions

Figure 4.2: Spectral and geometrical views of three diffusion geometries, corresponding to the neighborhood parameters $\varepsilon = 1, 5$, and 25. Figure (a) shows that the larger $\varepsilon$, the faster the spectrum decays. An immediate consequence is shown in Fig. (b) that shows relation between the number of significant components and $\varepsilon$. Figure (c) shows the probability distribution of the diffusion distances. It is clear that large $\epsilon$ results in many short diffusion distances and vice-versa.

Figure 4.3 compares between the analytic bound (see Lemma 3.1), the minimal dimension of DM and the QR-based DM dimension that are required to achieve a certain distortion. It also demonstrates the above discussed relation between the neighborhood parameter $\varepsilon$ and the dimensionality of the embedding. Thus, the larger $\varepsilon$ the the fewer dimensions that are required to achieve a certain distortion.

[3]The eigenvalues of $P_\varepsilon$ are nonnegative, as the Gaussian kernel function $k_\varepsilon$ from Eq. 4.1 is positive definite, due to Bochner's theorem [49]. Thus, if the data points in $\mathcal{X}$ are all distinct, then $k_\varepsilon$ is strictly positive definite, and the eigenvalues of $P_\varepsilon$ are all positive.

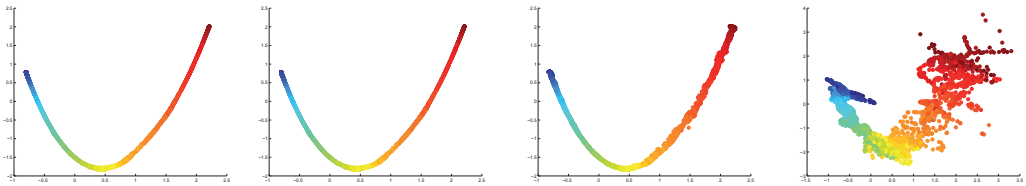



(a) Small neighborhood, $\varepsilon = 1$ (b) Medium neighborhood, $\varepsilon = 5$ (c) Large neighborhood, $\varepsilon = 25$

Figure 4.3: Number of components (dimensions, $y$-axis) required to preserve the Swiss roll diffusion geometry up to a distortion ($x$-axis) for three different neighborhood sizes. The continuous (green) graph denotes the analytic bound, provided by Lemma 3.1, the dashed (red) graph is the QR-based DM dimension, produced by Algorithm 2, and the dash-dotted (blue) graph is the minimal DM dimension, required to achieve a certain distortion.

### 4.1.2 Low-dimensional embedding

In this section a comparison of the classic DM and ICPQR-based DM is presented. Figure 4.4 shows the two-dimensional DM embedding of $\mathcal{X}$, and a two-dimensional view of an aligned versions of ICPQR-based DM, applied to $\mathcal{X}$ with three different distortion values.



(a) DM (b) ICPQR-DM, $\mu = 0.1$ (c) ICPQR-DM, $\mu = 1$ (d) ICPQR-DM, $\mu = 5$

Figure 4.4: Comparison of two-dimensional DM and projected aligned ICPQR-based DM of $\mathcal{X}$, with $\varepsilon = 3$, for the first diffusion time-step: (a) two most significant DM coordinates of $\boldsymbol{\Psi}^{(1)}$. (b)-(d): two most significant coordinates of the aligned version of $\mathbf{h}_s^{(1)}$, with (b) $\mu = 0.1$, $s = 1,246$, actual distortion 0.01, (c) $\mu = 1$, $s = 752$, actual distortion 0.23, and (d) $\mu = 5$, $s = 382$, actual distortion 3.91. Data coloring is consistent with Fig. 4.1.

It should be stressed that $\mathbf{h}_s^{(1)}$, the output of Algorithm 4, is an $s$-dimensional $2\mu$-distortion of the $n$-dimensional DM $\boldsymbol{\Psi}^{(1)}$, as defined in Definition 1.1. Therefore, it is unlikely that low-dimensional projections (lower than $s$) would represent similar geometries. Nevertheless, to demonstrate the notion of low-rate distortion, Fig. 4.4 shows a two dimensional view of an aligned version of $\boldsymbol{\Psi}^{(1)}$ with $\boldsymbol{\Psi}^{(1)}$. For that purpose, the DM of $\mathcal{X}$ was explicitly computed. The utilized alignment algorithm is described in Appendix A.

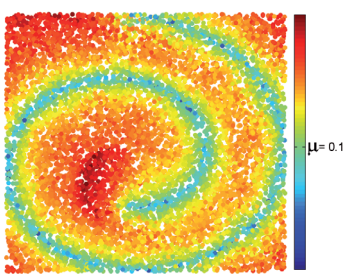
### 4.1.3 Out of sample extension and anomaly detection

Application of Algorithm 4 to $\mathcal{X}$ with $K_\varepsilon$, $\varepsilon = 3$, $t = 1$ and $\mu = 0.1$ was resulted in an $s$-dimensional $2\mu$-distortion of $\boldsymbol{\Psi}^{(1)}$, $\mathbf{h}_s^{(1)} : \mathcal{X} \to \mathbb{R}^s$ with $s = 1,246$. An out-of-sample extension of $\mathbf{h}_s^{(1)}$ to $\bar{\mathcal{X}}$ is demonstrated in this section, as well as anomaly detection, $\bar{\mathcal{X}} \subset \mathbb{R}^3$ be a random subset of $10,000$ data points, uniformly sampled from the three dimensional bounding box of $\mathcal{X}$.

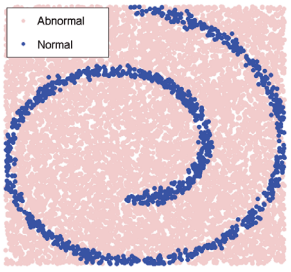
For that purpose, Algorithm 5 was applied to $\bar{\mathcal{X}}$. Beside its three first inputs, which are provided as outputs of Algorithm 4, a transition probabilities vector $\mathbf{p}^{(1)}(x) \in \mathbb{R}^n$ should be

defined for any $x \in \bar{\mathcal{X}}$. In this example $\mathbf{p}^{(1)}(x)$ was defined consistently with the kernel $K_\varepsilon$, using Eq. 3.11. This definition coincides with the definition of the transition probabilities matrix $P_\varepsilon$ from Eq. 4.2, and results in exact extension on $\mathcal{X}$, i.e. if $x = x_i$ for a certain $i \in [n]$, then $\mathbf{h}_s^{(1)}(x) = \mathbf{h}_s^{(1)}(x_i)$.

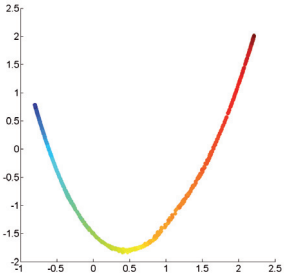
The results are shown in Fig. 4.5. Figure 4.5(a) shows a side view of the dataset $\bar{\mathcal{X}}$, each data point $x \in \bar{\mathcal{X}}$ is colored proportionally to its out-of-sample extension distortion rate $\boldsymbol{\mu}(x)$ (see step 3 of Algorithm 5.) Classification of $\bar{\mathcal{X}}$ to normal and abnormal classes is shown in Fig. 4.5(b). The normal class $\mathcal{N} \subset$ is darkly colored, and abnormal class $\bar{\mathcal{N}}$ is brightly colored. The classification was done according to Definition 2.2. Thus, $x \in \bar{\mathcal{X}}$ is considered normal if its distortion rate $\boldsymbol{\mu}(x) \leq \mu$. Otherwise, it is considered abnormal. Lastly, a two dimensional view of the out-of-sample extension of the normal class, namely $\mathbf{h}_s^{(1)}(x),\ x \in \mathcal{N}$, is shown in Fig. 4.5(c). Each embedded point is colored in the same color of its nearest neighbor from the embedding of $\mathcal{X}$, $\mathbf{h}_s^{(1)}(\mathcal{X})$. The shown coordinates-system is consistent with the one presented in Fig. 4.4.




(a) Distortion rate function $\boldsymbol{\mu} : \bar{\mathcal{X}} \to \mathbb{R}$.




(b) Classification of $\bar{\mathcal{X}}$ to normal ($\boldsymbol{\mu}(x) \leq 0.1$) and abnormal (otherwise) classes.




(c) Normal points in the embedded space. Points are colored identically to their nearest neighbor in the original space (see Figure 4.1) .

Figure 4.5: 10,000 out-of-sample points randomly selected in the bounding box of the original Swiss roll. Left and center: two-dimensional side view of the out-of-sample dataset, colored by their distortion rate and classification, respectively. Right: extension of the first two meaningful ICPQR-based DM embedded space coordinates to the normal out-of-sample points.

### 4.1.4 Comparison with $\mu$IDM

The $\mu$IDM algorithm in [42] is a dictionary-based method that provides a low rank $2\mu$-distortion for the first transition time step of the DM, $\boldsymbol{\Psi}^{(1)}$. The algorithm incrementally constructs an approximated map, using a single scan of the data. This algorithm is greedy and sensitive to the scan order. Typically, the growth rate of the dictionary is very high at the beginning and decays across time. Moreover, the resulted dictionary and the resulted embedding's dimension may be redundant. In each iteration of the $\mu$IDM, a newly processed data point is considered for inclusion in the dictionary that was constructed from previously scanned data points. At the beginning of every iteration, the dictionary elements are already embedded in a low-dimensional space (whose dimension equals to the size of the dictionary), where its geometry is identical to the diffusion geometry of the data, restricted to the dictionary. Then, a Nyström-type extension [5] is applied to the scanned data points, based on its affinities with the dictionary elements, in order to approximate the embedding of the newly processed data point. The exact DM of this data point, together with the dictionary, is then efficiently computed. The geometries of these two embeddings are identical for the dictionary therefore, at this stage these

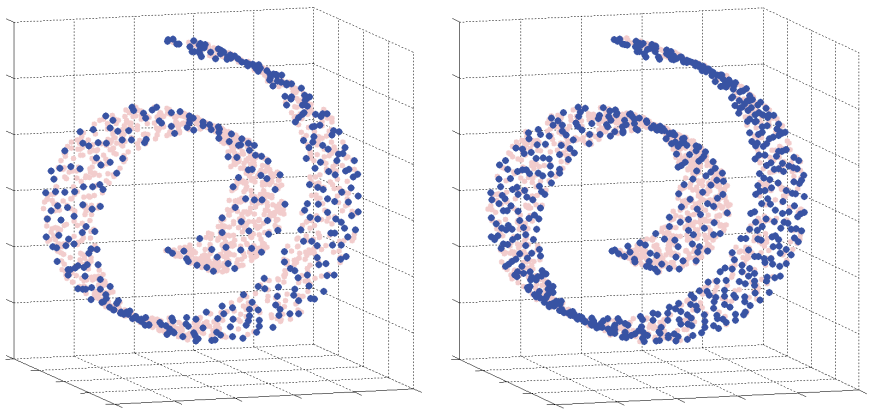
two geometries are aligned to coincide on the dictionary, and the distance of the extended map from the exact map of the examined data point is measured. If this distance is larger than $\mu$ then the examined data point is added to the dictionary.

The entire computational complexity of this iterative process is lower than the computational complexity of DM. The exact number of required operations depends on the required accuracy and on the dimensionality of the original ambient space.

The presently proposed QR-based DM method considers the entire dataset in each iteration and is not sensitive to the order of the dataset. Therefore, the resulted dictionary is more sparse, as demonstrated in Table 4.1 and Fig. 4.6.

| $\mu$ | | ICPQR-based DM | $\mu$IDM [42] |
|---|---|---|---|
| 0.1 | Dictionary size $s$ | $1,246$ | $2,305$ |
| | Execution time (sec.) | 43 | 7 hours |
| | Actual distortion | 0.01 | 0.03 |
| 1 | Dictionary size $s$ | 752 | $1,293$ |
| | Execution time (sec.) | 27 | 71 minutes |
| | Actual distortion | 0.23 | 0.61 |
| 5 | Dictionary size $s$ | 382 | 630 |
| | Execution time (sec.) | 17 | 15 minutes |
| | Actual distortion | 3.91 | 4.46 |
| 10 | Dictionary size $s$ | 190 | 284 |
| | Execution time (sec.) | 9 | 4 minutes |
| | Actual distortion | 12.81 | 13.24 |

Table 4.1: ICPQR-based DM and $\mu$IDM algorithms compared by dictionary size, execution time and actual distortion[4], w.r.t. $\mathbf{\Psi}^{(1)}$, for various distortion parameters. Clearly, the actual distortion is bounded by $2\mu$. Execution times are averaged over 10 runs of the algorithms.



(a) ICPQR-based DM Dictionary, 382 points

(b) $\mu$IDM Dictionary, 630 points

Figure 4.6: Data points admitted to ICPQR-based DM and $\mu$IDM dictionaries (dark points), with distortion parameter $\mu = 5$. The input was given to both algorithms ordered from the inner part of the Swiss roll, to the outer, larger radius.

[4]An actual distortion of a function $f$ w.r.t. $g$ is $\sup_{x,y\in\mathcal{X}} |\|f(x) - f(y)\| - \|g(x) - g(y)\||$.

## 4.2 QR-based DM analysis - real-world data

This section exemplifies a semi-supervised anomaly detection process, applied to a real-world dataset. The examined dataset $\bar{\mathcal{X}} \subset \mathbb{R}^{14}$ is the DARPA dataset [28] consists of $\bar{n} = 12,617$ data points. Each data point is a vector of 14 features that describes a computer network traffic, labeled as either normal and legitimate or abnormal that pertains to an attack and intrusion to the network. The dataset is divided into a training set $\mathcal{X} \subset \bar{\mathcal{X}}$ which contains $n = 6,195$ normally behaving samples and 5 testing datasets $\{\mathcal{T}_{mon}, \ldots, \mathcal{T}_{fri}\}$, which were collected during different days of the week, each contains normal and abnormal data points, as described in Table 4.2.

First, the training dataset was scaled to lay in the 14-dimensional unit box $[0,1]^{14}$. Then, the same scaling was performed to the testing datasets $\mathcal{T} = \mathcal{T}_{mon} \cup \ldots \cup \mathcal{T}_{fri}$. Application of Algorithm 4 to the training set $\mathcal{X}$, using the Gaussian kernel from Eq. 4.1 with $\varepsilon = 0.6$, $t = 1$ and $\mu = 10^{-7}$, produced an $s$-dimensional $2\mu$ distortion of the first time step DM $\boldsymbol{\Psi}^{(1)}$ of $\mathcal{X}$, $\mathbf{h}_s^{(1)} : \mathcal{X} \to \mathbb{R}^s$ with $s = 138$. The neighborhood size parameter $\varepsilon$ was chosen to be twice the median of all the mutual distances between the data points in $\mathbb{R}^{14}$. Such a selection is a common heuristic for that parameter determination in the context of DM. This concludes the training phase.

As a second stage, for each testing point $x \in \mathcal{T}$, the probabilities vector $\mathbf{p}^{(1)}(x)$ was defined as the natural extension of the Gaussian kernel from Eq. 4.1, using Eq. 3.11, with $\varepsilon = 0.6$. Any testing point which is distant (relatively to $\varepsilon$) from the training set $\mathcal{X}$ yields $k_\varepsilon(x, y) \approx 0,\ y \in \mathcal{X}$, and was a-priori classified as abnormal. The subset of distant testing points is denoted by $\mathcal{F}$.

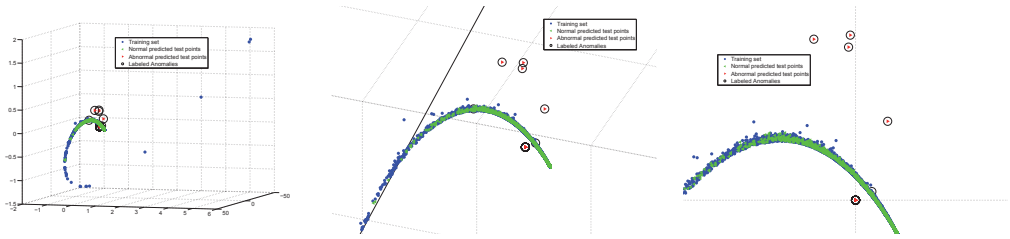
Finally, Algorithm 5 was applied to the elements of $\mathcal{T} \backslash \mathcal{F}$, using the previously computed probabilities vectors $\mathbf{p}^{(1)}(x),\ x \in \mathcal{T}$, to get an extension of $\mathbf{h}_s^{(1)} : \mathcal{T} \backslash \mathcal{F} \to \mathbb{R}^s$ and a distortion rate $\boldsymbol{\mu} : \mathcal{T} \backslash \mathcal{F} \to \mathbb{R}$. Then, strictly abnormal data points were detected, following Definition 2.2 with $\mu_{strict}$ from Eq. 3.10.

The anomaly detection results are summarized in Table 4.2, and some of them are demonstrated in Fig. 4.7. The rates in the accuracy percentage and false alarms columns are related to the original labeling of $\bar{\mathcal{X}}$.

| Set | Size | # of anomalies | Accuracy [%] | False Alarms [%] |
|---|---|---|---|---|
| $\mathcal{T}_{mon}$ | $1,321$ | 1 | 100 | 0.68 |
| $\mathcal{T}_{tue}$ | $1,140$ | 53 | 100 | 0.53 |
| $\mathcal{T}_{wed}$ | $1,321$ | 16 | 100 | 0.08 |
| $\mathcal{T}_{thu}$ | $1,320$ | 24 | 96 | 1.74 |
| $\mathcal{T}_{fri}$ | $1,320$ | 18 | 100 | 0.15 |

Table 4.2: Anomaly detection performances.

Figure 4.7 presents three-dimensional views of an aligned version of $\mathbf{h}_s^{(1)}$ of $\mathcal{X}$, as well as its extension to $\mathcal{T}_{thu} \backslash \mathcal{F}$, with the first three significant coordinates of $\boldsymbol{\Psi}^{(1)}$, the DM of $\mathcal{X}$ in the first time step. As can be seen in the figure, most of the test set located near the training set, while the abnormal data points are embedded further.




(a) (b) (c)

Figure 4.7: Three different angles and scales of an aligned version of three-dimensional projection of $\mathbf{h}_s^{(1)}$, applied to the training set $\mathcal{X}$ (blue points), its out-of-sample extension to $\mathcal{T}_{thu} \backslash \mathcal{F}$ (strictly-normal points are green, strictly-abnormal points are red), and labeled anomalies (black circles). (a) general view. (b) strictly-normal out-of-sample data points are mapped closely to the training set. (c) strictly-abnormal data points.

As in Section 4.1.2, the alignment was done only for visualization purposes, using the described algorithm in Appendix A.

## 4.3 Semi-supervised multi-class classification of high dimensional data

In this section a multi classification process, based on Algorithms 2 and 3 is presented. The analyzed data is the parametric $m$-dimensional ISOLET dataset [3] $\bar{\mathcal{X}} \subset \mathbb{R}^m$ with $m = 617$, consists of $7,797$ data points in $[-1,1]^m$. Each datum corresponds to a single human pronunciation of ISOlated LETters. The goal is to classify a testing subset $\mathcal{T} \subset \bar{\mathcal{X}}$ of $1,559$ letter-pronunciation samples spoken by 30 people to 26 classes, based on a training set $\mathcal{X} \subset \bar{\mathcal{X}}$ of $n = 6,238$ samples, already classified to $\mathcal{X}_\zeta$, $\zeta \in \mathcal{I} \triangleq \{A, B, \ldots, Z\}$, spoken by 120 different people.

For this purpose Algorithm 2 was applied to each of the training sets $\mathcal{X}_\zeta$ with distortion parameter $\mu = 4.7$, to produce 26 dictionaries $\mathcal{D}_\zeta \subset \mathcal{X}_\zeta$, $\zeta \in \mathcal{I}$. Then, for each of these dictionaries, Algorithm 3 was applied to each element in the testing set $\mathbf{x} \in \mathcal{T}$, to produce its distortion rate $\boldsymbol{\mu}_\zeta(\mathbf{x})$ (see Eq. 2.15.) Finally, each of the testing points was classified to the class whose dictionary described it the best, i.e. $\mathbf{x}$ was classified to class $\zeta_0$, where $\zeta_0 \triangleq \arg\min_{\zeta \in \mathcal{I}} \boldsymbol{\mu}_\zeta(\mathbf{x})$. We note that the parameter $\mu$ was determined by taking part of the train set to serve as a validation set. Then, several values of $\mu$ were applied on the (reduced) training set, and the validation set was classified based on these values. The chosen $\mu$ was the one which was optimal on the validation set.

The results are shown in Fig. 4.8. Out of $1,559$ test samples, 92% were classified correctly. The classification of the test data is presented in a confusion matrix in Fig. 4.8. It can be seen that the majority of test samples of each class were classified correctly, by looking at the shades of the diagonal. The sets $\{B, C, D, E, G, P, T, V, Z\}$ and $\{M, N\}$ are the most difficult letters to classify, due to high similarity in pronunciation of the letters within each set. The state-of-the-art classification accuracy of this dataset, 96.73%, was achieved in [22] by using 30-bit error correcting output codes based on neural networks. This method is far more complex than the solution proposed in this work.

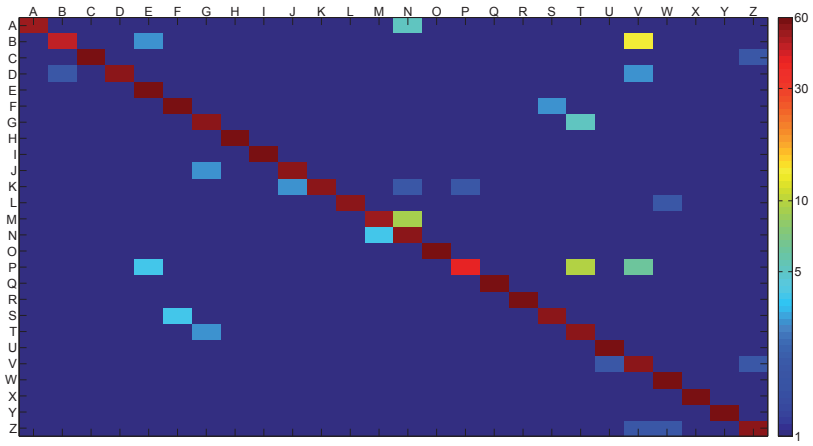



Figure 4.8: Classification of the test samples in ISOLET dataset. For each ordered couple $(i, j)$, the cell at row $i$ and column $j$ is colored according to the number of test samples belonging to class $i$ that were classified by the algorithm to the class $j$. The cells on the diagonal denote correct classification. For each letter a total of 60 test samples are provided (except for 'M' which is missing one sample due to recording difficulties).

# 5 Conclusions and Future Works

This work presents a complete framework for linear dimensionality reduction, out-of-sample extension and anomaly detection for high-dimensional parametric data, which is based on incomplete pivoted QR decomposition of the associated data matrix. The presented method preserves the high-dimensional geometry of the data, up to a user-defined distortion parameter. Such a low-dimensional data representation enables further geometrically-based data analysis which, due to the geometry-preservation, is still valid to the original data. The storage complexity of the method is extremely low compared to the classical PCA, and in the worst case its computational complexity is similar to that of the PCA. The method provides a dictionary, which is a subset of landmark data points that forms a basis for the projection of the entire data to a low dimension. Out-of-sample extension and anomalous point detection become simple tasks once the dictionary is computed. Although The suggested method is designated for parametric data analysis, in some cases it can be adapted to non-parametric data analysis frameworks, as was demonstrated to DM framework. The stability of our method to perturbations (noise) was proved, and its connection to matrix-approximation was presented.

Experimental results show that our method achieves a lower dimensional embedding than PCA achieves for a certain distortion. Moreover, analyses of both synthetic and real-world datasets were demonstrated, and showed good results of dimensionality reduction, out-of-sample extension, anomaly detection and multi-class classification.

Future works include randomized version of the presented framework, to provide a more computationally efficient method for a dictionary-based dimensionality reduction. In addition, a generalization of our method for non-parametric data analysis methods is considered, as well as a dynamic framework that enables to cope with datasets that vary across time. Finally, a parallel version of the algorithm, which simultaneously builds dictionaries for subsets of the data and then unifies them, is considered.

## Acknowledgment

This research was partially supported by the Israel Science Foundation (Grant No. 1041/10), by the Israeli Ministry of Science & Technology (Grants No. 3-9096, 3-10898), by US - Israel Binational Science Foundation (BSF 2012282) and by a Fellowship from Jyväskylä University.

# A Appendix: Alignment Algorithm

This section details the alignment algorithm that was utilized in Sections 4.1.2 and 4.2, for visualization purposes.

Suppose that $A$ and $B$ are two data $m \times n$ matrices of $n$ data points in $\mathbb{R}^m$. If the sizes of $A$ and $B$ are different, then the necessary zeros padding can be performed. Clearly, it will not change the geometry of the columns of these matrices. Let $\bar{A}$ and $\bar{B}$ be centralized versions of $A$ and $B$ around the columns means of each one of them. Then, the best orthogonal alignment of $\bar{B}$'s columns with $\bar{A}$'s columns is provided by the orthogonal matrix $Q = U_A U_B^*$, where $U_A$ and $U_B$ are the left singular vectors of $A$ and $B$, respectively. Then, the aligned centralized matrix $\tilde{B} = Q * B$ is decentralized by $A$'s columns mean. Obviously, since $Q$ is orthogonal, the geometry of $\tilde{B}$'s columns is unaffected. Algorithm 6 concludes the above.

**Algorithm 6:** Alignment Algorithm

**Input** : Two $m \times n$ matrices $A$ and $B$

**Output**: An $m \times n$ matrix $\tilde{B}$ whose columns geometry is identical to that of $B$'s columns, and $\tilde{B}$'s columns are optimally aligned with $A$'s columns.

1 centralize $A$: $\bar{A} = A - \mathbf{a}\mathbf{1}_n^*$, where $\mathbf{a} = \sum_{j=1}^{n} A_{(:,j)}$ and $\mathbf{1}_n \in \mathbb{R}^n$ is the all-ones vector

2 centralize $B$: $\bar{B} = B - \mathbf{b}\mathbf{1}_n^*$, where $\mathbf{b} = \sum_{j=1}^{n} B_{(:,j)}$

3 compute the left singular vectors of $\bar{A}$ and $\bar{B}$, $U_A$ and $U_B$, respectively

4 define $\tilde{B} = U_A U_B^* \bar{B} + \mathbf{a}\mathbf{1}_n^*$

# References

[1] D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003.

[2] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 557–563. ACM, 2006.

[3] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[4] Z-Z. Bai, I. S. Duff, and A. J. Wathen. A class of incomplete orthogonal factorization methods. i: Methods and theories. *BIT Numerical Mathematics*, 41(1):53–70, 2001.

[5] C. T. H. Baker. *The Numerical Treatment of Integral Equations*. Oxford: Clarendon Press, 1977.

[6] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, 2008.

[7] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[8] A. Bermanis, A. Averbuch, and R.R. Coifman. Multiscale data sampling and function extension. *Applied and Computational Harmonic Analysis*, 34(1):15 – 29, 2013.

[9] A. Bermanis, G. Wolf, and Averbuch. A. Cover-based bounds on the numerical rank of gaussian kernels. *Applied and Computational Harmonic Analysis*, 36(2):302 – 315, 2014.

[10] C. Boutsidis, M. W. Mahoney, and P. Drineas. Unsupervised feature selection for principal components analysis. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–69. ACM, 2008.

[11] C. Boutsidis, M. W. Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 968–977, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[12] C. Boutsidis, J. Sun, and N. Anerousis. Clustered subset selection and its applications on it service metrics. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 599–608, New York, NY, USA, 2008. ACM.

[13] C. Boutsidis and D. P. Woodruff. Optimal cur matrix decompositions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 353–362. ACM, 2014.

[14] M. Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20 – 30, 2006. Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems.

[15] P. Brard, G. Besson, and S. Gallot. Embedding riemannian manifolds by their heat kernel. *Geometric and Functional Analysis GAFA*, 4(4):373–398, 1994.

[16] T. F. Chan. Rank revealing qr factorizations. *Linear Algebra and its Applications*, 88–89(0):67 – 82, 1987.

[17] T. F. Chan and P. Hansen. Some applications of the rank revealing qr factorization. *SIAM Journal on Scientific and Statistical Computing*, 13(3):727–741, 1992.

[18] H. Cheng, Z. Gimbutas, P.G. Martinsson, and V. Rokhlin. On the compression of low rank matrices. *SIAM Journal on Scientific Computing*, 26(4):1389–1404, 2005.

[19] K. L. Clarkson. Tighter bounds for random projections of manifolds. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 39–48. ACM, 2008.

[20] R.R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[21] J.K. Cullum and R.A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations I: Theory*, volume 41 of *Classics in Applied Mathematics*. SIAM, 2002.

[22] T.G. Dietterich and G. Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *IN PROCEEDINGS OF AAAI-91*, pages 572–577. AAAI Press, 1991.

[23] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-based methods. In *In Proc. of the 10th RANDOM*, pages 316–326, 2006.

[24] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Relative-error cur matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30(2):844–881, 2008.

[25] A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

[26] W. Givens. Computation of plane unitary rotations transforming a general matrix to triangular form. *Journal of the Society for Industrial and Applied Mathematics*, 6(1):pp. 26–50, 1958.

[27] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, fourth edition, 2013.

[28] I. Graf, R. Lippmann, R. Cunningham, D. Fried, K. Kendall, S. Webster, and M. Zissman. Results of darpa 1998 offline intrusion detection evaluation. In *DARPA PI Meeting*, volume 15, 1998.

[29] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[30] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24, 1933.

[31] A. S. Householder. Unitary triangularization of a nonsymmetric matrix. *J. ACM*, 5(4):339–342, October 1958.

[32] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[33] A. Jennings and M. Ajiz. Incomplete methods for solving $a^t ax = b$. *SIAM Journal on Scientific and Statistical Computing*, 5(4):978–987, 1984.

[34] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[35] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[36] M. W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2):123–224, 2011.

[37] M. W. Mahoney and P. Drineas. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

[38] P.G. Martinsson, V. Rokhlin, and M. Tygert. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis*, 30(1):47–68, 2011.

[39] A. T. Papadopoulos, I. S. Duff, and A. J. Wathen. A class of incomplete orthogonal factorization methods. ii: implementation and results, 2002.

[40] J. R. Rice. Experiments on gram-schmidt orthogonalization. *Math. Comp.*, 20:pp. 325–328, 1966.

[41] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[42] M. Salhov, A. Bermanis, G. Wolf, and A. Averbuch. Approximately-isometric diffusion maps. *Applied and Computational Harmonic Analysis*, (0):–, 2014.

[43] B. Schölkopf, A. Smola, E. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[44] L. J. Schulman. Clustering for edge-cost minimization. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 547–555. ACM, 2000.

[45] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290:2319–2323, 2000.

[46] S. Vempala. *The random projection method*, volume 65. American Mathematical Soc., 2005.

[47] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17, 2007.

[48] S. Wang and Z. Zhang. Improving cur matrix decomposition and the nyström approximation via adaptive sampling. *The Journal of Machine Learning Research*, 14(1):2729–2769, 2013.

[49] H. Wendland. Scattered data approximation. Cambridge University Press, 2005.