Graph-based Methods for Visualization and Clustering.

THÈSE N^{O} 7952 (2017)

PRÉSENTÉE LE 13 OCTOBRE 2017
À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE TRAITEMENT DES SIGNAUX 2
PROGRAMME DOCTORAL EN GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Johann PARATTE

acceptée sur proposition du jury:

Dr F. Fleuret, président du jury
Prof. P. Vandergheynst, directeur de thèse
Prof. P. Borgnat, rapporteur
Dr B. Ricaud, rapporteur
Prof. P. Frossard, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2017

L'énigme a peur du mot ; l'infini semble à peine
Pouvoir contenir l'inconnu.
— Victor Hugo

A tous ceux qui regardent les étoiles.

Abbreviations

ACC	**A**verage **C**luster **C**oncentration
ACI	**A**verage **C**lusterability **I**ndex
ACN	**A**verage **C**luster **N**oise
ANN	**A**pproximate **N**earest **N**eighbours
BFS	**B**readth **F**irst **S**earch
CDR	**C**ompressive **D**imensionality **R**eduction
CGT	**C**lustering **G**raph **T**ree
CHD	**C**onvex **H**ull **D**iffusion
DFS	**D**epth **F**irst **S**earch
FEARS	**F**ast **E**igenspace **A**pproximation using **R**andom **S**ignals
GSP	**G**raph **S**ignal **P**rocessing
HSV	**H**ue **S**aturation **V**alue
i.i.d.	**i**ndependent and **i**dentically **d**istributed
KDD	**K**ernelized **D**iffusion **D**istance
LKD	**L**ocalized **K**ernel **D**istance
NN	**N**earest **N**eighbours
PCA	**P**rincipal **C**omponent **A**nalysis
PSD	**P**ower **S**pectral **D**ensity
RGB	**R**ed **G**reen **B**lue
RKHS	**R**eproducing **K**ernel **H**ilbert **S**pace
SVD	**S**ingular **V**alue **D**ecomposition
TBD	**T**ree **B**ranching **D**istance
TV	**T**otal **V**ariation

Symbols

$\mathbb{R}$	Space of real numbers
$\mathbb{C}$	Space of complex numbers
$\mathcal{G}$	Graph
$\mathcal{V}$	Vertex set
$\mathcal{E}$	Edge set
$\mathbf{W}$	Weight matrix
$\mathbf{L}$	Graph Laplacian matrix
$\mathbf{U}$	Graph Fourier matrix
$\boldsymbol{\lambda}$	Vector associated to the set of eigenvalues $\{\lambda_\ell\}_{\ell=0,\dots,N-1}$ with $\boldsymbol{\lambda}[\ell] = \lambda_\ell$
$\boldsymbol{\Lambda}$	Diagonal matrix of the eigenvalue vector $\boldsymbol{\lambda}$
$\mathbf{X}$	Input data matrix with N rows and D columns
$\mathbf{Y}$	Output data matrix of N rows and d columns
N	Number of vertices, $\lvert\mathcal{V}\rvert$, and number of data points in $\mathbf{X}$
v_i	Vertex indexed by i
i, j	Indices for the vertex set: $1, 2, \dots N$
ℓ	Indices for the frequency set: $0, 1, \dots N-1$
$g(\boldsymbol{\lambda})$	Vector composed of the kernel evaluated on the eigenvalue set, $g(\boldsymbol{\lambda})[\ell] = g(\boldsymbol{\lambda}[\ell])$
$\mathcal{T}_i g$	Localization of the kernel g at vertex i

Notation conventions

In this thesis, lowercase bold fonts are used for vectors (e.g. $\mathbf{x}$, $\boldsymbol{\lambda}$), uppercase bold fonts for matrices (e.g. $\mathbf{L}$, $\mathbf{U}$), calligraphic uppercase font is used for both sets and operators (e.g. $\mathcal{V}$, $\mathcal{E}$). Note that we use $|\mathcal{S}|$ to write the cardinality of a set $\mathcal{S}$.

Vectors, matrices and other discrete elements are indexed using square brackets (e.g. $\mathbf{x}[i]$) and continuous functions and operators with parentheses (e.g. $g(\lambda)$). For simplicity, and when it does not present an ambiguity, subscripts are used as aliases to square brackets (e.g. $\mathbf{x}[i] = \mathbf{x}_i$ or $\mathbf{W}[i,j] = \mathbf{W}_{i,j} = \mathbf{W}_{ij}$). Also, when an univariate function g is applied to a vector $\boldsymbol{\lambda}$, the result is a vector with the function applied element-wise, i.e. $[g(\boldsymbol{\lambda})]_i = g(\boldsymbol{\lambda}[i])$.

The letters $\mathbf{x}$, $\mathbf{y}$, $\boldsymbol{z}$ are, when not stated otherwise, reserved for data (e.g. graph signals, sample of data) and f, g, h for functions.

The symbol x^* denotes the complex conjugation of x for scalars and the transpose complex conjugation for vectors and matrices. We note $\mathcal{F}(\mathbf{x})$ or $\hat{\mathbf{x}}$ the (Graph) Fourier transform of $\mathbf{x}$. In addition, we use $\dot{\mathbf{x}}$ to denote an estimator of $\mathbf{x}$.

When not otherwise stated, the operator $\|\mathbf{x}\|$ denotes the ℓ_2-norm of $\mathbf{x}$. The ℓ_∞ uniform (sup) norm of $\mathbf{x}$ is defined as $\|\mathbf{x}\|_\infty = \max_i |\mathbf{x}_i|$ and we abusively use the ℓ_0-norm to count the number of non-zero elements in a vector.

The minimum and maximum eigenvalues of a matrix $\mathbf{A}$ are written $\sigma_{\min}(\mathbf{A})$ and $\sigma_{\max}(\mathbf{A})$ respectively. The induced norm of a matrix $\mathbf{A}$ is noted $\|\mathbf{A}\|_{op} = \sup_{\mathbf{x}\neq 0} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$ and the Froebenius norm $\|\mathbf{A}\|_F = \sqrt{\sum_i \sum_j \mathbf{A}_{ij}}$.

When measuring algorithmic complexity, time complexity is assumed unless otherwise stated. We use asymptotic Landau notations with the following definitions :

- $f(x) = \mathcal{O}(g(x))$, $(x \to a)$, if and only if

$$\limsup_{x\to a} \left| \frac{f(x)}{g(x)} \right| < \infty,$$

- $f(x) = \Omega(g(x))$, $(x \to a)$, if and only if

$$\limsup_{x\to a} \left| \frac{f(x)}{g(x)} \right| > 0,$$

- $f(x) = \Theta(g(x))$, $(x \to a)$, if and only if

$$f(x) = \mathcal{O}(g(x)) \quad \text{and} \quad f(x) = \Omega(g(x)).$$

When not specifically noted, it assumed that $a = +\infty$.

Acknowledgements

A doctoral thesis is a fantastic journey which is more about what you become than what you achieve. In this era of ubiquitous digital information, which is the focus of this work, we need to remember that we are living people in a tangible world, and I would not have gone far without the people around me.

The first person I would like to thank is Pierre Vandergheynst, who made this experience possible by welcoming me in his laboratory. I am very grateful for his support, ideas, spirit, and openness. The whole Geekroom concept originated from him and I hope it has been the chaotic, nerdy, open place that he expected.

Then, I would not have progressed much in the graph jungle without the help of Nathanaël Perraudin, who became quickly more a friend than a colleague. I must apologize because I repaid him quite poorly for his exceptional academic support by introducing him to Club Mate.

Having spent my Ph.D. in the (in)famous Geekroom (also less known as ELE 229), I must say it has been a working place quite incompatible with work. Luckily, the noise generally consisted mostly of laughter and interruptions were often unexpected philosophical discussions. But mostly, this place has been a reminder that work should be a social experience. Thank you, Fabien, Maximilien, Yann, Lionel, Nico, Basile and Alex for having shared those moments.

I do not forget, of course, all other members of the lab, with whom I shared this constant solidary quest towards becoming autonomous researchers. A special thought to Andreas, Benjamin, Nathanaël and Lionel who took the time to proofread this thesis.

This story actually began long ago with my parents, who instiled in me the love for knowledge and supported me while I pursued my never-ending educational journey. These thoughts extend, of course, to my brothers, my family and my friends.

Last but not least, I would like to thank my dear Audrey for her patience and constant support, during good and bad times. She helped me smile and get up every morning, and this thesis is a bit hers too.

Lausanne, August 2017 Johan Paratte

Abstract

The amount of data that we produce and consume is larger than it has been at any point in the history of mankind, and it keeps growing exponentially. All this information, gathered in overwhelming volumes, often comes with two problematic characteristics: it is complex and deprived of semantic context. The field of Data Science has emerged from this need to process and make sense of these large collections of data.

High-dimensionality being at the same time one of the main characteristics of modern data and a fundamental issue for efficient processing, the task of dimensionality reduction has become of great interest. Indeed, reducing the dimensionality allows to extract the relevant information from raw data and thus facilitate subsequent tasks such as, for example, clustering or visualization. The literature contains many dimensionality reduction methods such as PCA, a technique widely used for decades, or t-SNE, one of the most famous current state-of-the-art algorithms. The common idea underlying dimensionality reduction methods is that of providing a mapping which preserves the similarity (e.g. pairwise distances) between data points from their original space to a new one.

Measuring similarity between large sets of high-dimensional objects is, however, problematic for two main reasons: first, Euclidean distances tend to become meaningless in large dimensions and second, the number of pairwise distances between points is quadratic with respect to the amount of data points. The current consensus in the field is that the first issue, generally called the curse of dimensionality, can be overcome by exploiting the fact that data is naturally structured. The second problem is addressed by considering only closest neighbours instead of all possible candidates. These two solutions are actually the premises that motivate the use of nearest neighbours graphs to understand the structure in data. As a matter of fact, most dimensionality reduction methods use similarity matrices that can be interpreted as graph adjacency matrices.

Yet, despite this progress, dimensionality reduction is still very challenging when applied to very large datasets. Indeed, despite the fact that recent methods specifically address the problem of scalability, processing datasets of millions of elements remain a very lengthy process.

In this thesis, we propose new contributions which address the problem of scalability using the framework of Graph Signal Processing, which extends traditional signal processing to graphs. We do so motivated by the premise that graphs are well suited to represent the structure of the data.

Abstract

In the first part of this thesis, we look at quantitative measures for the evaluation of dimensionality reduction methods. Using tools from graph theory and Graph Signal Processing, we show that specific characteristics related to quality can be assessed by taking measures on the graph, which indirectly validates the hypothesis relating graph to structure.

The second contribution is a new method for a fast eigenspace approximation of the graph Laplacian. Using principles of GSP and random matrices, we show that an approximated eigensubpace can be recovered very efficiently, which be used for fast spectral clustering or visualization.

Next, we propose a compressive scheme to accelerate any dimensionality reduction technique. The idea is based on compressive sampling and transductive learning on graphs: after computing the embedding for a small subset of data points, we propagate the information everywhere using transductive inference. The key components of this technique are a good sampling strategy to select the subset and the application of transductive learning on graphs.

Finally, we address the problem of over-discriminative feature spaces by proposing a hierarchical clustering structure combined with multi-resolution graphs. Using efficient coarsening and refinement procedures on this structure, we show that dimensionality reduction algorithms can be run on intermediate levels and up-sampled to all points leading to a very fast dimensionality reduction method.

For all contributions, we provide extensive experiments on both synthetic and natural datasets, including large-scale problems. This allows us to show the pertinence of our models and the validity of our proposed algorithms. Following reproducible principles, we provide everything needed to repeat the examples and the experiments presented in this work.

Key words: data science, dimensionality reduction, big data, scalable processing, graph, graph signal processing, quality measure, sampling, transductive learning, embedding, clustering, visualization

Résumé

La quantité de données générées et consommées par la société humaine est à son point le plus haut de toute l'Histoire, et continue de croître exponentiellement. Ce déluge d'informations récoltées de toute part est souvent associé à deux caractéristiques problématiques : les données sont généralement complexes, d'une part, et pauvres en contexte sémantique, d'autre part.

La haute dimensionnalité étant à la fois une propriété intrinsèque de la majorité des données modernes et l'un des principaux obstacles à un traitement efficace, le problème de la réduction de dimension suscite naturellement un grand intérêt. En effet, réduire la dimensionnalité permet d'extraire les informations pertinentes des données brutes et facilite ainsi le traitement de tâches ultérieures telles que, par exemple, le clustering ou la visualisation. La littérature scientifique contient de nombreuses méthodes de réduction de dimension, telle que la PCA, une technique utilisée dans une large palette de problèmes depuis des décennies, ou t-SNE, une des méthodes modernes les plus populaires. Un objectif fondamental et commun à la quasi-majorité de ce corpus consiste à déterminer une transformation qui préserve la similarité entre les objets depuis leur espace d'origine vers leur espace de destination.

Cependant, établir des mesures de similarité entre des éléments en haute dimension est problématique pour deux principales raisons : d'une part, les distances Euclidiennes présentent un comportement contre-intuitif en grande dimension qui les rend peu fiables, et, d'autre part, le nombre de paires de points dans un ensemble de données croît de manière quadratique avec sa taille. Le consensus actuel dans le domaine concernant le premier problème, connu sous le nom de malédiction de la dimension, est qu'il peut être contourné en faisant l'hypothèse que les données possèdent une structure intrinsèque. Le second problème est quant à lui résolu en considérant uniquement les plus proches voisins au lieu de tous les candidats possibles.

Ces deux solutions sont en fait les prémisses qui motivent l'utilisation de graphes de plus proches voisins pour comprendre comment les données sont structurées. De fait, la plupart des algorithmes de réduction de dimensionnalité utilisent des matrices de similarité qui s'apparentent aux matrices d'adjacence utilisées dans les graphes.

Cependant, en dépit des progrès récents, la réduction de dimension reste problématique sur les grands ensembles de données. En effet, bien que la plupart des méthodes modernes tentent explicitement de remédier au problème de la mise à l'échelle, le traitement d'ensembles de

milliards d'objets reste globalement irréalisable.

Dans cette thèse, nous portons un intérêt particulier au problème de mise à l'échelle et proposons de nouvelles contributions dans le domaine de la réduction de dimension mettant à profit les concepts du Traitement de Signal sur Graphes (TSG), qui étend les concepts classiques du traitement de signal aux graphes. Cette approche est motivée par l'hypothèse que les graphes possèdent une aptitude particulière pour l'expression de la structure intrinsèque des données.

Dans une première partie, nous abordons des mesures quantitatives pour l'évaluation de la qualité des méthodes de réduction de dimension. En utilisant les outils fournis par la théorie des graphes et le TSG, nous montrons que différentes caractéristiques liées à la qualité peuvent être établies en utilisant des mesures prises sur les graphes, ce qui valide indirectement l'hypothèse reliant les graphes à la structure des données.

La deuxième contribution est une nouvelle méthode pour l'approximation de l'espace propre du Laplacien sur graphe. En utilisant des matrices aléatoires et les principes du Traitement de Signal sur Graphes, nous montrons qu'une bonne approximation du sous-espace propre peut être obtenue de manière très efficace, celui-ci pouvant être utilisé pour une version accélérée du clustering spectral ou pour de la visualisation.

Ensuite, nous décrivons une nouvelle technique générique pour accélérer tout algorithme de réduction de dimension. L'idée de base consiste combiner des principes d'échantillonnage compressé et d'inférence transductive sur les graphes. Plus précisément, après avoir calculé le plongement d'un petit sous-ensemble de points, l'information est propagée à l'ensemble de données complet. Les composantes clés de cette méthode sont une bonne méthode d'échantillonnage et l'application de l'apprentissage transductif sur des graphes.

Enfin, nous abordons le problème lié à la sur discrimination des espaces de données en proposant une structure de clustering hiérarchique combinée à des graphes multi-résolution. En utilisant des opérateurs de sur et sous-échantillonnage efficaces sur cette structure, nous montrons que des algorithmes de réduction de dimension peuvent être appliqués à des niveaux intermédiaires et ensuite raffinés en une solution globale générant un algorithme de réduction de dimension extrêmement efficace.

Pour toutes ces contributions, nous fournissons les résultats de nombreuses expériences effectuées à la fois sur des données artificielles et réelles, incluant des problèmes à large échelle. Dans le respect des principes de recherche reproductible, nous mettons à disposition l'ensemble des éléments nécessaires à la reproduction des exemples et expériences présentées dans ce travail.

Mots clefs : science des données, réduction de dimension, big data, traitement efficace, graphe, traitement du signal sur graphe, mesures de qualité, échantillonnage, apprentissage transductif, plongement, clustering, visualisation

Contents

The world is full of magic things,
patiently waiting for our senses to grow sharper.
— W.B. Yeats

Introduction

Originating at the dawn of written History, data is a testimony of human activities. From the clay tables used by Sumerians in the 3rd-millennium BC to the digital flow generated by today's particle-physics detectors, we, as a species, mark our presence in the world in multitudes of data fragments. And we do so at an exponential rate, giving rise to unprecedented amounts of information: statistics show that 90% of all human-generated data has been created in the last two years.[1]

In its modern form, nearly everything can be represented digitally, which implies the current ubiquity of digital information. Indeed, it encompasses our day-to-day souvenirs and entertainment (e.g. pictures, movies, music), our communications (e.g. texts, phone calls), our personal and public records (e.g. financial or medical files); the list goes on.

While using binary values to represent everything means a drastic low-level standardization, it also implies numerous complex data formalisms to represent what is considered meaningful, such as images, texts or sounds. In order to keep the semantic unity of such objects, they are necessarily encoded by lengthy bit streams. This fact implies that information in the digital age is fundamentally high-dimensional. Let us consider for example an image of a movie in full-HD. Represented as a vector of pixels (by concatenating all its lines or columns) such an image is already in a space of more than two million dimensions. Similarly, most of today's data is very high-dimensional.

Assuming all this, it happens that almost all the data we consume or manipulate has been or needs to be processed in some way. For example, removing noise from sensor data, inferring semantic information or cross-referencing various models to give predictions. The field of Data Science emerged from the needs to build methods able to deal with this data deluge. Those circumstances lead to the fundamental question addressed by this thesis:

How to organize large sets of complex data ?

To answer this question, we must first formulate it as a mathematical statement. *Organizing*

[1] Every two years we generate 10 times as much data as what existed before http://www.sintef.no/en/latest-news/big-data--for-better-or-worse/


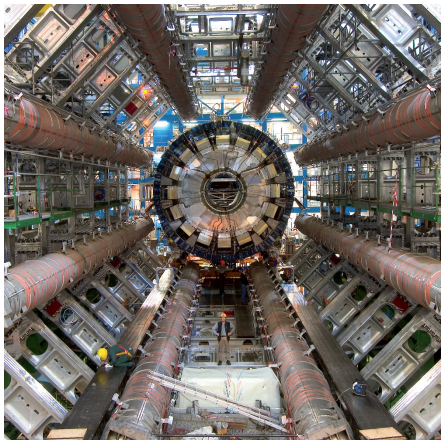

Figure 1 – **Ubiquitous information.** Data has been intrinsically linked with human activities for a long time, only its scale has changed. On the left, the Tablet V of *The Epic of Gilgamesh* dating back to the Babylonian period (2000 BC) which contains less than 5KB of text. On the right, the ATLAS detector of the Large Hadron Collider at CERN generating more than 1GB/s of experiment data when active.
Image credits : CC-BY-SA Wikimedia (left), CERN media (right).

data imply the ability to do pairwise comparisons (i.e. similarity measures); *large sets* means a large number of samples, leading naturally to the use of statistical approaches; *complex data*, as was hinted above, refers to its inherent high-dimensionality. This last property of the data proves to be very challenging when trying to apply statistical methods or compute similarity between samples. This issue is generally called the *curse of dimensionality*. It was originally coined by Bellman [23] to describe a simple fact related to high-dimensional data. Simply put, in order to be able to apply statistical methods to a dataset, the number of samples needs to grow exponentially with the dimensionality of the data to compensate for the expansion of the volume of the space itself. Conversely, the insufficient number of samples usually available leads to what is generally called the empty space problem (or empty space phenomenon [160]). The name comes from the fact that in high dimensions the samples occupy only fractions of the space, thus causing problems when trying to empirically estimate characteristics based on local averages (e.g. density functions).

The curse of dimensionality is also associated with other problematic characteristics of high-dimensional spaces, originating from the counter-intuitive topology encountered in such spaces. Let us take a few examples: in high dimensions, the volume of a ball is concentrated very close to its surface, the probability mass of an isotropic Gaussian distribution is mostly located in its tails and norms (and distances) concentrate. This last example is explained in more detail in Chapter 1, but, intuitively, it implies that points in high dimensions all appear similarly close, making the notion of nearest neighbour meaningless.

If high-dimensional data samples cannot be compared using usual distances, the hope to

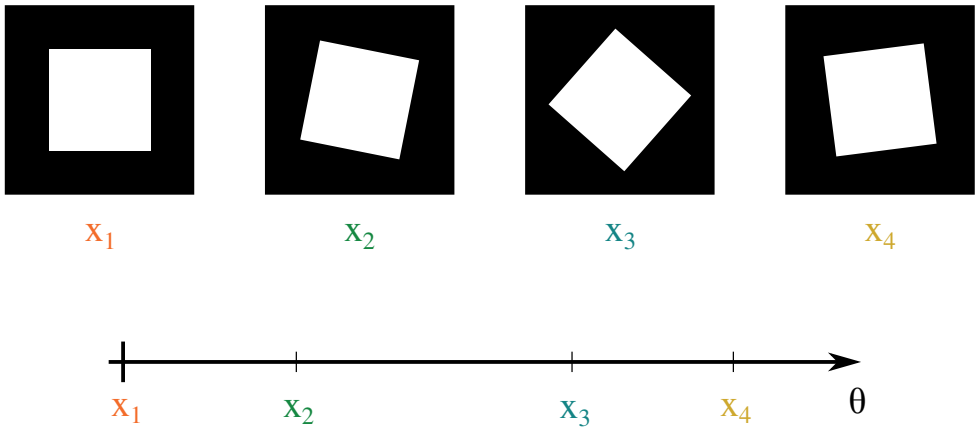

Figure 2 – **The manifold hypothesis.** High-dimensional data may possess very simple, low-dimensional, representations. Let us take as an example a series of images of size 400 by 400 pixels with a black background containing a rotating white square. Four realizations (top) and their one-dimensional representations (bottom) are shown in this figure. Those images live in a huge 160'000 dimensional space, but, in reality, they are far from randomly located in it. Knowing what the possible images are, and having set the size and position of the inner white square, we can parametrize the entire sample space with a single parameter, the angle of rotation of the white square (depicted as the line parametrized by θ). In short, because of its structure, this series of high-dimensional images can be perfectly embedded in a one-dimensional space.

establish any kind of similarity seems compromised. But, as the curse of dimensionality may seem an inescapable problem in general, it is manifest that real data, independently of its dimensionality, is most probably not randomly positioned in the high-dimensional space and thus possesses some geometrical structure. This simplifying assumption of structure is generally used to claim that data is not truly high-dimensional. The hypothesis that the underlying intrinsic dimension of the data is much lower than the extrinsic dimension is a consensus in the data analysis community (see e.g. [26, 204, 162]). The concept of intrinsic dimension refers to the number of parameters of the data, its actual space of variability. The converse also implies that there is a lot of redundancy between the different coordinates of the high-dimensional representation of the data. An illustration of this principle is given in Figure 2.

Emerging from these facts, dimensionality reduction encompasses all the techniques used to extract the low-dimensional structure of the data from its high-dimensional, redundant, representation. Adopting an information theoretical point of view, this can be formulated as to extract useful information from noisy data. But, it is, foremost, a way to bypass the curse of dimensionality by providing a low-dimensional representation in which distances between data points are not subject to the concentration phenomenon.

The consensus around the low intrinsic dimension of the data is generally formulated using

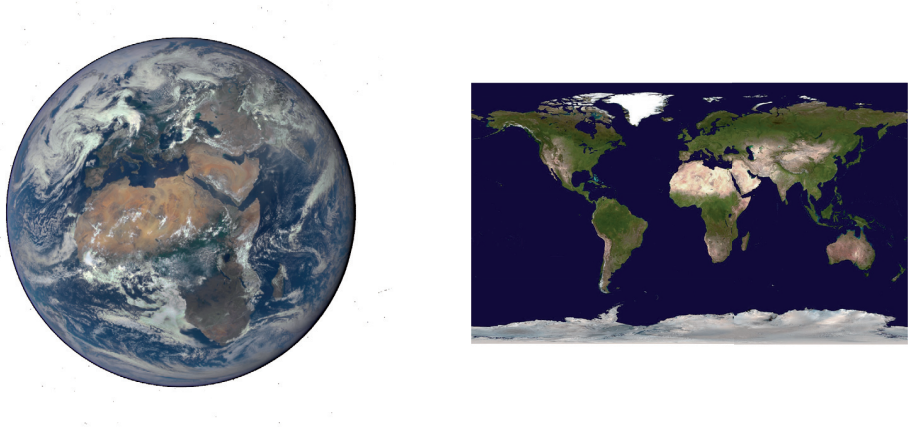

Figure 3 – **The mapping problem.** The question of creating a 2D map of the surface of the Earth is a very old problem which has occupied cartographers and travellers from a long time. Mathematically speaking, the idea is to map the surface of a 3D sphere to a 2D plane while preserving geometric properties. Sadly, this task is impossible to do without making a cut, i.e. a strong discontinuity in the mapping, allowing to flatten the surface of the sphere. In addition, not all geometrical features can be preserved in a single mapping, which has inspired many different projections. For example, the equirectangular projection (the one shown on the right) preserves distances along meridians, the Mercator projection was used for navigation as it makes lines of constant bearing straight or the Mollweide projection which is equi-area displays meridians as ellipses. As we see, considered globally, mapping a sphere is problematic, but, when considering maps locally, they may be faithful representations of the original surface of the sphere.
Image credits : CC-BY-SA NASA.

geometrical and topological terms using the manifold hypothesis: the data is well represented by a smooth or (piece-wise smooth) manifold of dimension corresponding to the intrinsic dimension, embedded in the original high-dimensional space. But since data points are discrete elements by nature, they can only represent a sampling of the continuous manifold, up to some residual noise. While the continuous manifold is both hard to capture and to formalize, the graph created by connecting the closest data points together is a good discrete equivalent as far as the geometrical aspects are concerned. Indeed, as the number of samples grow, essential operators of a nearest neighbours graph are proven to converge to the ones associated to the sampled manifold [21, 20].

The need for nearest neighbours graphs, or equivalently, sparse similarity matrices, which happen to be at the core of many dimensionality reduction methods, arises as a solution to an issue unrelated to the curse of dimensionality. Indeed, the need to do pairwise comparisons (i.e. similarity measures or distance computations) between all data points implies necessarily a quadratic number of such comparisons. For large datasets, even if the computational cost is low, the memory requirements can become unmanageable very rapidly. As an example: a complete similarity matrix for a dataset containing 10^9 data points would have more than 400

Petabytes memory footprint. Since this is intractable, a simple solution to this issue is to only consider the closest (or most similar) neighbours, which is also sound in a geometrical point of view, as we just saw above. A major difficulty that remains is to efficiently find the closest neighbours without performing exhaustive searches. This obstacle is generally mitigated by only looking for approximate nearest neighbours as we will detail in Chapter 1.

The technique used to reveal the structure of data common to most of the core dimensionality reduction methods is the spectral decomposition of similarity (or covariance) matrices (e.g. PCA [139], Isomap [188], Laplacian Eigenmaps [18]). In order to cope with large datasets, sparsity is needed, but the eigendecomposition still provides valid results. Developing from this context where graphs and eigendecomposition meet, most of the work of this thesis is naturally rooted to Graph Signal Processing. This is a relatively new field which is founded on the use of the spectral decomposition of the graph Laplacian to define a Fourier basis, filters and operators pendant to traditional signal processing tools.

Supplementary to the issues already mentioned, the question we are trying to answer raises another problem related to complex data. Until now, the assumption was that *complex* only meant high-dimensional, but an additional hidden hypothesis was implicitly made: regularity of the data formalism. In other words, it was assumed that the dimensionality was constant over all the samples of a dataset, although this assumption is rarely true for natural data. A simple example is to consider a set of images of a monument taken from the Internet: the images will almost never contain the same number of pixels. As such their vectorized form is irregular and establishing a notion of distance or similarity between them is non-trivial. The general solution to this problem is feature extraction: representing irregular data using a regular formalism containing sufficient information so that the discriminatory power needed to compare objects is kept. Feature extraction is traditionally domain-specific as the relevant information varies between the different data of interest; indeed, texts, images or sounds are naturally very dissimilar and need specialized extraction procedures. While specialized techniques for feature crafting were the rule, the emergence of advanced deep-learning architectures (e.g. deep auto-encoders) is sketching a new trend of generalized and automatic feature extraction.

Taking into account the ramifications entailed by the question asked at the beginning of this section, we can summarize the main issues at hand: high-dimensional data means facing the curse of dimensionality, large datasets imply a quadratic number of similarity measures and similarity measures are not well defined for irregular objects. Taking into account the current solutions to these problems, we will address the original question of this thesis using the following framework:

Data agnostic graph-based embedding.

This framework is illustrated in Figure 4 and we examine it in more details in the following paragraphs in order to match the issues and their proposed solutions. First, *data agnosticism*

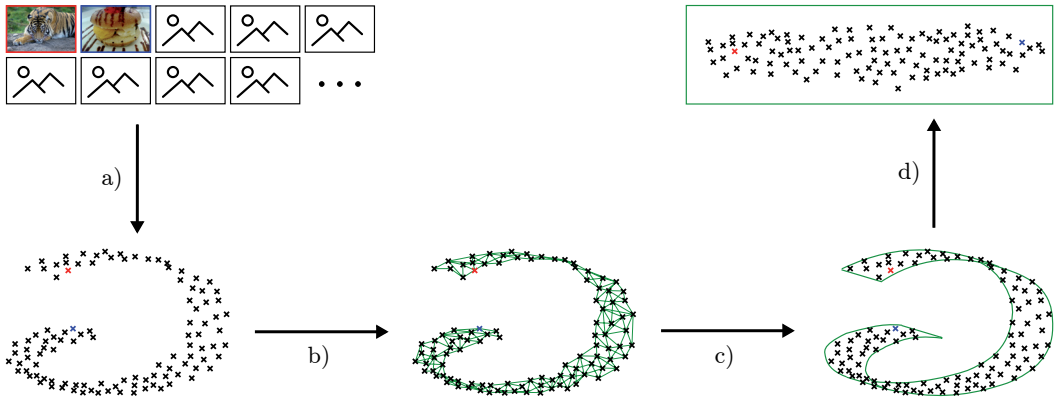

Figure 4 – **Data agnostic graph-based embedding.** In this figure, we illustrate the complete pipeline of the framework we use throughout the thesis using the example of embedding a collection of images. The first step (a) corresponds to the feature extraction process mapping complex images to high-dimensional vectors. Step (b) depicts the construction of the nearest-neighbours graph. The step (c) illustrates the intrinsic structure of the data revealed by graph and the step (d) shows the actual dimensionality reduction procedure to get the final mapping. In the figure, we also show two images, the first one of a tiger and the second one of a dessert, highlighted in red and blue respectively. This pair of pictures happen to have similar visual features: yellow-orange colours with brown-black stripes, however, they are semantically very dissimilar. This notion is displayed in the pipeline with the red and blue crosses which are quite close in the general space, but actually far away following the structure of the data.

is related to the fact that the processing of the data is always done by considering the feature space as the original space, i.e. assuming that feature extraction has been used to regularize the raw data if needed. It will not be a central topic of this thesis, and we will always assume that the features have been standardized using a feature-extraction procedure. Next, the concept of *graph-based embedding* solves both the curse of dimensionality and the quadratic size of the similarity matrix. Indeed, as it was hinted above, nearest neighbours graphs encode the manifold information hidden in the data and are inherently sparse.

Considering from now that this work is rooted in this framework, we can look further for what is done in the state-of-the-art and what issues remain.

- First, let us acknowledge that there exist many algorithms for dimensionality reduction today, and some of them are very effective (see Chapter 1). Nevertheless, there are only a few comparative studies of dimensionality reduction methods in the literature and they rely on a limited set of quantitative measures. In the context of visualization, the quality is still often assessed using visual inspection.
- Second, most traditional techniques (e.g. [18, 154]) provide an embedding of some

target dimension without specializing for the task at hand, while it would make sense that embeddings used for visualization or clustering, for example, do not have identical constraints. Indeed, reducing the dimension for a clustering task only minimizes the k-means cost, while an embedding for visualization needs especially to avoid concentration around zero. While those constraints are linked, their importance could benefit from being made task-specific.

- Third, all dimensionality reduction methods suffer from scalability issues, which are inherent to the complexity of the problem at hand. Hence, methods which today are deemed efficient still take hours to process datasets containing the order of 10^6 points (see e.g. [197, 187]). This time complexity makes it unfeasible to tackle problems of size 10^9 or bigger, which exist today.
- Fourth, another issue related to the scalability problems just mentioned, is the excessive discriminatory power of the feature space. Since features often live in high-dimensional spaces, very similar points are still considered as two distinct elements, and thus contribute to the size of the dataset. This very fine discrimination is however rarely needed and more probably even problematic. Let us take an example: consider extracting audio features from different guitar songs. If we assume that the features consist of the frequency content extracted from a small time window, identical chords played at different times will most probably be assigned slightly dissimilar frequency values. The consequence is that two chords that one would consider identical perceptually (or semantically) will be considered as two distinct points in the feature space.
- Finally, the feature-extraction process loses some information, by definition. Those losses are two-fold: on one side, the amount of information kept is reduced to obtain a regular representation, on the other side contextual information is often completely discarded. In other words, extracting features is generally done without considering their original organization. Let us take a few examples: when extracting keypoints descriptors in images, their location is not part of the feature, when extracting audio features from songs, their position in time is not kept, when extracting text features, the position in the global article is generally discarded. All this original structure is usually lost while applying dimensionality reduction to the features while it could provide key insights into further processing.

The goal of this thesis is to address the issues we described above using the data-agnostic graph-based embedding framework. In particular, the proposed solutions are:

- quantitative measures to assess the quality of embeddings using fine metrics.
- efficient and scalable dimensionality reduction methods for both clustering and visualization.
- organization and simplification of the feature-space to gain control on both its size and discriminatory level using a combination of tree structures and hierarchical graphs.

The next section details how those main items are distributed in the different chapters.

Thesis structure and contributions

In **Chapter 1** we cover the topics on which the rest of the thesis is founded. More specifically, the different elements of the data agnostic graph-based embedding framework, introduced in the previous section, are detailed. The two main sections cover dimensionality reduction and graph signal processing respectively, since they are at the core of the contributions we make throughout this work. None of those two topics is covered exhaustively, but all the elements used subsequently are explained in a comprehensive way.

In **Chapter 2** we address quantitative evaluations of embeddings and dimensionality reduction. We start by reviewing the state-of-the-art in both unsupervised and supervised formal measures for dimensionality reduction algorithms. Then, we propose new supervised techniques to assess different quality metrics related to various properties of embeddings. In particular, our contributions are the following: first, a global measure of quality based on clusterability and balanced graph cuts. Then, we introduce a measure of class-based spread using the length of active random walks. The latter is made possible using a new distance and an active sampling technique based on the graph localization operator. Finally, we provide a way to compute the amount of sparse noise based on statistics of the norm of the graph localization operator. Our contributions are backed by various experiments on both synthetic and natural datasets.

Chapter 3 focuses on the problem of estimating the eigenspace of the graph Laplacian. While this specific problem has attracted a lot of interest, we introduce a method based on random signal filtering. In this context, our chapter proposes various improvements to the field, whose main contributions are: a very efficient scheme for the estimation of eigenspaces using filtering of random graph signals, a proven tight bound for the number of random signals needed for perfect recovery, algorithms and implementations with practical considerations regarding filter design, fast filtering, and numerical stability and an accelerated method for the count of eigenvalues in a given range. The complexity of our technique is compared to related methods and we present experiments in both clustering and visualization tasks showing the superior scalability of this method compared to the state of the art.

In **Chapter 4** we approach the scalability issue of dimensionality reduction in a very generic way. We propose a scheme inspired by compressive sampling in which only a fraction of the data points are embedded and used to infer information on all samples. The sampling procedure used to select the nodes is first presented in detail and theoretical bounds are provided. The diffusion of the information is presented as a transductive learning problem on graphs and different solutions are proposed. Our main contributions are: graph sampling schemes and theorems stating the minimum number of samples necessary to capture energy everywhere and new transductive learning algorithms to extend the embedding information computed on the samples to all data points using localized low pass graph filters. Finally, experiments on various real-world datasets are used to show the validity of the method.

Finally, the main topic of **Chapter 5** is a hierarchical structure, called the Clustering Graph

Tree, proposed to organize the feature-space. It combines a traditional tree structure to a multi-resolution graph hierarchy. First, this framework is proposed to provide control on the discriminatory level of the features by using the meta-features found on intermediate tree levels. Second, we show that it can be equipped with very efficient coarsening and refining operators allowing to down-sample and up-sample graph signals between different levels. These operators are also used to create a new meta-algorithm for dimensionality reduction which can be used to accelerate any other embedding algorithm. Third, we contribute a feature-encoding scheme based on tree traversal that can be used to establish structured colour mappings and a one-dimensional embeddings. Finally, we provide a wide range of experiments to demonstrate the validity of our contributions.

There is a single light of science,
and to brighten it anywhere
is to brighten it everywhere.
— Isaac Asimov

1 Background

In this chapter, we establish the scientific background on which the different contributions of this thesis are based. Since the main focus of this work is on dimensionality reduction, it forms the first of the two main background topics. We start by introducing different point of view to motivate the need for dimensionality reduction in general. Then we briefly introduce several methods, from traditional ones to the current state-of-the-art solutions. We conclude the section by highlighting the principles underlying the different algorithms presented.

As we will see, those principles will lead us to graphs, which are used both as theoretical concepts to model the structure of the data and as the tools used in several dimensionality reduction methods. The claim we make in this work is that their underlying presence is not coincidental but rather an emerging evidence that graphs are integral in the understanding of hidden structures in data. For this reason, they will be used as the main tools, both theoretically and experimentally, to approach the different themes we will explore. To achieve this goal, we follow a conceptual approach called Graph Signal Processing (GSP), that we introduce, starting from the basic concepts, in the second section of this chapter.

The motivation for using graphs in dimensionality reduction problems is that they are well suited to represent pairwise relationships in a set of objects; and they indeed are naturally present in many algorithms, as we will see in our overview. The reason for this presence comes from the link between graphs and the hypothesis that data lives on manifolds. Nevertheless, despite the fact that graphs are often used (sometimes implicitly) in dimensionality reduction, modern tools from GSP have not been applied much in this setting and this work aims at filling this gap.

In the second section, we give an introduction to GSP starting from basic definitions in spectral theory to its fundamental concepts and tools: the graph Fourier transform, filtering, localization, and optimization. We conclude the section by introducing the problem of efficient nearest neighbour search and presenting different solutions, which are incidental in the creation of k-Nearest Neighbours (kNN) graphs.

1.1 Dimensionality reduction

Dimensionality reduction is a broad topic but its objective can be stated first using a literal interpretation: given a set of points living in a space of dimension D, find a set of points in a target dimension $d < D$ such that every original point is associated to a point in dimension d.

More formally, we call the original set of points $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with $|\mathcal{X}| = N$ and $\mathbf{x}_i \in \mathbb{R}^D$ and the new set of points $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_M\}$ with $|\mathcal{Y}| = M$ and $\mathbf{y}_i \in \mathbb{R}^d$. The sets $\mathcal{X}$ and $\mathcal{Y}$ are associated to the matrices $\mathbf{X}$ and $\mathbf{Y}$ respectively, where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{N \times D}$ and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N] \in \mathbb{R}^{M \times d}$. Note that it is assumed that points are described by real numbers, but most of the theory applies identically to other formalisms (e.g. complex or integers numbers). Also, for most of this thesis, it is considered that $M = N$. The set $\mathcal{X}$ and matrix $\mathbf{X}$ will be called interchangeably input, high-dimensional or original data and the set $\mathcal{Y}$ and matrix $\mathbf{Y}$ will be called output, low-dimensional data or embedding.

1.1.1 Motivation

Now that the basic definitions of dimensionality reduction are stated, the motivation to use such techniques comes from three different angles. The first point of view is the one of information, or more precisely of compression: considering that data contains redundant parts, one can search for a more compact and efficient way to represent it. The second point of view, which will be central in this work, is the geometric (or manifold) view ; it comes from the fact that high-dimensional data possess some structure which can be difficult to extract and could be better expressed using its intrinsic dimension (i.e. only its degrees of freedom) instead of its original extrinsic dimension. The third point of view is the one of visualization whose goal is to lower the dimension of the data in order to be represented in accessible ways (typically 2D or 3D). Since all of these three motivations for dimensionality reduction are relevant for this work and provide key insights in the following chapters, we analyse them in more depth.

Information Theory

The first focus is the one of information theory. The idea is to extract only the essential information from the data, with the central concept of entropy defining a frontier between lossless and lossy reduction. Using this prism, preserving the geometry of the data is not a constraint and any entropy achieving or lossy compression algorithm can be used to encode the data in a more compact form.

Lossless compression schemes usually aim to be close to the Shannon entropy of the data and they do so by adapting to a bit stream of interest. That is, if we consider using some implementation of the Lempel–Ziv algorithm [220], either files are considered independently and each is assigned to a dictionary, or they are considered in batch and thus possess one

global dictionary. In each case, the bit streams themselves are compressed which results in the file size (or dimension) being reduced globally. The compressed representations are inherently non-regular as they only rely on the empirical statistics of the original data. Furthermore, even assuming identical target dimensions, measuring distances in the compressed space is meaningless, since the geometry of the data is never taken into account.

Lossy schemes may also not well preserve the geometrical aspects of the original data (e.g. isometries). Indeed, if we take JPEG compression[1] for example, the sub-sampling and block-DCT are both meaningful when viewed as geometrical transformations, but the last encoding step (Huffman coding) is not. Nonetheless, some hashing algorithms provide the notable exception of dimensionality reduction algorithms originating from information theory principles whose goal is to preserve pairwise distances. They are broadly separated into two classes: Locality-Sensitive Hashing [71, 38, 113] and Locality-Preserving Hashing [87, 194]. The former being independent of the data and the latter not.

As we see, dimensionality reduction from the information theory perspective is not originally designed to preserve geometric isometries or, more specifically, pairwise distances between samples. In this work, the notion of compactness of representation or, compression, is only a byproduct of the dimensionality reduction procedure, as the main goal will be to preserve some geometrical properties.

Geometry

The geometrical point-of-view in dimensionality reduction (and in data processing in general) simply states that data possess an intrinsic geometrical structure which ought to be extracted or preserved by a dimension reduction procedure. The principle has been derived and specialized in many ways but two trends will be used throughout this work: the distance preservation and the manifold hypothesis.

First, the distance (or similarity) preservation hypothesis is quite intuitive and states that dimensionality reduction should preserve pairwise distances between data points (up to a renormalization). It is indeed a sound principle, when looking for geometric structure, to look for isometries. Indeed, by making few assumptions on the geometry one can derive very general methods while still preserving a fundamental topological property.

The *manifold hypothesis* makes a stronger assumption about the data in that it assumes that the points are sampled from (or live close) to a manifold of intrinsic dimension d embedded in a space of extrinsic dimension D (with $d < D$). It is often assumed that the manifold is smooth, but this additional constraint is not necessary. Indeed, the manifold can be non-Riemannian or piece-wise smooth and still stand as a valid hypothesis for dimensionality reduction. This assumption is more specific than distance preservation since it puts a stronger geometrical constraint on the data. Its key component is the notion of intrinsic dimension which could be

[1] http://www.itu.int/rec/T-REC-T.81

defined as the minimal number of parameters (or degrees of freedom) needed to be able to represent the data without ambiguity.

Both the distance-preservation and manifold hypotheses are based on a meaningful notion of distance between data points. Now, as it was motivated in the introduction, data is often (very) high-dimensional. The first and most problematic consequence of this fact is known as the *curse of dimensionality*.

In order to better understand the fundamental problem of high-dimensional spaces, we must consider the phenomenon of concentration of norms and distances. To do so, we first state the following lemma adapted from [27, Proposition 2] and [204, Theorem 1.1].

Lemma 1. *Let $\boldsymbol{x} = [x_1, \ldots, x_D]' \in \mathbb{R}^D$ be a random vector whose components x_k, $1 \leq k \leq D$, are independent and identically distributed (i.i.d) with a finite fourth-order moment.*

Then the mean $\mathbb{E}[\|\boldsymbol{x}\|] = \mu_{\|\boldsymbol{x}\|}$ and variance $\mathrm{Var}[\|\boldsymbol{x}\|] = \sigma^2_{\|\boldsymbol{x}\|}$ of the Euclidean norm of $\boldsymbol{x}$ are

$$\mu_{\|\boldsymbol{x}\|} = \sqrt{aD - b} + \mathcal{O}(D^{-1}), \tag{1.1}$$

$$\sigma^2_{\|\boldsymbol{x}\|} = b + \mathcal{O}(D^{-\frac{1}{2}}), \tag{1.2}$$

respectively, where a and b are polynomials of order two of the four first moments of $\boldsymbol{x}$.

This implies that

$$\lim_{D \to \infty} \frac{\sigma_{\|\boldsymbol{x}\|}}{\mu_{\|\boldsymbol{x}\|}} = 0. \tag{1.3}$$

This result shows that, as the dimensionality increases, the relative standard deviation of the norm tends to zero. It is generally referred to as the norm concentration in high-dimensional spaces because of its impact on the ratio between $\|\boldsymbol{x}\|$ and $\mu_{\|\boldsymbol{x}\|}$. Indeed, we see that, for any $\epsilon > 0$,

$$\mathbb{P}\left(\frac{\|\boldsymbol{x}\|}{\mu_{\|\boldsymbol{x}\|}} - 1 \geq \epsilon\right) = \mathbb{P}\left(\|\boldsymbol{x}\| - \mu_{\|\boldsymbol{x}\|} \geq \epsilon \mu_{\|\boldsymbol{x}\|}\right) \leq \frac{\sigma^2_{\|\boldsymbol{x}\|}}{\epsilon^2 \mu^2_{\|\boldsymbol{x}\|}} \to 0, \text{ as } D \to \infty, \tag{1.4}$$

using Chebyshev's inequality and Lemma 1. This shows that, in probability, the ratio $\frac{\|\boldsymbol{x}\|}{\mu_{\|\boldsymbol{x}\|}}$ tends to 1 as the dimensionality increases, i.e. high-dimensional norms concentrate around the mean. This result also implies that high-dimensional random vectors are distributed very close to the surface of a ball of radius $\mu_{\|\boldsymbol{x}\|}$.

Intuitively, we see that this atypical distribution of the mean norm may impact distances. To see that, it is common to analyse the behaviour of the maximum and minimum distance between a query point and random points in high dimension. To simplify the equations, and without loss of generality, we assume that the query point is located at the origin. The

following lemma, adapted from [81, Theorem 1 and 2] and [26, Theorem 1], shows the impact of high-dimensionality on the distances.

Lemma 2. *Let* $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ *with* $\mathbf{x}_i = [x_1, \ldots, x_D]' \in \mathbb{R}^D$ *be a set of random vectors whose coordinates* x_k, $1 \leq k \leq D$, *are independent and identically distributed (i.i.d). Let the maximum and minimum distances between the origin and the set of points be defined as* $d_{\max} = \max_{i \in [1,N]} \|\mathbf{x}_i\|$ *and* $d_{\min} = \min_{1 \in [1,N]} \|\mathbf{x}_i\|$ *respectively. Provided that*

$$\lim_{D \to \infty} \frac{\sigma_{\|\mathbf{x}\|}}{\mu_{\|\mathbf{x}\|}} = 0,$$

then the two following statements hold :

$$\frac{d_{\max} - d_{\min}}{d_{\min}} \to_p 0, \tag{1.5}$$

and

$$\lim_{D \to \infty} \mathbb{E}[d_{\max} - d_{\min}] = C, \tag{1.6}$$

with C *a constant and* $\to_p$ *meaning convergence in probability.*

These results, generally referred to as the distance concentration in high dimensions, state that, for the Euclidean distance, the relative contrast $\left(\frac{d_{\max} - d_{\min}}{d_{\min}}\right)$ tends to zero and the contrast $(d_{\max} - d_{\min})$ tends to a constant when the dimensionality grows. In other words, it means that for very high dimensions, the ratio between the smallest and largest distances from a query point is insignificant and that all points appear to be similarly close. The main consequence of this surprising fact is that the notion of nearest neighbour becomes meaningless.

In addition, while those results are asymptotic, the concentration of distances may occur even for relatively low (e.g. < 15) dimensions [26]. Although those facts seem to indicate that accomplishing any distance-based measures in high dimensions is meaningless, let us recall that the underlying assumption was i.i.d distribution of the samples. Fortunately, it happens that real-world data is not randomly distributed in space and is generally structured, which can be sufficient to overcome this problem. Indeed, it happens that nearest neighbours are still meaningful when data is clustered [26], or more generally when the samples are concentrated in a subspace of smaller intrinsic dimension [26, 204, 79].

Assuming that data lives on a manifold of intrinsic dimension d, one may want to directly estimate its geometry. However, since we only have access to (possibly noisy) samples of the continuous underlying manifold, estimating its parameters, even assuming d is known, is a very ill-posed problem. To understand this fact, let us consider high-dimensional data which we know live in a one-dimensional manifold (i.e. a line), then any number of smooth one-dimensional curves can be fitted to go through all the high-dimensional points. Inversely, it might be easier to try to estimate the intrinsic dimension d without estimating the precise geometry of the manifold.

The particular problem of intrinsic dimension estimation has been well studied and is of particular importance as a pre-processing task for further dimensionality reduction. Indeed, most of the techniques we will present in the next section take the target dimension as a parameter. Assuming that the manifold is linear and the samples are noiseless, d can be easily estimated using Principle Components Analysis (PCA). However, when one of the two assumptions is not valid, especially when the manifold is non-linear, PCA overestimates d [198]. To cope with non-linearity, other intrinsic dimensionality estimation use nearest neighbours, either via volumes of d-dimensional spheres to estimate fractal dimensions [63, 95] or using multi-scale analysis [115, 116, 108]. The latter has a huge advantage over the fractal-dimension approaches as only $\mathcal{O}(d \log d)$ samples are needed instead of $\mathcal{O}(2^d)$.

Although estimating the intrinsic dimension is feasible, it is not always necessary depending on the application. Indeed, in visualization tasks, the dimension is usually fixed to two or three. For clustering tasks, the target dimension can be set according to the number of clusters needed.

Visualization

The last point-of-view regarding dimensionality reduction is the one of visualization. It is different from the two previous ones as it is not usually defined using quantitative constraints. The main objective comes from the fact that we, as humans, are only able to visualize data when it is represented as low dimensional objects: basically from one to three dimensions. A few extra dimensions can be accommodated using variations over time, colour or size, for example.

Related to its non-formal definition, there is no precise goal for visualization, since it may vary from task to task. A very broad goal is to help understand the data. More specifically, this translates to revealing relationships between data points: inferring similarity from proximity, interpreting communities from clusters aggregation, and so on.

Dimensionality reduction for visualization is used in many domains such as : graph drawing [206, 149], biomedical data [114, 32], wood inspection [136], geospatial data [184], genetic analysis [212, 189, 61], etc.

1.1.2 Methods

Now that the general context and motivation for dimensionality reduction are given, we will take a look at the techniques, from the seminal ones to the state-of-the-art. This review does not pretend to be exhaustive and its purpose is only to lay the foundations on which the rest of this work builds. The goal of the overview given here is to cover the fundamental principles of the most famous or successful techniques. We refer the interested reader to [109, 198] for more complete reviews of dimensionality reduction literature.

We start by briefly explaining the different methods and then analyse their unifying principles. For more details, please refer to the original works. For all the following methods we will consider that $\mathbf{X} \in \mathbb{R}^{N \times D}$ and $\mathbf{Y} \in \mathbb{R}^{N \times d}$. To simplify the algebra, and without loss of generality, we assume that $\mathbf{X}$ is centred, i.e. $\mathbb{E}[\mathbf{X}] = 0$.

PCA [139, 83]

The Principal Components Analysis (PCA), a seminal work in the field, dates back to the early 20^{th} century. Essentially, it is an orthogonalization procedure, which is also known as the empirical version of the Karhunen–Loève theorem, i.e. the Karhunen-Loève Transform (KLT). It is not, by definition, a dimensionality reduction method, since it yields a new basis spanning the original space. The reduction occurs when projecting only on a subset of the resulting orthogonal components. More formally, PCA works as follows: first, the sample covariance matrix $\mathbf{C_X} = \mathbf{X}^T\mathbf{X}$ is computed. Second, it is decomposed as $\mathbf{C_X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ using the Singular Value Decomposition (SVD), which provides the solution to the eigenproblem $\mathbf{C_X}\mathbf{U} = \mathbf{\Lambda}\mathbf{U}$. Here, $\mathbf{U}$ is the matrix of eigenvectors and $\mathbf{\Lambda}$ the diagonal matrix of eigenvalues of $\mathbf{C_X}$. Finally, the orthogonal representation is simply obtained by a linear mapping $\mathbf{Z} = \mathbf{XU}$. The dimensionality reduction is done when using only the first d components, i.e. $\mathbf{Y} = \mathbf{XU}_d$, where $\mathbf{U}_d$ is the matrix containing the first d columns of $\mathbf{U}$ (sorted in decreasing order of the eigenvalues). By construction, such a reduction procedure is the one maximizing the retained variance of the data while minimizing the ℓ_2 reconstruction error $\|\mathbf{X} - \mathbf{YU}^T\|_2^2$.

MDS [101, 190]

Another traditional technique happens to provide the same solution set as the one given by PCA : classical Multi Dimensional Scaling (MDS). The goal of MDS is to provide a linear mapping $\mathbf{Y} = \mathbf{XP}$ which minimizes the cost function :

$$E(\mathbf{Y}) = \sum_{ij} (\|\mathbf{x}_i - \mathbf{x}_j\|^2 - \|\mathbf{y}_i - \mathbf{y}_j\|^2). \tag{1.7}$$

It has been shown that the matrix minimizing $E(\mathbf{Y})$ is given by the eigendecomposition of the matrix $\mathbf{XX}^T$. That is, if we have $\mathbf{XX}^T = \mathbf{V}\Sigma\mathbf{V}^T$, the minimizer of $E(\mathbf{Y})$ is given by $\mathbf{P} = \sqrt{\Sigma}\mathbf{V}$. To establish the link between PCA and MDS, let us first see that, assuming $\mathbf{X} = \mathbf{USV}^T$:

$$\mathbf{XX}^T = (\mathbf{USV}^T)(\mathbf{VSU}^T) = \mathbf{US}^2\mathbf{U}^T, \tag{1.8}$$

$$\mathbf{X}^T\mathbf{X} = (\mathbf{VSU}^T)(\mathbf{USV}^T) = \mathbf{VS}^2\mathbf{V}^T. \tag{1.9}$$

First we see that the eigenvalues are identical, i.e. $\Lambda = \Sigma = \mathbf{S}^2$. Then, we see that the eigenvectors of MDS and PCA are related by $\mathbf{XV} = \mathbf{SU}$, which is why both problems give very similar solution sets.

Note that the matrix $\mathbf{XX}^T$ is an inner-product matrix which is, in fact, a doubly centred Eu-

clidean distance matrix, as shown in [198]. The ability to replace the distance matrix with other similarity or distance matrices is why MDS is considered more of a class of methods rather than a specific algorithm.

Isomap [188]

The concept of Isomap is very closely related to the general formulation of MDS but uses a non-linear model. Indeed, instead of using a Euclidean distance matrix, the similarity is based on geodesic distances. The method works as follows: first, a kNN graph $\mathcal{G}$ is built from the data $\mathbf{X}$ (see Section 1.2.4 for more details on the graph construction). A complete matrix of pairwise geodesic distances is computed using traditional shortest-path algorithms such as Djikstra [57] or Floyd-Warshall [67, 207]. Then, the MDS procedure is followed, i.e. the distance matrix is doubly centred and its eigenvectors extracted by spectral decomposition.

Laplacian Eigenmaps [17]

The use of the graph structure in Isomap has two main scalability issues: first, the distance matrix is dense and second, exact all-pairs shortest paths have a high complexity. Keeping the graph structure while trying to avoid those issues, the original authors proposed to use the graph Laplacian $\mathbf{L}$ of the kNN graph $\mathcal{G}$ obtained from the data to solve the generalized eigenproblem $\mathbf{Ly} = \lambda\mathbf{Dy}$, with $\mathbf{D}$ the degree matrix of $\mathcal{G}$ (see Section 1.2.1 for precise definitions of the graph nomenclature). Here, the solution $\mathbf{y}$ is called the Laplacian Eigenmaps (LE). This eigenproblem is actually proven to be the solution of the minimization of the Dirichlet energy $\mathbf{Y}^T\mathbf{LY}$, with the additional constraint $\mathbf{Y}^T\mathbf{DY} = 1$ to avoid the trivial solution $\mathbf{Y} = 0$.

An alternative view of this method is to see it as computing an eigenspace of the random walk Laplacian. Indeed, if we define the random walk Laplacian as $\mathbf{P} = \mathbf{D}^{-1}\mathbf{L}$, then the equation above can be rewritten as $\mathbf{Py} = \lambda\mathbf{y}$. Thus, Laplacian Eigenmaps aims to find the eigenspace of $\mathbf{P}$ and use it as an embedding for visualization. The dimensionality reduction occurs when using only a subset of the columns of $\mathbf{P}$.

LLE [154]

On the track of methods leveraging graphs, Local Linear Embedding (LLE) uses the neighbourhood structure to estimate and preserve the inherent manifold structure of the data locally. In short, the method considers each data point as a linear combination of its neighbours. It does so by fitting a hyperplane to the point and its neighbours. Due to the locality and linearity of such a linear model, its reconstruction weights are preserved by linear embedding on lower dimensions.

More formally, assuming that the high-dimensional points are defined as $\mathbf{x}_i = \sum_j^k w_{ij}\mathbf{x}_j$, with k the parameter defining the number of nearest neighbours and w_{ij} the reconstruction weights, the goal is to find a good embedding $\mathbf{Y}$ such that $\|\mathbf{y}_i - \sum_j^k w_{ij}\mathbf{y}_j\|_2^2$ is minimized. Similarly to Laplacian eigenmaps, there is a trivial solution $\mathbf{Y} = 0$ which imposes a constraint on $\mathbf{Y}$, e.g.

$\|\mathbf{Y}_{.,j}\|^2 = 1, \forall j$.

The original authors showed that the solution to this problem is given by the eigendecomposition of the matrix $\mathbf{K} = (\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})$ where $\mathbf{W}$ is the matrix containing the reconstruction weights $\mathbf{W}_{i,j} = w_{ij}$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected and $\mathbf{W}_{i,j} = 0$ otherwise.

An interesting interpretation of this solution is to consider $\mathbf{W}$ as a normalized weight matrix for a graph $\mathcal{G}$. Thus, by definition, the matrix $(\mathbf{I}-\mathbf{W})$ is the normalized Laplacian of $\mathcal{G}$, i.e. $\mathbf{L}_n = (\mathbf{I}-\mathbf{W})$. Using the spectral decomposition of the Laplacian we have $\mathbf{L}_n = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$. This allows us to reformulate the solution as:

$$\mathbf{K} = (\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W}) = \mathbf{L}_n^T\mathbf{L}_n = (\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T)(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T) = \mathbf{U}\mathbf{\Lambda}^2\mathbf{U}^T. \tag{1.10}$$

This result shows that the eigenvectors provided by LLE are the ones of the normalized Laplacian corresponding to the graph whose weighted adjacency matrix corresponds to the reconstruction weights.

Sammon Mapping [155]

All the methods presented above were related to the resolution of convex problems, i.e. whose cost functions were convex. Convex energy minimization in dimensionality reduction has two main issues: first, it needs artificial constraints to avoid trivial solutions (e.g. $\mathbf{Y} = 0$) and second, ℓ_2 cost functions are sensitive to the scale of the data. For example, the cost function of MDS (1.7) penalizes more the discrepancies for large pairwise distances than for small ones. This can have non-desirable effects since, similarly motivated by the manifold hypothesis, closest points are the ones for which distances make more sense.

Using this assumption, a non-convex variant of MDS, called Sammon Mapping, was proposed using a cost function normalized by the high-dimensional distances. More formally, the cost function can be written as :

$$E(\mathbf{Y}) = \frac{1}{\sum_{i<j}\|\mathbf{x}_i-\mathbf{x}_j\|}\sum_{i<j}\frac{(\|\mathbf{x}_i-\mathbf{x}_j\|-\|\mathbf{y}_i-\mathbf{y}_j\|)^2}{\|\mathbf{x}_i-\mathbf{x}_j\|}. \tag{1.11}$$

Since the problem is non-convex, the minimization of the cost function is usually done using steepest descent or pseudo-Newton methods.

t-SNE [118, 197]

In recent years, other non-convex approaches have been proposed using probabilistic modelling. One of the first such methods was Stochastic Neighbours Embedding (SNE) [82], whose principle is as follows: the high-dimensional points are modelled using a pairwise probability matrix $\mathbf{P}$ for which close points are assigned high transition probabilities and distant points low values. Similarly, the embedded points are modelled using another probability matrix $\mathbf{Q}$, constructed with the same principle. The objective is then to minimize the Kullback-Leibler

divergence between the two probability distributions. In the original work, both **P** and **Q** are constructed using Gaussian distributions and the optimization is done using gradient descent.

More recently, t-distributed Stochastic Neighbour Embedding (t-SNE), a now famous state-of-the-art technique for visualization has been proposed with the goal of improving SNE. It leverages the use of a symmetric version of the SNE cost function (as previously introduced in Symmetric SNE [52]) and, more importantly, proposes the use of a heavy-tail distribution for **Q**. This change allows solving the main issue of most visualization techniques: the crowding problem. This refers to the concentration around zero of the embedded points allegedly caused by two effects: the tendency to be close to the trivial solutions of convex problems (e.g. by enforcing the constraints via degenerated solutions) and, as previously mentioned, by not sufficiently preserving the small pairwise distances.

More formally, t-SNE uses a Gaussian distribution for **P**:

$$\mathbf{P}_{ij} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2)/2\sigma_i^2}{\sum_{k=l}\exp(-\|\mathbf{x}_k - \mathbf{x}_l\|^2)/2\sigma_i^2}. \tag{1.12}$$

This definition is identical to Symmetric SNE and the variance of the Gaussians (i.e. σ_i^2) is determined in the same way as for the original SNE, i.e. by adapting the variance in order to get a constant perplexity over the samples.

The low-dimensional distribution **Q** is defined, unlike other methods, using a heavy tail, t-Student distribution:

$$\mathbf{Q}_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k=l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}. \tag{1.13}$$

The justification for the use of such a distribution is that it behaves as an inverse square law for large pairwise distances. This means that the joint probabilities are not really affected for points that are already sufficiently far apart.

Finally, the cost function is simply the Kullback-Leibler divergence between P and Q :

$$E(\mathbf{Y}) = \mathrm{KL}(\mathbf{P}\|\mathbf{Q}) = \sum_i \sum_j \mathbf{P}_{ij} \log \frac{\mathbf{P}_{ij}}{\mathbf{Q}_{ij}}, \tag{1.14}$$

whose gradient is :

$$\frac{\partial E(\mathbf{Y})}{\partial \mathbf{y}_i} = 4\sum_j (\mathbf{P}_{ij} - \mathbf{Q}_{ij})(\mathbf{y}_i - \mathbf{y}_j)(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}. \tag{1.15}$$

This gradient, which is derived in [118], happens to be simpler than the ones of SNE and Symmetric SNE. Indeed, it can be efficiently computed in matrix form using a graph formalism. Let us define $\mathbf{W} = (\tilde{\mathbf{P}} - \tilde{\mathbf{Q}})$, with $\tilde{\mathbf{P}_{ij}} = \mathbf{P}_{ij}(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$ and $\tilde{\mathbf{Q}_{ij}} = \mathbf{Q}_{ij}(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$ and consider it as a weight matrix. Let $\mathbf{L} = \mathbf{D} - \mathbf{W}$ be the combinatorial Laplacian associated to **W**.

Then the gradient (1.15) simply becomes :

$$\frac{\partial E(\mathbf{Y})}{\partial \mathbf{Y}} = 4\mathbf{L}\mathbf{Y}. \tag{1.16}$$

The optimization method used in [118] is a gradient descent with two modifications: an adaptive learning rate is used (as proposed in [88]) and both the momentum term and the high-dimensionality probability matrix $\mathbf{P}$ are artificially amplified for the early iterations.

Due to the use of two full probability matrices $\mathbf{P}$ and $\mathbf{Q}$, the original implementation has an $\mathcal{O}(N^2)$ complexity, which is quickly unmanageable for large values of N. The authors of t-SNE thus proposed an accelerated version, which we will call Barnes Hut t-SNE (BH t-SNE), running in $\mathcal{O}(N\log(N))$ using two optimizations. The first one is an approximation of the input similarities $\mathbf{P}$ by sparsification: only the k nearest neighbours are considered to compute the probabilities and the other are set to zero. The kNN search is performed using vantage-points trees to support more than simple Euclidean distances (see Section 1.2.4). The second optimization is to use the Barnes-Hut approach [15] for the low-dimensional probability matrix $\mathbf{Q}$. Essentially, it consists of using a hierarchical subdivision of the low-dimensional space using tree structures (e.g. a quadtree) and then computing the similarities using averaged cells instead of individual points. Note that the use of a quadtree on the embedded points imposes a two-dimensional output.

LargeVis [187]

Following the success and wide use of t-SNE, an other method, called LargeVis, was proposed to address the remaining speed issues of Barnes-Hut t-SNE. While not fundamentally changing the $\mathcal{O}(N\log N)$ complexity, which appears to be a lower bound as long as pairwise similarity needs to be computed, this new method proposes improvements in various directions. At its core LargeVis is similar to Barnes-Hut t-SNE: it uses probabilistic modelling with a sparse kNN graph for input similarities and an approximated probability matrix for the embedding.

Its differences with Barnes-Hut t-SNE are the following : first, a very efficient scheme for the graph construction is used at the cost of approximate nearest neighbours. Second, the embedding similarities are modelled using an inverse square law :

$$\mathbf{Q}_{ij} = \frac{1}{1+\|\mathbf{y}_i - \mathbf{y}_j\|^2}. \tag{1.17}$$

Third, the cost function is defined using a log likelihood of the embedding similarities :

$$E(\mathbf{Y}) = \sum_{(i,j)\in\mathcal{E}} \mathbf{P}_{ij}\log\mathbf{Q}_{ij} + \sum_{(i,j)\in\bar{\mathcal{E}}} \gamma\log(1-\mathbf{Q}_{ij}), \tag{1.18}$$

where $\mathbf{P}_{ij}$ has the same definition than for t-SNE, $\mathcal{E}$ is the set of edges in the graph, $\bar{\mathcal{E}}$ is all the non-edges and γ is a parameter controlling the weight of the non-edges.

Maximizing the first term of the cost function increases the similarity of close points and the second one pushes dissimilar points apart. Since there are $\mathcal{O}(N^2)$ elements in $\bar{\mathcal{E}}$, the second sum is evaluated using random sampling, more specifically, negative sampling techniques [126]. Finally, the last improvement is to optimize the cost function using a variant of stochastic gradient descent based on edge sampling [186].

1.1.3 Principles

This quick overview of dimensionality reduction methods from the seminal ones to the state-of-the-art reveals fundamental principles common to different techniques. In particular, a few characteristics can be used to categorize a large number of techniques, including the ones we presented. We summarize them using the following classification keys: type of pairwise similarity, sparsity, spectral decomposition and convexity of the objective. As we will show, those characteristics are directly linked to the temporal and spatial complexity.

- **Similarity**
  The concept of pair-wise similarity between data samples is underlying all of the methods covered above. The similarity matrices fall globally in the following categories: pairwise ℓ_2-distance (or inner product), kernelized ℓ_2-distance, covariance or other pairwise distance matrices (e.g. shortest-path distances).
- **Sparsity**
  The similarity matrices are divided into two main categories: dense versus sparse. The sparsity is mostly associated with the notion of nearest neighbours (possibly approximate) and thus is roughly equivalent to a thresholding of a dense matrix. Dense and sparse are considered to contain $\mathcal{O}(N^2)$ and $\mathcal{O}(N)$ elements respectively.
- **Eigen decomposition**
  A large proportion of the dimensionality reduction techniques leverages the eigendecomposition of the similarity matrices. The eigenvectors associated to the biggest (or smallest) eigenvectors are then used as a linear mapping for the low-dimensional representation. While the SVD is generally mentioned for simplicity to provide the eigenspace, more efficient and specialized methods can be used when only a few eigenvectors need to be extracted. The sparsity of the matrix to be diagonalized is also an important factor. For more details on those specific methods, see [112, 2].
- **Convexity**
  Most of the early methods are formulated as convex problems, which are relatively easy to optimize. In fact, there methods formulated as convex problems are generally the ones using spectral decomposition. Non-convex problems, on the other hand, are often solved using different variants of gradient descent.
- **Complexity**
  The time and memory computational complexities of the different methods are mainly related to N, i.e. the amount of input samples and is stated in a number of operations.

For both time and memory, complexities in the quadratic regime or above are considered not to be able to handle large-scale datasets.

In Table 1.1, we list the different methods mentioned in the previous section, categorize them using the properties just mentioned and report their time and space complexity. The values are taken either directly from the original papers or from reviews [109, 198], without extra justification.

Method	Similarity	Sparsity	Optimization	Time	Memory
PCA	covariance	dense	diagonalization	$\mathcal{O}(D^3)$	$\mathcal{O}(D^2)$
MDS	ℓ_2-distance	dense	diagonalization	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$
Isomap	geodesic	dense	diagonalization	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$
LLE	kernel + ℓ_2-distance	sparse	diagonalization	$\mathcal{O}(\lvert\mathcal{E}\rvert N)$	$\mathcal{O}(\lvert\mathcal{E}\rvert N)$
LE	kernel + ℓ_2-distance	sparse	diagonalization	$\mathcal{O}(\lvert\mathcal{E}\rvert N)$	$\mathcal{O}(\lvert\mathcal{E}\rvert N)$
Sammon	kernel + ℓ_2-distance	sparse	gradient descent	$\mathcal{O}(nN^2)$	$\mathcal{O}(N^2)$
t-SNE	kernel + ℓ_2-distance	sparse	gradient descent	$\mathcal{O}(nN^2)$	$\mathcal{O}(N^2)$
BH t-SNE	kernel + ℓ_2-distance	sparse	gradient descent	$\mathcal{O}(nN\log N)$	$\mathcal{O}(\lvert\mathcal{E}\rvert N)$
LargeVis	kernel + ℓ_2-distance	sparse	gradient descent	$\mathcal{O}(nN\log N)$	$\mathcal{O}(\lvert\mathcal{E}\rvert N)$

Table 1.1 – This table summarizes the properties of the dimensionality reduction presented in this section. The last two columns describe the time and memory complexity, where $|\mathcal{E}|$ is the number of edges in the kNN graph of similarity (i.e. is related to sparsity level) and n the number of iterations in gradient descent techniques.

1.2 Graph Signal Processing

As we saw in our short review of dimensionality reduction algorithm, graphs seem to arise naturally as a model for the similarity (i.e. structure) of the data. In this section, we present the field of Graph Signal Processing, which we choose as the main conceptual approach used in our contributions.

To begin with, we consider here that the word graph refers to abstract mathematical objects. They are able to represent both regular and irregular structures and occur naturally in the form of networks in many domains: transportation networks, social networks, energy grids, brain connectomes, sensors networks, and so on. In each case, the central concepts are the presence of a collection of objects linked to each other, which is modelled abstractly as nodes connected by edges. Information can exist both in the node and edge domains. For example, in a social network, the node is a person and a great deal of data is associated with it (e.g. name, birthdate, hobbies, etc.) and the edges model connections between people and are associated with other types of data (e.g. friendship, family or work relation, similar interests, etc.). In this thesis, we make the choice of representing data on the nodes, which we will call later graph signals, and simplify the information present on the edges by quantifying it as the strength of the connections, which we will soon call edge weights.

In addition to networks, graphs can be used to model pairwise relations for any collection of data. This can apply to theoretically any set of objects for which a similarity measure can be made, and it is, for example, commonly used with images, sounds, medical data, biological data, and so on. In this context, the nodes are simply references to the objects of interest and the weight of the connections are similarity measures between them.

Graphs are not recent mathematical models and there are very established fields which have produced fundamental results in both graph theory and algorithms. Numerous examples exist of problems formulated with graphs such as graph colouring, minimum spanning tree, travelling salesmen, shortest path, set covering, and so on (see [49, 210, 208] for references on graph theory). We will refer to this theoretical background as classical graph theory and, while using it sporadically, focus on a more recent field, also rooted in graph structures, called Graph Signal Processing (GSP).

GSP is at the convergence of graphs and traditional signal processing. The field is now established in two communities, the first one is based on a generalization of discrete differential operators to graphs, and eigendecomposition of the graph Laplacian, in particular [48, 167, 175, 217, 22, 20, 147], and the second one is based on a generalization of discrete signal processing using the Jordan decomposition of the adjacency matrix [157, 156]. In this work, we develop and use the first of the two formalisms.

1.2.1 Spectral Graph Theory

Since the GSP framework is central in this thesis and still not widely known, we introduce its key components thoroughly from the basic graph definitions to more advanced tools. For more complete reviews of the tools and concepts presented here, we refer the interested reader to [48, 167]. We start with an overview of spectral graph theory, with the goal of constructing the tools in an intuitive way by connecting the GSP concepts to their traditional signal processing counterparts.

Nomenclature

First, let us give formal definitions of the different terms we will use. Note that, for simplicity, we will restrict ourselves to undirected, connected and weighted graphs. Nevertheless, the definitions and tools presented in this paper can be extended to directed graphs [46, 47, 218] and hypergraphs [219]. We restrict ourselves to connected graphs since unconnected graphs can be split into independent connected components that can be processed as independently connected graphs.

Let us define $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ as an undirected weighted *graph* where $\mathcal{V}$ is the set of *vertices* and $\mathcal{E}$ the set of *edges* representing connections between nodes in $\mathcal{V}$. Note that the terms nodes and vertices are used interchangeably. The vertices $v \in \mathcal{V}$ of the graph are ordered from 1 to $N = |\mathcal{V}|$ and the edges are pairs of vertices $e = (v_i, v_j) \in \mathcal{E}$ with $v_i, v_j \in \mathcal{V}$. The edges can be represented

using a matrix $\mathbf{A}$ for which $\mathbf{A}_{i,j} = 1$ if $(v_i, v_j) \in \mathcal{E}$ and 0 else. This matrix is called the *adjacency matrix* of the graph $\mathcal{G}$. Here we consider a generalization of the adjacency matrix: the matrix $\mathbf{W}$, which is symmetric and positive, and is called the weighted adjacency matrix of the graph $\mathcal{G}$ (also called *weight matrix*). The weight $\mathbf{W}_{ij}$ represents the weight of the edge between vertices v_i and v_j and a value of 0 means that the two vertices are not connected. A graph for which the weight is either 0 or another constant value (usually 1) is called a binary graph. It is useful to give a representation of unweighted graphs using this formalism. By convention, we use the same symbol N as the number of vertices in $\mathcal{G}$ and the number of samples in a dimensionality reduction problem, since there will often be a one-to-one relationship between the two.

The *degree* $d(i)$ of a node v_i is defined as the sum of the weights of all its edges $d(i) = \sum_j \mathbf{W}_{ij}$. The degree measures the connectivity of the node (e.g. for a binary graph, it represents the number of its neighbours). The diagonal matrix $\mathbf{D}$, with $\mathbf{D}_{ii} = d(i)$, is called the *degree matrix*.

A formal definition of a *graph signal* is a function $f : \mathcal{V} \to \mathbb{R}$ (or $\mathbb{C}$) assigning one value to each vertex. For convenience, we work with an alternative vector definition and consider a graph signal $\mathbf{x}$ as a vector of size N with $\boldsymbol{x}[i] = f(v_i)$. Using this notation, we can define the scalar product of two graph signals $\mathbf{x}$ and $\mathbf{y}$ as:

$$\langle \mathbf{x}, \mathbf{y} \rangle := \sum_i \mathbf{x}[i]\mathbf{y}[i]^*. \tag{1.19}$$

Graph Laplacian

Most of the Graph Signal Processing concepts are derived from one of the most fundamental objects associated to graphs: the graph Laplacian operator (or simply *Laplacian*). As we will see, it can have various definitions, but one of the most common is the combinatorial Laplacian:

$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \tag{1.20}$$

Note that here we abusively mix notions of matrices and operators and $\mathbf{L}$ will be used to define the Laplacian matrix. Before using it to define the rest of the theory, let us motivate why it is called a Laplacian.

First, let us recall that the discrete Laplacian in the classical case is defined as

$$\Delta\mathbf{x}[i] = (\mathrm{div}\nabla\mathbf{x})[i] = 2\mathbf{x}[i] - \mathbf{x}[i-1] - \mathbf{x}[i+1], \tag{1.21}$$

with $\nabla f[i] = f[i+1] - f[i]$ the discrete gradient operator and $\mathrm{div} f[i] = f[i-1] - f[i]$ its adjoint operator, the divergence.

Let us now define the graph Laplacian using a similar construction, i.e. $\mathbf{L} = -\mathrm{div}_{\mathcal{G}}\nabla_{\mathcal{G}}$, where $\mathrm{div}_{\mathcal{G}}$ and $\nabla_{\mathcal{G}}$ are the divergence and gradient operators defined in the graph setting.

For this definition to make sense, we need to define the gradient and divergence on the graph in a suitable way. The graph gradient is defined as a linear operator which assigns a value to each edge. More formally, for a signal $\mathbf{x}$, its gradient $\nabla_G \mathbf{x} : \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}^{|\mathcal{E}|}$ is defined as :

$$\nabla_G \mathbf{x} = \left(\frac{\partial \mathbf{x}}{\partial_e}(i,j) \right)_{i,j \text{ s. t. } (v_i,v_j)\in\mathcal{E}} \tag{1.22}$$

where $\frac{\partial \mathbf{x}}{\partial_e} : \mathcal{E} \to \mathbb{R}$ is the edge derivative, which is defined as:

$$\frac{\partial \mathbf{x}}{\partial_e}(i,j) = \sqrt{\mathbf{W}_{i,j}} \left(\mathbf{x}[j] - \mathbf{x}[i] \right). \tag{1.23}$$

We see that the edge derivative is actually a weighted finite difference taking the strength of the connection into account. In other words, this definition of the gradient is an adaptation of the finite differences to the graph setting.

Similarly to the classic case, the graph divergence is defined as the adjoint of the gradient, i.e. we need to have $\langle \nabla_G \mathbf{x}, \mathbf{y} \rangle_{\mathcal{E}} = \langle \mathbf{x}, \mathrm{div}_G \mathbf{y} \rangle_{\mathcal{V}}$, for $\mathbf{x}$ a graph signal on the vertices and $\mathbf{y}$ a signal defined on the edge set. The divergence operator $\mathrm{div}_G \mathbf{y} : \mathbb{R}^{|\mathcal{E}|} \to \mathbb{R}^{|\mathcal{V}|}$ is thus defined as:

$$\left(\mathrm{div}_G \mathbf{y}\right)[i] = \frac{1}{2} \left(\sum_{j \text{ s. t. } (v_j,v_i)\in\mathcal{E}} \sqrt{\mathbf{W}(i,j)}\mathbf{y}(i,j) - \sum_{i \text{ s. t. } (v_i,v_j)\in\mathcal{E}} \sqrt{\mathbf{W}(i,j)}\mathbf{y}(j,i) \right). \tag{1.24}$$

These operators being defined, we can now verify the definition of the Laplacian. The combinatorial graph Laplacian $\mathbf{L} : \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}^{|\mathcal{V}|}$ applied to a graph signal $\mathbf{x}$ is given by :

$$\mathbf{L}\mathbf{x} = (\mathbf{D} - \mathbf{W})\mathbf{x} = -\mathrm{div}_{\mathcal{G}} \nabla_{\mathcal{G}} \mathbf{x}. \tag{1.25}$$

The complete derivation of Eq. (1.25) is given in Appendix A.1.1. For more details on the construction of the graph Laplacian, and the graph and divergence operators, we refer the reader to [62, 74].

As previously mentioned, other definitions exist for the graph Laplacian. One very common alternative is the normalized Laplacian $\mathbf{L}_n = \mathbf{D}^{-\frac{1}{2}} \mathcal{L} \mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{\frac{1}{2}}$. With this definition, the goal is to remove the influence of the degree of a node, hence the normalization using the degree matrix. An interesting consequence is that the spectrum of the normalized Laplacian is bounded by 2. Other Laplacian definitions exist for directed graph, e.g. degree normalized Laplacian, distribution normalized Laplacian [218, 47, 46]. Each definition is associated with a different definition of the edge derivative. Provided the latter is linear, the graph Laplacian will be a symmetric positive semi-definite operator. This fact allows us to derive the graph spectral theory without differentiating between the different cases.

In addition, asymmetric variants are also used in the literature. In particular, the random walk Laplacian $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$, has interesting properties despite the fact that it is not symmetric.

Indeed, $\mathbf{P}$ is adjoint to a symmetric matrix :

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{W} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{I} - \mathbf{L}_n)\mathbf{D}^{\frac{1}{2}} \tag{1.26}$$

and has thus a suitable spectral decomposition. Its left and right eigenvectors are even directly linked with ones of the normalized Laplacian. This Laplacian definition is interesting because $\mathbf{P}$ can be viewed as a probability matrix of the Markov chain for the random walk on the vertices of the graph, hence the name. For more details, see [164, 17, 102, 130]. Practically, the choice of the Laplacian highly depends on the application.

Spectral Theory

From here, the only assumption about the graph Laplacian that it is symmetric and positive semi-definite, independently of the definition. Using this hypothesis, we can apply the spectral theorem on $\mathbf{L}$ to decompose it into an orthonormal basis of *graph eigenvectors* noted $\{\mathbf{u}_\ell\}_{\ell=0,1,\dots,N-1}$ with corresponding *graph eigenvalues* $\{\lambda_\ell\}_{\ell=0,1,\dots,N-1}$. For convenience, and without loss of generality, we order the eigenvectors in ascending order of the eigenvalues $0 = \lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_{N-1} = \lambda_{\max}$. When the graph is connected, there is only one zero eigenvalue. In fact, the multiplicity of the zero eigenvalue is equal to the number of connected components (see for example [48]). In matrix form we can write this decomposition as $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^*$ with $\mathbf{U} = [\mathbf{u}_1|\mathbf{u}_2|\dots|\mathbf{u}_{N-1}]$ the matrix of eigenvectors and $\mathbf{\Lambda}$ the diagonal matrix containing the eigenvalues in ascending order.

Given a graph signal $\mathbf{x}$, we define its *graph Fourier transform* $\hat{\mathbf{x}}$ as the projection onto the set of eigenvectors $\{\mathbf{u}_\ell\}_{\ell\in[0,N-1]}$:

$$\hat{\mathbf{x}}[\ell] = \mathcal{F}(\mathbf{x})[\ell] = \langle \mathbf{x}, \mathbf{u}_\ell \rangle = \sum_{j=1}^{N} \mathbf{x}[j]\mathbf{u}_\ell^*[j], \quad \ell \in [0, N-1]. \tag{1.27}$$

Moreover, since $\mathbf{U}$ is an orthonormal basis, we can define the *inverse graph Fourier transform* as :

$$\mathbf{x}[i] = \mathcal{F}^{-1}(\hat{\mathbf{x}})[i] = \sum_{\ell=0}^{N-1} \hat{\mathbf{x}}[\ell]\mathbf{u}_\ell[i], \quad i \in [1, N]. \tag{1.28}$$

In matrix form, we have $\hat{\mathbf{x}} = \mathbf{U}^*\mathbf{x}$, and $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$. It is called a Fourier transform by analogy to the continuous Laplacian whose spectral components are Fourier modes, and the matrix $\mathbf{U}$ is sometimes referred to as the graph Fourier matrix (see e.g., [48]). By the same analogy, the set $\{\sqrt{\lambda_\ell}\}_{\ell=0,1,\dots,N-1}$ is often seen as the set of graph frequencies [165]. This definition of the Fourier transform possesses different properties described in [168]. An interesting one, which will prove useful later, is the Parseval relation :

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle \hat{\mathbf{x}}, \hat{\mathbf{y}} \rangle. \tag{1.29}$$

While there are other definitions for the graph Fourier transform (see e.g. [157]), we use the one presented in this section for two main reasons. First, it is a natural generalization of the classical case for very regular graphs. Indeed, on the ring graph, the graph Fourier transform has been proven to be equivalent to the Discrete Fourier Transform [180] and the eigenvectors of the path graph are equivalent to the Discrete Cosine Transform modes. Second, when applying this definition of the graph Fourier transform to regular graphs, the Fourier modes are the oscillating modes for some frequencies. In particular, lower values are associated to low oscillations and higher values to higher oscillations [165, 168]. This fact supports the intuition that low eigenvalues are associated to low frequencies and higher eigenvalues to higher frequencies. The analogy breaks for high frequencies where the eigenvectors may be localized in the vertex set, for non-regular graphs.

Graph Filtering

In traditional signal processing, filtering (i.e. convolution) can be carried out by a pointwise multiplication in Fourier. Thus, since the graph Fourier transform is defined, it is natural to consider a filtering operation on the graph using a multiplication in the graph Fourier domain. To this end, we define a *graph filter* (or graph kernel) as a continuous function $g : \mathbb{R}_+ \to \mathbb{R}$ directly in the graph Fourier domain.

If we consider the filtering of a signal $\mathbf{x}$, whose graph Fourier transform is written $\hat{\mathbf{x}}$, by a filter g the operation in the spectral domain is a simple multiplication :

$$\hat{\mathbf{y}}[\ell] = g(\lambda_\ell) \cdot \hat{\mathbf{x}}[\ell], \tag{1.30}$$

where $\mathbf{y}$ is the filtered signal and $\hat{\mathbf{y}}$ its graph Fourier transform. Taking the inverse Fourier transform we get an expression in the vertex domain:

$$\begin{aligned} \mathbf{y}[i] &= \sum_{\ell=0}^{N-1} g(\lambda_\ell)\hat{\mathbf{x}}(\ell)\mathbf{u}_\ell[i] \\ &= \sum_{j=1}^{N} \mathbf{x}[j] \sum_{\ell=0}^{N-1} g(\lambda_\ell)\mathbf{u}_\ell^*[j]\mathbf{u}_\ell[i]. \end{aligned}$$

Using the graph Fourier matrix $\mathbf{U}$, the explicit matrix formulation for graph filtering becomes:

$$\mathbf{y} = \mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^*\mathbf{x}, \tag{1.31}$$

where $g(\mathbf{\Lambda}) = \mathrm{diag}(g(\lambda_0), g(\lambda_1), \ldots, g(\lambda_{N-1}))$.

Finally, by defining the graph filtering operator as:

$$g(\mathbf{L}) = \mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^*, \tag{1.32}$$

we can rewrite the graph filtering equation as a vector-matrix operation :

$$\mathbf{y} = g(\mathbf{L})\mathbf{x}. \tag{1.33}$$

Since the filtering equation defined above involves the full set of eigenvectors $\mathbf{U}$, it implies the diagonalization of the Laplacian $\mathbf{L}$ which is costly for large graphs. To circumvent this problem, one can represent the filter g as a polynomial approximation, as we will see in Section 1.2.2.

Example : Adaptive denoising using graphs

In this example we give a practical application for graph filtering : image denoising. We consider a colour image corrupted with additive white noise and wish to remove it. An image is not associated to a graph naturally and a trivial way to do it would be to consider the 2-dimensional grid on which the pixels live, i.e. connecting each pixel to its four (or eight) neighbours. Such a graph would be very regular and not very different of the 2D Euclidean domain, so we prefer to view the graph as a collection of data and connect it using a kNN graph construction procedure.

In order to be more robust, the nodes are not represented only by their own pixel values, but by a small patch of pixels around it. The data used to create the graph is thus a collection of image patches connected in a kNN fashion. Such a graph is called non-local, as it connects the pixels not related to their location but only to their pixel content. In practice, we add small connection weight to patches of nearby pixels to regularize the graph and call the result a semi non-local graph.

Once the graph is constructed, the image in itself is represented by a signal containing the values of the pixels ; more precisely three signals associated to the RGB channels of the image. A simple way to denoise those signals is to filter them with a low-pass, for example a heat kernel $h(\boldsymbol{\lambda}) = \exp(-\tau\boldsymbol{\lambda})$. The benefit of this method is that, since the graph adapts to the structure of the image, the low-pass will not blur the edges or remove details.

The results in Figure 1.1 show that using the graph-based method is superior both perceptually and quantitatively to the wavelet method. In particular, in terms of chromatic aberrations and residual noise. This type of graph-based denoising is actually famous and called Non-Local Means [33], without the graph formulation, and we saw that it is a simple application of graph filtering. The principles illustrated in this simple example are developed in detail in [62].

Localization Operator

The concept of translation, which is well defined in traditional signal processing cannot be directly applied to graphs, as they are not provided with a natural ordering. However, inspired by the notion of translation, we can define the localization of a function $g : \mathbb{R}_+ \to \mathbb{R}$ defined on

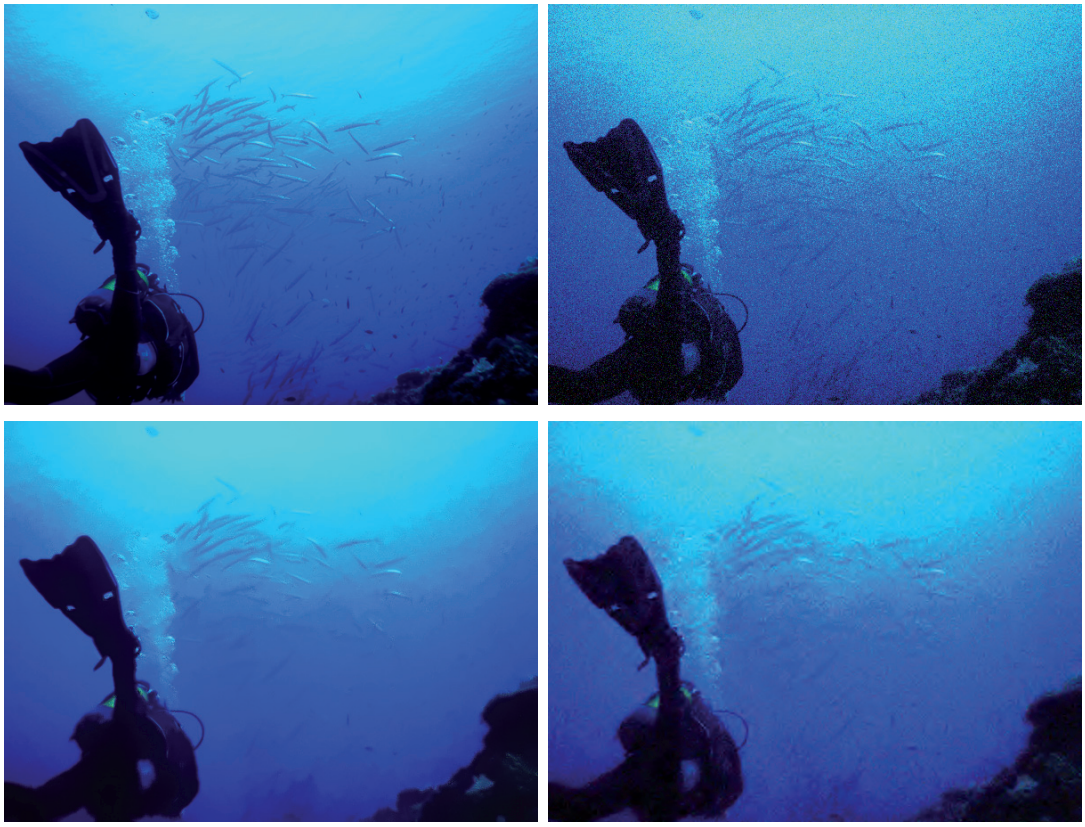

Figure 1.1 – **Adaptive denoising using graphs**. In this figure, the original image is on the top-left corner, the corrupted image in the top-right corner (SNR 13.07 dB). The image denoised using the filtering on graph is in the bottom left corner (SNR 17.65 dB), and for comparison, a denoising using a traditional state-of-the-art wavelet denoising is shown in the bottom right corner (SNR 16.24 dB).
Image : Sec des Gorgones, Porquerolles, 2016. Credits : Nathan Broquet

the graph spectrum as a multiplication with a Kronecker delta:

$$\mathcal{F}(\mathcal{T}_i g)[\ell] = g(\lambda_\ell) \cdot \hat{\delta}_i = g(\lambda_\ell) \cdot \mathbf{u}_\ell[i], \tag{1.34}$$

where $\mathcal{T}$ is called the *localization operator*, and $\mathcal{T}_i$ means localization at vertex i. Going back to the vertex domain, we get :

$$\mathcal{T}_i g[j] = \mathcal{F}^{-1}\left(g \cdot \hat{\delta}_i\right)[j] = \sum_\ell g(\lambda_\ell)\mathbf{u}_\ell^*[i]\mathbf{u}_\ell[j] = \left(g(\mathbf{L})\right)_{ij}. \tag{1.35}$$

The reason for calling $\mathcal{T}_i$ a localization operator comes from the fact that for regular functions g, the operator $\mathcal{T}_i g$ is concentrated. Here, regularity means that they can be well approximated by low order polynomials. Moreover, provided that g is low-pass, then $\mathcal{T}_i g$ is localized around the vertex i. The proof of these results and more information on the localization operator

can be found in [170, Theorem 1 and Corollary 2]. Localized filters are quite naturally called atoms since the result of filtering a signal $\mathbf{x}$ using a filter g can be expressed as $\mathbf{y}[i] = \langle \mathbf{x}, \mathcal{T}_i g \rangle$. The localization operator possesses a few interesting properties that we state in the following lemma.

Lemma 3. *Let $\mathcal{T}_i g[j]$ be the filter g localized at vertex i and evaluated at vertex j as defined in Eq.* (1.35)*. Then the following hold, for a real kernel g:*

- $\mathcal{T}_i g[j] = \mathcal{T}_j g[i]$,
- $\langle \mathcal{T}_i g, \mathcal{T}_j g \rangle = \mathcal{T}_i g^2[j]$,
- $\mathcal{T}_i g^2[j] \leq \left\| \mathcal{T}_i g \right\| \left\| \mathcal{T}_j g \right\|$.

The proof is given in Appendix A.1.2.

As we will see later, the ℓ_p-norms, and especially the ℓ_2-norms of the localization operator are useful to reveal characteristics of the graph. First, they are useful to measure the concentration (or sparsity) of localized filters.

Concentration

Traditionally, concentration (or sparsity) of vectors can be measured using ratios of ℓ_p-norms. For simplicity, we choose to use the ℓ_1-concentration, and define the concentration to be :

$$c(\mathbf{x}) = \frac{\|\mathbf{x}\|_2}{\|\mathbf{x}\|_1}. \tag{1.36}$$

For any vector $\mathbf{x} \in \mathbb{R}^N$, $c(\mathbf{x}) \in [\frac{1}{\sqrt{N}}, 1]$ with values close to 1 indicating high sparsity and low values weak concentration. Using this definition, we naturally define the concentration of a filter g localized at vertex i to be :

$$c(\mathcal{T}_i g) = \frac{\|\mathcal{T}_i g\|_2}{\|\mathcal{T}_i g\|_1} \tag{1.37}$$

with $\|\mathcal{T}_i g\|_p = \left(\sum_j |\mathcal{T}_i g_h[j]|^p\right)^{\frac{1}{p}}$.

The issue with this definition is that, in general, computing the ℓ_p-norms of localized filters amounts to N filtering operations, which is quite costly for large graphs. Nevertheless, the ℓ_2-norm can be computed efficiently using random signals. Let us start by stating the following lemma (from [140, Lemma 3]).

Lemma 4. *Let $\mathbf{r}$ be a random vector with i.i.d entries. Let also $\mathbf{r}$ be drawn from a zero-mean and unit variance distribution. Then,*

$$\mathbb{E}\left[(\langle \mathbf{r}, \mathcal{T}_i g \rangle)^2\right] = \|\mathcal{T}_i g\|_2^2. \tag{1.38}$$

This interesting result shows that the average of random signals filtered with g is actually

the squared ℓ_2-norm. Inspired by this fact, we can design an estimator $\dot{T}_K$ which gives the empirical average of K random signals $\{\mathbf{r}_1, \dots, \mathbf{r}_K\}$ filtered with g :

$$\dot{T}_K = \frac{1}{K} \sum_{k=1}^{K} (g(\mathbf{L})\mathbf{r}_k)^2. \tag{1.39}$$

The characteristics of this estimator are stated in the following lemma, adapted from [140, Theorem 5].

Lemma 5. *Let $\{\mathbf{r}_1, \dots, \mathbf{r}_K\}$ be a set of random vectors with i.i.d entries, drawn from a zero-mean distribution with unit variance and bounded fourth moment (written κ). Then, the estimator $\dot{T}_K$ is unbiased, i.e. :*

$$\mathbb{E}\left[\dot{T}_K[i]\right] = \|\mathcal{T}_i g\|_2^2, \tag{1.40}$$

and has variance

$$\mathrm{Var}\left[\dot{T}_K[i]\right] = \frac{1}{K}\left(\|\mathcal{T}_i g\|_4^4(\kappa - 3) + 2\|\mathcal{T}_i g\|_2^4\right). \tag{1.41}$$

The first interesting point is that the estimator is unbiased. Second, his variance decreases linearly with the number of random vectors used for filtering. We also see that the variance is influenced by the tails of the distribution since it is linearly correlated with κ, the kurtosis.

Two specific distributions for $\mathbf{r}_k$ give better insights : the Gaussian and balanced Bernoulli distributions. If $\mathbf{r}_k \sim \mathcal{N}(0, \mathbf{I}_N)$, $\kappa = 3$ and thus the variance becomes simply :

$$\mathrm{Var}\left[\dot{T}_K[i]\right] = \frac{2}{K}\|\mathcal{T}_i g\|_2^4 = \frac{2}{K}\left(\|\mathcal{T}_i g\|_2^2\right)^2. \tag{1.42}$$

This means that the variance of the estimator is affected by the square of the quantity of interest.

The Bernouilli distribution with states $[-1, 1]$ and success probability $\frac{1}{2}$ is the distribution with the lowest kurtosis. This is interesting since for all values of κ smaller than 3 the first term will lower the variance. If the random vectors are drawn from such a distribution, then the variance becomes :

$$\mathrm{Var}\left[\dot{T}_K[i]\right] = \frac{1}{K}\left(\|\mathcal{T}_i g\|_4^4(-2) + 2\|\mathcal{T}_i g\|_2^4\right) = \frac{2}{K}\left(\|\mathcal{T}_i g\|_2^4 - \|\mathcal{T}_i g\|_4^4\right). \tag{1.43}$$

which provides the lowest variance for the estimator $\dot{T}_K$.

Example : Low-pass filter localization

In this example we illustrate how localized filters are related to the local parts of the graph. We consider a semi non-local kNN graph constructed from image patches. We will look at the localization of a low-pass kernel since its behaviour is quite intuitive. In particular, we

Figure 1.2 – **Low-pass filter localization**. The original image from which the graph is created is on the left, with the three coloured dots corresponding to the different vertices at which the kernel is localized. On the right, the localized atoms are displayed with colours corresponding to the left image.
Image : Cabane de Moiry, Valais, 2016. Credits : Johan Paratte

consider the heat kernel $h(\boldsymbol{\lambda}) = \exp(-\tau\boldsymbol{\lambda})$. We expect the localized atoms to be concentrated and close to their anchor vertex. In order to understand the effect of the graph topology, we localize the filter at three locations having different characteristics.

In Figure 1.2, the red point corresponds to a quite uniform part of the image, the green one to an edge and the yellow one to a textured part. The resulting atoms display strikingly different geometrical aspects which illustrate a very interesting point : since the graph adapts to the image structure, the localized atoms of a generic kernel can have very different characteristics and in particular, adapt to the graph, and by extension, to the data. The red atom is isotropic since it corresponds to a uniformly looking part of the image, the green one follows the edge and the yellow responds exactly to the texture.

In this example, we saw two things: first, as expected, localized low-pass filters are close to their anchor vertices on the graph and they are concentrated in the vertex domain. Second, since the graph adapts to the data, generic graph kernels specialize in function of the graph topology.

1.2.2 Fast Filtering

All filtering operations in the previous section are based on the Graph Fourier transform and thus require an explicit computation of the Fourier basis (i.e. diagonalization of the Laplacian) which is very expensive computationally $O(N^3)$ and memory consuming $O(N^2)$. This exact method is thus only applicable for small graphs. In order to tackle problems of bigger size, we need more efficient methods. The ones we introduce in this section rely on the use of Chebyshev or Lanczos polynomials and are presented in detail in [76] and [183] respectively.

Filtering in the vertex domain

Since we have no direct access to the graph Fourier transform, we need to approach filtering in the vertex domain. To achieve that, we will leverage the natural filtering nature of the Laplacian. Indeed, let us apply the Laplacian directly to a signal $\mathbf{x}$:

$$\mathbf{L}\mathbf{x} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^*\mathbf{x} = \mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^*\mathbf{x}. \tag{1.44}$$

This equality holds if $g(\boldsymbol{\lambda}) = \boldsymbol{\lambda}$. This result shows that multiplying a signal with the Laplacian is equivalent to filtering with the linear kernel $g(\lambda_\ell) = \lambda_\ell$. Using the same reasoning, we see that applying a power of the Laplacian to $\mathbf{x}$ will be equivalent to filtering with a polynomial kernel, i.e. $\mathbf{L}^k\mathbf{x} = g(\mathbf{L})\mathbf{x}$ with $g(\lambda_\ell) = \lambda_\ell^k$.

Knowing that we have a simple access to polynomials, we can filter with any kernel of the form $g(\boldsymbol{\lambda}) = a_0 + a_1\boldsymbol{\lambda} + a_2\boldsymbol{\lambda}^2 + \cdots + a_K\boldsymbol{\lambda}^M$ using powers of the Laplacian :

$$\mathbf{y} = \mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^*\mathbf{x} = \left(a_0\mathbf{I} + a_1\mathbf{L} + a_2\mathbf{L}^2 + \cdots + a_M\mathbf{L}^M\right)\mathbf{x} = g(\mathbf{L})\mathbf{x}. \tag{1.45}$$

When the kernel is not a polynomial, we can use a polynomial approximation instead. Since the order of the approximation is necessarily finite, some error will be present and the filtering will necessarily an approximation. The quality of this approximation depends on the choice of the polynomials and the order of approximation.

Chebyshev polynomial approximation

Since we saw that filtering could be performed using polynomial approximation, the idea is to find suitable polynomials. Chebyshev polynomials of the first kind were proposed in [76] because they have several good properties.

To start, let us write $P_m(y)$ a Chebyshev polynomial of order m in y. Such polynomial satisfy:

$$P_m(y) = \cos(m \arccos(y)), \tag{1.46}$$

for $y \in [-1,1]$. This means that the Chebyshev polynomials are bounded, i.e. $P_m(y) \in [-1,1]$ and are orthogonal with respect to the weight $\frac{1}{\sqrt{1-y^2}}$. Thus, any function $h \in L^2\left([-1,1], \frac{dy}{\sqrt{1-y^2}}\right)$ has a convergent series given by :

$$h(y) = \sum_{m=0}^{\infty} c_m P_m(y) = \frac{1}{2}c_0 + \sum_{m=1}^{\infty} c_m P_m(y), \tag{1.47}$$

where $\{c_m\}_{m\in[0,\infty[}$ are the Chebyshev coefficients given by :

$$c_m = \frac{2}{\pi}\int_0^{\pi} \cos(m\theta)h(\cos(\theta))d\theta \tag{1.48}$$

Since the kernels we want to approximate are defined over the Laplacian spectrum, we need to shift and scale the domain from $[-1,1]$ to $[0,\lambda_{max}]$ using a simple change of variable $\lambda = (\frac{\lambda_{max}}{2}(y+1))$. This gives :

$$h(\lambda) = \frac{1}{2}c_0 + \sum_{m=1}^{\infty} c_m \tilde{P}_m(\lambda) \tag{1.49}$$

that is valid for $\lambda \in [0,\lambda_{max}]$ with $\tilde{P}_m(\lambda) = P_m(\frac{2\lambda}{\lambda_{max}} - 1)$.

Another interesting property of Chebyshev polynomials of the first order is that they can be generated using the recurrence relation $P_m(y) = 2yP_{m-1}(y) - P_{m-2}(y)$ with $P_0(y) = 1$ and $P_1(y) = y$. Inserting the change of variable we get :

$$\tilde{P}_m(\mathbf{L}) = 2\left(\frac{2\mathbf{L}}{\lambda_{max}} - \mathbf{I}\right)\tilde{P}_{m-1}(\mathbf{L}) - \tilde{P}_{m-2}(\mathbf{L}). \tag{1.50}$$

This allows us to write the approximated filtered signal as :

$$\mathbf{y} = g(\mathbf{L})\mathbf{x} = \frac{1}{2}c_0\mathbf{I}\mathbf{x} + \sum_{m=1}^{\infty} c_m \tilde{P}_m(\mathbf{L})\mathbf{x} \tag{1.51}$$

For more details on the precision of this approximation and how to derive those results, see [148, 76]. In practice, of course, the approximation is kept to a maximum order M, such that :

$$\mathbf{y} = g(\mathbf{L})\mathbf{x} \approx \frac{1}{2}c_0\mathbf{I}\mathbf{x} + \sum_{m=1}^{M} c_m \tilde{P}_m(\mathbf{L})\mathbf{x} \tag{1.52}$$

where the M is a parameter which gives control on the tradeoff between the precision of the approximation and the computational cost. Due to the recursive form of Chebyshev polynomials, the implementation of this fast filtering method only costs m multiplications of a vector and the Laplacian. Assuming it is sparse, this means that the overall complexity is $\mathcal{O}(m|\mathcal{E}|)$.

Lanczos polynomial approximation

Another method has been proposed for fast filtering based on polynomial approximation. The method is based on a result given in [70] stating that:

$$g(\mathbf{L})\mathbf{x} \approx \|\mathbf{x}\|_2 V_m g(H_m) e_1, \tag{1.53}$$

where V_m is an orthonormal basis of the Krylov subspace $\mathcal{K}_{m+1}(\mathbf{L},\mathbf{x}) = \text{span}\{\mathbf{x},\mathbf{L}\mathbf{x},\ldots,\mathbf{L}^m\mathbf{x}\}$, $H_m = V_m\mathbf{L}V_m$, m is the order of the approximation and $e_1 \in \mathbb{R}^m$ the first unit vector.

The authors of [183] have proposed a technique that leverages the Lanczos method to give an approximation of V_m. Similarly to the Chebyshev approximation, the Lanczos method [72] uses m iterations of multiplications between a vector and the Laplacian. Its complexity is

thus also $\mathcal{O}(m|\mathcal{E}|)$. The difference between Chebyshev and Lanczos approximations is that the latter tends to adapt to the distribution of the eigenvalues and thus can provide better approximations for irregular spectra, while being provably at least as good as Chebyshev (see [183, Theorem 1]).

1.2.3 Optimization on graphs

Since the graph is supposed to encode the structure of the data, it is interesting to include it when solving optimization problems. The main way to do it is via graph regularization terms. In general settings, most problems are of the form:

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) + \gamma h_{\mathcal{G}}(\mathbf{x}) \tag{1.54}$$

where $f(\mathbf{x})$ is a data fidelity term and $h_{\mathcal{G}}(\mathbf{x})$ a graph regularization term with γ controlling its strength.

Usual choices for the data fidelity $f(\mathbf{x})$ depend on the application at hand. We list a few common ones based on ℓ_2 losses.

- **Denoising** : Solving the problem $\mathbf{y} = \mathbf{x} + \boldsymbol{n}$ with $\boldsymbol{n}$ a realization of white noise leads naturally to :

  $$f(\mathbf{x}) = \|\mathbf{x} - \mathbf{y}\|_2^2 \tag{1.55}$$

  with $\mathbf{y}$ a graph signal to be denoised.
- **Inpainting** : Solving the problem $\mathbf{y} = \mathbf{M}\mathbf{x} + \boldsymbol{n}$ with $\boldsymbol{n}$ a realization of white noise and $\mathbf{M}$ the sampling (or masking) matrix gives :

  $$f(\mathbf{x}) = \|\mathbf{M}\mathbf{x} - \mathbf{y}\|_2^2 \tag{1.56}$$

  with $\mathbf{y}$ a sampled graph signal (i.e. coming only from observed values).
- **Inverse problem** :

  More generally, we can assume that the data is generated using a linear operator : $\mathbf{y} = \mathbf{H}\mathbf{x} + \boldsymbol{n}$ with $\boldsymbol{n}$ a realization of white noise and $\mathbf{H}$ any linear operator. In which case the ℓ_2 loss is again of the form

  $$f(\mathbf{x}) = \|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 \tag{1.57}$$

  with $\mathbf{y}$ the observed signal.

Many more data fidelity terms can be encountered and we will not list them here as it is not the main focus of this section. See [30, 51] for more details.

The use of graphs in regularization terms is still quite recent, while being more and more

used, see e.g. [175, 19, 90, 159, 161, 119]. The common point to all methods is to say that signals must have a sense of regularity on the graph, the most common assumption being smoothness.

- **Tikhonov** : The most commonly used graph regularizer is the Dirichlet energy (i.e. Sobolev norm or ℓ_2-norm of the graph gradient) :

$$h_{\mathcal{G}}(\mathbf{x}) = \|\nabla_G \mathbf{x}\|_2^2. \tag{1.58}$$

  This regularizer is usually suggested because it is supposed to enforce the smoothness of $\mathbf{x}$ on $\mathcal{G}$. Let us analyse it in more details :

$$\begin{aligned}
\|\nabla_G \mathbf{x}\|_2^2 &= \sum_{i=1}^{N}\sum_{j=1}^{N} \mathbf{W}[i,j](\mathbf{x}[i]-\mathbf{x}[j])^2 \\
&= \mathbf{x}^T\mathbf{L}\mathbf{x} \\
&= \mathbf{x}^T(\mathbf{U}\mathbf{\Lambda}\mathbf{U})\mathbf{x} \\
&= \hat{\mathbf{x}}^T\mathbf{\Lambda}\hat{\mathbf{x}} \\
&= \sum_{\ell=0}^{N-1} \lambda_\ell \hat{\mathbf{x}}^2
\end{aligned}$$

  This chain of equalities allows us to see the vertex and frequency based point of view. This term is at the same time weighted squared finite differences (i.e. squared gradient) and a filtering with $g(\lambda) = \lambda$. The frequency interpretation is interesting because it shows that this smoothness term actually penalizes high frequencies.

  Also note that the minimization of this loss gives an explicit form for the solution, i.e. since $\|\nabla_G \mathbf{x}\|_2^2 = \langle \mathbf{x}, \mathbf{L}\mathbf{x}\rangle = \mathbf{x}^T\mathbf{L}\mathbf{x}$ the solution $\dot{\mathbf{x}}$ to :

$$\arg\min_{\mathbf{x}} \|\mathbf{x}-\mathbf{y}\|_2^2 + \|\nabla_G \mathbf{x}\|_2^2 \tag{1.59}$$

  is given by :

$$\mathbf{y} = (\mathbf{I} + 2\gamma\mathbf{L})\dot{\mathbf{x}} \tag{1.60}$$

  and thus

$$\dot{\mathbf{x}} = (\mathbf{I} + 2\gamma\mathbf{L})^{-1}\mathbf{y}. \tag{1.61}$$

  We can recognize this last result as being a simple low-pass filtering with $g(\lambda_\ell) = \frac{1}{1+2\gamma\lambda_\ell}$.

- **Total Variation**: Another quite common regularizer is the graph Total Variation (TV) norm :

$$h_{\mathcal{G}}(\mathbf{x}) = \|\nabla_G \mathbf{x}\|_1 = \sum_{(v_i,v_j)\in\mathcal{E}} \sqrt{\mathbf{W}[i,j]}|\mathbf{x}[i]-\mathbf{x}[j]|. \tag{1.62}$$

  This term can be understood intuitively as the standard TV adapted to the graph : it

favours a solution which is piecewise-constant (or at least piecewise-smooth) on the graph.

- **Data-adapted prior**: As we saw with the graph Tikhonov regularization, the smoothness is enforced by penalizing the high frequencies with a linear kernel. Now, assuming the signal to be recovered is known to have some specific frequency content (e.g. is stationary), one can generalize it for a regularization targeting any frequency band, for example :

$$h_{\mathcal{G}}(\mathbf{x}) = \mathrm{tr}(\mathbf{x}^T g(\mathbf{L})\mathbf{x}) \tag{1.63}$$

where g can be any filter. Note that such a prior favour frequencies not contained in $g(\mathbf{L})$. But doing so is pointless as the goal is to drive the minimization to a class of solutions. Intuitively, we want to penalize the frequency content that we do not expect in the solution.

Very recently, a new approach was proposed for regularization making use of this principle. In [140], it was suggested to use the expected Power Spectral Density (PSD) of the signal $\mathbf{x}$ to regularize the minimization, generalizing Wiener estimation to graphs (i.e. the PSD is the expected frequency content of $\mathbf{x}$). More precisely, assuming $\mathbf{x}$ is Graph Wise Sense Stationary with PSD $s^2(\boldsymbol{\lambda})$ and with $\sigma^2(\boldsymbol{\lambda})$ the PSD of the noise, then:

$$h_{\mathcal{G}}(\mathbf{x}) = \| w(\mathbf{L})(\bar{\mathbf{x}})\|_2^2, \tag{1.64}$$

with $w(\boldsymbol{\lambda}) = \| \frac{\sigma(\boldsymbol{\lambda})}{s(\boldsymbol{\lambda})} \|$ and $\bar{\mathbf{x}} = \mathbf{x} - \mathbb{E}[\mathbf{x}]$. This regularizer provides the MAP estimator for the problem $\hat{\mathbf{x}}|\mathbf{y} = \arg\min_{\mathbf{x}} \|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + h_{\mathcal{G}}(\mathbf{x})$ and is also the linear MMSE (see [140, Theorems 17 and 18]).

1.2.4 Nearest Neighbours Graphs

As we saw above, graphs are globally divided in two categories : some occur naturally, often in the form of networks emerging from relations (or connections) : social networks, road networks, power grids, co-authorship, etc. The second class regroups all graphs created by connecting (the most) similar objects. This approach is motivated by the manifold hypothesis stating that the graph topology approaches the intrinsic geometrical structure of the data. In this section, we delve a bit into how such nearest neighbours graphs are constructed.

In the following, since we want to be general, and assuming we want to find neighbourhoods on $\mathbf{X}$, we consider the distance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ to be $d(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{R}$ which we write $d(i, j)$ for simplicity. For the following definitions to make sense, we assume that $d(.,.)$ is at least a semimetric on $\mathbf{X}$.

Common choices for d are ℓ_p-norms (in particular ℓ_2) or cosine distances, but more complex ones such as Hamming, Wasserstein or Minkowski distances can be used. The choice may impact the efficiency of the graph construction either directly (e.g. Wasserstein distances are

costly) or indirectly, by making approximate methods inapplicable (see Section 1.2.4).

Since edge weights are supposed to indicate the strength of connections between nodes, using the distance directly does not make sense as the bigger the distance the larger the weight of the edge between them would become. The weight function (or kernel) applied on the distances is generalized in the following definitions using $f : \mathbb{R} \to \mathbb{R}$. The traditional choice for this function is the Gaussian (or exponential, or heat) kernel :

$$f(x) = \exp\left(-\frac{x^2}{\sigma}\right). \tag{1.65}$$

It is the most well funded theoretically, as the convergence results towards manifolds are done using this kernel with the ℓ_2 distance (for more details see [48, 20, 21]).

ϵ-NN Graphs

An ϵ-Nearest Neighbours (ϵ-NN) graph connects every point to other points being at a distance less than ϵ. More precisely is weight matrix is defined as :

$$\mathbf{W}[i,j] = \begin{cases} f(d(i,j)) & d(i,j) \leq \epsilon \\ 0 & d(i,j) > \epsilon \end{cases} \tag{1.66}$$

The use of ϵ-NN graphs comes from a first sparsification step of a fully connected graph. Assuming a Gaussian kernel and the Euclidean distance, the weight matrix created using the ϵ-NN definition is a truncated version of a fully connected weight matrix, and thus very close to it. The good theoretical properties of convergence are thus minimally affected.

There are, however, two main issues with ϵ-NN graphs: first, the parameter ϵ needs to be chosen carefully. Indeed, if it is too small, the graphs end up in many disconnected components and if it is too large it tends to a fully connected graph. The second issue is that there are no guarantees that the resulting graph is sparse and there is no easy and direct relation between ϵ and the sparsity level.

kNN Graphs

A k-Nearest Neighbours (kNN) graph is very similar to an ϵNN graph except that only the k closest neighbours of a points are connected to it. More precisely, the weight matrix is defined as :

$$\mathbf{W}[i,j] = \begin{cases} f(d(i,j)) & j \in \mathcal{N}_i(k) \\ 0 & \text{otherwise} \end{cases} \tag{1.67}$$

where $\mathcal{N}_i(k)$ is the set of the k closest point to i using the distance d.

The main advantage of kNN graphs is that they are inherently sparse and the sparsity level can be easily controlled with k. In fact, for such a graph, we have $|\mathcal{E}| = \mathcal{O}(kN)$. While being good to limit the memory footprint and make fast filtering computationally worthwhile, the difference between the kNN adjacency matrix and its fully connected counterpart potentially breaks the theoretical guarantees of convergence to the manifold.

Another issue arising with the kNN condition for connecting two nodes is that it is not necessary symmetric : $j \in \mathcal{N}_i(k) \not\Rightarrow i \in \mathcal{N}_j(k)$. Since $\mathbf{W}$ needs to be symmetric for the Laplacian to be symmetric, there are usually two ways to symmetrize it using either $\frac{1}{2}(\mathbf{W} + \mathbf{W}^T)$ or $\max(\mathbf{W}, \mathbf{W}^T)$. Note that the symmetrized version is still sparse (i.e. $\mathcal{O}(kN)$).

Nearest Neighbour Search

The common need of all these graph definitions is an ability to access sorted pairwise distances between the data points, i.e. to perform Nearest Neighbours (NN) searches. The naive approach to solve this problem is to compute the complete pairwise distance matrix. It is often called *exhaustive search* since for every point we need to pass through all other points. The obvious drawback of this approach is its complexity, as it needs $\mathcal{O}(N^2)$ distance computations.

The need for faster methods has induced the creation of many algorithms. The fundamental aspect to approach this problem is to use hierarchical structures to divide the space, allowing for $\mathcal{O}(\log(N))$ operations to find the closest point to a target. In this section we review some well-known and state-of-the-art such hierarchical partitioning schemes.

Quadtrees and Octrees [66, 125]

Quadtrees and octrees are space partitioning tree structures for data in 2 (respectively 3) dimensions. They both follow the same principle : each node in the tree is defined by a point and has 4 (respectively 8) children which represents the quadrants (respectively octants) around it. These spatial regions are generally called cells or buckets and have a list of points they contain. Starting from the root, each time a cell contains too many points, the space is split and the recursion continues.

A nearest neighbour search in such a structure is done by Depth-First Search (DFS) using comparisons to the node splitting points to descend into the tree. Note that a single DFS is sufficient to get the closest point to a target unambiguously, since the space is split around points and not hyperplanes. The time complexity for an all-point NN-search is $\mathcal{O}(N\log(N))$.

The main drawback of these structures is that the trees can be very unbalanced and have a large height (typically more than $\log(N)$) if the data concentration is varying around the space. The second issue is that they are restricted to 2 or 3 dimensional data. While the generalization to d dimensions is possible, it would imply a tree with a branching factor of 2^d, with most of the branches empty.

Quadtrees are typically used in Barnes-Hut schemes [15] where the intermediate nodes are used directly as representative of the points in their cell. This allows typically to approximate the full t-SNE embedding similarity matrix efficiently, at the cost of imposing a target dimensionality of 2.

k-d trees [24]

Also in the class of spatial partitioning trees, k-dimensional trees (k-d trees) are well known structures used for fast NN-search. In contrary to quad and octrees, k-d trees are binary and split the space using hyperplanes instead of points.

More precisely, each node of the tree is a k-dimensional point associated with a hyperplane. The hyperplanes are aligned on one axis of the k-dimensional space meaning the two half-spaces it defines can be decided using only one coordinate. When the height of the tree is bigger than k, they are simply reused in cycles.

Usually, the point selected as anchor for the hyperplane splitting is chosen based on a median of the points in the subtree (or random estimation of the median, for efficiency). The reason for this choice is that it tends to produce balanced trees [31]. The search is then performed by first doing a DFS on the tree to get the first candidate and then traversing back to the root by updating the best target point for each hyperplane and descending into the tree if necessary. In terms of complexity : it takes $\mathcal{O}(N \log N)$ both for construction and for all-point NN-search.

Compared to quadtrees and octrees, k-d trees have two advantages : they can naturally deal with more than 3 dimensions and they produce more balanced trees. While k-d trees can in theory handle data of any dimensionality, they prove to be quite inefficient for high-dimensional data. The reason usually advanced is that the backtracking to the root can take a lot of time as the first best candidate is rarely the good one and the search can become close to linear instead of being $\mathcal{O}(\log(N))$ [172].

To counter this undesirable effect, a few schemes have been proposed in two directions : on one hand, instead of applying a simple backtracking of the nodes encountered in the DFS, the authors of [7] introduced the notion of priority queue. While descending in the tree the nodes are added to the priority queue which is sorted by increasing distance to the query point. Once the bottom of the tree is reached, the nodes are evaluated in the order given by the priority queue, the subtrees are processed in a DFS manner and candidates added to the priority queue. Another approach, proposed in [16], is to allocate a finite budget (in number of evaluations or time) for the number of nodes to visit. This is called Best-Bin-First (BBF) and simply states that the nearest neighbour is approximate : it often is the exact closest point and otherwise, points that are not too far.

Vantage-point trees [215]

Vantage-point trees (independently proposed in [195] as metric trees) are also part of spatial

partitioning trees. They are similar to the other techniques presented in this section in that they are binary trees where the splitting points, called vantage points, are nodes of the tree.

While being very similar to k-d trees, vantage-point trees present two main differences : first, the splitting is done using hyperspheres instead of hyperplanes and second, it can naturally handle other metrics than the Euclidean distance.

Similarly to the k-d tree construction, vantage points and the radius of the high-dimensional balls are selected using the median of the points. The left subtree then contains the set of points outside the ball and the right subtree the points on the inside. If a child node contains too many points, the process is repeated recursively. Using the same method for search than for k-d trees, the complexity of both the construction and all-pairs search is also $\mathcal{O}(N\log(N))$.

Vantage-point trees and k-d trees are constructed and perform similarly. They differ mainly only in that the former works by coordinate projection and the latter using high-dimensional distance thresholding. This last fact is the main advantage of vantage-point trees in that they can handle any metrics, since only distance comparisons are needed. This is the reason why this method is used to compute the high-dimensional sparse similarity matrix in BH t-SNE.

Randomized k-d trees [172]

In order to alleviate the inefficiency of k-d trees in high dimensions, randomized k-d trees were introduced. They used BBF and priority queues, which thus provides approximate nearest neighbours. In addition, three major improvements were adopted : first, the construction of m (globally) independent trees that are searched in parallel. Second, the total number of comparisons (i.e. DFS and backtracking) is limited to n evaluations in total in a shared priority queue. Third, the trees can be made independent by rotating them using PCA so that the different trees are aligned to the principal axes of the data.

The first and last improvements can be implemented independently, i.e. by randomly rotating the trees or by using a single PCA aligned tree. For simplicity, the trees are not actually rotated but built on rotated data, i.e. on $\mathbf{R}_i\mathbf{X}$ with $\mathbf{R}_i$ a rotation matrix. Using this technique, after construction of the different trees, there is no need of keeping the rotated instances of the data, as an evaluation can be made with $\mathbf{R}_i\mathbf{x}$ as a target.

The speed improvements come from a priority queue for the search spanning all the m trees. By searching in this priority queue the best candidate across all trees is evaluated and thus removed from the search in all other trees. In [128] it was shown that the performance improves significantly up to $m = 20$ trees evaluated. The reason advanced for the speed improvement is that if the first best candidate encountered after the DFS in the priority queue is not the true nearest neighbour, then the true best candidate must lie on the other side of one of the separating hyperplanes, which is why the randomization increases the probability that those two points are not separated in an other tree, leading to a very fast convergence without much backtracking.

This algorithm has been included in the Fast Library for Approximate Nearest Neighbours (FLANN) [128] and is the default method we use in this thesis to construct kNN graphs.

Hierarchical k-means [128]

The authors of FLANN proposed an alternative to randomized k-d trees which they called priority search k-means trees. The goal of this algorithm is to cluster data in high-dimension in order to recursively construct the tree.

More precisely, the data $\mathbf{X}$ is split in k clusters using the k-means algorithm [117]. Then, the centroid of each cluster is considered to be a node in the tree and its k children are defined by reapplying k-means on its cluster. The recursion is stopped once there are no more than k points in a cluster.

Its main differences with k-d trees is that all dimensions are used for every comparison instead of only one at a time. In addition, the branching of the tree is k, and thus its height is necessarily well constrained. The search is done similarly to other trees with a DFS, BBF and a priority queue, which makes it an approximate algorithm. Its complexity is thus also $\mathcal{O}(N\log(N))$.

Random Projection trees [54]

Also starting from the fact that k-d trees poorly adapt to high dimensions, the authors of [54] proposed a variant of k-d trees called Random Projection (RP) trees. Basically, instead of splitting the data with a hyperplane aligned on a coordinate axis of the data, the normal of the plane is randomly picked as a high-dimensional vector centred at the origin. Then instead of splitting data at the median, a small random noise is added to the centre of the hyperplane.

Such a tree happens to adapt well to the intrinsic dimensionality of the data. Indeed, the original authors proved that the cell size adapts to the intrinsic dimensionality of the data (see [54, Theorem 3]). Using a compressive sensing point of view, RP trees work on the fact that high-dimensional sparse data can be reconstructed from few random projections.

Another way to see the solutions provided by RP trees is to analyse the problem using the growth of the tree height needed to keep the cell size constant (defined by the hyperplanes) in function of the dimension. In fact, we see that the tree height needs to grow linearly with the dimension, or the data to grow exponentially, so that the cell size keeps its discriminatory power (see [54] for a proof).

Modern implementations of RP trees are in the most efficient NN-search algorithms. For example, Annoy is a library for approximate nearest neighbour search used by Spotify.[2]

Graph-based methods

Nearest neighbour methods, either for general Approximate NN (ANN) search method or for

[2] https://github.com/spotify/annoy

*k*NN graph construction, tend to more and more use graphs. The principle behind this trend is simply that *neighbours of neighbours are probably also neighbours*.

NN-Descent [60] is a scheme which iteratively improves an approximation of a *k*NN graph using any similarity metric. Essentially, the NN-descent starts with a random *k*NN graph and then explores sequentially the 2-hop neighbourhood for each node in order to improve its accuracy. The iterations can be stopped once no improvements are made. The shrinking of the diameter for each neighbourhood is formulated as a gradient descent, hence the name.

The LargeVis method [187], described in the dimensionality reduction methods, also brings a scheme for an efficient construction of an approximate *k*NN graph. Their method is to use multiple RP trees with low precision parameters to build a first rough graph, and then use local exploration techniques to refine it. More specifically, once a first approximate graph is constructed, the neighbours are refined using its 2-hop neighbours. In fact, the LargeVis method can be seen as applying one iteration of NN-descent on a graph created from coarse RP trees.

Another very recent method called Hierarchical Navigable Small World graphs (HNSW) [121] use graphs to approach the ANN search problem. The technique is based on multi-layer graphs that are created by iteratively inserting nodes and finding its neighbours in a poly logarithmic greedy search. Once the multi-layer graph is constructed, the search is done in a top-down manner and refines the best neighbours at each level by following node-level edges between levels and 2-hop step exploration at the graphs level.

The question is not what you look at, but what you see.
— Henry David Thoreau

2 Embedding quality evaluation

Evaluating the quality of a dimensionality reduction algorithms depends on what properties are thought to be preserved by the transformation. As we saw in our short review in Section 1.1.2, this includes pairwise distances and local neighbourhood preservation. Both concepts can be generalized to a similarity-preserving mapping, independent of a notion of distance, where the similarity can be evaluated on labels associated to the data points.

In the context of visualization, simple visual inspection is often used to assess quality (generally implying an access to labelled data). When labels are not available, a common practice is to generate the labels using a clustering of the points in high dimension [187]. The main issue with visual inspection is that, in addition of being a bit subjective, it is often the only provided quality assessment of the data. In fact, a recent review of dimensionality reduction and visualization techniques showed that 40% of the corpus they analysed did not use any kind of quantitative evaluation [202].

As we will see in the next section, quantitative measures do exist to compare different embeddings. However, they are often quite general and do not assess specific characteristics, e.g. concentration, cluster splits or noise level. In addition, some involve very costly operations (e.g. all-paths shortest paths), or are not based on very sound theoretical foundations.

In this chapter we propose three quantitative measures of the quality of embeddings, and, by extension, of dimensionality reduction algorithms. The first one is used to assess the global quality using graph cuts and their relation to the clustering problem, the second one measures cluster concentration (or splitting) using active sampling and random walks and the last one captures the noise level using the norm localized filters.

The organization of this chapter is as follows: in Section 2.1 we review the related work in formal measures for quality evaluation. Then, in Section 2.2 we propose new graph-based measures for quantitative quality assessment of embeddings and then devote Section 2.3 for experiments on both synthetic and natural datasets. Finally, we give a brief summary and future directions in Section 2.4.

This chapter expands the work presented in a paper written in collaboration with Nathanaël Perraudin and Pierre Vandergheynst [138] which introduced early results on the subject. The content presented in this chapter strengthen the theoretical background, introduce new methods and present extended experiments compared to the original paper. In addition, the active sampling algorithm presented in Section 2.2.2 is the result of unpublished work done in collaboration with Nathanaël Perraudin.

2.1 Related work

In this section, we review some of the existing approaches for empirical performance comparison of dimensionality reduction methods. We refer the interested reader to [110, 202, 127] for more detailed reviews of quality evaluation in dimensionality reduction. We focus here on methods which allow comparisons (e.g. measuring the effect of parameters in a method, or comparing the embedding produced by different algorithms) and we do not discuss the quality evaluation of one specific visualization instance. Reviews of such fine-grained studies are presented in [196, 8, 127].

Similarly to most techniques in machine learning and data analysis, we can categorize quality evaluation methods as either supervised or unsupervised, i.e. using labels or not. We start by discussing the main unsupervised techniques, since they form the major part of the literature.

A first very natural formal measure is to consider the reconstruction error. Assuming that dimensionality reduction is performed using some mapping $\mathbf{Y} = \mathbf{XM}$, the reconstruction error can be expressed by measuring the error between the original data points and their reconstructed version $\mathbf{YM}^{-1}$. Minimizing the reconstruction error is actually the objective of some methods such as PCA or auto-encoders. However, except for those specific examples, measuring the reconstruction error is often not applicable in practice because the inverse mapping is generally not available. In addition, assuming that high-dimensional data is noisy or redundant, the notion of inverse mapping is ill-defined. Indeed, different high-dimensional representations generated by the same intrinsic information can have very different reconstruction errors simply due to the lack of the unnecessary ability to reconstruct a specific realization of noise.

A second approach is derived from local neighbourhood preservation. Simply stated, it corresponds to measuring the difference of local properties between the high and low-dimensional representations. Many techniques use this principle and the state-of-the-art is mainly divided in two trends: distance versus rank. Distance-based measure quantify the disparities between distances in high and low dimensions while techniques focusing on rank only consider the order of neighbours sorted by increasing distance.

The principle of distance-based preservation is used directly in dimensionality reduction algorithms such as MDS or Sammon mapping, but they are not the most popular because of two issues. First, they are subject to the scaling problem: monotonic transformation of

distances may impact the measures while not fundamentally affecting an embedding. Second, high-dimensional data may be accessible only through a similarity matrix or non-Euclidean distances which make the comparison to the embedding space unfeasible. Nevertheless, some methods (e.g. precision and recall) proposed in [202] use distance preservation for quality assessment.

In order to avoid the two main issues of distance agreement between high-dimensional data and its embedding, ranks are often preferred. Intuitively, the rank of a query point to a target is the number of points closer to the query than the target. More formally, the high-dimensional rank R_{ij} between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ is $R_{ij} = |\mathcal{N}^{\mathbf{X}}_{i<j}|$ with $\mathcal{N}^{\mathbf{X}}_{i<j}$ the set of points in $\mathbf{X}$ closer to $\mathbf{x}_i$ than $\mathbf{x}_j$. Similarly, the low-dimensional rank between $\mathbf{y}_i$ and $\mathbf{y}_j$ is given by $r_{ij} = |\mathcal{N}^{\mathbf{Y}}_{i<j}|$. The disparity between both is then simply measured by the rank error $R_e(i,j) = r_{ij} - R_{ij}$. While numerous measures use ranks, a unifying framework has emerged using the so-called co-ranking matrix, which can be seen as a histogram of rank errors. More precisely, the co-ranking matrix $\mathbf{C}$, as originally defined in [111] has entries of the form :

$$\mathbf{C}_{kl} = \left|(i,j) : R_{ij} = k, r_{ij} = l\right|. \tag{2.1}$$

A perfect rank preservation would imply having non-zero entries only on the diagonal of the matrix $\mathbf{C}$. Formal measures based on the co-ranking matrix thus use the deviation from the diagonal as a quality criterion. The authors of [110] have actually shown that many measures can be expressed using weighted sums of entries in the co-ranking matrix. In particular, traditional measures such as trustworthiness and continuity [93, 200, 201] can be expressed directly using this framework. Other examples exist such as local continuity meta-criterion (LCMC) [41], mean relative rank errors [109] or point-wise co-ranking matrix [127]. In addition to these general methods, ranks are also used in different contexts such as Principal Manifold Analysis [73] or non-linear divergences [182].

Although being attractive and providing meaningful measures, rank-based methods are impeded by one major issue: scaleability. Indeed, the reported complexity of computing the co-ranking matrix, using its original definition in [111], is $\mathcal{O}(N^2 log N)$ because it consists of $2N$ sorting operations. A point-wise co-ranking matrix is cheaper to compute but still has a $\mathcal{O}(N^2)$ complexity. As a matter of fact, the timing experiments performed in [127] are restricted to less than 10′000 data points in $\mathbb{R}^{10}$ due to computational constraints.

The last class of unsupervised quantitative measures is closely related to the graph drawing problem which is: knowing the adjacency matrix of a graph, the goal is to get 2D coordinates for the nodes so that drawing the edges is meaningful. Studies analysed the link between the properties of 2D graph embeddings and their quality (assessed by perceptual and cognitive issues for humans). The results of [149] and [206] tend to show that edge crossings is one of the most important factors for the quality of graph drawings. Indeed, minimizing the number of crossings is correlated with a good quality assessment. Those measures, while interesting because they only depend on a graph or similarity matrix as input, are very difficult to measure

and apply in practice and are rarely used in the context of dimensionality reduction, especially for large-scale data.

Much less supervised formal measures have been proposed in the literature. And while some techniques simply extend unsupervised measures to labelled data (e.g. group compactness based on ranks [73]), quality evaluation is generally viewed using another angle. In the supervised case, the quality of the agreement between the high-dimensional data and its embedding is less important than the class separability of the data.

Based on this idea, the most common approach to assess the quality of embeddings, assuming labels are associated to the data, is to use the low-dimensional points to train a classification algorithm and then measure its generalization error. Doing so indeed assesses the separability of the emdedding, which is assumed to be related to its quality. An obvious choice is to use a kNN classification algorithm, as it does not apply any transformation on the low-dimensional data, and simply provides the majority label of a node's neighbourhood. To increase the variance of the method (i.e. to avoid the smoothing effect of large neighbourhoods), 1-NN classifiers are often used in practice, see e.g. [118, 198, 158, 187].

2.2 Graph-based measures

In this section, we will propose new supervised formal quality measures for embeddings. While unsupervised methods are often preferred since they do not rely on extra information, we think that supervised methods can extract more precise characteristics while still being applicable to almost any data. Indeed, despite the fact that we consider the problem settings for which the data points are associated to some categorical information, data points with no label or multiple labels can nevertheless be accommodated. The former by computing an artificial labelling using clustering of the high-dimensional data, and the latter without special adaptation except a right normalization. Lastly, since dimensionality reduction is often used for visualization, labelled (or partially labelled) data is quite common since it frequently gives pertinent insights.

The objective for the measures we propose is focused on providing tools to compare embeddings, not to evaluate a single realization. In particular, a main goal is to be able to finely compare different algorithms (or different parameters for a specific one) using the evaluation of the embeddings they produce. In order to have meaningful measures, some normalization is systematically performed.

All the techniques we propose share a common point in that they are based on a similarity graph constructed between the points in the embedded domain, that we will call $\mathcal{G}_e$ to distinguish from $\mathcal{G}$. The graph $\mathcal{G}_e$ is a simple kNN graph constructed from the embedding $\mathbf{Y}$ using the Euclidean distance. Note that $\mathcal{G}$ and $\mathcal{G}_e$ share the same vertex set $\mathcal{V}$. In addition, we write the set of categorical labels (also called classes) $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$. For each class c_i we note V_{c_i} the subset of vertices of $\mathcal{G}_e$ having the label c_i. We write the number of vertices of a label

c_i as $N_{c_i} = |\mathcal{V}_{c_i}|$ and the total number of labelled points $N_{\mathcal{C}} = \sum_{c_i \in \mathcal{C}} N_{c_i}$. Note that in some situations we may have $N_{\mathcal{C}} \neq N$; for example, $N_{\mathcal{C}} < N$ for partially labelled sets or $N_{\mathcal{C}} > N$ with multiple labels per node.

The first method we introduce is based on balanced graph cuts and measures the clusterability of $\mathcal{G}_e$. The second technique uses the norm of localized filters to estimate the number of positional outliers. The third method uses the combination of an active sampling of the nodes and the length of random walks on the samples to estimate the amount of cluster split (i.e. if classes spread on multiple parts of the embedding).

2.2.1 Supervised Graph Cuts

In order to use graph cuts, we start with a few definitions. A cut partitions a graph $\mathcal{G}$ in two complementary sets of vertices S and S^c with $V = S \cup S^c$ and $S \cap S^c = \emptyset$. The graph cut operator is then defined as

$$Cut(S, S^c) = \sum_{i \in S} \sum_{j \in S^c} \mathbf{W}_{ij} \tag{2.2}$$

which represents the total weight of the edges between S and S^c, or the weight of the edges trimmed by the cut.

In order to define balanced cuts, we also need to use the volume operator which is defined as

$$Vol(S) = \sum_{i \in S} d(i) \tag{2.3}$$

where $d(i)$ is the degree of the vertex v_i.

The first interest of cuts in the context of clustering is that the minimization of Eq. (2.2) happens to be a solution to the clustering problem [211]. The minimal cut is, however, rarely used in practice as it tends to favour small sets of isolated vertices. Due to this fact, the focus shifted to balanced cuts, which are normalized by the volume and thus equalize the size of the clusters. Two of the most popular balanced cuts are the Cheeger cut [39] and the Normalized cut [164].

The Cheeger cut is related to the Cheeger constant which is defined as :

$$h_c(\mathcal{G}) = \min_{S \subseteq V} \frac{Cut(S, S^c)}{\min(Vol(S), Vol(S^c))} \tag{2.4}$$

for a graph $\mathcal{G}$.

Similarly, the normalized cut is defined as :

$$h_n(\mathcal{G}) = \min_{S \subseteq V} \frac{Cut(S, S^c)}{Vol(S)} + \frac{Cut(S, S^c)}{Vol(S^c)}. \tag{2.5}$$

Both numbers give a measure of the optimal clusterability of $\mathcal{G}$, i.e. it is small if there exist two subsets of vertices separated with a strong bottleneck and large otherwise.

Average Clusterability Index

The Cheeger cut and Cheeger constant imply a minimization in order to find the best partitions, but in our case, we already have them since they are derived from the labels. We can thus reformulate Eq. (2.4) to define a Cheeger score for a class c_i as :

$$h_c(\mathcal{G}_e, c_i) = \frac{Cut(\mathcal{V}_{c_i}, \mathcal{V}_{c_i}^c)}{\min(Vol(\mathcal{V}_{c_i}), Vol(\mathcal{V}_{c_i}^c))}, \tag{2.6}$$

and respectively a Normalized cut score :

$$h_n(\mathcal{G}_e, c_i) = \frac{Cut(\mathcal{V}_{c_i}, \mathcal{V}_{c_i}^c)}{Vol(\mathcal{V}_{c_i}} + \frac{Cut(\mathcal{V}_{c_i}, \mathcal{V}_{c_i}^c)}{Vol(\mathcal{V}_{c_i}^c)}. \tag{2.7}$$

Computing the above quantity for a given class gives a measure of its clusterability from which we can define the Average Clusterability Index (ACI) as an average weighted by the classes cardinality :

$$\text{ACI} = \frac{1}{N_{\mathcal{C}}} \sum_{c_i \in \mathcal{C}} N_{c_i} h(\mathcal{G}_e, c_i), \tag{2.8}$$

where $h(\mathcal{G}_e, c_i)$ corresponds to one of the class cuts defined in Eq. (2.6) and Eq. (2.7). This score, inspired by the balanced cuts, inherits its properties: small values mean that the classes are well separated in the graph and large values mean that the classes are more mixed.

Since it is entirely based on sums of the weighted adjacency matrix, the time complexity of this method is bounded by the graph creation time. As we saw in Section 1.2.4, the best approximate kNN search methods have all a $\mathcal{O}(N \log(N))$ time complexity.

2.2.2 Active Random Walks

The ACI introduced in the previous section is useful to evaluate the clusterability of the data from the labels (i.e. if labels provide a good partitioning of $\mathcal{G}_e$). However, this metric, as well as others (such as 1-NN generalization error) will not be very sensitive to cluster splitting. Take for example a dataset with ten classes which should be separated in ten clusters. A dimensionality reduction algorithm may provide an embedding with more than ten clusters, meaning that at least one class is split in more than one cluster. The ACI between a perfect separation and an over-separation will be almost indistinguishable, as both embedding scenario will result in highly clusterable classes.

In order to evaluate this cluster splitting effect, we need to measure the overall concentration of all points in a class, i.e. that all points in a class are reasonably close to each other. An idea to

measure this effect could be to compute the average of all geodesic pairwise distances between points in a given class. Indeed, it should be small if a class is well concentrated and larger if a class is split around different cluster centres. The main problem of this approach is that it would imply $\mathcal{O}(N_{c_i}^2)$ distance computations for each class c_i, which makes it infeasible in practice. This complexity issue could be solved by random sampling pairs of points, but leaves the problem of the geodesic distance evaluation unresolved, which by itself is computationally expensive.

Starting with this global idea, we propose a measure which solves both issues while still providing a good estimator for class-based spreading. In order to avoid considering all pairwise distances, we will use a sampling strategy. But in order to be efficient, we use an active sampling scheme instead of simple uniform sampling. Second, we solve the problem coming from the high cost of geodesic distance computation by introducing an approximate geodesic metric. Both schemes make use of localized filters, and thus their combined cost is not bigger than their individual ones.

The first part of this section is dedicated to the description of the active sampling strategy. In a second part we introduce a new function called the Kernelized Diffusion Distance, which we prove to be a metric on the graph. Finally, we combine both concepts to introduce the concept of Active Random Walks, whose length will be used to define class-based concentration scores.

Active sampling

In this section, we introduce a sampling scheme on graphs using localized kernels. More specifically, we want a method which provides samples following the graph structure, with the goal of providing the best cover of the nodes using the fewer possible atoms. By cover, we mean that the energy diffused from the sampled nodes using atoms localized at the samples is non-zero for every node.

Here, we also allow for the sampling to be active, i.e. samples are iteratively picked, with the knowledge of all previous picks. The motivation behind this approach is to quickly maximize an objective. More precisely, assuming the set of N_s samples is written $\mathcal{S} = \{s_0, s_1, \dots, s_{N_s}\}$, with $\mathcal{S} \subset \mathcal{V}$, let us define the cumulative energy at node j to be :

$$E_{\mathcal{S}}[j] = \sum_{s_i \in \mathcal{S}} \frac{\mathcal{T}_{s_i} g^2[j]}{\|\mathcal{T}_j g\|_2^2}. \tag{2.9}$$

Our main objective can be written as:

$$\min_j E_{\mathcal{S}}[j] \geq \epsilon \tag{2.10}$$

for some $\epsilon \geq 0$. This optimization can be reformulated in two ways: first, assuming we have a

fixed ϵ, we seek the set of samples of smallest cardinality, i.e.

$$\mathcal{S}^* = \arg\min_{\mathcal{S}} |\mathcal{S}| \quad \text{with} \quad \min_j E_{\mathcal{S}}[j] \geq \epsilon. \tag{2.11}$$

Second, assuming a fixed number of samples N_s, we want the set of samples which provides the largest ϵ. We thus look for an optimal set of samples $\mathcal{S}^*$ such that

$$\mathcal{S}^* = \arg\max_{\mathcal{S}} \left(\min_j E_{\mathcal{S}}[j] \right) \quad \text{with} \quad |S| = N_s. \tag{2.12}$$

While keeping in mind these optimization problems, we do not give here a formal proof that our proposed scheme is optimal and leave it as future work. Instead, we will give a theoretical intuition and solve the problem in a greedy fashion.

To initiate the sampling procedure we propose to focus on the concentration of the localized atoms. If we wish to select a node on which the energy of localized atoms spreads, we can look for nodes which have low concentration values. Following this idea, we can select the node with the largest spread to be the first sample. More formally, it can be computed as:

$$s_0 = \arg\min_i \frac{\|\mathcal{T}_i g\|_2}{\|\mathcal{T}_i g\|_1}. \tag{2.13}$$

The first sample being defined, we want to actively take its contribution into account for the next pick. The most intuitive choice is to take a greedy pick: choose as next sample the one with the lowest cumulative energy, i.e. $s_1 = \arg\min_i E_{\mathcal{S}}[i]$. Using this scheme, we make the best local optimization, by ensuring that the minimum (or at least the number of minimal elements) is increased at each step. Combining everything together, we derive our active sampling method which is detailed in Algorithm 1.

Algorithm 1 Active Sampling (Original)

Require: $\mathcal{G}_e$, N_s

1: Compute the initial sample $s_0 = \arg\min_i \frac{\|\mathcal{T}_i g\|_2}{\|\mathcal{T}_i g\|_1}$
2: Compute the initial weight $c_0[i] = \left(\mathcal{T}_{s_0} g[i]\right)^2$
3: **while** $n < N_s$ **do**
4: Compute the new sample as $s_n = \arg\min_i c_{n-1}[i] / \|\mathcal{T}_i g\|_2^2$
5: Update the weight $c_n(i) = c_{n-1}(i) + \left(\mathcal{T}_{s_n} g[i]\right)^2$
6: **end while**
7: return $\mathcal{S} = \{s_0, s_1, \ldots, s_{N_s-1}\}$

This algorithm, while reasonably well motivated theoretically, has one major practical issue : computing ℓ_p-norms of $\mathcal{T}_i g$ is costly (i.e. require N filtering). As we introduced in the previous chapter (see Lemma 5), the quantity $\|\mathcal{T}_i g\|_2^2$ can be well estimated using random signals. However, the problem remains for $\|\mathcal{T}_i g\|_1$. We will solve this specific issue by working with a filter possessing interesting properties : the heat kernel. His first remarkable characteristic is stated in the following lemma.

Lemma 6. *For any function of the form of a heat kernel* $g_h(x) = \gamma e^{-\tau x}$ *with* $\tau \geq 0$ *and* $\gamma > 0$, $\forall i \in \{1,\dots,N\}$, $j \in \{1,\dots,N\}$ *we have :*

$$(\mathcal{T}_i g_h)[j] \geq 0. \tag{2.14}$$

The proof of this lemma is given in Appendix A.1.2.

This result shows that localized heat kernels are always nonnegative on the entire vertex set, which happens to be useful for the issue at hand. Indeed, we can rewrite the ℓ_1-norm of the heat kernel

$$\|\mathcal{T}_i g_h\|_1 = \sum_j |\mathcal{T}_i g_h[j]| = \sum_j \mathcal{T}_i g_h[j], \tag{2.15}$$

using the positivity property for the second equality.

Now that the absolute value is removed, we can derive a very simple expression:

$$\|\mathcal{T}_i g_h\|_1 = \sum_j \mathcal{T}_i g_h[j] = g(0) = 1, \tag{2.16}$$

where the second equality comes from the third property of [170, Corollary 1]. This surprising result happens to completely solve our problem since the quantity $\|\mathcal{T}_i g_h\|_1$ does not imply any computation. It also means that the concentration of heat kernels can be computed by a direct evaluation of $\|\mathcal{T}_i g\|_2^2$, which allows us to compute the initial sample using simply

$$s_0 = \arg\min_i \|\mathcal{T}_i g\|_2, \tag{2.17}$$

by taking the square root of the fast estimator $\dot{T}_K$ defined in (1.39).

Regarding the greedy steps after initialization, it is arguable that ensuring an increase of the minimum cumulative energy does not necessarily mean that the new sample diffuses sufficiently to its neighbours. In order to regularize by taking the spread into account, the sample selection at line 5 of Algorithm 2 could be replaced with :

$$s_n = \arg\min_i \frac{c_{n-1}[i]}{\|\mathcal{T}_i g\|_2^2} + \gamma \|\mathcal{T}_i g\|_2 \tag{2.18}$$

where $\gamma \geq 0$ is a regularization parameter.

In Figure 2.1 we show the effect of this regularization on a random sensor graph with a fixed N_s and varying γ. The first measure we report is the value of $\epsilon = \min_j E_{\mathcal{S}}[j]$. For fairness, we also measure the percentage of nodes under a fixed cumulative energy threshold ϵ_t. That is, if we define the set of nodes whose cumulative energy is below ϵ_t as $V_{\epsilon_t} = \{v_j : E_{\mathcal{S}}[j] < \epsilon_t\}$, then we report $\frac{1}{N}|V_{\epsilon_t}|$.

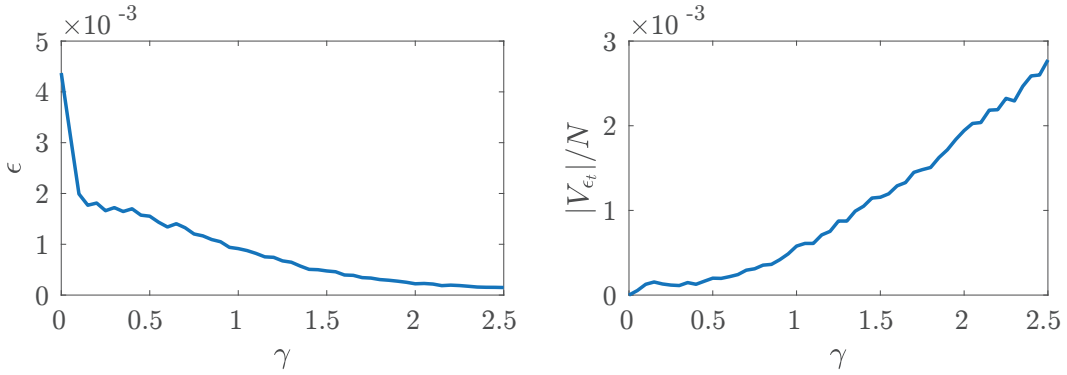

Figure 2.1 – **Effect of regularization in active sampling**. Application of the active sampling algorithm with $N_s = 10$ on a random sensor network of 1000 nodes. All values reported are averages over 2000 runs of the algorithm. On the left, the minimum value of cumulative energy $\epsilon = \min_j E_S[j]$ is shown for varying values of γ. On the right, the ratio of nodes below a cumulative energy threshold to the number of vertices $|V_{\epsilon_t}|/N$ is reported for varying γ.

As expected, ϵ is at his maximum at $\gamma = 0$ and then decreases monotonically (up to randomness effects) with higher values of γ. For the second measure, we could expect that regularizing has a good impact on the percentage of nodes above ϵ_t, since nodes with larger spread are favoured. Contrarily to this intuition, we see that the lowest percentage of nodes below threshold is attained at $\gamma = 0$ and then it monotonically increases with higher values of γ. This small experiment tends to show that the best results are obtained with $\gamma = 0$, i.e. without any regularization at all. We will see, however, that while no regularization seem to be needed in noiseless cases, it can be needed when sampling data corrupted with sparse noise. The definitive version of our active sampling algorithm using the heat kernel and regularization is provided in Algorithm 2.

Algorithm 2 Active Sampling (Regularized)

Require: $\mathcal{G}_e, N_s, \gamma$
Ensure: $\gamma \geq 0$, g is a heat kernel
1: Compute the initial sample $s_0 = \arg\min_i \|\mathcal{T}_i g\|_2$
2: Compute the initial weight $c_0[i] = \left(\mathcal{T}_{s_0} g[i]\right)^2$
3: **while** $n < N_s$ **do**
4: Compute the new sample as $s_n = \arg\min_i c_{n-1}[i] / \|\mathcal{T}_i g\|_2^2 + \gamma \|\mathcal{T}_i g\|_2$
5: Update the weight $c_n(i) = c_{n-1}(i) + \left(\mathcal{T}_{s_n} g[i]\right)^2$
6: **end while**
7: return $\mathcal{S} = \{s_0, s_1, \ldots, s_{N_s-1}\}$

The overall complexity of our proposed active sampling algorithm is $\mathcal{O}(K)$ filtering operations for the initialization step using the estimator for $\|\mathcal{T}_i g\|_2^2$ followed by $\mathcal{O}(N_s)$ filtering for the active updates. Assuming an order m polynomial approximation for the fast filtering, we get a global $\mathcal{O}((K + N_s)m|\mathcal{E}|)$ complexity.

Kernelized Diffusion Distance

Before tackling the problem of measuring the cluster splitting, we make a small detour to introduce a new method to approximate geodesic distances. The main problem with true geodesic distances is that they are computed with algorithms of quite high complexity (e.g. all pairs shortest path using Dijkstra or BFS) which do not scale well for large datasets.

In this section, we propose a metric that provides a good approximation of a geodesic distance and scales well with respect to the number of data points. In fact, we will prove that what we propose is a generalization of diffusion distances. The key tool for this new metric is the localization operator.

The idea of our approach is to use localized atoms to define distances by measuring the norm of the difference between a filter localized at two different nodes. We call it the Kernelized Diffusion Distance (KDD) and define it as:

$$\mathrm{KDD}(i,j) = \|\mathcal{T}_i g - \mathcal{T}_j g\|, \tag{2.19}$$

where g is a kernel defined in the graph spectral domain. Before going further, and as it will be useful later, let us derive a corollary definition of Eq. (2.19)

$$\mathrm{KDD}(i,j) = \sqrt{\sum_\ell g(\boldsymbol{\lambda}_\ell)^2 (\boldsymbol{u}_\ell^*[i] - \boldsymbol{u}_\ell^*[j])^2}. \tag{2.20}$$

This alternative definition is derived in Appendix A.2.1. Let us now examine the properties of the KDD by stating the following theorem.

Theorem 1. *The space* $(\mathcal{V}, \mathrm{KDD})$ *with* $\mathcal{V}$ *the vertex set of a graph and the* KDD *as defined in Eq.* (2.19) *is a pseudometric space, that is, for every* $x, y, z \in \mathcal{V}$*:*

1. $\mathrm{KDD}(x,y) \geq 0$
2. $\mathrm{KDD}(x,y) = \mathrm{KDD}(y,x)$
3. $\mathrm{KDD}(x,z) \leq \mathrm{KDD}(x,y) + \mathrm{KDD}(y,z)$

The proof of this theorem is given in Appendix A.2.1. From there, the only missing property to prove that the KDD is a metric is the identity of the indiscernibles, i.e. $\mathrm{KDD}(i,j) = 0 \iff i = j$. This can be achieved by using an additional hypothesis on g, which is formulated in the following theorem.

Theorem 2. *The space* $(\mathcal{V}, \mathrm{KDD})$ *with* $\mathcal{V}$ *the vertex set of a graph and* KDD *as defined in Eq.* (2.19)*, with* $g(\mathbf{L})$ *being full rank (i.e.* $g(\lambda_\ell) > 0$, $\forall \lambda_\ell \in \Lambda$*) is a metric space, that is, for every* $x, y, z \in \mathcal{V}$*:*

1. $\mathrm{KDD}(x,y) \geq 0$
2. $\mathrm{KDD}(x,y) = \mathrm{KDD}(y,x)$

3. $\mathrm{KDD}(x,z) \leq \mathrm{KDD}(x,y) + \mathrm{KDD}(y,z)$

4. $\mathrm{KDD}(x,y) = 0 \Leftrightarrow x = y$

The proof of this theorem is given in Appendix A.2.1.

Diffusion distance

As was hinted in the name, the Kernelized Diffusion Distance happens to be a generalized diffusion distance. Indeed, taking its spectral formulation we have :

$$\mathrm{KDD}(i,j) = \sqrt{\sum_{\ell} g(\boldsymbol{\lambda}_\ell)^2 (\boldsymbol{u}^*_\ell[i] - \boldsymbol{u}^*_\ell[j])^2} = D_t(i,j), \tag{2.21}$$

where $D_t(i,j)$ is the diffusion distance associated to specific kernels depending on t (i.e. the diffusion parameter). If we take two common definitions of the diffusion distance, the original works of [130, 50] use a kernel of the form $g(x) = x^t$ and the Graph Diffusion Distance defined in [77] uses the heat kernel $g(x) = e^{-tx}$.

The interest of diffusion distances is that they are an interesting proxy for geodesic distances and an alternative to exact shortest-path computation, which is computationally expensive. While similar, they provide a different view of geodesic distance, in that shortest paths are more local (i.e. consider only the edges on the path to compute a distance) and diffusion distances are more global (i.e. consider multiple possible paths between two nodes). In fact, diffusion distances can be seen as the integration of all possible paths between two nodes.

This particular property of diffusion distances has two consequences: they provide a poorer discrimination but at the same time are much more robust to noisy edges. This is particularly interesting when facing what is called short-circuiting problems. Let us take an example: imagine a kNN graph is created between samples of a manifold and that globally, the edges follow the actual structure. If only one erroneous edge is added between two points that should be far apart following the manifold structure, then it provides a short-circuit and completely disrupt the notion of shortest-path. Diffusion distances are, however, much less affected since the amount of energy diffused in one edge is limited, and by consequence they better preserve actual geodesic distances.

In addition, using the KDD to estimate diffusion distances imply the computation of only two graph filtering, i.e. it has a $\mathcal{O}(|E|)$ complexity. Assuming that the graph is sparse it costs $\mathcal{O}(N\log(N))$ less than Dijkstra. An all-pair diffusion distance using the KDD thus costs $\mathcal{O}(N|E|)$ operations, which is still quite costly for large N, but still better than Floyd-Warshall or N iterations of Dijkstra.

Average Cluster Concentration

Now that we walked through all the prerequisites, we can describe the actual method to measure cluster (or class) concentration. The initial idea being to measure the spread of nodes through distances in the embedding, we propose to use active sampling to get a small set of well-distributed samples in a class. Then we need to use those samples to compute the class spread.

Since computing the length of all possible paths is combinatorial in N_s, it is clearly infeasible even for relatively small problems. We propose two other options: the first one is to measure the average of a fixed number of random walks on the samples. The second one is to use the order of the samples to define the path and measure its length. In any case, the path length of such walks can be normalized by the global scale of the embedding.

The first method to compute the Average Cluster Concentration (ACC), using random walks is given in Algorithm 3 and the second method using a unique path in Algorithm 4

Algorithm 3 Average Cluster Concentration (Randomized)

Require: $\mathcal{G}_e$, N_s, N_r, d_e, f_σ
1: Compute the global scale $\sigma = f_\sigma(\mathcal{G}_e)$
2: **for** $c_i \in \mathcal{C}$ **do**
3: Compute N_s samples $\mathcal{S}_{c_i}$ using Algo 2
4: Compute all pairwise distances $D_{\mathcal{S}_{c_i}}$, $d_e(s_i, s_j)$, with $s_i, s_j \in \mathcal{S}_{c_i}$
5: Compute μ_{c_i} the average of N_r random walks on $D_{\mathcal{S}_{c_i}}$
6: $\mathrm{ACC}(c_i) \leftarrow \frac{\mu_{c_i}}{\sigma}$
7: **end for**
8: return $\frac{1}{N_\mathcal{C}} \sum_{c_i \in \mathcal{C}} N_{c_i} \mathrm{ACC}(c_i)$

Let us detail the different parameters in Algorithms 3 and 4: $\mathcal{G}_e$ is the kNN graph on the embedding, N_s is the number of samples, N_r the number of random walks on the samples, d_e the distance function used for the samples and f_σ the function used to compute the global scale.

The distance function and the global scale computation are intrinsically linked and can be computed in three domains :

- the Euclidean space : the distance function d_e is simply the ℓ_2-norm, with the complete set of pairwise distances computed explicitly. The scale f_σ can be computed using the bounding box given by minimum and maximum coordinates :

$$f_\sigma = \| \left[\max(\mathbf{y}_{i,0}) - \min(\mathbf{y}_{i,0}), \ldots, \max(\mathbf{y}_{i,d}) - \min(\mathbf{y}_{i,d}) \right] \|_2. \tag{2.22}$$

- the unweighted embedding graph : the distance function d_e is the shortest path computed with a BFS and the complete set of pairwise distances, needing N_s BFS in total. The scale f_σ is the diameter of the unweighted graph $\mathcal{G}_e$. Note that the unweighted

adjacency matrix $\mathbf{A}$ can be trivially computed from $\mathbf{W}$ using $\mathbf{A} = (\mathbf{W} > 0)$. Since a naive computation of the graph diameter is costly, we use the fast approximate method proposed in [152].

- the weighted graph : we use the KDD for the distance function as a proxy for weighted geodesic distance. Normally, it would cost N_s filtering operations for a complete pairwise distance matrix. However, there is a very good choice for the kernel function which lessens drastically the cost : the heat kernel. First, let us note that powers of heat kernels are still heat kernels, in particular, $g_h^2(\lambda) = \exp(-\tau\lambda)^2 = \exp(-2\tau\lambda) = g_h(2\lambda)$. Second, the active sampling procedure computes the atoms $\left(\mathcal{T}_{s_i} g\right)^2$. Using the fact that $\left(\mathcal{T}_{s_i} g_h\right)^2 = \left(\mathcal{T}_{s_i} g_h^2\right)$, and $g_h^2(\lambda) = g_h(2\lambda)$, we see that we can keep the atoms already computed for the active sampling and reuse them to compute the KDD for no supplementary cost. In addition, $g_h(\mathbf{L})$ is full rank, because g_h is monotonically decreasing and strictly larger than 0. The KDD is thus a metric on $\mathcal{V}$ by Theorem 2. About the global scale, the approach is different than for the other two domains, since the KDD is only bounded above by the energy in the filter g_h. A notion of global scale does not make much sense, but the distances can be normalized by the filter energy : $\|g_h(\lambda)\|_2^2 = \sum_i \|\mathcal{T}_i g\|_2^2$.[1] Again, $\|\mathcal{T}_i g\|_2^2$ can be stored when computed in the active sampling procedure.

Algorithm 4 Average Cluster Concentration (One Shot)

Require: $\mathcal{G}_e$, N_s, N_r, d_e, f_σ
1: Compute the global scale $\sigma = f_\sigma(\mathcal{G}_e)$
2: **for** $c_i \in \mathcal{C}$ **do**
3: Compute N_s samples $\mathcal{S}_{c_i}$ using Algo 2
4: Compute $(N_s - 1)$ pairwise distances in order$\{d_e(s_0, s_1), d_e(s_1, s_2), \ldots, d_e(s_{N_s-2}, s_{N_s-1})$.
5: Compute the length of the sampling path $\mu_{c_i} = \sum_{i=0}^{N_s-1} d_e(s_i, s_{i+1})$.
6: $\mathrm{ACC}(c_i) \leftarrow \frac{\mu_{c_i}}{\sigma}$
7: **end for**
8: return $\frac{1}{N_\mathcal{C}} \sum_{c_i \in \mathcal{C}} N_{c_i} \mathrm{ACC}(c_i)$

As it can be expected, the ACC is very sensitive to the quality of the samples returned by the active sampling procedure. In particular, the sampling can be biased by extreme conditions of the $\mathcal{T}_i g$ norms. Indeed, they are strongly affected by isolated (or disconnected) nodes.

We show a simple illustration of this concept in Figure 2.2 where we compute the ℓ_2-norm of the $\mathcal{T}_i g$ for noiseless random data, and data corrupted with sparse noise. First, we see that the range of values for $\|\mathcal{T}_i g\|_2^2$ is much smaller for noiseless data compared to noisy data. Second, we see that the additive sparse noise creates isolated nodes which have very large values of $\|\mathcal{T}_i g\|_2^2$, one order of magnitude bigger than in the noiseless case.

In the second line, we can see the effect of sparse noise on active sampling. And, although the procedure works as expected in the noiseless case and gives a sampling path which spans well the nodes, all sampled nodes in the noisy case, except the initial sample, are the noisy

[1] $\|g_h(\lambda)\|_2^2 = \sum_\ell |g_h(\lambda_\ell)|^2 = \sum_\ell |g_h(\lambda_\ell)|^2 \sum_i |\mathbf{u}_\ell[i]|^2 = \sum_i \sum_\ell |g_h(\lambda_\ell)\mathbf{u}_\ell[i]|^2 = \sum_i \|\mathcal{T}_i g\|_2^2$

(isolated) ones. The resulting path is thus completely perturbed by the noise and off the core set of points.

In the third line, the regularized sample selection is used with a small value of γ and both sample paths for the noiseless and noisy cases span well the class. In presence of sparse noise, we see that the regularization is superior to the basic sample selection process.

Complexity

In terms of time complexity, the main costs come from the graph creation time and the active sampling. The former is $\mathcal{O}(N\log(N))$ and the latter $\mathcal{O}((K+N_s)m|\mathcal{E}|)$. In addition, the random walk computation costs $\mathcal{O}(N_s^2)$ for Euclidean space, $\mathcal{O}(N_s^2+N_s(N+|\mathcal{E}|))$ for the geodesic space (i.e. N_s BFS) and $\mathcal{O}(N_s^2)$ for the KDD (by using the atoms of the active sampling).

Since N_s, K and m are small constants, and assuming the graph is sparse (i.e. $|\mathcal{E}|=\mathcal{O}(N)$), then the overall complexity is $\mathcal{O}(N\log(N))$ for all cases (i.e. still bounded by the graph creation time). Of course, in practice, the constants will have an impact on the actual computational time of this method.

2.2.3 Average Cluster Noise

The problem of sparse noise contamination that we introduced in Figure 2.2 may be ignored by other measures focusing on clusterability. Indeed, the two first metrics we introduce here, as well as the 1-NN generalization error, provide good proxies towards cluster purity but do not provide good measures of the noise level into account due to averaging effects (and randomization).

Because noisy points are isolated, they have large $\|\mathcal{T}_i g\|_2^2$ values. Motivated by this fact, we can design an outlier detector using simple statistics of the $\mathcal{T}_i g$ ℓ_2-norm since we have a fast estimator for it. In Figure 2.3 we see that outliers have very large $\|\mathcal{T}_i g\|_2^2$ values compared to the bulk of inliers. Using this fact, we can design a simple indicator for outliers :

$$I_g[i]=\begin{cases}1 & \|\mathcal{T}_i g[i]\|_2^2>\mu+\rho\sigma\\ 0 & \text{otherwise}\end{cases} \tag{2.23}$$

with $\mu=\mathbb{E}\left[\|\mathcal{T}_i g[i]\|_2^2\right]$, $\sigma=\sqrt{\mathrm{Var}\left[\|\mathcal{T}_i g[i]\|_2^2\right]}$ and $\rho>0$ a parameter controlling how far from the mean the threshold should be.

In order to apply the outlier detection to classes, the global graph $\mathcal{G}_e$ might not be sufficient. Indeed, if an outlier of some class is moved by noise in a dense region of another class, it will go undetected. In fact, using the outlier indicator on the whole graph can be done in an unsupervised manner, and has the potential to provide interesting information. Nevertheless, since we have access to the labels, we can adapt the technique to make it class-specific.

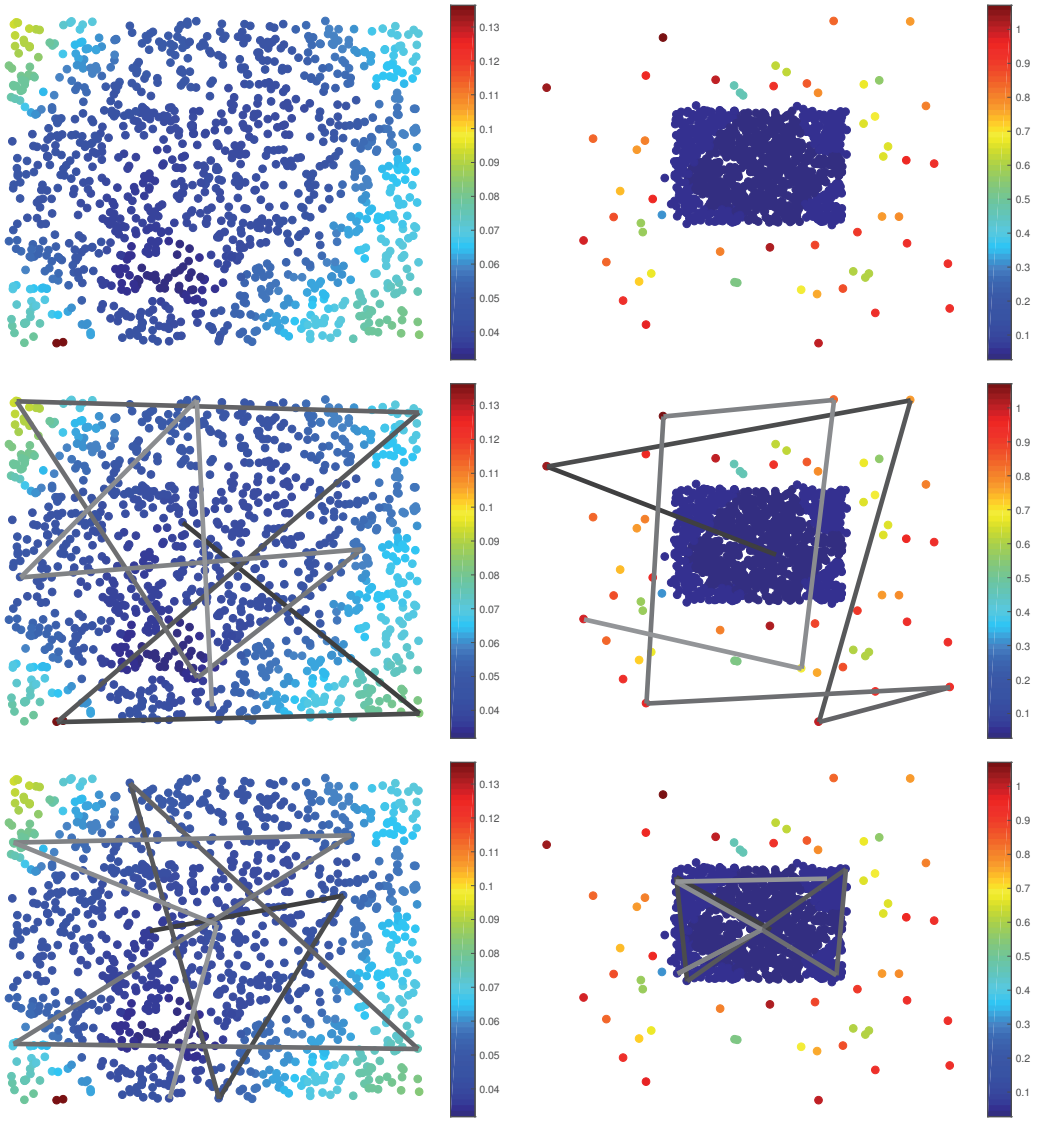

Figure 2.2 – **Effect of sparse noise on active sampling** . In this figure, we show how the presence of isolated nodes influences the $\|\mathcal{T}_i g\|_2^2$ and thus, the active sampling. On the left column, the graph is constructed from 2D points randomly drawn in $[0,1] \times [0,1]$, on the right column, 10% of the points are corrupted with additive white Gaussian noise of variance $\sigma = 1$. The graphs are kNN graphs with $k = 10$ created from the points. The vertices are drawn with dots and, to avoid visual clutter, the edges are not displayed. The colors of the dots is provided by a colourmap of the $\|\mathcal{T}_i g\|_2^2$. The first line displays only the $\|\mathcal{T}_i g\|_2^2$, the second line shows the result of the active sampling procedure with $N_s = 10$. The grey lines link the samples in order from s_0 to s_{N_s-1}. The colour code associated starts from black (s_0) to light grey (s_{N_s-1}). The third line shows the active sampling with $N_s = 10$ using the regularized sample update (Eq. (2.18)) with $\gamma = 0.3$.

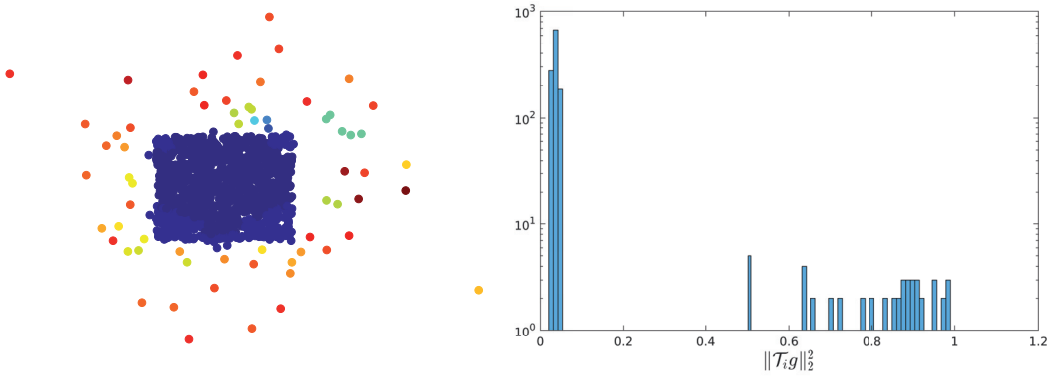

Figure 2.3 – **Statistics of $\mathcal{T}_i g$ ℓ_2-norm.** In this figure, we show how noisy points stand out in $\|\mathcal{T}_i g\|_2^2$ histograms. On the left we display a dataset corrupted with sparse noise where the signal is $\|\mathcal{T}_i g\|_2^2$, with g a heat kernel and the colour map ranging from blue (low values) to red (high values). On the right, the histogram of the signal is shown.

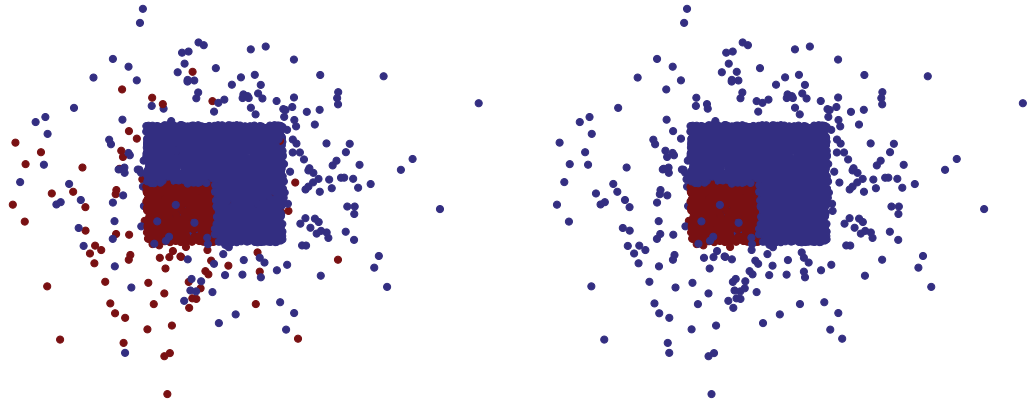

Figure 2.4 – **Subgraph outliers detection**. In this figure we show how the outliers detection using the indicator defined in Eq. (2.23) with $\rho = 0.1$ works on a class subgraph. On the left the selected class is displayed in red and the other points in blue. On the right, only the inliers (i.e. points not reported by the indicator) of the selected class are shown in red.

The easiest way to achieve it is to compute subgraphs for each class and apply the indicator on each one separately. Doing so is likely to produce disconnected subgraphs, but it is not problematic since the different connected components are actually processed independently, and extreme cases such as one-node components will have maximum $\mathcal{T}_i g$ norms. An example of outlier detection applied on a class is provided in Figure 2.4 and we see that it is very effective.

Those ideas are summarized in the measure that we call the Average Cluster Noise (ACN) and compute using Algorithm 5. Since the values are ratios of outliers to the total number of points,

they are bounded between 0 and 1.

Algorithm 5 Average Cluster Noise

Require: $\mathcal{G}_e$, g, ρ
1: **for** $c_i \in \mathcal{C}$ **do**
2: Compute the subgraph $\mathcal{G}_{c_i}$ using $\mathbf{W}(\mathcal{V}_{c_i}, \mathcal{V}_{c_i})$.
3: Compute the norm $\|\mathcal{T}_i g\|_2^2$ of $\mathcal{G}_{c_i}$.
4: Define the set of outliers $\mathcal{R}_{c_i} = \{v_j : I_g[j] = 1, v_j \in \mathcal{V}_{c_i}\}$
5: $\text{ACN}(c_i) \leftarrow |\mathcal{R}_{c_i}|/|\mathcal{V}_{c_i}|$
6: **end for**
7: return $\frac{1}{N_\mathcal{C}} \sum_{c_i \in \mathcal{C}} N_{c_i} \text{ACN}(c_i)$

Complexity

In terms of time complexity, the main costs come from the graph creation time and estimation of the $\mathcal{T}_i g$ norm. The former is $\mathcal{O}(N\log(N))$ and the latter $\mathcal{O}(mK|\mathcal{E}|)$, i.e. K graph filtering operations. Assuming the graph is sparse, we get once again a complexity bounded by $\mathcal{O}(N\log(N))$.

2.3 Experiments

In order to assess the validity of the quantitative measures proposed in this chapter, we first present experiments with controlled synthetic datasets to evaluate our hypotheses. Then, we provide a reference benchmark for all datasets and dimensionality reduction algorithms presented in Appendix B for which we compute all the metrics proposed in this chapter, in addition to traditional ones.

2.3.1 Synthetic datasets

In this section, we use controlled synthetic datasets which exhibit the characteristics we would like to measure. In order to be able to interpret the measures and relate them to their spatial arrangements, we designed the datasets as two-dimensional point clouds with labels. They are dynamic and can be deformed continuously between different conformations by varying a parameter $\lambda \in [0,1]$. In Figure 2.5 all datasets are displayed for different values of λ.

A unique construction principle was used with different topological arrangements (i.e. bands, disc and checkerboard). The idea is that for $\lambda = 0$ the different classes are well separated in clusters, with a greater number of clusters than the number of classes. For $\lambda = 1$ the classes are well separated with a one-to-one mapping between classes and clusters. For intermediate values, the classes are mostly mixed as the points move between $\lambda = 0$ and $\lambda = 1$ conformations. The checkerboard pattern has an intermediate non-mixed conformation at $\lambda = 0.5$.

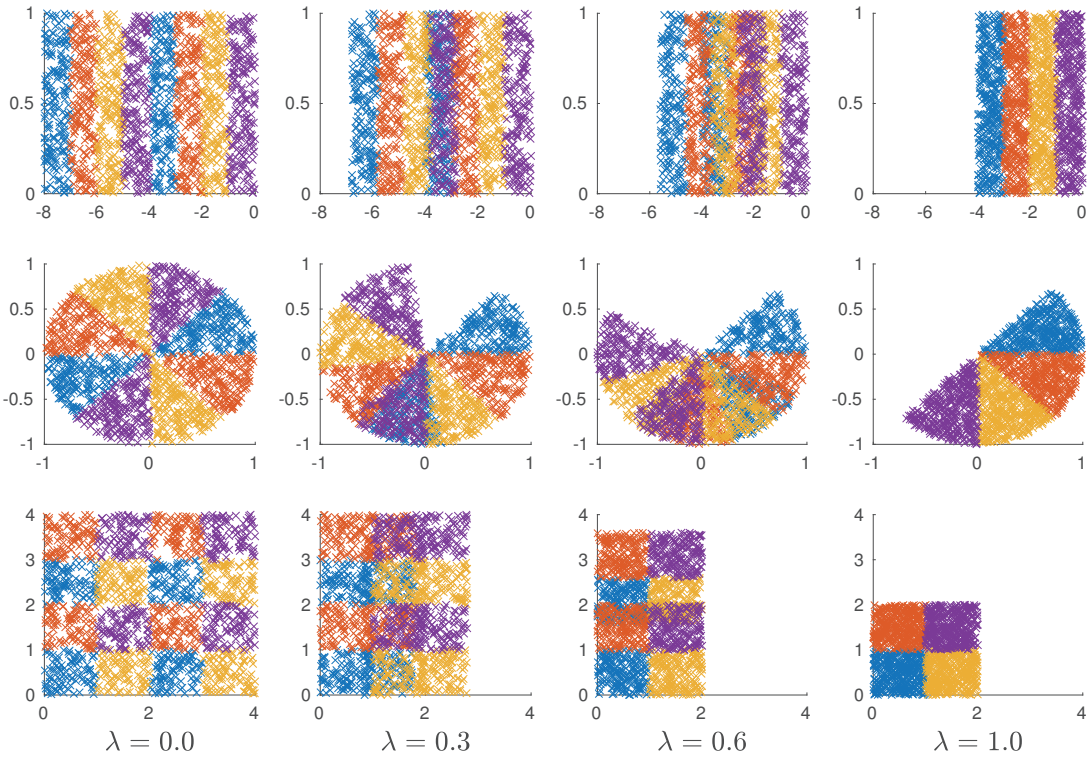

Figure 2.5 – Synthetic datasets with four classes displayed at dynamics $\lambda = 0, 0.3, 0.6, 1$ (one value per column). On the top row, clusters form bands and move horizontally, on the medium row clusters form disc parts and rotate to form a half-disc and finally on the bottom row clusters are small squares in a larger one, move horizontally until $\lambda = 0.5$ and then vertically.

Note that, due to the randomness of the data generation process and the evaluation methods all reported results are averages over multiple realizations.

ACI

In this section, we expect to verify that the ACI detects when classes are well clusterized. The results of the ACI scores computed using either the Cheeger cut or the Normalized cut for the three synthetic datasets, using the full dynamic $\lambda \in [0, 1]$ and for different numbers of classes, is shown in the first two rows of Figure 2.6. The last row is the same experiment with additive Gaussian noise on the coordinates.

As expected both extreme dynamics ($\lambda = 0$ and $\lambda = 1$ for bands and circle, and additionally $\lambda = 0.5$ for checkerboard) display low ACI scores for Cheeger and Normalized cuts, and the intermediate values correspond to the amount of mixing between the classes. In addition, more classes mean a steeper increase of ACI as the classes mix. As a last remark, we can confirm that the ACI is not sufficient to distinguish between split clusters and unified clusters ($\lambda = 0$ and $\lambda = 1$ respectively) which was the main reason for proposing the ACC. The Cheeger

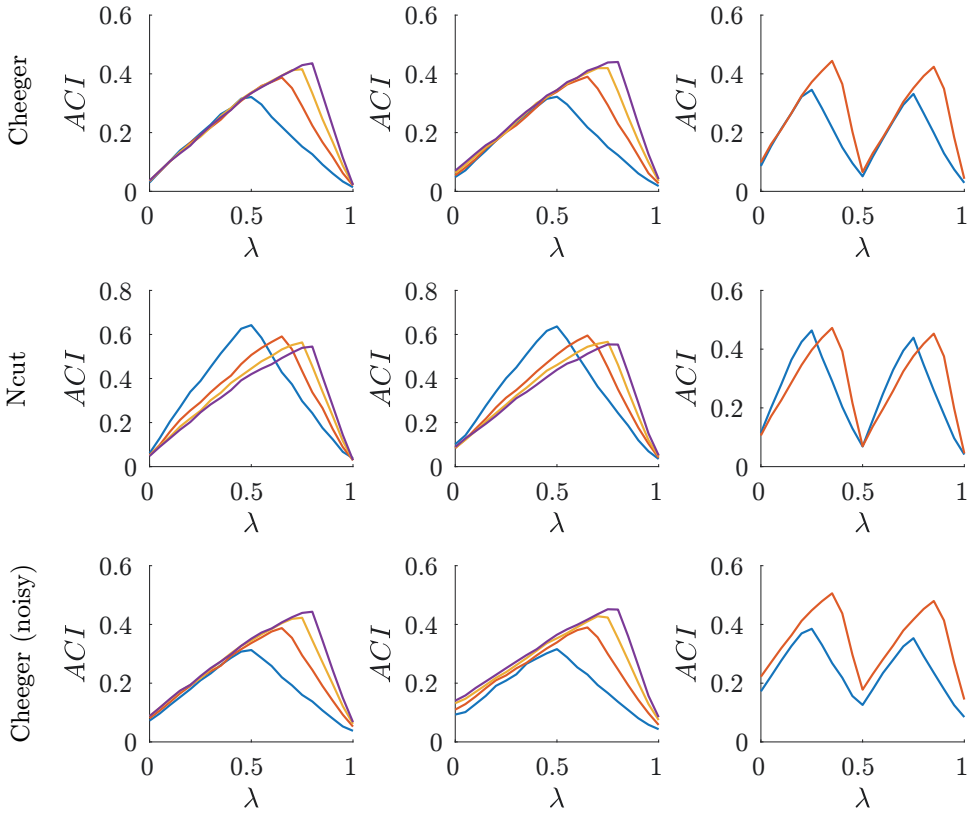

Figure 2.6 – **ACI results on synthetic data**. The synthetic datasets are placed on columns : bands (left column), circle (middle column) and checkerboard (right column). The colours indicate the number of classes, blue = 2, orange = 3, yellow = 4 and purple = 5 for the left and middle column, and blue = 4 and orange = 16 on the right. The first row shows the results of the ACI score with the Cheeger cut, the second row, the results of the ACI score with the Normalized cut and the last row the ACI score with the Cheeger cut on data corrupted with additive white Gaussian noise.

and Normalized cuts display very similar results, except that the Normalized cut maximum values are invariant to the number of classes, which can be an interesting property.

We also see that sparse noise does not influence the ACI score much. Indeed while there are differences between the first and last row of Figure 2.6, they are not striking and the trends are very similar.

Since 1NN misclassification scores are the most frequently reported values for supervised quality evaluation, we provide their performance scores on all artificial datasets in Figure 2.7. First, we see that 1NN scores perform very similarly to the ACI with Cheeger cuts. It provides a good measure of clusterability and is not much affected by noise, but it also does not detect cluster splitting and is not invariant to the number of classes.

All characteristics considered, all three measures provide a good evaluation of clusterability.

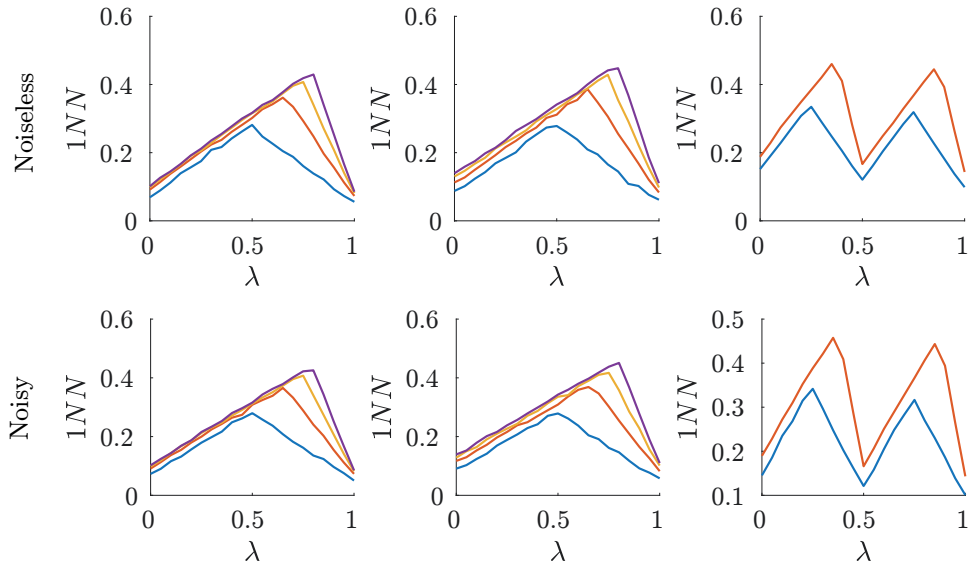

Figure 2.7 – 1**NN classification error on synthetic data.** The columns correspond to the different synthetic datasets and the colours describe the number of classes, similarly to Figure 2.6. The first row reports the 1NN classification error in a noiseless case and the second row in presence of additive white Gaussian noise.

The ACI score with Normalized cut is, however, the only one to be invariant to the number of classes, which can be advantageous when comparing different labelling of a same dataset, for example.

ACC

Now that we showed that clusterability can be faithfully detected using the ACI or 1NN scores, we want to test if the ACC is able to capture the notion of split clusters. Using the same synthetic datasets with varying dynamics as for the ACI experiments, we test how the ACC performs using either Algorithm 3 or 4, and compute the length of the random walks in the three different domains described in Section 2.2.2. The results of the experiments using the Euclidean, geodesic or KDD domains, are shown in Figure 2.8, Figure 2.9 and Figure 2.10 respectively.

The first remarkable observation is that both algorithms produce very similar results, which can be given the following interpretation : the path corresponding to the active sampling order is very representative of the overall scale. Since Algorithm 4 only takes one iteration and provides similar results to the randomized version, it appears to be the best choice.

Looking at Figure 2.8, we see that the path length computed in the Euclidean domain is related to increases in λ for each dataset. Indeed, the ACC score decreases with the class spanning multiple clusters to a single one, which is the effect we wanted to detect. While there are a few

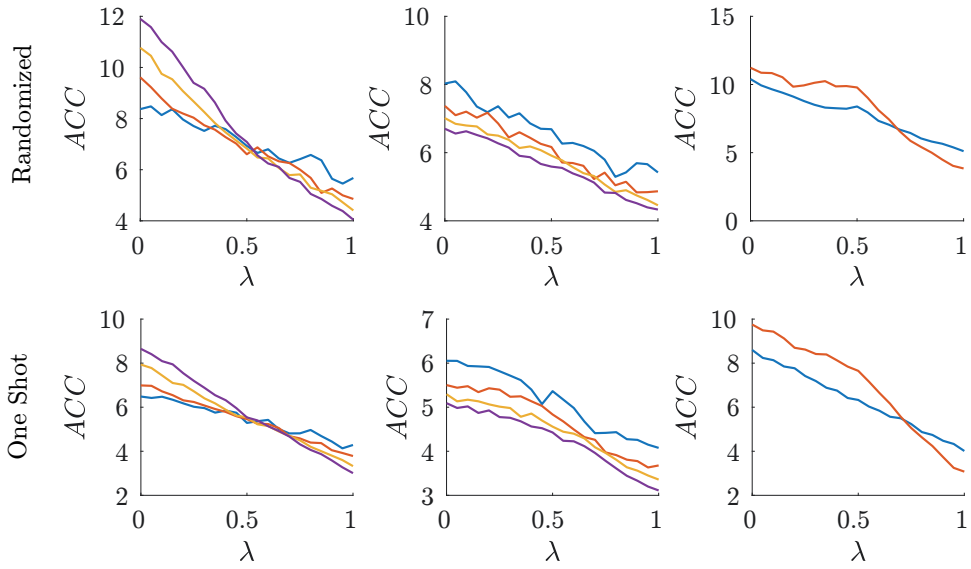

Figure 2.8 – **ACC results on synthetic data (Euclidean).** The columns correspond to the different synthetic datasets and the colours describe the number of classes, similarly to Figure 2.6. The first row reports the ACC score computed with Algorithm 3 ($\gamma = 0.3$, $N_s = 10$ and $N_r = 10$) and the second row the ACC score computed with Algorithm 4 ($\gamma = 0.3$ and $N_s = 10$), both in Euclidean space.

differences related to the number of classes and the topological structure, the results show clearly that the ACC allows us to discriminate between split and concentrated clusters.

In Figure 2.9 we see that the using the unweighted geodesic space to compute path length gives worse results than for the Euclidean case. Indeed, although the ACC is globally correlated to λ, the values are noisy and the correlation is not very strong. The most plausible explanation for these inferior results is that unweighted geodesic path lengths miss the actual embedding scale by only taking into account the number of hops.

Next, we report the results for the ACC with the KDD in Figure 2.10. Compared to the two other measures, we see that the topology has a strong impact, since the behaviour is different between the three datasets. For the bands, the reported ACC decreases with the value of λ, as we expect, although the effect is much stronger with an increasing number of classes. The circle dataset is more problematic since for very low values of λ the score is not related to the cluster splitting, in fact, both extreme conformations ($\lambda = 0$ and $\lambda = 1$) give similar scores. Outside this range, the trend is similar to the band dataset. This bad performance could be explained by the fact that the points of the disc parts are always very close, creating a path between the two clusters. The checkerboard dataset is also partially problematic since the score does not provide much information for the smallest number of classes. Finally, similarly to the other two ACC distances, both Algorithm 3 and 4 produce very similar results.

The first three experiments being noiseless we want to see if the regularization of the active

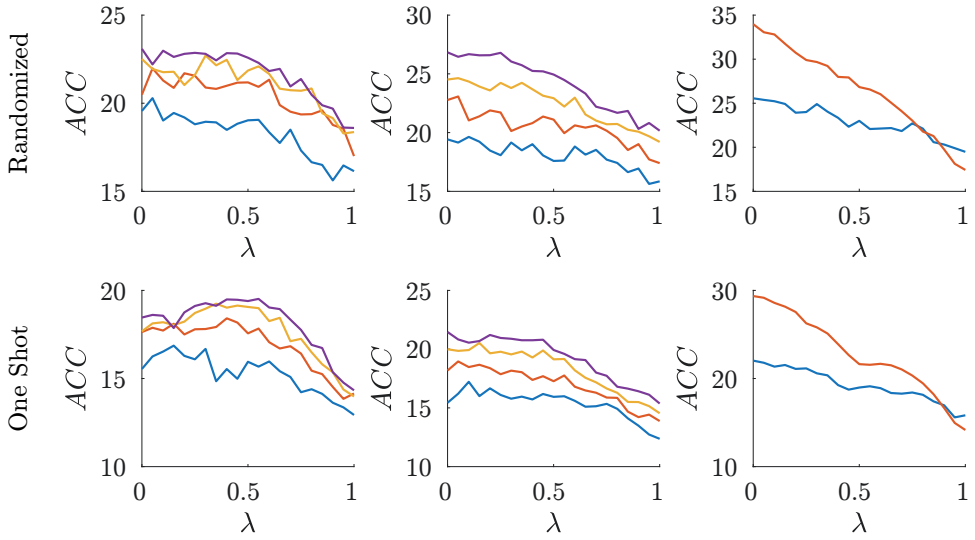

Figure 2.9 – **ACC results on synthetic data (geodesic).** The columns correspond to the different synthetic datasets and the colours describe the number of classes, similarly to Figure 2.6.The first row reports the ACC score computed with Algorithm 3 ($\gamma = 0.3$, $N_s = 10$ and $N_r = 10$) and the second row the ACC score computed with Algorithm 4 ($\gamma = 0.3$ and $N_s = 10$), both in geodesic space.

sampling is sufficient to avoid the problems that can be caused by sparse noise (as shown in Figure 2.2). In Figure 2.11 we reproduce the experiment of Figure 2.8 with the data corrupted by additive white Gaussian noise. We see that the trends in both figures are nearly identical, which shows the success of the regularization procedure to provide invariance to noise.

Although it might be surprising that the path length in Euclidean space performs well, we must say that the experiment design was slightly biased in its favour. Indeed, in those settings, since the data is well concentrated in convex hulls, the Euclidean distance is very similar to the weighted geodesic distance. In more complex settings, computing the path length using the KDD might better reveal cluster splitting.

ACN

To evaluate the performance of the ACN in detecting sparse noise levels, we must change the experiment design compared to the two previous scores. Indeed, the conformation dynamic is not relevant in this case and it would be better to see the behaviour with varying noise levels and percentage of corrupted points with a fixed λ.

In Figure 2.12 we show the ACN scores for different noise settings for all synthetic datasets. In the first row, we see that the ACN quickly rises from 0 to a higher level and then stays constant. This behaviour is close to the perfect case since the reported ACN is strongly correlated to the noise percentage for increasing noise levels. Indeed, it converges quite precisely to the actual

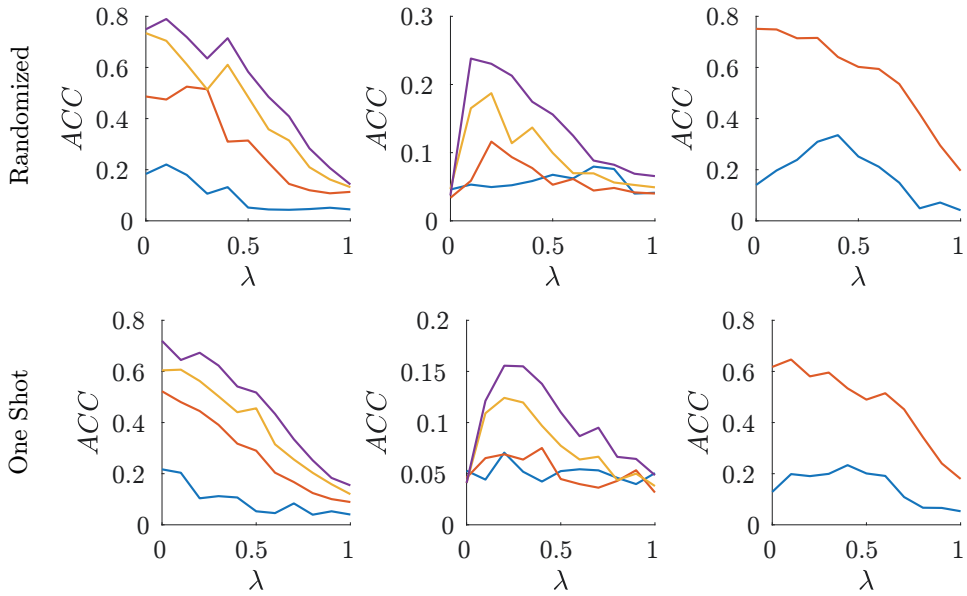

Figure 2.10 – **ACC results on synthetic data** (KDD)**.** The columns correspond to the different synthetic datasets and the colours describe the number of classes, similarly to Figure 2.6. The first row reports the ACC score computed with Algorithm 3 ($\gamma = 0.3$, $N_s = 10$ and $N_r = 10$) and the second row the ACC score computed with Algorithm 4 ($\gamma = 0.3$ and $N_s = 10$), both in KDD space.

noise percentages, also in the noiseless case. The only imprecision is for values above 30% of corrupted points where saturation effects begin to occur. This effect can be expected since, in any case, the maximum detectable percentage of noise is 50%. Indeed, above this threshold, the noise becomes predominant and the actual data may be statistically considered to be the noise. We can also note that the higher the noise level the more precise the detected noise percentage.

In the second row of Figure 2.12 we see that the ACN increases with the percentage of noise. The correlation is almost linear, and the higher the noise level, the closer to a perfect correlation between the reported ACN and the actual noise percentage. Both results are very positive as they show that the ACN measures the noise percentage very accurately.

In Figure 2.13 we report the computational time needed to compute the different scores we presented for exponentially increasing datasets sizes. Since all methods have an asymptotic time complexity bounded by the knn-search time $\mathcal{O}(N\log(N))$, we expect the trends to be similar, which is verified experimentally. Indeed, all methods have a similar profile, and are separated by a constant difference (except for very small values of N). Unsurprisingly, the $1NN$ has the lowest cost (it has no overhead with respect to the NN search) and the ACC the highest cost (due to its higher overhead operations). Both ACI and ACN have very close values between the two extremes. Practically, at $\mathcal{O}(10^3)$ seconds for $10^6 < N < 10^7$, even the most costly score evaluation remains reasonable.

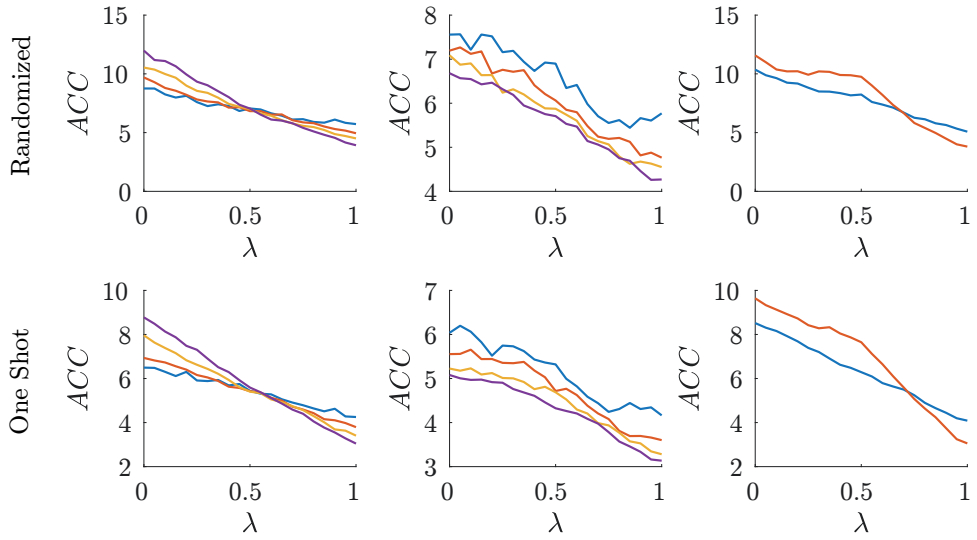

Figure 2.11 – **ACC results on noisy synthetic data (Euclidean)** The columns correspond to the different synthetic datasets and the colours describe the number of classes, similarly to Figure 2.6. The first row reports the ACC score computed with Algorithm 3 ($\gamma = 0.3$, $N_s = 10$ and $N_r = 10$) and the second row the ACC score computed with Algorithm 4 ($\gamma = 0.3$ and $N_s = 10$), both in KDD space.

2.3.2 Natural datasets

In this section, we compute quality metrics for different dimensionality reduction algorithms applied to natural, real-world, datasets. This allows us to evaluate both the pertinence of the formal measures we have established as well as the actual performance of different dimensionality reduction algorithms. In order to assess the pertinence of the quantitative scores, we cast the dimensionality reduction as a visualization task, i.e. the target dimension is set to 2.

Here we consider the same quality measures as before : ACI (with Ncut), 1NN, ACC (with Algo 4 in Euclidean space) and ACN. We also provide a simple unsupervised metric to ensure that all the supervised methods do relate to the actual quality of the embedding in general and are not biased by very imbalanced labels distributions. To do so, we chose to report the Nearest Neighbours Precision (NNP), which is the average number of common neighbours preserved between the original space and the embedding. In contrary to rank-based methods, NNP is an unsupervised metric which is very efficient to compute.

The evaluated embeddings are produced using the following methods : PCA, Laplacian Eigenmaps, Local Linear Embedding, Barnes-Hut t-SNE and LargeVis. This subset of algorithms was chosen mainly using a scaleability criterion. In addition, t-SNE and LargeVis are currently considered the best algorithms for visualization. The datasets we use come from various sources and encompass many different types of data. More details on the algorithms chosen and the datasets are given in Appendix B.

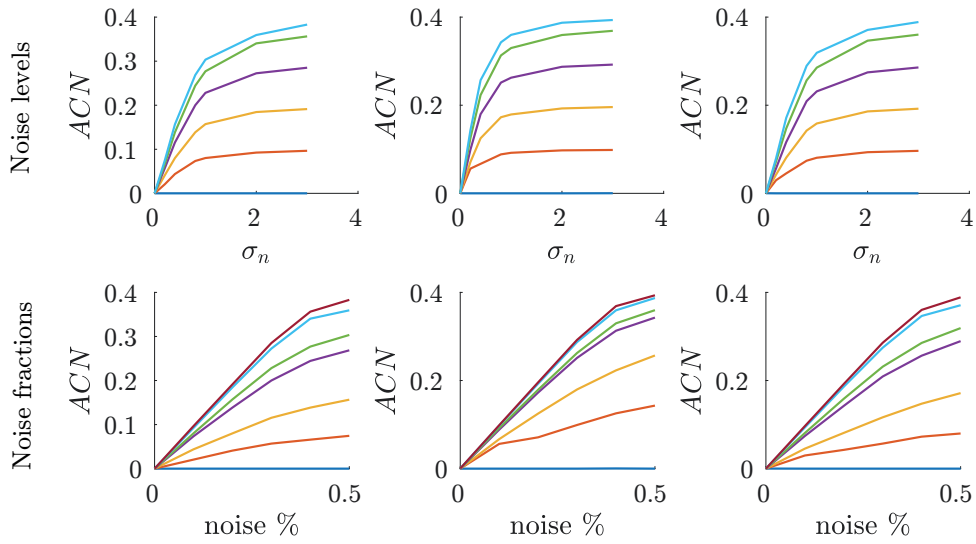

Figure 2.12 – **ACN results on noisy synthetic data** The columns correspond to the different synthetic datasets similarly to Figure 2.6. In the first row, the ACN is measured for varying noise levels (σ_n) and the colours correspond to different noise percentages $[0, 0.1, 0.2, 0.3, 0.4, 0.5]$. In the second row, the ACN is measured for varying noise percentages and the colours correspond to different noise levels.

We present the results for all metrics on the natural datasets in Tables 2.1-2.6, and highlight a few selected datasets in Figures 2.14-2.16. In all tables, the values are given for different algorithms and for different datasets (corresponding to the columns and lines of the tables, respectively). Green indicates the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination statuses are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

First, let us examine the CPU time reported in Table 2.1 for the different algorithms considered. We see that the fastest method is PCA, except for the two sparse datasets, since it scales with D and not N. Then Laplacian Eigenmaps performs second overall. LargeVis scores third, but exhibits a different pattern than the other algorithms. Indeed, it has a very large base cost and then scales well with the input size. This effect can be observed by the fact that it performs second best for most large datasets and worst for smaller scale ones. Then LLE and t-SNE have the largest overall execution times. In addition, we see that LLE and Laplacian Eigenmaps did not terminate for all datasets due to lack of memory, time-outs or numerical stability issues (RE). All other methods were robust and did terminate on all datasets.

Moving on to the clusterability evaluation reported by ACI and the generalization error of 1NN in Tables 2.2 and 2.3, we confirm that the two measures strongly agree for all methods and datasets considered. In addition, they show that t-SNE and LargeVis both consistently outperform the other algorithms considered with very few exceptions. We also see in Table 2.4 that the NNP reports very similar trends to both supervised measures, which confirms that

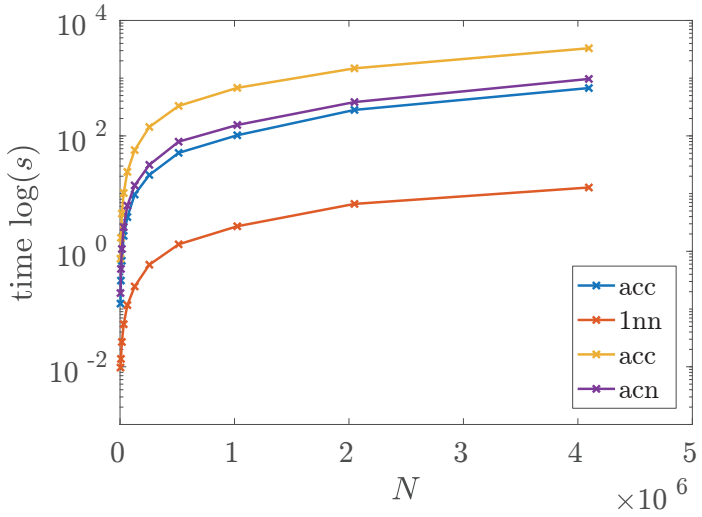

Figure 2.13 – **Time complexity** Total CPU time (in log(s)) for the ACI, 1NN, ACC and ACN scores for increasing values of dataset sizes N. The datasets consist of N randomly drawn 2-dimensional points.

the labels do not introduce misleading biases.

The noise measure given by the ACN is reported in Table 2.5. The values reported show that this measure is very consistent with the global quality indices reported by ACI and 1NN scores giving t-SNE and LargeVis as the algorithms with lowest sparse noise levels.

Finally, the ACC score is reported in Table 2.6. It is by far the most challenging measure to interpret since it does not concur with the other ones. Overall, LLE and Laplacian Eigenmaps are the two methods with lower ACC values and PCA the highest. A plausible explanation for this behaviour is that the ACC is focused on detecting cluster split, which can be completely unrelated to clusterability or crowding phenomena. For example, it is easy to imagine that it could be biased towards very packed data. Indeed, a degenerated solution with each class concentrated around one point should yield very low ACC scores. In addition, low-density, but uniformly distributed class embeddings of points may also produce low ACC scores provided they are not all overlapping.

The complexity of interpretation of the ACC makes it unsuitable to be a first-level indicator for quality. However, it can be a very interesting secondary level, finer measure. Indeed, it allows to discriminate between methods already considered good by other quality measures (e.g. ACI or 1NN). As a matter of fact, a combination of ACI and ACC for the full datasets shows LargeVis as slightly superior to t-SNE, quite consistently.

Finally, let us validate the analysis done so far with our quality indicators using visual inspection. We show the resulting 2D embedding of the MNIST dataset in Figure 2.14. First, we

Time [s]	pca	lle	le	tsne	largevis
caltech101-caffenet	**11.08**	16.54	**6.85**	**89.52**	**219.00**
caltech256-caffenet	**17.47**	**391.18**	**186.26**	**469.09**	256.87
cifar10-cnn	**1.54**	**3227.07**	**OM**	1410.91	**315.21**
cmupie	**0.70**	29.66	**14.83**	**130.22**	**219.71**
coil20	**1.26**	2.03	**0.42**	**10.82**	**216.03**
fma-echonest	**0.29**	60.54	**15.42**	**139.97**	**220.09**
fma-rosa	**1.13**	**TO**	1882.70	**2475.82**	**413.09**
hiva	**150.24**	**RE**	703.64	**949.41**	**279.98**
mnist	**0.74**	**2851.26**	**3448.48**	1271.24	**336.87**
norb	**92.06**	**247.61**	**296.17**	**884.97**	291.14
sylva	**0.32**	**OM**	**OM**	3336.83	**490.75**
usps	**0.18**	20.36	**9.58**	**94.30**	**215.55**
zebra	**0.20**	230.83	**TO**	**141.54**	**422.45**
20newsgroup	**23265.48**	**103.97**	**107.86**	**303.95**	236.83
nova	**9681.71**	**185.94**	**100.53**	**310.05**	233.49

Table 2.1 – **Embedding time** This table reports the embedding time (in seconds) for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

can immediately see that PCA provides the poorest separation, t-SNE and LargeVis display actual clusters, and Laplacian Eigenmaps and LLE show intermediate results which separate the classes with some overlap. This observation is well validated by looking at both the ACI and 1NN scores with the worst value achieved by PCA, LE and LLE being twice better and t-SNE and LargeVis one order of magnitude better. The sparse noise appears visually to be the highest in PCA, medium in LE and LLE and the lowest in t-SNE and LargeVis. This observation concurs perfectly with the ACN measure. In terms of ACC, while it clearly set PCA as the worst score, we need to consider it as a secondary differentiator for the remaining datasets. Both LE and LLE actually provide a perfect mapping between the number of classes and the number of clusters (i.e. continuous set of points) which translate as the lowest ACC measures. LargeVis and t-SNE do not achieve a perfect mapping, since there are 11 clusters for t-SNE and 12 for LargeVis for 10 classes. This effect is reported as middle ACC values for both embeddings. Other visualization are shown in Figures 2.15 and 2.16 which support this analysis.

2.4 Conclusion

In this chapter, we have introduced new supervised methods to quantify the quality of embeddings and, by extension, of dimensionality reduction algorithms. Their common core is the use of graphs to grasp the structure of the embedded points. Our three measures attempt to widen the characteristics measured by usual quality measures while keeping a good theoretical intuition. We have seen, both in synthetic controlled experiments and for natural datasets, that theses measures are meaningful. They will thus be used in the remaining chapters for comparative quality assessments, alongside traditional measures.

As we saw in the experiments, ACI and 1NN exhibit very similar trends, so they will be used interchangeably and form our principal quality criterion. The measure of the ACN generally concur with the first two, but can be used as a differentiator. Finally, the ACC is the most difficult metric to interpret, since, for example, it can report excellent values for quite bad embeddings, provided the mapping between the number of classes and the number of clusters is preserved.

The conclusion of both visual inspection and formal quality evaluation confirm t-SNE and LargeVis as the best dimensionality reduction algorithms very consistently. They are, however, the two slowest methods compared to the more traditional ones. The ideal goal would thus be to achieve similarly good embeddings with a reduced execution time. The following chapters will discuss different strategies to get closer to this goal.

The work presented here could be improved in different directions. First, one characteristic is not specifically addressed : class overlap. While it is indirectly reported in clusterability and generalization error metrics, overlaps can occur in various ways. A measure related to the amount and type of overlapping classes would be useful to give better insights when comparing embeddings. More specifically related to our contributions, we think that a formal theoretical analysis of our active sampling algorithm would be very valuable and leave it as future work. In addition, we used the KDD with only one kernel for optimization reasons, but its use with different kernels should be investigated, and it could prove to be a meaningful distance in other contexts.

ACI - Full	pca	lle	le	tsne	largevis
caltech101-caffenet	**0.73**	**0.61**	0.58	**0.14**	**0.13**
caltech256-caffenet	**0.95**	**0.91**	0.86	**0.39**	**0.44**
cifar10-cnn	**0.79**	**0.58**	OM	**0.28**	**0.27**
cmupie	**0.96**	**0.94**	0.94	**0.42**	**0.65**
coil20	**0.06**	**0.12**	**0.12**	0.06	**0.05**
fma-echonest	**0.94**	**0.94**	0.91	**0.87**	**0.88**
fma-rosa	**0.97**	TO	**0.96**	**0.92**	**0.93**
hiva	**0.90**	RE	**0.77**	**0.62**	**0.65**
mnist	**0.72**	0.22	**0.37**	**0.05**	**0.06**
norb	**0.97**	**0.96**	0.94	**0.52**	**0.60**
sylva	**0.99**	OM	OM	**0.35**	**0.33**
usps	**0.60**	**0.28**	0.21	**0.03**	**0.04**
20newsgroup	**0.95**	**0.86**	0.79	**0.25**	**0.26**
nova	0.16	**0.35**	**0.27**	**0.07**	**0.06**

Table 2.2 – **Embedding ACI.** This table reports the ACI score for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

1*NN* - Full	pca	lle	le	tsne	largevis
caltech101-caffenet	**0.71**	0.57	**0.57**	**0.18**	**0.19**
caltech256-caffenet	**0.94**	**0.90**	0.85	**0.47**	**0.53**
cifar10-cnn	**0.68**	**0.51**	OM	**0.28**	**0.29**
cmupie	**0.93**	0.85	**0.90**	**0.26**	**0.55**
coil20	0.08	**0.08**	**0.12**	**0.05**	**0.04**
fma-echonest	**0.75**	**0.75**	0.74	**0.70**	**0.71**
fma-rosa	**0.80**	TO	**0.80**	**0.77**	**0.79**
hiva	**0.06**	RE	**0.05**	**0.04**	**0.05**
mnist	**0.62**	0.24	**0.35**	**0.05**	**0.06**
norb	**0.82**	**0.80**	0.74	**0.38**	**0.46**
sylva	**0.11**	OM	OM	**0.05**	**0.04**
usps	**0.52**	**0.29**	0.23	**0.04**	**0.05**
20newsgroup	**0.87**	**0.80**	0.73	**0.21**	**0.24**
nova	**0.16**	**0.14**	0.07	**0.02**	**0.02**

Table 2.3 – **Embedding 1NN.** This table reports the 1NN score for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

NN Precision	pca	lle	le	tsne	largevis
caltech101-caffenet	**0.15**	**0.19**	0.21	**0.35**	**0.36**
caltech256-caffenet	**0.04**	**0.06**	0.09	**0.22**	**0.22**
cifar10-cnn	**0.03**	**0.04**	OM	**0.15**	**0.14**
cmupie	**0.25**	**0.11**	0.29	**0.39**	**0.40**
coil20	0.28	**0.22**	**0.28**	**0.29**	**0.27**
fma-echonest	0.28	**0.15**	**0.14**	**0.38**	**0.38**
fma-rosa	**0.15**	TO	**0.15**	**0.32**	**0.28**
hiva	**0.06**	RE	**0.16**	**0.41**	**0.42**
mnist	**0.05**	0.13	**0.11**	**0.32**	**0.29**
norb	**0.10**	**0.10**	0.13	**0.33**	**0.30**
sylva	**0.02**	OM	OM	**0.11**	**0.11**
usps	**0.21**	**0.22**	0.28	**0.49**	**0.49**
20newsgroup	**0.02**	**0.04**	0.07	**0.25**	**0.26**
nova	**0.04**	**0.06**	0.08	**0.20**	**0.21**

Table 2.4 – **Embedding NNP.** This table reports the NNP score for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

ACN - Full	pca	lle	le	tsne	largevis
caltech101-caffenet	**0.43**	0.35	**0.37**	**0.20**	**0.21**
caltech256-caffenet	**0.51**	0.49	**0.51**	**0.36**	**0.37**
cifar10-cnn	**0.35**	**0.30**	OM	**0.24**	**0.23**
cmupie	**0.54**	0.50	**0.51**	**0.36**	**0.36**
coil20	**0.10**	0.09	**0.10**	**0.04**	**0.02**
fma-echonest	**0.38**	**0.36**	**0.38**	**0.37**	0.38
fma-rosa	**0.42**	TO	**0.43**	**0.41**	**0.41**
hiva	**0.18**	RE	**0.20**	**0.31**	**0.26**
mnist	**0.32**	0.20	**0.24**	**0.07**	**0.06**
norb	**0.42**	0.36	**0.37**	**0.34**	**0.30**
sylva	**0.03**	OM	OM	**0.02**	**0.02**
usps	**0.28**	**0.21**	0.21	**0.06**	**0.06**
20newsgroup	OK	**0.43**	**0.40**	**0.30**	**0.26**
nova	OK	**0.11**	**0.15**	**0.32**	**0.06**

Table 2.5 – **Embedding ACN.** This table reports the ACN score for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

ACC - Full	pca	lle	le	tsne	largevis
caltech101-caffenet	**10.56**	**0.10**	**0.04**	**4.61**	2.87
caltech256-caffenet	**18.12**	**0.01**	**0.11**	**6.80**	4.72
cifar10-cnn	**32.14**	**0.34**	OM	**12.28**	**9.89**
cmupie	**252.20**	**0.85**	**0.64**	**30.77**	24.93
coil20	**36.54**	**0.21**	**0.05**	3.97	**14.50**
fma-echonest	**400.37**	**0.64**	**0.77**	**34.97**	26.62
fma-rosa	**181.50**	TO	**0.15**	**23.55**	**22.39**
hiva	**10.17**	RE	**0.31**	**30.29**	**25.94**
mnist	**109.82**	**0.14**	**0.13**	**13.08**	10.47
norb	**376.63**	**0.42**	**0.21**	26.93	**34.66**
sylva	**129.88**	OM	OM	**21.08**	**28.99**
usps	9.02	**0.29**	**0.13**	**13.67**	**10.75**
20newsgroup	**96.34**	**0.01**	**0.24**	**15.15**	8.77
nova	**160.20**	**0.41**	**0.29**	**29.75**	19.75

Table 2.6 – **Embedding ACC.** This table reports the ACC score for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

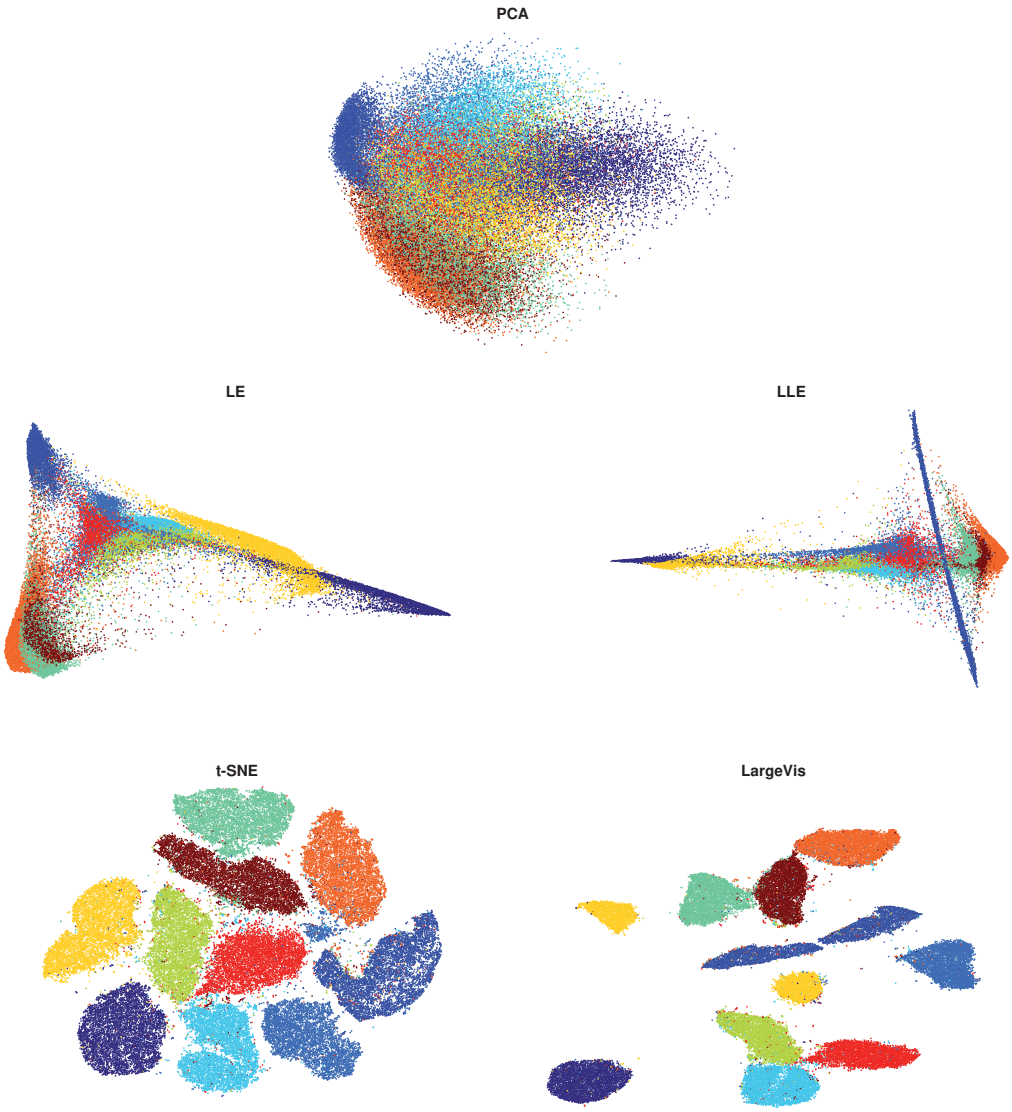

Figure 2.14 – Visualizations of the full MNIST dataset using different dimensionality reduction algorithms : PCA, Laplacian Eigenmaps (LE), LLE, t-SNE and LargeVis. The colour map corresponds to the labels.

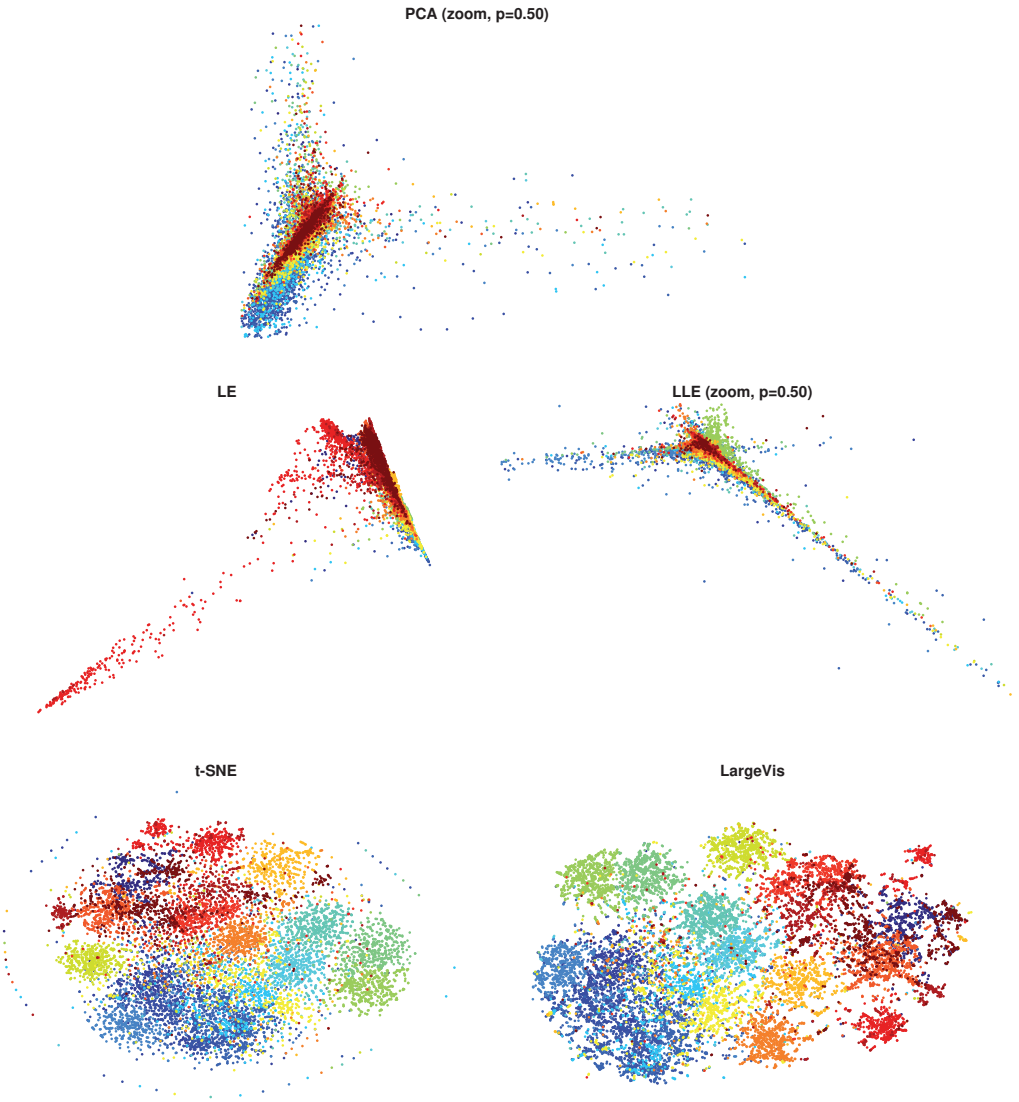

Figure 2.15 – Visualizations of the full 20Newsgroup dataset using different dimensionality reduction algorithms : PCA, Laplacian Eigenmaps (LE), LLE, t-SNE and LargeVis. The colour map corresponds to the labels. Zoomed-in versions are done by excluding points whose coordinates are in percentiles p and $(100 - p)$ of the data distribution.

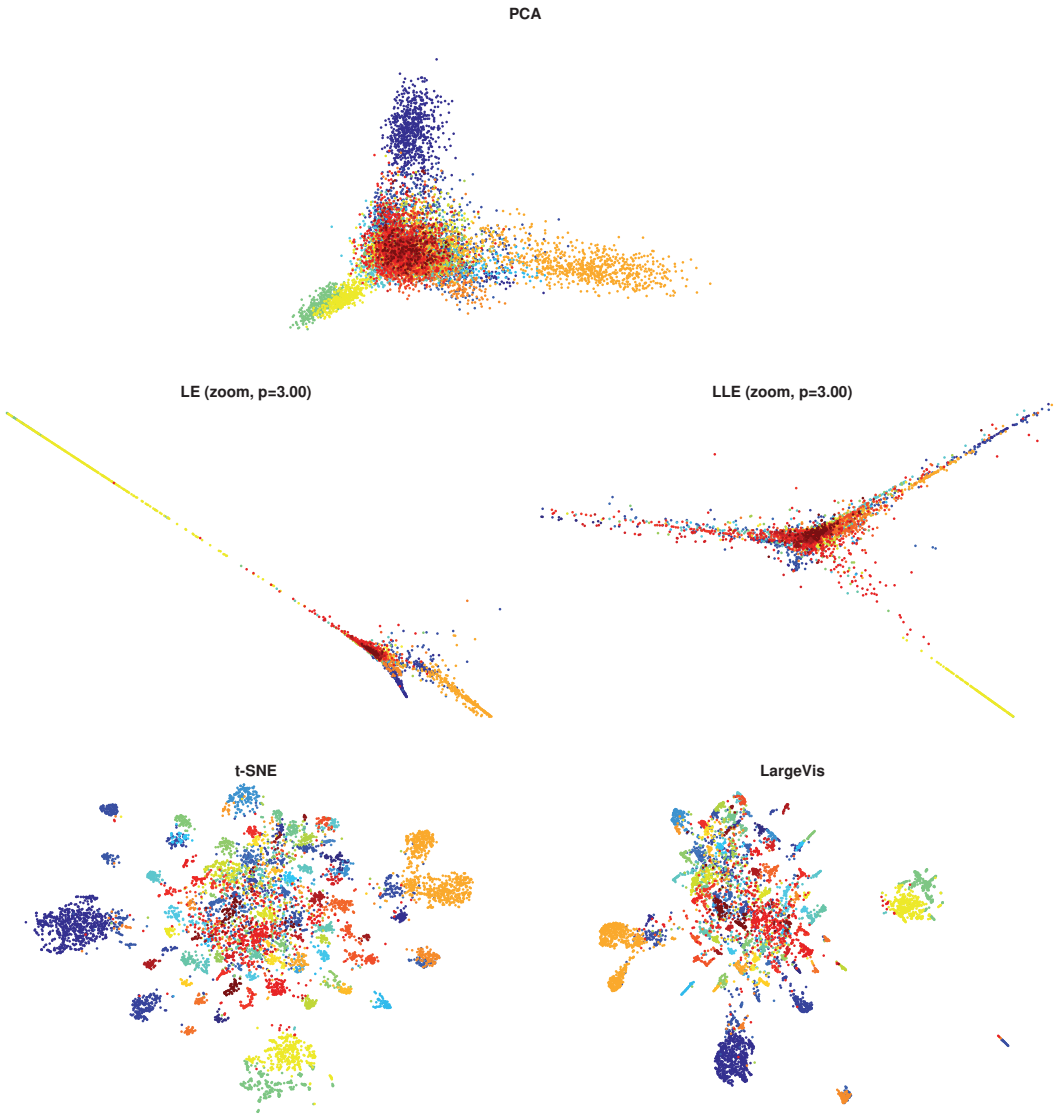

Figure 2.16 – Visualizations of the full Caltech101 dataset using different dimensionality reduction algorithms : PCA, Laplacian Eigenmaps (LE), LLE, t-SNE and LargeVis. The colour map corresponds to the labels. Zoomed-in versions are done by excluding points whose coordinates are in percentiles p and $(100-p)$ of the data distribution.

Nature uses only the longest threads to weave her patterns,
so each small piece of her fabric reveals the organization of the entire tapestry.
— Richard Feynman

3 Eigenspace Estimation

In this chapter we start to address the scaleability issue intrinsically related to dimensionality reduction by tackling a standard problem : the estimation of the first k eigenvectors of any graph Laplacian. We propose a method providing a good approximation by filtering Gaussian random signals on the graph. We prove that we only need k such signals to be able to recover as many of the eigenvectors corresponding to the smallest eigenvalues, regardless of the number of nodes in the graph. In addition, we address key issues in implementing the theoretical concepts in practice using accurate approximated methods.

As the main contribution, we present a new algorithm for Fast Eigenspace Approximation using Random Signals (FEARS) to estimate the first k eigenvectors of the graph Laplacian using random signal filtering techniques. Contrary to previous works (e.g. [29, 192]) that only focus on distance preservation, we are able to obtain the partial eigenspace, with an inferior total complexity.

We also present experiments which show the validity of our method in practice and compare it to state-of-the-art techniques for clustering and visualization both on synthetic small-scale datasets and larger real-world problems of millions of nodes. We show that our method allows for a better scaling than all previous methods, while achieving an almost perfect reconstruction of the eigenspace formed by the first k eigenvectors.

The underlying motivation for this work is to stress that, although the questions related to data analytics such as clustering or visualization have received a lot of attention in the past decades, the size of the datasets usually considered nowadays requires us to review old techniques with modern tools.

This chapter is organized as follows. First, we review the related work on the subject in Section 3.1. Then, Section 3.2 develops the main results of this chapter from the theoretical point of view while Section 3.3 presents its applied counterpart and the algorithms for fast spectral embedding and eigencount estimation. Next, in Section 3.4, we show the validity and

benefits of our method and compare it with the state of the art through several experiments. Finally, Section 3.5 discusses open problems in the domain as well as potential future work to address.

This chapter core content has been presented in a paper written in collaboration with Lionel Martin [137]. The additional content is mainly in the experiments section on natural datasets.

3.1 Related work

As we saw in our review of dimensionality reduction techniques in Chapter 1, eigendecomposition has been at the core of many famous techniques. It is commonly used for partitioning (e.g., spectral clustering [135, 164]), data visualization (e.g., Laplacian eigenmaps [17]), but also simply as a dimensionality reduction technique for preprocessing (e.g., PCA). The main drawback of all the aforementioned techniques is that they do not to scale well as they have a rather high complexity (e.g., a partial eigendecomposition being $\mathcal{O}(kN^2)$), with k the number of eigenvectors to be recovered.

The usual way to recover the eigenspace of a symmetric matrix $\mathbf{L}$ is to diagonalize it as $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^*$, and take the first k columns of $\mathbf{U}$. The diagonalization is typically done using a singular value decomposition (SVD), which is $\mathcal{O}(N^3)$ for a symmetric matrix of size N. Since this is intractable even for medium scale N, a great deal of work has been done on faster ways to compute eigenvalues and eigenvectors of $\mathbf{L}$ efficiently (see [11] for a review). The fastest methods are variants of Arnoldi or Lanczos iteration methods [6, 104] such as Implicitly Restarted Arnoldi Method (IRAM) [177] or Implicitly Restarted Lanczos Method (IRLM) [35]. The preferred method for graph Laplacians is the IRLM since the matrix is symmetric and generally assumed to be sparse. The IRLM has a worst case complexity of $\mathcal{O}(h(|\mathcal{E}|k+k^2N+k^3))$, with h the number of iterations to reach convergence and assuming there are $\mathcal{O}(k)$ extra Lanczos steps [11]. If we consider sparse graphs with $|\mathcal{E}| = \mathcal{O}(N)$ and a fixed k independent of the value of N, the complexity of the IRLM is bounded by the term $\mathcal{O}(k^2N)$.

Since the exact computation of the eigenspace proves to be expensive, several angles were considered to approximate the result. Physicists came with a solution to the problem of eigenspace determination using contour integration techniques for the reduction of the matrices on which to apply the eigendecomposition [146], that allows improving the complexity with almost no loss of precision. Meanwhile, with the additional constraint that the matrix should contain a subset of the columns of the original matrix, Boutsidis et al. [28] propose a fast method to approximate low-rank matrix reconstruction (whose optimal solution is the eigenspace generated by the first eigenvectors). Some works, such as [120], focus on the determination of the first non-trivial eigenvector only. Finally, Bai [10] proposes a solution for the approximation of eigenvectors using tridiagonalization of sparse matrices that requires "efficiently" sparse matrices as input. Although this might not necessarily apply in practice depending on the data set at hand, it proved to be efficient in various problems involving modelling physical phenomena with strong locality properties.

Instead of computing the eigensubspace, finding features which provide similar pair-wise distances can be considered sufficient depending on the application. Indeed, for tasks such as clustering, supposing an algorithm such as k-means is performed as the final assignment step, the preprocessing for dimensionality reduction only requires pairwise distances between points to be preserved in the new space. In this mind, [151] presents a clustering algorithm that avoids the computation of an SVD by computing polynomial approximations and using the Johnson-Linderstrauss lemma.

On the same line, the authors of [29] show that the power method (computing powers of the normalized weight matrix) gives a good approximation of the eigenvectors for distance preservation. They give a bound on the power required to obtain a good approximation of the clustering. This is among the first works, to our knowledge, to use random signal multiplied by powers of the weighted adjacency matrix.

Even more recently, [192] proposed a fast algorithm for graph clustering which is provably as good as spectral clustering. The first half of their work uses random signal filtering and provides a result similar to the one presented in [29]. Moreover, they additionally show that only a subset of the nodes must be assigned with k-means and that the rest can be inferred from the graph structure by solving an optimization problem. They state bounds on the number of signals required and the number of nodes to label with k-means.

3.2 Eigenspace estimation using random signals

In this section, we lay out the theoretical findings on which our method is based. The content is centred around Theorem 3 which is our main result, and its mathematical context : random matrices and filtering.

In short, the goal of our method is to get the best estimation of the subspace formed by the k first eigenvectors of the graph Laplacian $\mathbf{L}$, denoted $\mathbf{U}_k$, for the lowest computational cost. In a similar approach to [192] and [29], we consider the filtering of random signals. In this contribution, we use an ideal low-pass filter $g(\mathbf{L}) = \mathbf{U}_k\mathbf{U}_k^*$. In this section, we show that filtering only k Gaussian random signals with this filter is sufficient to span $\mathbf{U}_k$ and that a good approximation of the subspace can be obtained using a tall SVD of the filtered signals.

3.2.1 Exact eigenspace recovery with random signals

Let us now construct our result step by step. First, assuming we pack d Gaussian random signals with i.i.d. entries $\sim \mathcal{N}(0, \frac{1}{d})$ in a Gaussian random matrix $\mathbf{R} \in \mathbb{R}^{N\times d}$, the result of the filtering using $g(\mathbf{L}) = \mathbf{U}_k\mathbf{U}_k^*$ can be written as

$$\mathbf{F} = g(\mathbf{L})\mathbf{R} = \mathbf{U}_k\mathbf{U}_k^*\mathbf{R} = \mathbf{U}_k\mathbf{R}_k, \tag{3.1}$$

with $\mathbf{R}_k = \mathbf{U}_k^*\mathbf{R}$.

We start by analysing $\mathbf{R}_k$ and then use it directly to compute the projection with $\mathbf{U}_k$. As we state in the following lemma, it happens that the matrix $\mathbf{R}_k$ is a Gaussian random matrix.

Lemma 7. *Let* $\mathbf{U}$ *be an orthonormal basis and denote* $\mathbf{U}_k$ *a subset of* k *of its rows.*

Then, $\mathbf{R}_k$*, the projection of a Gaussian random matrix* $\mathbf{R} \sim \mathcal{N}(0, \sigma^2 I)$ *onto* $\mathbf{U}_k$*, preserves all the Gaussian properties of* $\mathbf{R}$*.*

The proof of this lemma is given in Appendix A.3.1. Using Lemma 7, we know that $\mathbf{R}_k \in \mathbb{R}^{k \times d}$ is i.i.d. Gaussian of zero mean and variance $\frac{1}{d}$. The next step is to show that $\mathbf{R}_k$ is full rank, which we state in the following lemma.

Lemma 8. *Let* $\mathbf{R}_k \in \mathbb{R}^{k \times d}, d \geq k$ *be a Gaussian random matrix with i.i.d. entries.*

Then, the matrix $\mathbf{R}_k$ *is full rank with probability 1. That is,* $\text{rank}\{\mathbf{R}_k\} = k$ *since* $d \geq k$*.*

The proof of this lemma is given in Appendix A.3.2. Now that we confirmed that $\mathbf{R}_k$ is full rank, we analyse the span of the complete projection $\mathbf{F} = \mathbf{U}_k\mathbf{R}_k$ in the following lemma.

Lemma 9. *Let* $\mathbf{F} = \mathbf{U}_k\mathbf{R}_k$ *a matrix of size* $N \times d$*, with* $\mathbf{U}_k$ *and* $\mathbf{R}_k$ *as defined above. The two following statements are true:*

$$\forall x \in \mathbb{R}^k, \exists y \in \mathbb{R}^d : \mathbf{U}_k x = \mathbf{F} y. \tag{3.2}$$

$$\forall y \in \mathbb{R}^d, \exists x \in \mathbb{R}^k : \mathbf{U}_k x = \mathbf{F} y. \tag{3.3}$$

That is, $\mathbf{F}$ *and* $\mathbf{U}_k$ *share the same column space.*

Proof. Since $\mathbf{R}_k$ is full rank, its span is able to generate any matrix of $\mathbb{R}^{k \times d}$. Then, the projection of this full space onto $\mathbf{U}_k$ can form any matrix generated by the span of $\mathbf{U}_k$. □

Now that we have all the necessary pieces, we state our main result in the following theorem.

Theorem 3. *Let* $g(\boldsymbol{\lambda})$ *be an ideal low-pass filter of cutoff frequency* λ_k*, let* $\mathbf{R} \in \mathbb{R}^{N \times d}$ *be a random matrix formed of entry-wise i.i.d. Gaussian random variables* $\sim \mathcal{N}(0, \frac{1}{d})$*. Let* $\mathbf{L}$ *be the Laplacian of any graph* $\mathcal{G}$ *and* $\mathbf{F} = \mathbf{U}_k\mathbf{R}_k$*. Further assume that* $\mathbf{F}$ *is factorized as* $\mathbf{F} = \mathbf{B}\Sigma\mathbf{V}^\top$*.*

Then, for any $d \geq k$*, the matrix* $\mathbf{B}$ *provides the first* k *eigenvectors of* $\mathbf{L}$ *altered only by a rotation in* $\mathbb{R}^k$*.*

The proof of this Theorem is given in Appendix A.3.3. Note that, although the only assumption used is $d \geq k$, we suggest using $d = k$ in practice since this is the minimal value for which the result holds and thus the one that will require the fewer computations.

Although we specifically addressed the eigenspace of the graph Laplacian, we would like to stress the fact that the theory described here does not use any specific assumption made on $\mathbf{L}$

except that it is symmetric and positive semi-definite. Thus, the statements we make are also true for any matrix for which there exists a spectral decomposition. However, the sparsity of this matrix is key to a fast implementation using graph filtering as we will show later.

3.2.2 F as an approximation of $\mathbf{U}_k$

The matrix $\mathbf{B}$ has been shown to approximate $\mathbf{U}_k$ up to a rotation, which is perfectly fine for all common applications (e.g., embedding, spectral clustering, etc.). In the following lines, we wanted to present the quality of $\mathbf{F}$ as a direct approximation of $\mathbf{U}_k$. In the discussion below, we show that it could be enough in some situations to stop the procedure before the SVD step and reduce further the complexity of the algorithm.

We have shown in Lemma 8 that $\mathbf{F}$ and $\mathbf{B}$ share the same column space (i.e., span$\{\mathbf{U}_k\}$) in Theorem 3 and have the same shape. The major difference between the matrices is that only the latter is composed of normalized columns. However, the distribution of the singular values of $\mathbf{F}$ is well-known: it is the same as that of $\mathbf{R}_k$ since $\mathbf{U}_k$ has unitary columns. Moreover, the works of Marchenko and Pastur [123] contain a large number of results regarding the study of Gaussian ensembles and Wishart matrices. They showed, among other things, that the eigenvalues of Wishart matrices follow a quarter circle law, which means that the distribution of any singular value of $\mathbf{F}$ is a normalized quarter circle of support $[0;2]$ when $d = k$. On top of that, they proved that the expected value and the standard deviation of those eigenvalues tend to 1 as N becomes large. This means that in average, even with $d = k$, $\mathbf{F}$ is a very good candidate for the approximation of the subspace. The problem is that with the variance on the eigenvalue distribution, random samples hardly benefit from the expectation.

Meanwhile, the Johnson-Linderstrauss lemma says that with $d = \mathcal{O}(\log(N))$, the distances between rows of $\mathbf{U}_k$ and rows of $\mathbf{F}$ are almost preserved, up to a $(1+\varepsilon)$ multiplicative factor, with high probability. Thus, it seems intuitive that picking more random signals would improve the repartition of the eigenvalues between 0 and 2 and provide concentration around the mean. In fact, from the definition of the Marchenko-Pastur distribution, we have the following result (adapted from [203, Corollary 5.35]).

Lemma 10. *Let $\boldsymbol{A}$ be an $N \times n$ matrix whose entries are independent standard normal random variables. Then for every $t \geq 0$, with probability at least $1 - 2\exp(-t^2/2)$ one has*

$$\sqrt{N} - \sqrt{n} - t \leq \sigma_{\min}(\boldsymbol{A}) \leq \sigma_{\max}(\boldsymbol{A}) \leq \sqrt{N} + \sqrt{n} + t. \tag{3.4}$$

with $\sigma_{\min}(\boldsymbol{A})$ and $\sigma_{\max}(\boldsymbol{A})$ the minimum and maximum eigenvalue of $\boldsymbol{A}$ respectively.

In our case, the entries of $\mathbf{R}_k$ are Gaussians of variance $\frac{1}{N}$. Applying Lemma 10, we get the following bound on the extreme eigenvalues :

$$1 - \sqrt{\frac{k}{d}} - \frac{t}{\sqrt{d}} \leq \sigma_{\min}(\mathbf{R}_k) \leq \sigma_{\max}(\mathbf{R}_k) \leq 1 + \sqrt{\frac{k}{d}} + \frac{t}{\sqrt{d}}. \tag{3.5}$$

We conclude that the more the matrix $\mathbf{R}_k$ is flat (i.e., $d > k$), the more its eigenvalues are concentrated around 1 in good probability, which confirms our intuition. However, we see that the convergence to the mean is only $\mathcal{O}(\sqrt{d})$, which is quite slow. Indeed, compared to the bound of the Johnson-Lindenstrauss lemma, we see that estimating the eigenspace by using only concentration results converges much slower than distance preservation. This makes sense since the latter is a simpler problem. To summarize, we see that sidestepping the SVD is not interesting when looking for the eigenspace approximation, but is meaningful to generate features used, for example, in spectral clustering.

3.3 Computational aspects of subspace approximation

Now that our main theoretical result is established, we look into its practical implementation, while focusing on efficient solutions. First, we verify that the argument about the rank made in Lemma 8 is valid in practice. Next, we present a solution on how to find the cutoff eigenvalue λ_k. Then, we show our choice for the actual filter design using polynomials enabling fast filtering operations while limiting the problems caused by the approximation. Finally, we describe our algorithms and analyse their complexity.

3.3.1 Numerical limits of rank approximation

As Lemma 8 is critical to the proof, we make a small digression and discuss its numerical approximation. Indeed, we proved that the matrix $\mathbf{R}_k$ is full rank and this means that the smallest singular value of $\mathbf{R}_k$ is strictly positive. However, while computing the singular value decomposition, numerical approximations are performed and the singular values below a given threshold are assimilated to linearly dependent columns. In other words, we need to make a stronger statement and ensure that the smallest singular value stays above a numerical precision threshold in good probability. To this end, we state the following lemma (adapted from [124, Lemma 3.15]).

Lemma 11. *Suppose that k and ℓ are positive integers with $k \leq \ell$. Suppose further that G is a real $\ell \times k$ matrix whose entries are i.i.d. Gaussian random variables $\sim \mathcal{N}(0,1)$, and β is a positive real number, such that*

$$1 - \frac{1}{\sqrt{2\pi(\ell-k+1)}}\left(\frac{e}{(\ell-k+1)\beta}\right)^{\ell-k+1} \tag{3.6}$$

is nonnegative.

Then, the least (that is, the kth greatest) singular value of G is at least $\frac{1}{\sqrt{\ell}\beta}$ with probability not less than the amount in (3.6).

Since $\mathbf{R}_k$ in our case is a Gaussian random matrix of size $k \times k$ (i.e. assuming we take $d = k$), zero mean and variance $\frac{1}{k}$, then $\sigma_{\min}(\mathbf{R}_k)$ equals $\frac{\lambda_{\min}}{\sqrt{k}}$ with $\lambda_{\min}$ the smallest singular value of

a matrix whose entries match the lemma above. Thus from this result, we can state that the cumulative density function of $\sigma_{\min}(\mathbf{R}_k)$ is:

$$\mathbb{P}\left(\sigma_{\min}(\mathbf{R}_k) < \frac{1}{\beta}\right) < \frac{e}{\beta\sqrt{2\pi}} \tag{3.7}$$

In practice, we need to ensure that the minimal singular value is above the predefined threshold of the rank estimate, that usually is around 10^{-13} (i.e. machine precision). Knowing that the probability of $\sigma_{\min}(\mathbf{R}_k)$ being below the numerical threshold is less than $\frac{e}{\sqrt{2\pi}}10^{-13} \approx 10^{-13}$, we can conclude that the claim we made theoretically for the rank still holds in practice with very high probability.

3.3.2 Estimation of λ_k

The computation of $\mathbf{F}$ described above depends on the quality of the filter g and the determination of its cutoff frequency λ_k which is not known a priori. A standard method is to use eigencount techniques such as the one proposed in [56]. In this work, the authors used the fact that the energy retained by an ideal low-pass filtering of random signals with cutoff frequency λ_k, called g_{λ_k}, is proportional to the number of eigenvalues that are smaller than λ_k. Mathematically, we have:

$$\mathbb{E}\left[\|g_{\lambda_k}(\mathbf{L})\mathbf{R}\|_F^2\right] = |\{\lambda : \lambda \leq \lambda_k\}|\,. \tag{3.8}$$

Thus, by dichotomy, one could approximate the desired threshold value λ_k for our filter, since we want it to capture exactly k eigenvalues. Unfortunately, each step of the dichotomy requires $\mathcal{O}(k)$ filterings and the dichotomy must be applied $\mathcal{O}(\log(N))$ times, without making strong assumptions on the distribution of the eigenvalues over the spectrum. Thus, the estimation of λ_k such as defined and used in [192] costs $\mathcal{O}(k\log(N))$ filtering operations, which is more than the actual eigenspace estimation procedure.

To circumvent this issue, we propose an accelerated version of the eigencount technique for the determination of λ_k that will not increase the complexity of the complete algorithm. To do so, we first assume that the eigenvalues are distributed evenly over the spectrum (between 0 and $\lambda_{\max}$). Thus, on average, the k^{th} eigenvalue should be $\mathbb{E}[\lambda_k] = \frac{k}{N}\lambda_{\max}$. However, one will not find the exact count systematically on the first guess, due to the randomness of the process and the non-uniformity of the eigenvalue distribution in practice. We suggest thus to iterate with the assumption of local uniformity of the distribution of the eigenvalues until the goal is reached. In practice, this means that our initial pick will be $\lambda^{(0)} = \frac{k}{N}\lambda_{\max}$. Then, assuming the value at iteration t is noted $\lambda^{(t)}$, we count the actual number of eigenvalues below it $n_{\lambda^{(t)}}$ using Eq. (3.8). We then iterate using the simple linear estimation $\lambda^{(t+1)} = \frac{k}{n_{\lambda^{(t)}}}\lambda^{(t)}$ until the targeted count is achieved with good precision. See Algorithm 7 for the detailed procedure.

Note that, since the number of iterations does not depend on N but only of the local eigenvalue distribution, a good precision can be achieved with a constant number of iterations. The cost

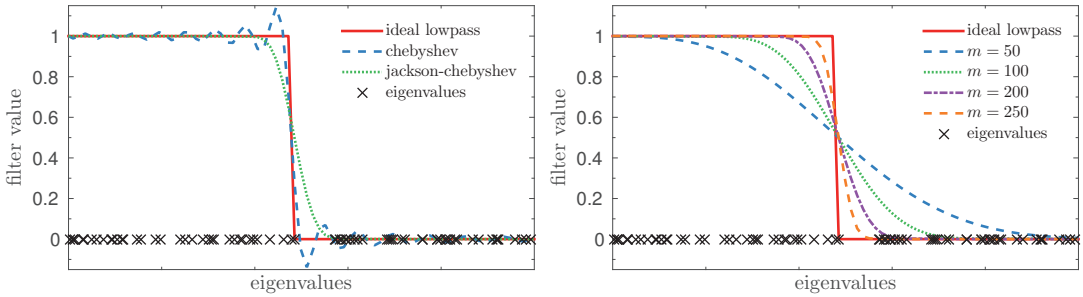

Figure 3.1 – The effect of approximating a step function with polynomials. The solid red line is the ideal step function. The black crosses represent the eigenvalues. The approximation using Jackson-Chebyshev polynomials (dotted line) is compared with Chebyshev polynomial approximation (dashed line) of same order m in (a). Jackson-Chebyshev approximations with different orders m are compared in (b).

in number of operations of this accelerated version is thus $\mathcal{O}(k)$ filterings which is acceptable since it is of the same order than the remaining of our method.

3.3.3 Acceleration using fast filtering

The construction of the matrix $\mathbf{F}$ in the previous section requires the knowledge of the first k eigenvectors of the graph Laplacian. This knowledge is very costly for large graphs (N large) since it requires a partial SVD of a $N \times N$ matrix, which we try to avoid in the first place. Fortunately, as we explained before, the product $\mathbf{U}_k\mathbf{U}_k^\top$ corresponds to a graph filtering with $g(\mathbf{L})$, an ideal low-pass filter

$$g(\lambda) = \begin{cases} 1 & \lambda \leq \lambda_k, \\ 0 & \lambda > \lambda_k. \end{cases} \tag{3.9}$$

Since we cannot afford the cost of exact filtering which would imply the computation of the SVD we try to avoid, we use a polynomial approximation of the filter $g(\mathbf{L})$ (see Section 1.2.2). For the task at hand, the Jackson-Chebyshev [56] polynomial approximation is the best suited to approximate the step function of $g(\mathbf{L})$ since it avoids the Gibbs effect of Chebyshev polynomials, depicted in Fig. 3.1 (left).

The quality of the approximation is based on the order of the polynomial, directly related to the number of coefficients to compute. If we define m as the highest degree of the polynomial, we can show that the approximation error decreases as m increases. This effect is shown in Fig. 3.1 (right) where we can see the convergence to the ideal low-pass with an increasing value of m. Since the complexity of the filtering increases linearly with m one cannot let it become too large. In particular, we cannot let m be $\mathcal{O}(N)$ since it would have a huge impact on the overall complexity.

Let us remind here that the filter approximation needs to be correct only on the discrete values given by the eigenvalues. Indeed, the approximation does not need to fit closely $g(\lambda)$ as long as the discrete values that the filter takes on the eigenvalues are correct. In our case, since we only want to approximate a step function, we need the value of the filter to be equal to 1 for $\lambda_0, \lambda_1, \ldots, \lambda_{k-1}$ and 0 for $\lambda_k, \lambda_{k+1}, \ldots, \lambda_{N-1}$. The non-respect of this condition can be caused by two approximation errors: the estimated cutoff eigenvalue can be wrong or the order of the polynomial can be too small. Let us examine both situations in detail.

If the order m is too small, then, as can be seen in Fig. 3.1 for $m = 100$, the filter will be below 1 for a few eigenvalues below λ_{k-1}, and above 0 for a few eigenvalues after λ_k. If the estimated cutoff eigenvalue is a bit off, a similar situation happens, with a shift towards lower or higher frequencies. In both cases, the value of the filter still equals 1 (up to some eigenvalue λ_j), then monotonically decreases to 0 (up to some eigenvalue λ_l) and finally is equal to 0 (up to λ_{N-1}) ; with $\lambda_j < \lambda_k < \lambda_l$. In such a case, the filter has non-zero coefficients in the range $[\lambda_k, \lambda_l]$ and thus, $\mathbf{F}$ will be contaminated by some elements of the space $\mathbf{U}_{[k+1,l+1]}$.

However, the energy of these erroneous contributions is not too large since the coefficients of the filter for the eigenvalues bigger than λ_k are smaller than all coefficients of the range $[\lambda_0, \lambda_{k-1}]$. Since our final approximation $\mathbf{B}$ is done using an SVD of $\mathbf{F}$, then $\mathbf{B}$ will be the best rank k approximation of $\mathbf{F}$ minimizing the energy of the residuals, thus discarding the undesired contributions. Indeed, as one can verify in the experiments of Section 3.4, $\mathbf{B}$ provides features that remain very good for the various applications that we develop, even with a low polynomial order.

3.3.4 Algorithms

We propose in this section to summarize the procedure to obtain the approximation of the subspace $\mathbf{U}_k$ based, on one side, on the theoretical development of Section 3.2.1, and, on the other side, on the practical considerations of Sections 3.3.2 and 3.3.3.

Algorithm 6 summarizes the steps of our method to approximate the eigenspace $\mathbf{U}_k$ from $\mathbf{L}$. If a graph is not provided with the data, a k-NN graph can be constructed and its associated Laplacian computed beforehand. The algorithm takes a graph and a number k as input and outputs a set of k approximated eigenvectors of the graph Laplacian.

Algorithm 6 Eigenspace Approximation

1: Generate $\mathbf{R}$ with $d = k$ as defined in Section 3.2.1
2: Estimate λ_k using Algorithm 7
3: Compute the approximated graph filter $g(\lambda)$ cf. Section 3.3.3
4: Apply filtering: $\mathbf{F} = g(\mathbf{L})\mathbf{R}$
5: Compute an economic SVD: $\mathbf{USV} = \text{SVD}(\mathbf{F})$
6: Return the left singular vectors $\mathbf{U}$

Algorithm 7 presents in detail the strategy described in section 3.3.2 for the accelerated

estimation of λ_k. The main assumption here is that the distribution of the eigenvalues is uniform by part over the spectrum. Since some parts of the spectrum can be empty due to eigengaps for some classes of graphs, we implemented a dichotomic step to get a broad spectrum distribution estimate if the search does not progress.

Algorithm 7 Estimation of λ_k

Require: k, λ_{max} and $\mathbf{L}$
Initialize: $\lambda_{lb}, c_{lb}, iter, c_{est} \leftarrow 0$
$\lambda_{ub} \leftarrow \lambda_{\max}, c_{ub} \leftarrow N$
$\lambda_{est} \leftarrow k\frac{\lambda_{\max}}{N}$
Generate $\mathbf{R}$ with $d = k$
while $c_{est} \neq k$ and $iter < max_{iter}$ **do**
    Compute approximated graph filter g with $\lambda = \lambda_{est}$
    $c_{est} \leftarrow \|g(\mathbf{L})\mathbf{R}\|_F^2$
    if $c_{est} < k$ **then**
        $\lambda_{lb} \leftarrow \lambda_{est}$
    else
        $\lambda_{ub} \leftarrow \lambda_{est}$
    end if
    if $c_{lb} = c_{est}$ or $c_{ub} = c_{est}$ **then**
        $\lambda_{est} \leftarrow \frac{\lambda_{lb}+\lambda_{ub}}{2}$
    else
        if $c_{est} < k$ **then**
            $c_{lb} \leftarrow c_{est}$
        else
            $c_{ub} \leftarrow c_{est}$
        end if
        $\lambda_{est} \leftarrow \lambda_{lb} + (k - c_{lb})\frac{\lambda_{ub}-\lambda_{lb}}{c_{ub}-c_{lb}}$
    end if
end while
return λ_{est}

3.3.5 Complexity analysis

Steps 1 and 3 of Algorithm 6 are nonsignificant in the analysis of the overall complexity. We focus here on steps 2, 4 and 5 for which the number of operations is studied in detail. Using fast filtering operations, applying our method consists of k graph filtering operations at step 4, which is $\mathcal{O}(m|\mathcal{E}|k)$, with m the order of the polynomial approximation of the filter. The SVD performed in step 5 has an additional cost of $\mathcal{O}(k^3)$ for a tall matrix of size N by k as we have here. Finally, step 2 takes $\mathcal{O}(m|\mathcal{E}|k)$ if we consider the improvement proposed in Section 3.3.2. Thus, the total complexity of our method is $\mathcal{O}(m|\mathcal{E}|k + k^3)$.

Comparison with IRLM [35]

As reminded above, the complexity of IRLM is $\mathcal{O}(h(|\mathcal{E}|k+k^2N+k^3))$ with h a convergence factor. Thus, assuming h and m have similar orders, the IRLM needs at least $\mathcal{O}((h-1)k^2N)$ more operations than our method. In any reasonable application, we will have either $k<N$ or $k \ll N$, thus, the term $\mathcal{O}(hk^2N)$ will be larger than the term $\mathcal{O}(hk^3)$.

Comparison with CSC [192]

Although the method presented in [192] is not directly an eigenspace estimation method, it does use the same mechanics of filtered random signals on the graph to obtain spectral features. The number of filtering needed is d, which has to be larger than a threshold given in [192, Theorems 3.2 and 3.4]. To simplify, we can say that $d=\gamma\log(\alpha k\log(k))$ where γ and α are influenced by the precision of the distance preservation and the probability that the distance is preserved. Note that even with medium precision (e.g., 10^{-1}), the constants γ and α will be large (i.e., 10^3). The overall complexity for the spectral features estimation will cost $\mathcal{O}(m|\mathcal{E}|\gamma\log(\alpha k\log(k)))$ operations. Finally, the $\mathcal{O}(\log(N))$ filterings required to estimate λ_k have an added cost of $\mathcal{O}(m|\mathcal{E}|\log(N))$.

If we compare the complexity of FEARS with CSC, we get the difference in numbers of operations:

$$\begin{aligned}\Delta &= m|\mathcal{E}|k+k^3-m|\mathcal{E}|(d+\log(N))\\ &= m|\mathcal{E}|(k-d-\log(N))+k^3\\ &= m|\mathcal{E}|(k-\gamma\log(\alpha k\log(k))-\log(N))+k^3.\end{aligned}$$

For sparse graphs we can assume $|\mathcal{E}|=c_dN$, with c_d the average node degree, which gives:

$$\Delta = mc_dN(k-\gamma\log(\alpha k\log(k))-\log(N))+k^3.$$

In order to finish the comparison, we now need to make hypotheses on the relation between k and N.

If we assume that $k=\mathcal{O}(log(N))$, then, for N large we have

$$\begin{aligned}\Delta &= mc_dN(\log(N)-\gamma\log(\alpha k\log(k))-\log(N))+\log^3(N)\\ &= \log^3(N)-mc_dN\gamma\log(\alpha k\log(k))\\ &< 0,\end{aligned}$$

with the last step following from the fact that $\log(\alpha k\log(k))>1$ and $\mathcal{O}(\log^3(N))<\mathcal{O}(N)$. This means that for this regime, our method is cheaper than CSC, for large N.

If we assume that $k = \mathcal{O}(\sqrt{N})$, then, for N large we have

$$\begin{aligned}\Delta &= mc_d N\left(\sqrt{N} - \gamma\log(\alpha N^{\frac{1}{2}}\log(N^{\frac{1}{2}})) - \log(N)\right) + \sqrt{N^3} \\ &= N\left(\sqrt{N}(mc_d+1) - \gamma\log\left(\frac{\alpha\log(N)}{2}\right) - \frac{\gamma+2}{2}\log(N)\right) \\ &> 0,\end{aligned}$$

with the last step coming from the fact that $\gamma > 1$ and $\mathcal{O}(\sqrt{N}) > \mathcal{O}(\log(N))$. This means that for this regime CSC will be cheaper than our method for large enough N.

From the two cases described above we can see that, if $\mathcal{O}(1) \leq k \leq \mathcal{O}(\log(N))$, our method is cheaper; and if $\mathcal{O}(\sqrt{N}) \leq k \leq \mathcal{O}(N)$, then CSC is cheaper. Note that in both cases the order of the filter m is kept constant, but that both results hold for any m, even with $m = \mathcal{O}(N)$.

3.4 Experiments

In this section, we present experiments whose objective is to show how our proposed methods behave in practice. First, we want to ensure that our algorithms do fulfil their goals, i.e., that they provide accurate enough results and do so efficiently. Second, both as illustrations and practical applications, we show the performance of our eigenspace approximation method on typical clustering and visualization tasks.

3.4.1 Time performance analysis

Since the complexity analysis in Section 3.3.5 only covers asymptotically large N, it is also interesting to look at the cost of the algorithms for actual implementations and realistic values of N and k. In addition to the eigenspace estimation with IRLM and the k-dimensional spectral features of Compressive Spectral Clustering (CSC) mentioned in Section 3.3.5, we consider the power method described in [29].

The data on which the different methods are evaluated consists of N points of small intrinsic dimension which are randomly drawn. In addition, a kNN graph with 10 neighbours is constructed from the data points. Each method is run with fixed parameters and the time is measured in total CPU time to completion.

In Figure 3.2 we report the time needed in function of k with N fixed. The first observation is that the power method does not scale well with k and is exceedingly time-consuming for everything other than very small values of k, for which it performs well. Since it is order an order of magnitude slower for the parameters used in the other experiments, it is not displayed in the remaining figures to keep readability. We see, as expected in accordance with the complexity analysis, that, our method performs better than IRLM and worse than CSC above a threshold corresponding to $\sqrt{N}$ (i.e., 100).

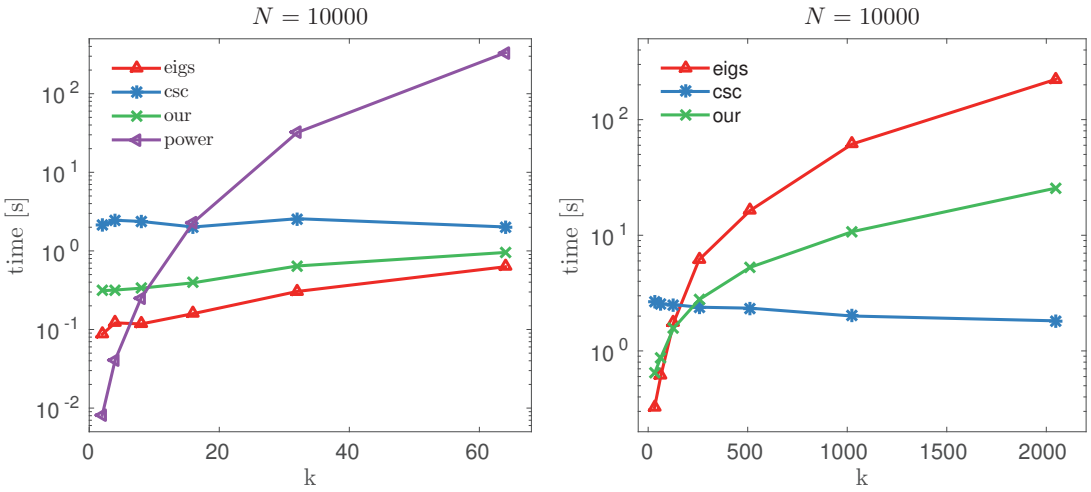

Figure 3.2 – Comparison of CPU time needed between different methods for the estimation of an eigensubspace of dimension k. The time is reported for increasing values of k and with N fixed. Small values of k are displayed on the left and larger ones on the right. The method 'power' is only reported for small values since it scales badly. The time axis is in log scale.

Figure 3.3 shows the results for an exponentially growing N and two regimes, $k = \log(N)$ and $k = \sqrt{N}$. In the former, our method outperforms both eigs and CSC for all values of N. In the latter our method performs best up to $N = 10^6$. Above this value, CSC is best. Note that results above $N = 10^6$ for this regime are not shown due to memory limitations for eigs.

Altogether, those results confirm the conclusions drawn from the complexity analysis of Section 3.3.5. First, except for very small values of k, eigs is the most time-consuming method, even though it benefits from very optimized implementations. Second, for the $\log(N)$ regime, our method performs best for all values of N. For the $k = \sqrt{N}$ regime, our method is cheaper than CSC for $N < 10^6$. Above the limit $k = \sqrt{N}$, CSC is the cheapest method. As a final remark on these results, we need to point out that, contrarily to the other methods considered in this experiment, CSC does not compute an eigensubspace per se but only k-dimensional features allowing good pairwise distance measurements between data points.

As a last remark on timing, we want to call attention to the fact that when filtering multiple random signals, all filtering operations are independent. Indeed, the signals are independent by definition and both the polynomial coefficients of the filter and the Laplacian are unaltered by the successive filtering operations. The filtering operations in our algorithms could thus easily benefit from a parallel implementation.

3.4.2 Quality of approximation for various graphs

In this section, we measure the accuracy of our algorithms for different classes of graphs and for different values of k and N. In particular, we wish to evaluate two things: on one hand, the

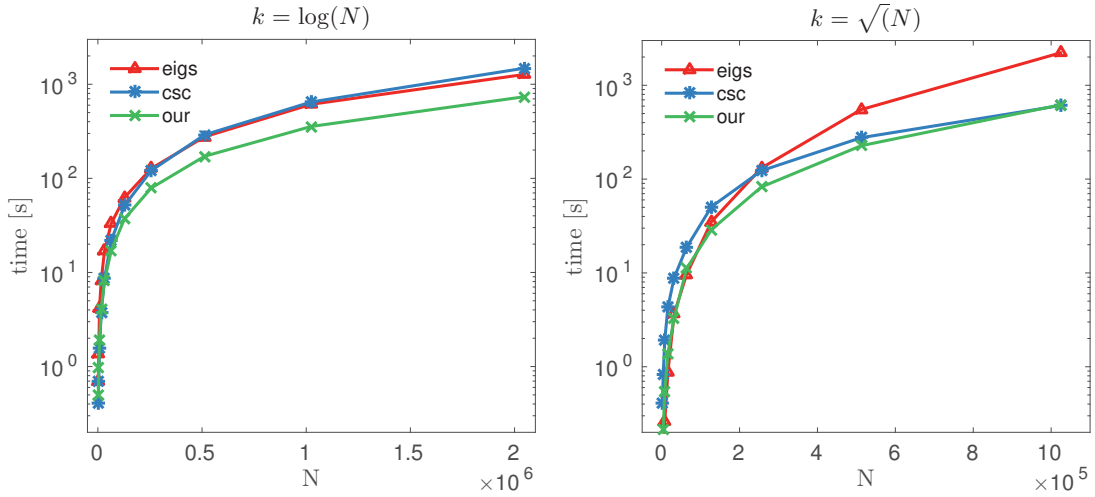

Figure 3.3 – Comparison of CPU time needed between different methods for the estimation of an eigensubspace of dimension k. The time is reported for increasing values of N with two regimes : $k = \log(N)$ on the left and $k = \sqrt{N}$ on the right.

quality of approximation of the eigenspace $\mathbf{U}_k$ with Algorithm 6 and, on the other hand, the precision and efficiency of our accelerated eigencount method with Algorithm 7.

The graphs chosen for this experiment are commonly used in the GSP community and are known to have various spectral properties. Here is a list of all graphs with short descriptions:

- **Sensor network**: A graph of a synthetic sensor network, which represents randomly positioned sensors connected in a kNN fashion.
- **SBM**: Stochastic Block Model represent social networks or community graphs and are known to be clusterable (and thus possesses eigengaps).
- **Swissroll**: This graph is a kNN graph of the famous Swissroll manifold, a point cloud drawn from a rolled 2D surface in 3D.
- **Bunny**: This graph is the knn graph constructed from the 3D point cloud of the Stanford bunny.
- **Image graph**: This graph is created by connecting the pixels of an image using similarity of patches. The image of interest is the grayscale image of Barbara, a natural image often used in image processing.
- **Road network**: This graph represents the Minnesota road network (originally from the MatlabBGL library).

In order to measure the quality of the approximated eigenspace (up to a rotation), we introduce a measure of the amount of energy which is preserved when the approximated eigenspace is projected on the real eigenspace computed with exact methods. If we note the approximated eigenspace as $\mathbf{B}_k$ and the exact eigenspace $\mathbf{U}_k$, the normalized energy kept by the projection

		Sensor network	SBM	Swiss-roll	Bunny	Image	Road network	
		N = 10000	N = 10000	N = 10000	N = 2503	N = 16384	N = 2642	
ME	exact	0.86 ± 0.01	1.00 ± 0.01	0.86 ± 0.02	0.99 ± 0.01	0.91 ± 0.01	0.93 ± 0.01	0.92
	standard	0.80 ± 0.03	0.95 ± 0.05	0.79 ± 0.03	0.94 ± 0.05	0.86 ± 0.04	0.90 ± 0.05	0.87
	fast	0.80 ± 0.03	0.96 ± 0.04	0.79 ± 0.03	0.95 ± 0.04	0.86 ± 0.04	0.90 ± 0.04	0.88
IT	standard	14.62 ± 0.90	**5.32** ± 1.58	4.68 ± 0.62	8.74 ± 1.77	13.06 ± 1.58	11.34 ± 1.22	11.29
	fast	**3.02** ± 0.71	9.36 ± 1.06	**2.86** ± 0.70	**4.48** ± 1.25	**3.12** ± 0.75	**3.06** ± 0.51	**4.31**
KD	standard	0.60 ± 0.53	2.46 ± 4.92	0.52 ± 0.58	0.36 ± 0.53	0.34 ± 0.48	0.36 ± 0.48	0.79
	fast	**0.00** ± 0.00	**1.00** ± 1.01	**0.00** ± 0.00	**0.00** ± 0.00	**0.00** ± 0.00	**0.00** ± 0.00	**0.17**

Table 3.1 – **Quality of approximation.** This table reports three quality measures of our proposed algorithms for eigenspace estimation and λ_k estimation. For all experiments, the following parameters were used: $m = 500$, $k = 25$, $\epsilon = 10^{-1}$ for the standard eigencount method and 10 maximum number of iterations in the fast method. Green bold face numbers are the best score between two lines for the eigencount comparison. The last column is the average over all graphs. The average mean energy is written ME and computed as in Eq. (3.10). The ME measure is between 0 and 1 and higher values are better. The average number of iterations is noted IT for which smaller values are better. The mean squared deviation from k is noted KD for which smaller values are better. In ME, "exact" denotes the score computed using the true λ_k. For everything else, λ_k is estimated either with the dichotomy method proposed in [192] (standard) or with our proposed method as in Algorithm 7(fast).

is:

$$E(\mathbf{B}_k, \mathbf{U}_k) = \frac{1}{k} \|\mathbf{B}_k^T \mathbf{U}_k\|_F^2. \tag{3.10}$$

We chose to use the normalized energy to score the quality of the estimated eigenspace as it gives a number between 0 and 1 where higher values mean better approximation.

In order to compare our accelerated eigencount method with the reference dichotomy implementation of [192] (abbreviated 'fast' and 'standard' respectively in Table 3.1), we used two measures. First, the number of iterations required until convergence, which is adequate since the workload per iteration is the same in the two algorithms. Finally, we measure how close the algorithm converged to the true value of k (using the mean squared deviation from the target k). This last measure is useful to check if the method is able to converge with respect to the current random matrix used for estimation, not with respect to the actual value of λ_k.

The results of all measures for the various graphs described above are reported in Table 3.1. Due to the randomness of the different methods, all experiments are averaged over 50 realizations and the standard deviation is indicated for all measures.

If we first focus on the upper part of Table 3.1 we can see that the measure of the energy (ME) using the true cutoff λ_k shows an average above 90% of precision over all graphs with a perfect score for very clusterable graphs (such as SBM) and lower values for more difficult graphs (such as Sensor network). The trend is similar using estimated values for λ_k both with the standard and fast methods. Using the approximated cutoffs lowers the score of about 5%. Using the fast method leads to marginally better results. One very interesting fact regarding

these results is that both the λ_k estimation step and the eigenspace approximation contribute to the lost energy in approximately equal amounts. This seems to indicate that it is important to balance the computational effort between the two steps and not favouring one against the other.

On the middle part of Table 3.1, we saw that the number of iterations needed to compute λ_k (IT) is lower with the fast method for all graphs but the SBM. On average, the fast method is 2.5 times faster than the standard method. For the SBM graph, 'fast' is close to its maximum number of iterations meaning that the eigencount hardly converged. This result can be easily explained by the fact that the eigenvalue distribution for SBM is known to be highly non-uniform, especially for low frequencies, which is partly incompatible with the local uniformity hypothesis assumed by the fast method.

On the lower part of Table 3.1 the precision of the estimated k (KD) is reported. Both the fast and standard methods converge most of the time, with a better overall convergence of the former which converges exactly to the true value, except for SBM. This could be expected from the high number of iterations needed for this specific graph.

From those results, we can see that the quality of the estimated subspace computed using our proposed method is decent, while not perfect. The imprecision coming both from the approximation in the filter design and cutoff eigenvalue estimation. Our scheme for accelerated λ_k estimation is faster than the reference method and provides very good results.

3.4.3 Clustering

In this section, we want to test the capability of our filtered signals to produce a good clustering assignment for data points. We will compare the results obtained by our method (FEARS) to Spectral Clustering (SC) [164] and Compressive Spectral Clustering (CSC) [192]. We will also see that the compressive step of the latter can be used with k filtered signals instead of d. Before presenting the results, we briefly present the two techniques used for comparison.

Spectral clustering is a very famous method that follows directly from the relaxation of NCut for k classes. It states that the k eigenvectors associated with the smallest eigenvalues of $\mathbf{L}$ represent the optimal solution of the optimization problem of NCut. Thus, by computing the eigensubspace $\mathbf{U}_k$ one easily gets a very good assignment for the data partitioning problem since the k-means solution over the rows of the matrix $\mathbf{U}_k$ gives a standard discrete partition of the data points. However, computing spectral clustering on large graphs is not to be considered due to the runtime complexity of the method ($\mathcal{O}(N^3)$ for exact methods, $\mathcal{O}(k^2N)$ with IRLM).

In Compressive Spectral Clustering [192], the authors replaced the features formed by the eigenvectors with filterings of random signals on the graph $\mathcal{G}$. They propose a minimal number of signals to filter in order to preserve the distances between any two points in the data set. Then they apply k-means on the filtered signal to obtain an assignment identical to spectral clustering. Their second contribution is to show that k-means can be compressed, in the sense

that only a subset of the nodes needs to be assigned with this costly method. The remaining labels can be inferred by solving an optimization problem using graph regularization.

Synthetic case: Stochastic Block Model

For this experiment, we use a Stochastic Block Model (SBM) with $N = 5\,000$ nodes and $k = 20$ clusters. We set the average degree of the nodes to $s = 16$ and the nodes are associated at random with a particular class (the ground truth for the assignment). Then, an edge between two nodes exists with probability p if the two nodes belong to the same class (intra-cluster probability) and with probability $q < p$ if they belong to different clusters (inter-cluster probability). We generate several graphs with different ratios $\varepsilon = \frac{q}{p}$ (the larger ε, the harder the community detection) to evaluate our clustering capabilities.

The evaluation of the different methods is performed using the Adjusted Rand similarity Index [84] between the SBM ground truth and the resulting assignments. All results presented here are averaged over 50 realizations in each setup.

By looking at Figure 3.4 we can first observe that our method is the one that approximates the best the results of SC. Note that SC is not necessarily the method achieving the best rand index as ε increases since the ground truth is set before the edges are created. Thus, for relatively large values of ε, it might not make sense to keep this assignment for clustering purposes. For this reason, in our view, spectral clustering is the target to fit at best.

In addition, notice that the order of the polynomial approximations alters the result of the clustering in both our method and CSC. Finally, CFEARS represents the result of our features assigned with the compressive step of CSC instead of the full k-means. We see that k-means is more faithful to spectral clustering than the regularized label diffusion on the graph.

Real-world example: Amazon co-purchasing network

In addition to the synthetic SBM graphs, we want to go further and show that this also works well for real-world data sets. To this end, we consider the problem of clustering the Amazon co-purchasing network [214] that has also been evaluated for the study of CSC. The graph is composed of 334 863 nodes and 925 872 edges.[1] No clear ground truth can be used since the given information is the belonging of products to categories with overlaps. We decided to reproduce the experiment published in [192], adding our method to the benchmark. We evaluated the resulting assignments with two measures: the modularity score [134], used to determine whether a given partition is separating the network efficiently, and the adjusted Rand similarity index compared to the result of SC, used to identify the resemblance of the two assignments.

In Table 3.2 we first show the performance of the different algorithms with three different

[1] Available at http://snap.stanford.edu/data/com-Amazon.html

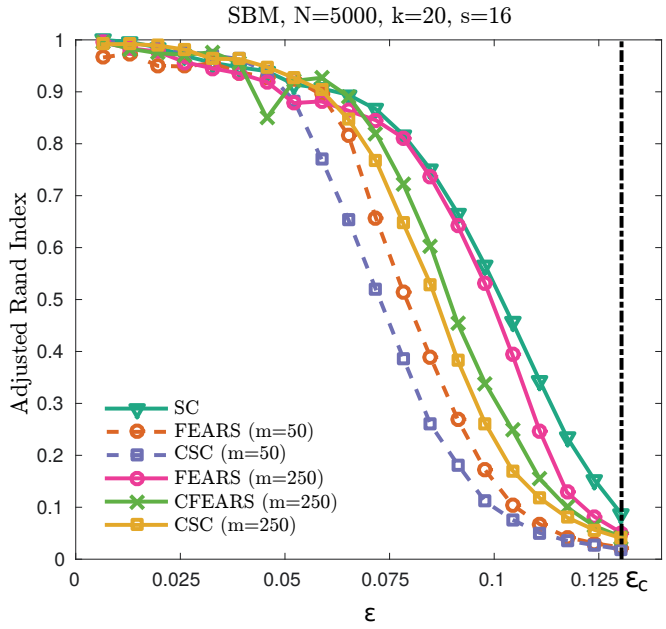

Figure 3.4 – Study of the clusterability of Stochastic Block Models for various values of ε, representing how well the graph can be split into clusters. Our method is the best to approximate the result of spectral clustering.

numbers of clusters: 250, 500 and 1 000. We split the timing into two parts, one for the feature extraction process and the other for the assignment based on these features. We see that consistently the features extracted using random signal filtering are faster to compute than those requiring partial eigendecomposition. We also notice that until $k = 500$, k-means is an efficient method for the assignment of the points to the clusters, it is even 5 times faster than the compressive assignment for $k = 250$ in our experiment. However, when k becomes large, using the compressive method of CSC (also applied in CFEARS) is helping greatly to reduce the overall time of the computation, earning a factor two speedup between FEARS and CFEARS.

Next, we consider the efficiency of the clustering reported in Table 3.3, where two important observations stand out. On one hand, the best modularity is achieved using CSC and we see that our method, with the use of the compressive step, tends to similar results with increasing k. On the other hand, the adjusted Rand similarity index clearly shows that our method is assigning the nodes very similarly to SC. This is an expected behaviour since the goal of our method is to reconstruct the set of the k first eigenvectors used as features in SC.

3.4.4 Visualization

In these last experiments, we show how our method can be used in the context of high-dimensional data visualization, since eigenspaces are commonly used for dimensionality reduction in this context. We wish to see how our proposed method behaves first in a very

	$k = 250$	$k = 500$	$k = 1\,000$
SC	14.37min + 2.13h	25.09min + 14.96h	55.63min + 106.87h
FEARS	0.12min + 2.55h	0.19min + 22.75h	0.52min + 104.82h
CFEARS	0.12min + 11.36h	0.19min + 17.22h	0.52min + 58.46h
CSC	2.34min + 9.74h	3.73min + 21.07h	2.61min + 35.47h

Table 3.2 – **Timing of clustering for the Amazon data set.** All values represent one experiment and the order of the polynomial approximation is $m = 500$. Each experiment is split into two steps: the computation of the features (in minutes), and the assignment from the features to a cluster (in hours).

	SC	FEARS		CFEARS		CSC	
	mod	mod	rand	mod	rand	mod	rand
$k = 250$	0.344	0.387	0.884	0.588	0.711	0.764	0.509
$k = 500$	0.507	0.605	0.818	0.759	0.677	0.818	0.586
$k = 1\,000$	0.663	0.638	0.851	0.815	0.780	0.798	0.749

Table 3.3 – **Evaluation of the clustering for the Amazon data set.** All values represent one experiment and the order of the polynomial approximation is $m = 500$. The modularity score ([134]) is noted mod and the adjusted Rand similarity index ([84]) rand.

simple synthetic example and second for real-world data sets of larger size. For this task we compare FEARS to some dimensionality reduction algorithms presented in the previous chapter.

Toy example: the Swissroll

In this first small experiment, we wish to assess the validity of using our proposed method of eigenspace estimation for visualization on a simple toy example. We will compare the results obtained by our method only with Laplacian Eigenmaps as we would like to verify that we get similar results.

For this experiment, we use a traditional Swissroll graph for which we compute a 2 dimensional embedding. The Swissroll is computed by sampling its continuous manifold in the following way: given a set of randomly drawn angles θ in $[a\pi, b\pi]$ the coordinates are set using $x = \theta\cos(\theta)$, y drawn uniformly in $[0,1]$ and $z = \theta\sin(\theta)$. A kNN graph with 10 neighbours is constructed from the data points. For this experiment, the normalized Laplacian was used for all methods.

The resulting embeddings are shown in Figure 3.5. The colour map is a linear function of θ. The first thing to notice is that all embeddings are very smooth with respect to θ. The second interesting fact is that $\mathbf{B}$ indeed seems to be a good approximation of the Laplacian eigenmaps up to a rotation because they have very similar shapes. This tends to validate that the method indeed provides a good approximation of $\mathbf{U}_k$. In addition, in this specific

Figure 3.5 – **Visualization of the Swissroll point cloud.** The 3D point cloud of 10 000 points is shown in the top left corner. Its 2D embeddings using Laplacian eigenmaps is depicted in top-right position, our proposed fast eigenspace estimation method prior to the SVD step in the bottom left corner, and the result using FEARS is bottom right. The colour map is a linear function of the angle θ.

example, while embedding with $\mathbf{M}$ gives a smooth result, the normalization step provided by the SVD is necessary to get a good enough visualization. This observation makes sense since for visualization very few random signals are used to get $\mathbf{M}$, which, as discussed in Section 3.2.2, is not sufficient to have an expectation effect smoothing the variance on the eigenvalues. This scaling is normalized by the final SVD step, which is not costly for visualization (i.e. k is small).

Natural datasets

In this second experiment, we will consider the datasets presented in Appendix B and compare our method with t-SNE and LargeVis, the best performing methods evaluated in the previous chapter, as well as Laplacian Eigenmaps, which is expected to provide embeddings similar to FEARS.

LiveJournal	tsne	largevis	le	fears
time [s]	**150130.39**	**3283.26**	**OM**	**941.24**
aci	**0.35**	**0.36**	OM	**0.37**

Table 3.4 – 2D Embedding computation time. The default implementation of LargeVis uses parallelism. The value for Eigenmaps on LiveJournal is not reported because it exceeded the maximum memory available (128 GB).

The quality evaluation is done using the metrics presented in Chapter 2 which are reported in Tables 3.5-3.10. If we first look at the execution time in Table 3.5, FEARS is the fastest method on all considered datasets, with up to two orders of magnitude gain with respect to t-SNE and LargeVis on some instances. It is also faster than Laplacian Eigenmaps and handles large dataset more robustly due to its lower memory footprint. As we saw in the previous chapter, LargeVis presents a high base cost but scales well which puts it second for large datasets and last for smaller ones. Finally, t-SNE is consistently the slowest or second-slowest method presented here.

The global cluster quality is evaluated in Tables 3.6 and 3.7, which both give similar results. As expected, LargeVis and t-SNE clearly outperform FEARS and Laplacian Eigenmaps. Both eigenspace methods yield very similar scores with LE slightly better than FEARS quite consistently. The result is not surprising since our method only provides an approximated subspace. This analysis is confirmed by the unsupervised NN precision reported in Table 3.8.

The noise level is reported in Table 3.9 and globally concur with the above results : t-SNE and LargeVis provide the lowest numbers. However, this metric tends to show that FEARS is less prone to noise than Laplacian Eigenmaps, and that it even performs best on some datasets.

Finally, the ACC is reported in Table 3.10 and show similar charcteristics to what was presented in Chapter 2. That is, t-SNE and LargeVis have the highest values, with LargeVis performing slightly better, while both LE and FEARS have the best ACC scores.

A few selected 2D embeddings of Laplacian Eigenmaps and FEARS are shown in Figures 3.6-3.8. They show that visually, they are able to provide a reasonably good separation of the classes but are still quite limited as generic methods for visualization.

In addition, the results of the 2D embedding of the LiveJournal dataset are reported in Table 3.4. On this large dataset, FEARS is much faster than all other methods and provides a clusterability index close to t-SNE. Laplacian Eigenmaps was unable to terminate due to its bad memory scaleability. We do not provide the 2D embeddings for visual inspection due to an extreme imbalance in labels distribution for this dataset.

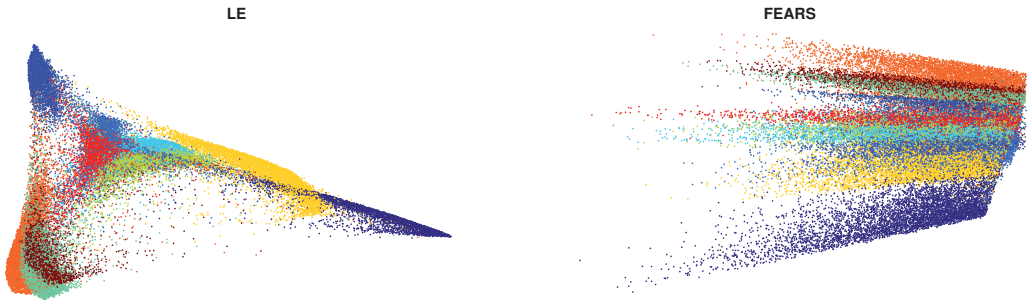

Figure 3.6 – Visualization of the MNIST ($N = 70000, D = 784$) dataset using Laplacian Eigenmaps and FEARS. The colour map corresponds to the labels. Zoomed-in versions are done by excluding points whose coordinates are in percentiles p and $(100 - p)$ of the data distribution.

3.5 Conclusion

In this chapter, we have presented a theoretical way to recover exactly the set of k smallest eigenvectors of a graph Laplacian. We have shown an accelerated algorithm for the approximation of the eigenspace of the Laplacian $\mathbf{L}$ solely based on Gaussian random signals filtering. We proved the bound on the number of signals to be as tight as possible. In addition, we proposed an accelerated eigenvalue estimation algorithm based on eigencount techniques. We presented different applications and compared the efficiency against the state of the art, showing the ability of our method to scale well with very large N. While usable for visualization tasks, we think that FEARS cannot be of much interest except if the speed is of paramount importance. Indeed, as all other eigenspace dimensionality reduction approaches, it does not provide visualization of a quality comparable to the state of the art. However, FEARS is well suited to provide good features for clustering tasks very efficiently.

Overall, we think that this chapter presents an interesting result for the field of graph signal processing and many further questions arise in this context. Among them, the design of the filter could be reconsidered. Could we gain even more efficiency by using a naturally polynomial function for the filter instead of the approximation of an ideal low-pass filter? We suggest using exponentially decreasing kernels, which are low-pass and infinitely differentiable and will assign to the eigenvalues an energy proportional to its position in the spectrum. One could wonder whether such design could allow stopping the computation before the SVD step.

Time [s]	tsne	largevis	le	fears
caltech101-caffenet	**89.52**	**219.00**	**6.85**	**1.54**
caltech256-caffenet	**469.09**	**256.87**	**186.26**	**9.15**
cifar10-cnn	**1410.91**	**315.21**	**OM**	**176.46**
cmupie	**130.22**	**219.71**	**14.83**	**3.56**
coil20	**10.82**	**216.03**	**0.42**	**0.34**
fma-echonest	**139.97**	**220.09**	**15.42**	**3.73**
fma-rosa	**2475.82**	**413.09**	**1882.70**	**71.20**
hiva	**949.41**	**279.98**	**703.64**	**91.47**
mnist	**1271.24**	**336.87**	**3448.48**	**38.71**
norb	**884.97**	**291.14**	**296.17**	**19.83**
sylva	**3336.83**	**490.75**	**OM**	**440.76**
usps	**94.30**	**215.55**	**9.58**	**1.96**
20newsgroup	**303.95**	**236.83**	**107.86**	**24.35**
nova	**310.05**	**233.49**	**100.53**	**36.31**

Table 3.5 – **Embedding time.** This table reports the embedding time (in seconds) for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

ACI	tsne	largevis	le	fears
caltech101-caffenet	**0.14**	**0.13**	**0.58**	**0.64**
caltech256-caffenet	**0.39**	**0.44**	**0.86**	**0.92**
cifar10-cnn	**0.28**	**0.27**	OM	**0.82**
cmupie	**0.42**	**0.65**	**0.94**	**0.94**
coil20	**0.06**	**0.05**	**0.12**	**0.15**
fma-echonest	**0.87**	**0.88**	**0.91**	**0.94**
fma-rosa	**0.92**	**0.93**	**0.96**	**0.97**
hiva	**0.62**	**0.65**	**0.77**	**0.91**
mnist	**0.05**	**0.06**	**0.37**	**0.40**
norb	**0.52**	**0.60**	**0.94**	**0.86**
sylva	**0.35**	**0.33**	OM	**0.99**
usps	**0.03**	**0.04**	**0.21**	**0.26**
20newsgroup	**0.25**	**0.26**	**0.79**	**0.97**
nova	**0.07**	**0.06**	**0.27**	**0.39**

Table 3.6 – **Embedding ACI.** This table reports the ACI score for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

1NN	tsne	largevis	le	fears
caltech101-caffenet	**0.18**	**0.19**	**0.57**	**0.63**
caltech256-caffenet	**0.47**	**0.53**	**0.85**	**0.91**
cifar10-cnn	**0.28**	**0.29**	OM	**0.74**
cmupie	**0.26**	**0.55**	**0.90**	**0.87**
coil20	**0.05**	**0.04**	**0.12**	**0.15**
fma-echonest	**0.70**	**0.71**	**0.74**	**0.75**
fma-rosa	**0.77**	**0.79**	**0.80**	**0.81**
hiva	**0.04**	**0.05**	**0.05**	**0.06**
mnist	**0.05**	**0.06**	**0.35**	**0.37**
norb	**0.38**	**0.46**	**0.74**	**0.69**
sylva	**0.05**	**0.04**	OM	**0.11**
usps	**0.04**	**0.05**	**0.23**	**0.26**
20newsgroup	**0.21**	**0.24**	**0.73**	**0.91**
nova	**0.02**	**0.02**	**0.07**	**0.11**

Table 3.7 – **Embedding 1NN.** This table reports the 1NN scores for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

NNP	tsne	largevis	le	fears
caltech101-caffenet	**0.35**	**0.36**	**0.21**	**0.20**
caltech256-caffenet	**0.22**	**0.22**	**0.09**	**0.06**
cifar10-cnn	**0.15**	**0.14**	OM	**0.02**
cmupie	**0.39**	**0.40**	**0.29**	**0.26**
coil20	**0.29**	**0.27**	**0.28**	**0.19**
fma-echonest	**0.38**	**0.38**	**0.14**	**0.15**
fma-rosa	**0.32**	**0.28**	**0.15**	**0.11**
hiva	**0.41**	**0.42**	**0.16**	**0.07**
mnist	**0.32**	**0.29**	**0.11**	**0.14**
norb	**0.33**	**0.3**	**0.13**	**0.15**
sylva	**0.11**	**0.11**	OM	**0.01**
usps	**0.49**	**0.49**	**0.28**	**0.30**
20newsgroup	**0.25**	**0.26**	**0.07**	**0.02**
nova	**0.20**	**0.21**	**0.08**	**0.05**

Table 3.8 – **Embedding NNP.** This table reports the NNP scores for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

ACN	tsne	largevis	le	fears
caltech101-caffenet	**0.20**	**0.21**	**0.37**	**0.35**
caltech256-caffenet	**0.36**	**0.37**	**0.51**	**0.48**
cifar10-cnn	**0.24**	**0.23**	OM	**0.40**
cmupie	**0.36**	**0.36**	**0.51**	**0.51**
coil20	**0.04**	**0.02**	**0.10**	**0.10**
fma-echonest	**0.37**	**0.38**	**0.38**	**0.36**
fma-rosa	**0.41**	**0.41**	**0.43**	**0.40**
hiva	**0.31**	**0.26**	**0.20**	**0.15**
mnist	**0.07**	**0.06**	**0.24**	**0.18**
norb	**0.34**	**0.30**	**0.37**	**0.32**
sylva	**0.02**	**0.02**	OM	**0.03**
usps	**0.06**	**0.06**	**0.21**	**0.15**
20newsgroup	**0.30**	**0.26**	**0.40**	**0.49**
nova	**0.32**	**0.06**	**0.15**	**0.17**

Table 3.9 – **Embedding ACN.** This table reports the ACN scores for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

ACC	tsne	largevis	le	fears
caltech101-caffenet	**4.61**	**2.87**	**0.04**	**0.32**
caltech256-caffenet	**6.80**	**4.72**	**0.11**	**0.76**
cifar10-cnn	**12.28**	**9.89**	OM	**0.15**
cmupie	**30.77**	**24.93**	**0.64**	**0.88**
coil20	**3.97**	**14.50**	**0.05**	**0.35**
fma-echonest	**34.97**	**26.62**	**0.77**	**0.48**
fma-rosa	**23.55**	**22.39**	**0.15**	**0.06**
hiva	**30.29**	**25.94**	**0.31**	**0.24**
mnist	**13.08**	**10.47**	**0.13**	**0.05**
norb	**26.93**	**34.66**	**0.21**	**0.23**
sylva	**21.08**	**28.99**	OM	**0.00**
usps	**13.67**	**10.75**	**0.13**	**0.14**
20newsgroup	**15.15**	**8.77**	**0.24**	**0.42**
nova	**29.75**	**19.75**	**0.29**	**0.47**

Table 3.10 – **Embedding ACC.** This table reports the ACC scores for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

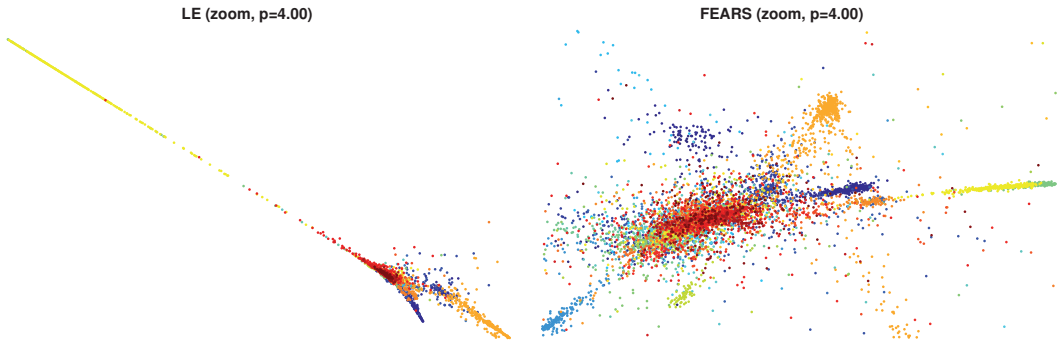

Figure 3.7 – Visualization of the Caltech101 ($N = 9145, D = 4096$) dataset using Laplacian Eigenmaps and FEARS. The colour map corresponds to the labels. Zoomed-in versions are done by excluding points whose coordinates are in percentiles p and $(100 - p)$ of the data distribution.

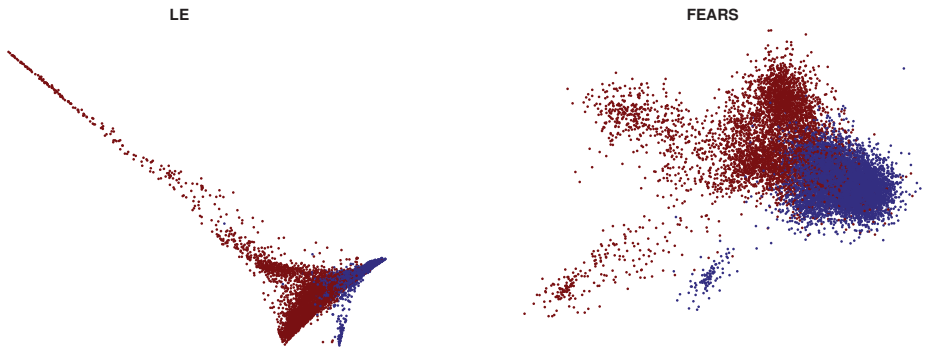

Figure 3.8 – Visualization of the NOVA ($N = 19466, D = 16969$) dataset using Laplacian Eigenmaps and FEARS. The colour map corresponds to the labels.

When one tugs at a single thing in nature,
he finds it attached to the rest of the world.
— John Muir

4 Compressive dimensionality reduction

In this chapter, we approach the scaleability of dimensionality reduction techniques in a very generic way and provide a method which can be used to accelerate any existing dimensionality reduction algorithm. The basic idea is to first assume that a subset of data points have been embedded and then see the complete embedding as a transductive learning problem, i.e. inferring information on a specific set of points provided with a few training examples.

Leveraging once more the GSP framework, we formulate this method as a graph compressive technique. More precisely, we start by building a kNN graph from the data, then we select a subset by sampling the nodes, apply any dimensionality reduction algorithm on the subset and finally extend the information to all nodes using transductive learning on the graph. Our first contribution is a graph sampling strategy based on the norm of the localization operator for which we establish theoretical bounds giving guarantees that the energy of the localized atoms on the samples is well distributed on the graph. Second, we propose a new distance on the graph using the localization operator that allows us to define a new transductive inference method on graphs. Finally, we show the efficiency of our method in experiments covering many natural datasets.

This chapter is organized as follows. Section 4.1 gives an overview of the related work on the subject. We give an outline of our contributions in Section 4.2. Then, in Section 4.3 we develop the results on our sampling method based on the energy of localized kernels. Section 4.4 describes the different methods to extend the information from the sampled nodes to all data points. Section 4.5 describes our proposed method for compressive embedding using the results from the previous sections. In Section 4.6, we show the validity and benefits of our method and compare it to the state-of-the-art through several experiments. Finally, Section 4.7 discusses open problems in the domain as well as potential future work to address.

This chapter extends the work presented in a paper written in collaboration with Nathanaël Perraudin and Pierre Vandergheynst [138] which introduced the first results on the subject. The additional content presented in this chapter focuses on the experiments.

4.1 Related work

The idea of compressive sampling, i.e., that assumptions made about the signals, such as smoothness, allow to go below the Shannon's theorem limit, has had great success in signal processing in recent years. See for example [36, 69] for surveys of the compressive sensing theory. To extend this theory to graphs, one need to combine smoothness priors to sampling and reconstruction schemes.

Transferring traditional sampling principles to graphs is, however, a non-trivial task, since regular sampling is not directly applicable except for very particular classes of graphs [131]. Because of this fact, the research has been focused on either irregular or random sampling.

Most of the works in irregular graph sampling have used the assumption that the signals to be sampled were k-bandlimited, meaning that their graph Fourier coefficients are non-zero only for the first k eigenvalues. Using this hypothesis, the existence of special sets, called uniqueness sets, have been defined in [144, 145]. Those sets are such that if two signals have identical values on the set, then they are equal on the complete set of vertices. Further works have shown that such sampling sets of size k always exist for k-bandlimited signals [3, 4, 43, 42]. The main issue with these definitions being either the need to compute the first k eigenvectors of the Laplacian [4] or the use of combinatorial searches [5] which are intractable for large graphs.

The other class of sampling which can be applied on graphs is random sampling. The simplest example of it is uniform sampling without replacement in the vertex set. It has the great advantage of being almost free to compute, but does not provide any guarantee for reconstruction. Nevertheless, it is still useful in practice since, as shown in [192], sampling $\mathcal{O}(k\log(k))$ vertices uniformly at random is sufficient to provide a clustering assignment equivalent to spectral clustering. In order to take the graph structure into account, variable density sampling of k-bandlimited signals inspired by traditional compressed sensing as been proposed in [44, 150]. They show in [150] that k-bandlimited signals are stably embedded using the graph weighted coherence and that $\mathcal{O}(k\log(k))$ samples are sufficient to ensure a stable reconstruction of the signals.

Various other sampling techniques have been proposed, such as ones focusing on multiresolution analysis [166, 191] or specific classes of graphs [43, 131].

Reconstructing the signal from the samples is intrinsically linked to the assumptions made on the signal. Since smoothness is the most common prior, Tikhonov regression is very often used (such as described in Section 1.2.3). It is used for example in [192] with an ℓ_2 fidelity term and in [150] with a fidelity term on the space of k-bandlimited signals. A more recent approach defines the signal reconstruction as the solution of an optimization problem using a Reproducible Kernel Hilbert Space defined on the graph [153].

Other schemes based on graph compressive sampling have been proposed in different works

such as multi-resolution image fusion based on compressed sensing and graph cuts [78], edge-signal reconstruction [97] and graph reconstruction [213, 37] or clustering [45] from random sketches.

4.2 Outline

In this chapter, we propose a general framework inspired by compressive sensing methods for accelerating any embedding algorithm. The first step is to create a kNN graph encoding the input data similarity. Then, using graph sampling techniques we select a subset of the data on which to apply an embedding algorithm. Finally, we diffuse the embedding from the sketch to all data using transductive inference on the graph. The two crucial points are the need to provide a sampling scheme along with a reconstruction method which guarantees that all datapoints will be considered.

Our main contributions are,

- an adaptive sampling scheme based on the norm of the localization operator,
- theoretical guarantees that provide the number of samples needed to allow the diffusion of the information from the samples to any other point,
- a transductive diffusion method which uses distances based on localized filters for accurate reconstruction.

We call this complete scheme Compressive Dimensionality Reduction (CDR).

4.3 Random sampling on graphs

In this section, we first define a graph sampling scheme based on the norm of the localization operator and then prove related theoretical limits. In particular, we try to define the number of samples needed in order to diffuse energy on every node by localizing filters on the samples. We will prove that the number of samples needed is directly linked with the rank of the filter (i.e., the number of eigenvalues on which it is non-zero).

The results we provide in this section are similar to those of [150], except that we do not assume that we deal with k-bandlimited signals but instead formulate more general assumptions using only the sampling kernel.

In addition, let us emphasize that we do not plan to use this sampling strategy to actually sample complete signals, but only to create a compressed signal on the sampled nodes. Therefore, the scheme discussed here needs to be designed so that any reconstructed point can be affected by a few samples. This does not have any direct consequence on the sampling procedure in itself but gives us the motivation for the theoretical limits we will prove.

Inspired by the works on data-adapted priors and stationarity [140], we can see the sampling

kernel as the expected PSD content of the signal we want to reconstruct. Adopting this point of view, we want the samples to give enough information about all points so that the reconstructed signal spectrum is close to its expected PSD.

4.3.1 Adaptive sampling scheme

As we have seen in the previous chapters, the ℓ_2-norm of the localization operator is related to the structure of the graph (see Section 1.2.1 and 2.2.2). We thus want to use it to create a random sampling strategy which adapts to the graph, while enforcing our data prior by a careful selection of the kernel g.

In this context, the sampling strategy we would like to use should sample few points in very connected areas of the graph, since information can diffuse very efficiently. Inversely, we need to sample more points in areas which are less connected since it is difficult to infer information on nodes which have few neighbours. Indeed, let us take the extreme example of an isolated (i.e. disconnected) node, a sampling strategy must sample it because no information can be diffused from other nodes. The norm of $\mathcal{T}_i g$ will help us achieve such a sampling strategy, since their value is related to the connectivity (see Figure 2.2 or [143, Figure 7]). For example, in the case of an isolated node, $\mathcal{T}_i g$ will be entirely localized on the node.

Based on this idea, we propose the following adaptive sampling for which we define the probability distribution $\mathcal{P}$ represented by a vector $\boldsymbol{p} \in \mathbb{R}^N$, given by

$$\boldsymbol{p}_i = \frac{\left\|\mathcal{T}_i g\right\|_2^2}{\left\|g(\boldsymbol{\lambda})\right\|_2^2}. \tag{4.1}$$

Remember that we have $\|\mathcal{T}_i g\|_2^2 = \sum_j \sum_\ell g(\lambda_\ell) \mathbf{u}_\ell^*[i] \mathbf{u}_\ell[j]$ and $\|g(\boldsymbol{\lambda})\|_2^2 = \sum_i \|\mathcal{T}_i g\|_2^2$, implying that $\sum_i \boldsymbol{p}_i = 1$. Let us associate to $\boldsymbol{p}$ the matrix

$$\mathbf{P} = \operatorname{diag}(\boldsymbol{p}) \in \mathbb{R}^{N \times N}. \tag{4.2}$$

Using this probability distribution, we draw independently (with replacement) N_s indices $\Omega = \{\omega_1, \dots, \omega_{N_s}\}$ from the set $\{1, \dots, N\}$ according to the probability distribution $\boldsymbol{p}$. We have

$$\mathbb{P}(\omega_j = i) = \boldsymbol{p}_i, \quad \forall i \in \{1, \dots, N\}, \quad \forall j \in \{1, \dots, N_s\}. \tag{4.3}$$

For any signal $\boldsymbol{x} \in \mathbb{R}^N$, defined on the vertices of the graph, its sampled version $\boldsymbol{y} \in \mathbb{R}^{N_s}$ satisfies

$$\boldsymbol{y}_j = \boldsymbol{x}_{\omega_j} \quad \forall j \in \{1, \dots, N_s\}. \tag{4.4}$$

Finally, the downsampling matrix $\mathbf{M} \in \mathbb{R}^{N \times N_s}$ is defined as

$$\mathbf{M}_{ij} = \begin{cases} 1 & \text{if } i = \omega_j \\ 0 & \text{otherwise,} \end{cases} \tag{4.5}$$

for all $i \in \{1, \dots, N\}$ and $j \in \{1, \dots, N_s\}$. Note that $\boldsymbol{y} = \mathbf{M}\boldsymbol{x}$.

4.3.2 Embedding Theorems

In order to simplify the notation, we slightly abuse one of the established symbols in this section. Until now, the matrix $\mathbf{U}_k$ was defined as the matrix containing the first k columns of $\mathbf{U}$. In this section, given a kernel g, we define $\mathbf{U}_k$ as a $N \times k$ matrix made of the k columns of $\mathbf{U}$ where $g(\lambda_\ell) \neq 0$. Similarly, we denote $\mathbf{\Lambda}_k$ the $k \times k$ diagonal matrix containing the associated eigenvalues. Also note that we have

$$g(\mathbf{L}) = \mathbf{U} g(\mathbf{\Lambda}) \mathbf{U}^* = \mathbf{U}_k g(\mathbf{\Lambda}_k) \mathbf{U}_k^* = \mathbf{U}_k \mathbf{U}_k^* g(\mathbf{L}). \tag{4.6}$$

Now that our sampling strategy is defined, we want to find the theoretical limits for which the residual energy after sampling spans all vertices. More precisely, we want that the random projection $\mathbf{M}\mathbf{P}^{-\frac{1}{2}} g(\mathbf{L})\mathbf{x}$ preserves as much of the original energy $g(\mathbf{L})\mathbf{x}$ as possible. The first theorem shows that given enough samples, we indeed have a conservation between the two quantities. In this sense, given enough samples, $\mathbf{M}\mathbf{P}^{-\frac{1}{2}} g(\mathbf{L})\mathbf{x}$ is a good embedding of $g(\mathbf{L})\mathbf{x}$.

Theorem 4. *Given a graph $\mathcal{G}$ and a kernel g with a given rank $\|g(\boldsymbol{\lambda})\|_0 = k$, given $\delta > 0$ and using the sampling scheme of Section 4.3.1, if*

$$N_s \geq 2 \frac{1}{\delta^2} \frac{\|g(\boldsymbol{\lambda})\|_2^2}{\|g(\boldsymbol{\lambda})\|_\infty^2} \left(1 + \frac{\delta}{3}\right) \log\left(\frac{2k}{\epsilon}\right)$$

we have with, probability $1 - \epsilon$, for all $\mathbf{x}$:

$$\left| \frac{\frac{1}{M} \left\| \mathbf{M}\mathbf{P}^{-\frac{1}{2}} g(\mathbf{L})\mathbf{x} \right\|_2^2 - \|g(\mathbf{L})\mathbf{x}\|_2^2}{\|g(\boldsymbol{\lambda})\|_\infty^2} \right| \leq \delta \|\mathbf{U}_k^* \mathbf{x}\|_2^2 \leq \delta \|\mathbf{x}\|_2^2. \tag{4.7}$$

Note that the above expression is normalized by $\|g(\boldsymbol{\lambda})\|_\infty^2$ in order to remove the scaling factor of the kernel g. The proof of this theorem is given in Appendix A.4.1.

Let us now analyse the most important term of the bound:

$$\frac{\|g(\boldsymbol{\lambda})\|_2^2}{\|g(\boldsymbol{\lambda})\|_\infty^2} = \frac{\sum_\ell g^2(\lambda_\ell)}{\max_\ell g^2(\lambda_\ell)}. \tag{4.8}$$

It is a measure of concentration of the kernel on its support. It is maximized with the value

$\frac{\|g(\boldsymbol{\lambda})\|_2^2}{\|g(\boldsymbol{\lambda})\|_\infty^2} = k$ when g is a rectangle. In general, it will be small for concentrated kernels. For example, a rapidly decreasing kernel such as the heat kernel ($g(\lambda) = e^{-\lambda\tau}$) will lead to a very small ratio.

Note that contrarily to other bounds available in the literature this bound does not require the kernel to be low rank but only concentrated. For a comparison [150, Corollary 2.3] requires

$$N_s \geq \frac{3}{\delta^2} k \log\left(\frac{2k}{\epsilon}\right)$$

for k-bandlimited signals.

Although Theorem 4 already gives an interesting result, it states that the overall energy is well preserved by the sampling procedure. We seek a more precise bound on how much energy is present for every node. Thus, building on top of Theorem 4, we establish a lower bound on the number of samples required to have some residual energy at every node. In other words, we want to capture enough information from each node with a given confidence level. It will ensure that the information diffused from the samples can reach all other vertices.

Theorem 5. *Using the sampling scheme described in Section 4.3.1, for $\delta > 0$, a graph $\mathcal{G}$ and a kernel g such that $\|g(\boldsymbol{\lambda})\|_0 = k$, each node i is guaranteed to have, with probability $1-\epsilon$,*

$$\frac{\frac{1}{M}\left\|\mathbf{M}\mathbf{P}^{-\frac{1}{2}}\mathcal{T}_i g\right\|_2^2}{\|\mathcal{T}_i g\|_2^2} \geq 1-\delta,$$

given that the number of samples satisfies

$$N_s \geq \frac{2a}{\delta^2}\left(1+\frac{\delta}{3}\right)\log\left(\frac{k}{\epsilon}\right),$$

where $a = \frac{\|g(\boldsymbol{\lambda})\|_2^2 \|g(\boldsymbol{\lambda})\|_\infty^2 \|\mathbf{U}_k^* \boldsymbol{\delta}_i\|_2^4}{\|\mathcal{T}_i g\|_2^4}$.

This theorem states that given enough samples N_s, atoms localized at the sampled nodes capture with some probability $1-\epsilon$ (close to 1), at least a good percentage of the energy at node i. The proof of this theorem is given in Appendix A.4.2.

Note that the factor a is always greater than 1 and varies depending on the shape of the kernel g and of the graph eigenvectors. However it is $\mathcal{O}(k)$ and exactly equal to k if g is a rectangular kernel. Indeed, a simple transformation shows that

$$a = \frac{\|g(\boldsymbol{\lambda})\|_2^2 \|g(\boldsymbol{\lambda})\|_\infty^2 \|\mathbf{U}_k^* \boldsymbol{\delta}_i\|_2^4}{\|\mathcal{T}_i g\|_2^4} = \frac{\sum_\ell g^2(\lambda_\ell)}{\max_\ell |g^2(\lambda_\ell)|}\left(\frac{\max_\ell |g^2(\lambda_\ell)| \sum_{\ell \in \mathcal{K}} \boldsymbol{u}_\ell^2[i]}{\sum_\ell g^2(\lambda_\ell)\boldsymbol{u}_\ell^2[i]}\right)^2.$$

The first term is smaller than k but is usually close to k for a kernel close to a rectangle. The second term is greater than 1 but close to 1 given that the kernel is close to a rectangle.

Note that this bound becomes loose if the kernel g has a large rank because of the term $\sum_{\ell \in \mathcal{K}} \boldsymbol{u}_\ell^2[i]$. To cope with this issue we can use another kernel g' that is a low-rank approximation of g, which is stated in the following theorem.

Theorem 6. *Given a graph $\mathcal{G}$, let g' (with $\left\|g'(\boldsymbol{\lambda})\right\|_0 = k$) to be the rank k approximation of the kernel g, i.e.,*

$$g(\lambda_\ell) = \begin{cases} g'(\lambda_\ell) & \text{for the the } k \text{ greatest values of } |g(\lambda_\ell)| \\ 0 & \text{otherwise.} \end{cases}$$

Using the sampling scheme described in Section 4.3.1 with the kernel g, for $\delta > 0$, each node i is assured with a probability $1-\epsilon$ to have

$$\frac{\frac{1}{N_s}\left\|\mathbf{M}\mathbf{P}^{\frac{1}{2}}\mathcal{T}_i g\right\|_2^2}{\left\|\mathcal{T}_i g\right\|_2^2} \geq 1 - \delta - \frac{\left\|\mathcal{T}_i\left(|g'| - |g|\right)\right\|_2^2}{\left\|\mathcal{T}_i g\right\|_2^2} \tag{4.9}$$

providing the number of samples satisfies[1]

$$N_s \geq 2\frac{1}{\delta^2}\frac{\left\|g'(\boldsymbol{\lambda})\right\|_2^2 \left\|g'(\boldsymbol{\lambda})\right\|_\infty^2 \left\|\mathbf{U}_k^* \boldsymbol{\delta}_i\right\|_2^4}{\left\|\mathcal{T}_i g\right\|_2^4}\left(1+\frac{\delta}{3}\right)\log\left(\frac{k}{\epsilon}\right). \tag{4.10}$$

Using Theorem 6 instead of Theorem 5 for high rank kernels, the number of samples N_s required can be highly reduced. Indeed, when the kernel g is well concentrated but not low rank, we trade some approximation error encoded by $\frac{\left\|\mathcal{T}_i\left(|g'|-|g|\right)\right\|_2^2}{\left\|\mathcal{T}_i g\right\|_2^2}$ (which is low if g is concentrated) but we need a smaller number of samples due to the fact that g' is low rank. This theorem can be interesting for a heat kernel for example.

4.4 Graph transductive learning

Now that we have established a sampling strategy, we want, in this section, to cast the problem of diffusing the information obtained on the samples in a transductive inference framework. In this setting, we observe a label field or signal $\boldsymbol{x}$ only at a subset of vertices $S \subset V$, determined by the sampling procedure, i.e $\boldsymbol{y}_i = \boldsymbol{x}[i]$, $\forall i \in S$, with $\boldsymbol{y}$ being the observed signal. The goal of transductive learning is to predict the missing signal/labels using both the observed signal and the remaining data points.

[1] Note that $\left\|\mathcal{T}_i g\right\|_2^2 \geq \left\|\mathcal{T}_i g'\right\|_2^2$.

4.4.1 Global graph diffusion

Solutions of transductive inference using graphs can be computed in a number of ways, for example using Tikhonov regression:

$$\arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \mathbf{M}\boldsymbol{x}\|_2^2 + \mu \boldsymbol{x}^t \mathbf{L} \boldsymbol{x}, \tag{4.11}$$

where $\mathbf{M}$ is the sampling operator and $\mathbf{L}$ the graph Laplacian. An alternative to the use of the Dirichlet smoothness constraint is to use graph Total Variation (TV). The regression would thus become :

$$\arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \mathbf{M}\boldsymbol{x}\|_2^2 + \mu \|\nabla_{\mathcal{G}} \boldsymbol{x}\|_1 \tag{4.12}$$

with $\nabla_{\mathcal{G}} \boldsymbol{x} = \left(\sqrt{\mathbf{W}_{i,j}}(\boldsymbol{x}[i] - \boldsymbol{x}[j])\right), \forall (v_i, v_j) \in \mathcal{E}$.

For large scale learning, solving the optimization problems as described above is easy for Tikhonov regression (see Section 1.2.3) but can be too expensive for TV and one typically uses accelerated descent methods. Another issue is that both approaches simply penalize some predefined sets of frequencies, i.e. the regularization in Eq. (4.11) simply penalizes the frequency with the function $g(\lambda) = \lambda$ and does not adapt to the data.

4.4.2 Reproducible Kernel Hilbert Space on Graphs

As we briefly stated in the introduction, an approach to include an aspect of data adaptation to the reconstruction can be to use methods based on general kernels and then set the kernels to the expected PSD of the signals we want to reconstruct. As a first attempt in this direction, we evaluate a kernel-based reconstruction method using Reproducible Kernel Hilbert Space on graphs.

What we want is to replace the smoothness term arising in Eq. (4.11) by constraining the solution to belong to the finite dimensional Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_{\mathbf{G}}$ corresponding to the graph kernel $\mathbf{G} = g(\mathbf{L})$, for some filter g. In this case, we solve the following problem:

$$\arg\min_{\boldsymbol{x} \in \mathcal{H}_{\mathbf{G}}} \|\boldsymbol{y} - \mathbf{M}\boldsymbol{x}\|_2^2. \tag{4.13}$$

In this section, we formulate transductive learning as a finite dimensional regression problem. This problem is solved by constructing a Reproducing Kernel Hilbert Space (RKHS) from a graph filter, which controls the expected spectral content of the solution and provides a fast algorithm to compute it. A similar RKHS construction for graphs has been proposed in [153].

An empirical RKHS for graphs

Let g be a smooth, strictly positive function defining a graph filter using the following matrix:

$$\mathbf{G}_{i,j} = g(\mathbf{L})\delta_{i,j} = \mathcal{T}_i g[j], \tag{4.14}$$

where $\mathcal{T}_i$ is the localisation operator at vertex i. Let us analyse $\mathbf{G}$ using its spectral representation

$$\begin{aligned}\mathbf{G} &= \mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^* \\ &= \mathbf{U}g(\mathbf{\Lambda})^{1/2}\big(\mathbf{U}g(\mathbf{\Lambda})^{1/2}\big)^*.\end{aligned}$$

Let $\boldsymbol{r}_i$ be the i-th row of $\mathbf{U}g(\mathbf{\Lambda})^{1/2}$, we immediately see that $\boldsymbol{r}_i^T \boldsymbol{r}_j = \mathbf{G}_{i,j}$. More explicitly, these vectors are written in terms of the kernel g:

$$\boldsymbol{r}_i[j] = \sum_\ell \sqrt{g(\boldsymbol{\lambda}_\ell)}\boldsymbol{u}_\ell[i]\boldsymbol{u}_\ell[j]. \tag{4.15}$$

These expressions suggest to define the Hilbert space $\mathcal{H}_\mathbf{G}$ as the closure of all linear combinations of localized graph filters $\mathcal{T}_i g$. Therefore, this space is composed of signals (or functions) of the form:

$$\boldsymbol{x} = \sum_{k\in\mathcal{V}} \alpha_k \mathcal{T}_k g, \tag{4.16}$$

with $\alpha_k \in \mathbb{R}$. Note that any $\boldsymbol{x} \in \mathcal{H}_\mathbf{G}$ has a well-defined graph Fourier transform:

$$\hat{\boldsymbol{x}}(\ell) = g(\boldsymbol{\lambda}_\ell) \sum_{k\in\mathcal{V}} \alpha_k \boldsymbol{u}_\ell[k]. \tag{4.17}$$

This allows us to equip $\mathcal{H}_\mathbf{G}$ with the following scalar product:

$$\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathcal{H}_\mathbf{G}} = \sum_\ell \frac{1}{g(\boldsymbol{\lambda}_\ell)} \hat{\boldsymbol{x}}(\ell)^* \hat{\boldsymbol{y}}(\ell). \tag{4.18}$$

Using this definition, we see that the vectors r_i form an orthonormal basis of $\mathcal{H}_\mathbf{G}$:

$$\begin{aligned}\langle \boldsymbol{r}_i, \boldsymbol{r}_j\rangle_{\mathcal{H}_\mathbf{G}} &= \sum_\ell \frac{1}{g(\boldsymbol{\lambda}_\ell)} \sqrt{g(\boldsymbol{\lambda}_\ell)}\boldsymbol{u}_\ell[i]^* \sqrt{g(\boldsymbol{\lambda}_\ell)}\boldsymbol{u}_\ell[j] \\ &= \sum_\ell \boldsymbol{u}_\ell[i]^* \boldsymbol{u}_\ell[j] \\ &= \delta_{i,j}.\end{aligned}$$

Let us now see that $\mathcal{H}_{\mathbf{G}}$ is a RKHS. First, we show that the scalar product with a $\mathcal{T}_i g$ in $\mathcal{H}_{\mathbf{G}}$ is the evaluation functional at vertex i:

$$\begin{aligned}\langle \mathcal{T}_i g, \mathcal{T}_j g \rangle_{\mathcal{H}_{\mathbf{G}}} &= \sum_{\ell} \frac{1}{g(\lambda_\ell)} g(\lambda_\ell)^2 u_\ell[i]^* u_\ell[j] \\ &= \mathcal{T}_i g[j].\end{aligned}$$

Then, by linearity of the scalar product and the definition of $\mathcal{H}_{\mathbf{G}}$ in Eq (4.16) we have:

$$\begin{aligned}\langle \mathcal{T}_i g, \boldsymbol{x} \rangle_{\mathcal{H}_{\mathbf{G}}} &= \sum_{k \in \mathcal{V}} \alpha_k \langle \mathcal{T}_i g, \mathcal{T}_k g \rangle_{\mathcal{H}_{\mathbf{G}}} \\ &= \sum_{k \in \mathcal{V}} \alpha_k \mathcal{T}_k g[i] \\ &= \boldsymbol{x}[i].\end{aligned}$$

Finally, for any $\boldsymbol{x} \in \mathcal{H}_{\mathbf{G}}$, $\boldsymbol{x} = \sum_{k \in \mathcal{V}} \beta_k \mathcal{T}_k g$, we have the following explicit form of their norm :

$$\begin{aligned}\|\boldsymbol{x}\|^2_{\mathcal{H}_{\mathbf{G}}} &= \langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\mathcal{H}_{\mathbf{G}}} \\ &= \sum_{\ell} \frac{1}{g(\boldsymbol{\lambda}_\ell)} g(\boldsymbol{\lambda}_\ell)^2 \sum_{i,j \in \mathcal{V}} \beta_i \beta_j^* \boldsymbol{u}_\ell[i] \boldsymbol{u}_\ell[j]^* \\ &= \sum_{i,j \in \mathcal{V}} \beta_i \beta_j^* \big(\sum_{\ell} g(\boldsymbol{\lambda}_\ell) \boldsymbol{u}_\ell[i] \boldsymbol{u}_\ell[j]^* \big) \\ &= \sum_{i,j \in \mathcal{V}} \beta_i \mathbf{G}[i,j] \beta_j^* \\ &= \beta^T \mathbf{G} \beta.\end{aligned}$$

Transductive learning with a RKHS

Now that we have established $\mathcal{H}_{\mathbf{G}}$ as a valid RKHS, we will seek to recover the full signal by solving the following problem :

$$\tilde{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in \mathcal{H}_{\mathbf{G}}} \sum_{k \in S} E(\boldsymbol{y}_k, \boldsymbol{x}[k]) + \mu \|\boldsymbol{x}\|_{\mathcal{H}_{\mathbf{G}}}, \tag{4.19}$$

where $E(.,.)$ is a loss function. Let us first decompose $\mathcal{H}_{\mathbf{G}} = \mathcal{H}_S \oplus \mathcal{H}_S^{\perp}$, where

$$\mathcal{H}_S = \left\{ \boldsymbol{x} \in \mathcal{H}_{\mathbf{G}} \text{ s.t. } \boldsymbol{x} = \sum_{k \in S} \alpha_k \mathcal{T}_k g \right\}, \tag{4.20}$$

and let us note that, for any $\boldsymbol{x} \in \mathcal{H}_S$,

$$\begin{aligned}\|\boldsymbol{x}\|^2_{\mathcal{H}_\mathbf{G}} &= \sum_\ell g(\boldsymbol{\lambda}_\ell) \sum_{i,j\in S} \alpha_i \alpha_j^* \boldsymbol{u}_\ell[i] \boldsymbol{u}_\ell[j]^* \\ &= \sum_{i,j\in S} \alpha_i \alpha_j^* \sum_\ell g(\boldsymbol{\lambda}_\ell) \boldsymbol{u}_\ell[i] \boldsymbol{u}_\ell[j]^* \\ &= \alpha^T \mathbf{K} \alpha\end{aligned}$$

where $\mathbf{K}[i,j] = \mathbf{G}[i,j]$, $i,j \in S$, is positive definite since it is a principal submatrix of a positive definite matrix.

Let $\boldsymbol{x} \in \mathcal{H}_\mathbf{G}$ be decomposed as $\boldsymbol{x} = \boldsymbol{x}_S + \boldsymbol{x}_{S^\perp}$, where $\boldsymbol{x}_S$ (resp. $\boldsymbol{x}_{S^\perp}$) is the orthogonal projection of $\boldsymbol{x}$ on $\mathcal{H}_S$ (resp. $\mathcal{H}_S^\perp$). Now it is immediate to check that :

$$\begin{aligned}\langle \mathcal{T}_k g, \boldsymbol{x}_{S^\perp} \rangle_{\mathcal{H}_\mathbf{G}} &= \boldsymbol{x}_{S^\perp}[k] \\ &= 0, \forall k \in S.\end{aligned}$$

Inserting this relationship back into Eq. (4.19), we see that :

$$\sum_{k\in S} E(\boldsymbol{y}_k, \boldsymbol{x}_S[k] + \boldsymbol{x}_{S^\perp}[k]) + \boldsymbol{\lambda} \|\boldsymbol{x}_S + \boldsymbol{x}_{S^\perp}\|^2_{\mathcal{H}_\mathbf{G}} \geq \sum_{k\in S} E(\boldsymbol{y}_k, \boldsymbol{x}_S[k]) + \lambda \|\boldsymbol{x}_S\|^2_{\mathcal{H}_\mathbf{G}}, \tag{4.21}$$

since $\boldsymbol{x}_{S^\perp}[k] = 0 \; \forall k \in S$ and adding $\boldsymbol{x}_{S^\perp}$ can only increase the norm of $\boldsymbol{x}_S$ in $\mathcal{H}_\mathbf{G}$. This shows that the minimizer of Eq. (4.19) is in $\mathcal{H}_S$ and therefore of the form

$$\tilde{\boldsymbol{x}} = \sum_{k\in S} \beta_k \mathcal{T}_k g \tag{4.22}$$

for some coefficients β_k. Moreover since $\|\tilde{\boldsymbol{x}}\|_{\mathcal{H}_\mathbf{G}} = \beta^T \mathbf{K} \beta$, we can rewrite Eq. (4.19) as a minimization only on those coefficients with $\tilde{\boldsymbol{x}} = \mathbf{K}\tilde{\beta}$ and

$$\tilde{\beta} = \arg\min_\beta \sum_k E(\boldsymbol{y}_k, (\mathbf{K}\beta)[k]) + \mu \beta^T \mathbf{K} \beta. \tag{4.23}$$

Finally, we observe that the recovered signal can be computed by filtering a stream of Kronecker deltas located at the observed values and weighted by the optimal coefficients computed in Eq. (4.23) :

$$\tilde{\boldsymbol{x}} = g(\mathbf{L}) \left\{ \sum_{k\in S} \tilde{\beta}_k \delta_k \right\}. \tag{4.24}$$

To summarize, in the case of the squared loss function $L(a,b) = (a-b)^2$, the transductive solution is given by first computing the optimal coefficients $\tilde{\beta} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \boldsymbol{y}$ and then computing the regression $\tilde{\boldsymbol{x}} = g(\mathbf{L}) \{\sum_{k\in S} \tilde{\beta}_k \delta_k\}$.

Note that in traditional ridge regression, the last step is usually given in terms of an explicit kernel that is easy to evaluate. In our case, this expression is also available from (4.24):

$$\begin{aligned}\tilde{\boldsymbol{x}}[i] &= \sum_{k\in S} \tilde{\beta}_k \mathbf{G}_{i,k} \\ &= \sum_{k\in S} \tilde{\beta}_k \mathbf{G}_{k,i} \\ &= \sum_{k\in S} \tilde{\beta}_k \mathcal{T}_k g[i]\end{aligned}$$

and, while the kernel does not have a simple analytical form, the sum can be efficiently computed via a graph filtering algorithm. In particular, it is sufficient to perform $|S|$ filterings to get $\mathcal{T}_k g, \forall k \in S$.

4.4.3 Convex hull diffusion

If we want to cast the general problem of transductive learning in a simpler framework, we can restrict ourselves to linear solutions of the form $\tilde{\mathbf{x}} = \mathbf{A}\mathbf{y}$. This means finding the coefficients such as :

$$\tilde{\mathbf{x}}[i] = \sum_{k\in S} \alpha_{i,k} \mathbf{y}_k, \tag{4.25}$$

with $\alpha_{i,k} = \mathbf{A}_{i,k}$.

In the previous section, we just saw how a RKHS built using a graph filter g allowed to weight the contributions of localized filters centred on a subset S of vertices. Writing the answer as a linear solution such as defined in (4.25) would give the following coefficients :

$$\alpha_{i,k} = \frac{\tilde{\beta}_k \mathcal{T}_k g[i]}{\mathbf{y}_k}. \tag{4.26}$$

Of course, this is a degenerate solution since the coefficients are normalized by $\boldsymbol{y}_k$ and the optimal coefficients already contain the information from $\boldsymbol{y}$. Nevertheless, coming back to the idea the the transductive inference should be kernel-based, we will derive a linear solution of the form of Eq. (4.25) using localized filters.

4.4.4 Localized Kernel Distance

Since localized filters are proven to be concentrated in the vertex domain (see [168, Theorem 1]), it seems natural to use them to get correlations between nodes directly. To this end, we introduce the Localized Kernel Distance (LKD), which we define as :

$$\mathrm{LKD}(i,j) = 1 - \frac{\mathcal{T}_i g^2[j]}{\|\mathcal{T}_i g\| \|\mathcal{T}_j g\|}. \tag{4.27}$$

Using Lemma 3, we know that we can derive an alternative form of Eq. (4.27) :

$$\mathrm{LKD}(x,y) = 1 - \frac{\mathcal{T}_i g^2[j]}{\|\mathcal{T}_i g\|\|\mathcal{T}_j g\|} = 1 - \frac{\langle \mathcal{T}_x g, \mathcal{T}_y g\rangle}{\|\mathcal{T}_x g\|\|\mathcal{T}_y g\|}. \tag{4.28}$$

Let us now examine its properties by stating the following theorem.

Theorem 7. *The space* $(\mathcal{V}, \mathrm{LKD})$ *with* $\mathcal{V}$ *the vertex set of a graph and* LKD *as defined in Eq.* (4.27) *is a pseudosemimetric space, that is, for every* $x, y \in \mathcal{V}$*:*

1. $\mathrm{LKD}(x,y) \geq 0$
2. $\mathrm{LKD}(x,x) = 0$
3. $\mathrm{LKD}(x,y) = \mathrm{LKD}(y,x)$

The proof of this theorem is given in Appendix A.2.2. Now that we proved that the LKD is a pseudosemi-metric, we need only to prove that $\mathrm{LKD}(x,y) = 0$ implies $x = y$ to make it a semi-metric. This can be done by using an additional hypothesis on g and it is formulated in the following theorem.

Theorem 8. *The space* $(\mathcal{V}, \mathrm{LKD})$ *with* $\mathcal{V}$ *the vertex set of a graph and* LKD *as defined in Eq.* (4.27)*, with* $g(\mathbf{L})$ *full-rank (i.e.* $g(\lambda) > 0, \forall \lambda$*, is a semimetric space, that is, for every* $x, y \in \mathcal{V}$*:*

1. $\mathrm{LKD}(x,y) \geq 0$
2. $\mathrm{LKD}(x,y) = 0 \Leftrightarrow x = y$
3. $\mathrm{LKD}(x,y) = \mathrm{LKD}(y,x)$

The proof of this theorem is given in Appendix A.2.2.

Convex Hull Diffusion

In this section we propose to use a notion of distances to the samples $\boldsymbol{y}$ to set the coefficients. More formally, by setting $\alpha_{i,k} \simeq d(x_i, y_k)$ for some distance function d in Eq. (4.25). Here, quite naturally, we propose to make use of the LKD we just defined. Since the coefficients $\alpha_{i,k}$ need to encode similarity between i and k, a reasonable choice is to set :

$$\alpha_{i,k} = 1 - \mathrm{LKD}(i,k) = \frac{\mathcal{T}_i g^2[j]}{\|\mathcal{T}_i g\|\|\mathcal{T}_j g\|}. \tag{4.29}$$

Using this definition, we know that the coefficients $\alpha_{i,k}$ have good properties derived from Theorems 7 and 8. First, since the LKD has values in $[0,1]$, the coefficients will also have values in this range. Second, $\alpha_{i,k} = \alpha_{k,i}$ which means that $\mathbf{A}$ is symmetric, square and non-negative. Finally, for any kernel g we have $\alpha_{i,i} = 1$ and, if we restrict ourselves to kernels as defined in Theorem 8, we have $\alpha_{i,j} = 0$ if and only if $i = j$. In general, we have the good property that the

coefficients $\alpha_{i,k}$ will be small if the vertices i and k are far apart on the graph and big if they are close.

In addition, knowing that a classical problem related to embedding data in low dimension, and more specifically to data visualization, is a concentration around zero, we wish to devise a method to prevent this effect from happening. It is reasonable to suppose that the problem of concentration is often related to a lack of information about some points or an absence of normalization. For example, if we take the linear combination as defined in Eq. (4.25), this could happen if for some i, all the coefficients $\alpha_{i,k}$ are small.

In order to avoid this problem, we propose to use a normalized version $\tilde{\mathbf{A}}$ of $\mathbf{A}$ that maps the points $\mathbf{x}$ in the convex hull of $\mathbf{y}$. This is done simply by normalizing each line of $\mathbf{A}$, that is :

$$\tilde{\alpha}_{i,k} = \frac{\alpha_{i,k}}{\sum_{k \in S} \alpha_{i,k}}, \tag{4.30}$$

with $\tilde{\alpha}_{i,k} = \tilde{\mathbf{A}}_{i,k}$.

4.5 Compressive Dimensionality Reduction

Building on what has been presented in the previous sections, we now propose our main contribution, a Compressive Dimensionality Reduction (CDR) scheme described in Algorithm 8. In the following, $\mathbf{X}$ denotes the original $N \times D$ data matrix, $\mathbf{Z}$ the high-dimensional sketch, which is an $N_s \times D$ matrix, subset of $\mathbf{X}$. $\mathcal{A}_e$ is any embedding algorithm, $\mathbf{E_Z}$ the low-dimensional sketch and $\mathbf{E_X}$ an embedding of the input data of dimension $N_s \times d$ and $N \times d$ respectively. $\mathcal{D}_\mathcal{G}$ is the transductive (i.e. diffusion) operator on the graph. We have $N_s < N$, $d < K$ and typically $d = 2$ or $d = 3$ when targeting visualization tasks.

Algorithm 8 Compressive Dimensionality Reduction

1: Compute a kNN graph $\mathcal{G}$ from the data $\mathbf{X}$
2: Sample N_s nodes of $\mathcal{G}$ cf. Section 4.3.1
3: Create a sketch $\mathbf{Z}$ from $\mathbf{X}$ using the sampled nodes
4: Apply $\mathcal{A}_e$ to $\mathbf{Z}$ to obtain an embedding $\mathbf{E_Z} = \mathcal{A}_e(\mathbf{Z})$
5: Solve the transductive learning problem to get $\mathcal{D}_\mathcal{G}$ c.f. Section 4.4
6: Apply the diffusion operator to obtain the final embedding $\mathbf{E_X} = \mathcal{D}_\mathcal{G}(\mathbf{E_Z})$

Let us detail Algorithm 8 step by step.

1. The graph construction can be carried out very efficiently by performing ANN searches in the data. See Section 1.2.4 for details.
2. Guided by the theoretical analysis of Section 4.3.1 we use low-pass concentrated kernels. Two choices are interesting, either a low-rank approximation (such as defined in Theorem 6) of a heat kernel $g(x) = e^{-\tau x}$.

   In Section 4.3.1 we defined theoretically the number of samples needed to be able to

sense and diffuse information from the sampled nodes to every other node. In practice, we were able to verify that $N_s = \mathcal{O}(\log(N))$, is sufficient for the diffusion process. When the number of classes $|\mathcal{C}|$ is available, $N_s = \mathcal{O}(|\mathcal{C}|\log(N))$ is also a good choice, since it gives at least $\mathcal{O}(\log(N))$ sampled points in each class. Otherwise $N_s = \mathcal{O}(d(\mathcal{G})\log(N))$ is a valid alternative, with $d(\mathcal{G})$ the diameter of the graph. All those choices for N_s are above the bounds defined in Section 4.3.1 for any choice of concentration of the kernels since $k < N$.

3. Since there is a trivial mapping between node indices and data points, creating the high-dimensional sketch $\mathbf{Z}$ is simply taking the subset of $\mathbf{X}$ corresponding to the samples indices, i.e. $\mathbf{Z} = [\mathbf{x}_{s_0}, \mathbf{x}_{s_1}, \ldots, \mathbf{x}_{s_{N_s}}]$ with $\{s_0, s_1, \ldots, s_{N_s}\} = \mathcal{S}$ the sample set.

4. The compressive embedding framework does not impose any constraint on the type of algorithm used. Indeed, any embedding algorithm $\mathcal{A}_e$ that can be applied on $\mathbf{X}$, can be applied on $\mathbf{Z} \subset \mathbf{X}$. We note the application of the embedding algorithm $\mathbf{E_S} = \mathcal{A}_e(\mathbf{Z})$.

5. The proposed transductive learning methods used for the diffusion defined in Section 4.4 need only graph filtering operations which are all carried out using fast-filtering such as defined in Section 1.2.2. The two operators that need to be computed are the localized filters $\mathcal{T}_i g$ and $\|\mathcal{T}_i g\|$. The former can be computed by filtering Kronecker delta centred on i, which means that exactly one filtering is needed to compute one $\mathcal{T}_i g$. The 2-norm $\|\mathcal{T}_i g\|$ is computed as defined in Eq. 1.39 and Lemma 4.

6. The final diffusion is a simple matrix-vector multiplication for both RKHS and CHD methods.

In terms of complexity, assuming that we write $c(\mathcal{A}_e(n))$ the complexity function of the embedding algorithm $\mathcal{A}_e$ for an input of size n, and m the order of the polynomial for fast-filtering, the complexity of Algorithm 8 is $\mathcal{O}(N\log(N))$ for step 1, $\mathcal{O}(m|\mathcal{E}|)$ for step 2, $c(\mathcal{A}_e(N_s))$ for step 4, $\mathcal{O}(m|\mathcal{E}|)$ for diffusion using Tikhonov, and $\mathcal{O}(mN_s|\mathcal{E}|)$ for RKHS and CHD diffusion for the last steps. This adds up to a $\mathcal{O}(N\log(N) + mN_s|\mathcal{E}| + c(\mathcal{A}_e(N_s))$ total complexity.

Assuming that the graph is sparse (i.e. $|\mathcal{E}| = \log(N)$) and that $c(\mathcal{A}_e(n)) = \mathcal{O}(n\log(n))$, which is the case for t-SNE and LargeVis, we obtain complexity of $\mathcal{O}\left(N\log(N) + N_s\log(N_s)\right)$ for their compressive versions using Algorithm 8. This result is interesting because the cost is asymptotically dominated by the graph construction for all $N_s < N$, and if $c(\mathcal{A}_e(n)) = \mathcal{O}(n\log(n))$, the CDR is asymptotically equivalent to the original embedding because of the graph construction. However, and as we will see in the experiences, non-asymptotic timing is in favour of the CDR algorithm.

4.6 Experiments

In this section, we provide experiments whose objective is to show how our proposed method behaves in practice. The first experiments examine the performance of the CDR with different sampling methods. The next one presents visualization results with different diffusion opera-

tors. In the last subsection, we show the results of the CDR routine used for visualization using t-SNE and LargeVis as the inner routines and compare them to their original implementation.

4.6.1 Sampling

In this experiment, we want to assess the impact of the sampling procedure on the overall performance of the CDR algorithm. To do so, we measure the quality of the embeddings produced using different sampling techniques: a uniform random sampling of the nodes without replacement, the adapted scheme presented in Section 4.3.1 and the active sampling technique presented in the previous chapter in Section 2.2.2.

In Figure 4.1 we show the 1NN scores and the execution times of the CDR algorithm run on a subset of the MNIST dataset in function of the number of samples N_s. First, we see that the uniform sampling scheme provides the worst results and improves slowly with an increasing number of samples. Second, we observe that both sampling schemes based on localized filters perform better and improve fast for low values of N_s. In addition, we see that the active scheme outperforms the adaptive sampling as the number of samples increases. While we have not proven that the active sampling scheme is optimal, it is not surprising that it performs better than instantaneous sampling methods and this validates its usability in practice. It has to be noted that, while the different sampling methods have a measurable impact on the performance, all 1NN values fall in a relatively short range.

The values reported in the left part of Figure 4.1 shows that both uniform and adaptive samplings have similar timings, and that the active sampling has a larger impact on the complete execution time. Finally, we see that the execution time increases almost linearly with the number of samples in this range of N_s.

4.6.2 Transduction

In this experiment, we compare the results of the CDR routine using the different graph transduction operators described in Section 4.4. Figure 4.2 shows 2D embeddings of the MNIST dataset both of the sketch $\mathbf{Z}$ used to start the diffusion and three full embeddings corresponding to the CHD, RKHS and Tikhonov transduction algorithms. All three methods provide a reasonably good class separation, with a slight advantage for the CHD. The ACI values reported in Figure 4.2 tend to confirm this evaluation. Finally, both the RKHS transduction and Tikhonov regression present scaling problems, i.e. points that are put far away from the bulk of the data. For those different reasons, the CHD will be used as the default diffusion operator of the CDR.

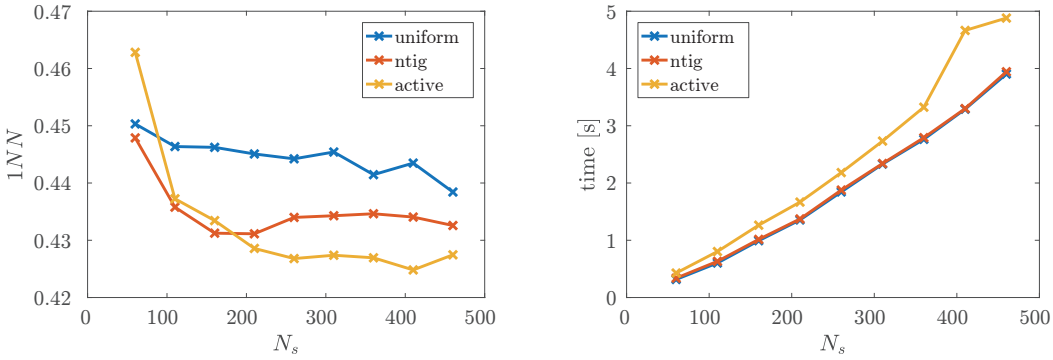

Figure 4.1 – **Sampling performance.** This figure shows the performance of the CDR method with three different sampling algorithms on a dimensionality reduction task ($d = 2$) with a subset of the MNIST dataset ($N = 1000, D = 784$). The inner embedding algorithm of the CDR is t-SNE. The first plot displays the generalization error with respect to the number of samples using the 1NN metric. The second plot shows the complete execution time of the algorithm with respect to the number of samples. The uniform sampling is labelled 'uniform', the adapted sampling scheme of Section 4.3.1 'ntig' and the active sampling technique of Chapter 2 'active'. All results are averaged over 200 realizations.

4.6.3 Natural datasets visualization

In this last experiment, we want to see the behaviour of state-of-the-art visualization algorithms (i.e. t-SNE and LargeVis) compared to their compressive versions. As for the previous chapters, we report the quality measures for all datasets in Tables 4.1-4.6 and show a few selected embeddings in Figures 4.3-4.5.

Starting with the timing results in Table 4.1, we see that compressive t-SNE is the fastest method overall, with an improvement of one order of magnitude over original t-SNE. It is however, sometimes slightly slower than LargeVis. Due to its large base cost, compressive LargeVis does not have a very good performance and only beats t-SNE, which as with the other experiments, is quite fast on smaller datasets but slow on larger ones.

Moving on the global clusterability and generalization error, reported in Tables 4.2 and 4.3, we observe that the original implementations perform better than their compressive counterparts. Both CDR methods have very similar scores, very slightly in favour of compressive LargeVis. The unsupervised NN precision reported in Table 4.4 displays a similar pattern. The average noise values given in Table 4.5 concur with these findings, with the difference that the score gap between the original algorithms and their compressive versions is lower in terms of ACN than of ACI or 1NN.

In terms of cluster split, all methods give very similar ACC values, as reported in Table 4.6, with the best performance given by the two CDR algorithms, with compressive t-SNE ranking slightly better. Getting similar ACC scores could be explained by the fact that all embeddings

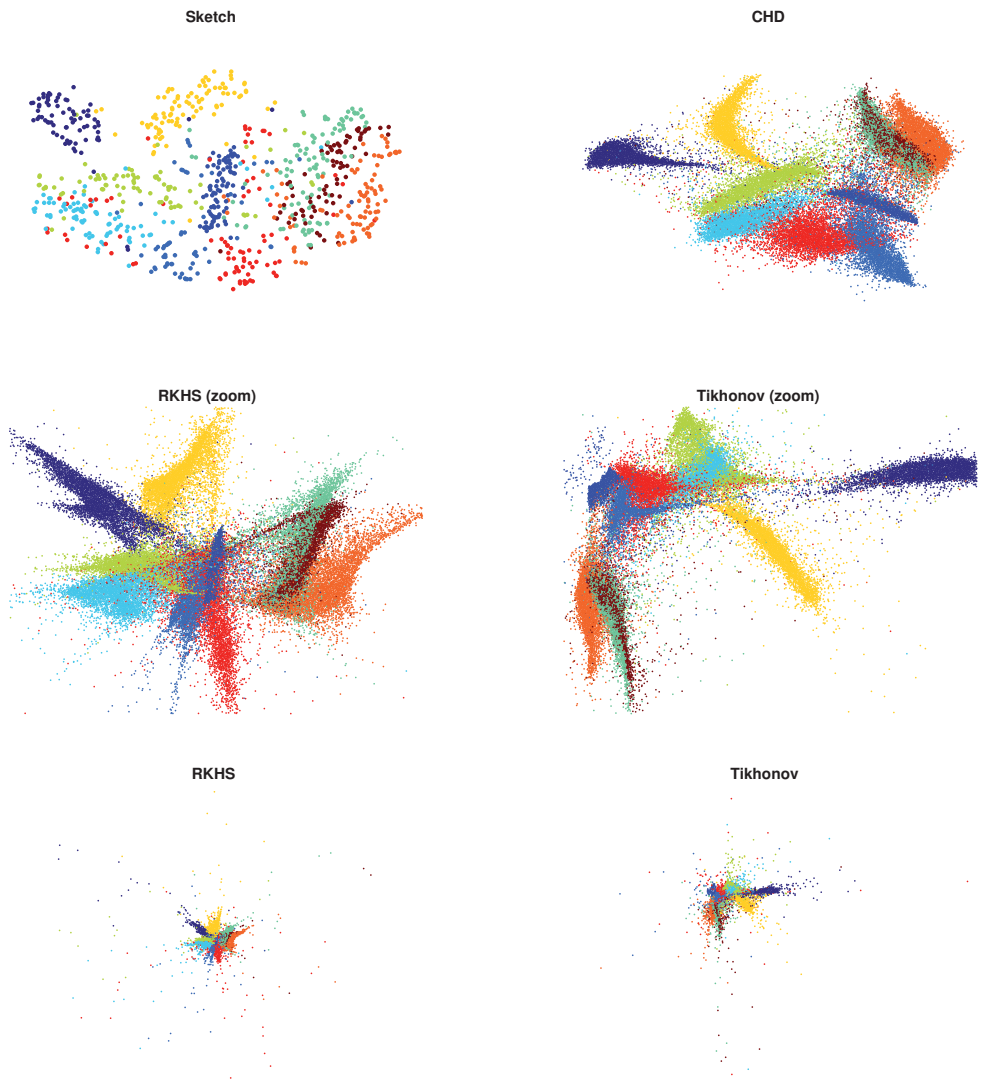

Figure 4.2 – **Diffusion operators.** In this figure, we show how the diffusion operators affect the CDR method. The CDR algorithm was used with t-SNE for the inner embedding and $N_s = 50\log(N)$. The task was the visualization of the full MNIST dataset ($N = 70000, D = 784$). The sketch $\mathbf{Z}$ is shown in the top left corner. Three complete embeddings created using the CHD, the RKHS reconstruction and Tikhonov regression are shown in top-right, middle-left and middle-right respectively. The last line shows the same embeddings as the middle line at their original scales. The ACI values computed for the three transductions are 0.238, 0.257 and 0.332 for CHD, RHKS and Tikhonov respectively. Zoomed-in versions are done by showing only points whose coordinates are between the 0.5 and 99.5 percentiles of the data distribution.

have close topologies.

Looking at Figure 4.3, we can confirm the analysis of the quality measures : first, the classes are better separated for the original implementations compared to their compressive versions. Second, the CDR variants exhibit more sparse noise. Finally, the classes of both CDR t-SNE and CDR LargeVis embeddings are, making abstraction of sparse noise, much more contained in continuous sets of points than in the original versions which have a lot of cluster splits, thus confirming the hypothesis drawn from the ACC observation. We show other embeddings in Figures 4.4 and 4.3 which display similar characteristics. Overall, the CDR algorithm yields very decent visualizations while being efficient to compute.

4.7 Conclusion

In this chapter, we have presented a general framework for the acceleration of dimensionality reduction algorithms. Our method is made possible by the use of similarity graphs, efficient sampling and graph diffusion. We showed how the method worked on visualization of natural datasets and that it gives satisfactory results while being faster than original implementations.

This work could be extended in several ways. First, while we made a theoretical analysis of the adapted sampling for a general kernel g, we simply used usual low-pass filters in the implementation, for simplicity. It would therefore be interesting to use data-driven kernels instead, to see the impact on the resulting embedding. In addition, although we have established precise bounds to compute N_s, we simply took a large upper bound instead of computing it explicitly. In fact, an exact evaluation of N_s using Theorems 5 or 6 implies non-trivial computations for certain quantities, and this could probably be estimated efficiently. We also saw in the experiments that the active sampling scheme empirically constructed in the previous chapter seems promising in the context of compressive sampling. To compensate the computational overhead of the sampling algorithm, one could re-use the localized filters for the diffusion process.

Time [s]	tsne	cdr:tsne	largevis	cdr:largevis
caltech101-caffenet	**89.52**	**27.31**	**219.00**	**232.24**
caltech256-caffenet	**469.09**	**118.71**	**256.87**	**324.24**
cifar10-cnn	**1410.91**	**346.35**	**315.21**	**552.65**
cmupie	**130.22**	**35.18**	**219.71**	**243.88**
coil20	**10.82**	**9.02**	**216.03**	**215.84**
fma-echonest	**139.97**	**41.60**	**220.09**	**249.65**
fma-rosa	**2475.82**	**542.23**	**413.09**	**791.51**
hiva	**949.41**	**197.30**	**279.98**	**402.83**
mnist	**1271.24**	**346.64**	**336.87**	**549.16**
norb	**884.97**	**190.00**	**291.14**	**394.10**
sylva	**3336.83**	**810.26**	**490.75**	**1024.58**
usps	**94.30**	**27.41**	**215.55**	**236.33**
20newsgroup	**303.95**	**64.08**	**236.83**	**270.77**
nova	**310.05**	**80.45**	**233.49**	**290.71**

Table 4.1 – **Embedding time.** This table reports the embedding time (in seconds) for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

ACI	tsne	cdr:tsne	largevis	cdr:largevis
caltech101-caffenet	**0.14**	**0.49**	**0.13**	**0.49**
caltech256-caffenet	**0.39**	**0.83**	**0.44**	**0.82**
cifar10-cnn	**0.28**	**0.48**	**0.27**	**0.47**
cmupie	**0.42**	**0.90**	**0.65**	**0.90**
coil20	**0.06**	**0.03**	**0.05**	**0.03**
fma-echonest	**0.87**	**0.93**	**0.88**	**0.94**
fma-rosa	**0.92**	**0.95**	**0.93**	**0.95**
hiva	**0.62**	**0.71**	**0.65**	**0.71**
mnist	**0.05**	**0.30**	**0.06**	**0.14**
norb	**0.52**	**0.74**	**0.60**	**0.73**
sylva	**0.35**	**0.96**	**0.33**	**0.94**
usps	**0.03**	**0.07**	**0.04**	**0.07**
20newsgroup	**0.25**	**0.59**	**0.26**	**0.60**
nova	**0.07**	**0.28**	**0.06**	**0.27**

Table 4.2 – **Embedding ACI.** This table reports the ACI score for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

1NN	tsne	cdr:tsne	largevis	cdr:largevis
caltech101-caffenet	**0.18**	**0.47**	**0.19**	**0.49**
caltech256-caffenet	**0.47**	**0.80**	**0.53**	**0.81**
cifar10-cnn	**0.28**	**0.44**	**0.29**	**0.44**
cmupie	**0.26**	**0.83**	**0.55**	**0.83**
coil20	**0.05**	**0.03**	**0.04**	**0.03**
fma-echonest	**0.70**	**0.74**	**0.71**	**0.74**
fma-rosa	**0.77**	**0.80**	**0.79**	**0.79**
hiva	**0.04**	**0.05**	**0.05**	**0.05**
mnist	**0.05**	**0.28**	**0.06**	**0.17**
norb	**0.38**	**0.59**	**0.46**	**0.58**
sylva	**0.05**	**0.10**	**0.04**	**0.10**
usps	**0.04**	**0.11**	**0.05**	**0.11**
20newsgroup	**0.21**	**0.57**	**0.24**	**0.58**
nova	**0.02**	**0.10**	**0.02**	**0.09**

Table 4.3 – **Embedding 1NN.** This table reports the 1NN scores for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

NNP	tsne	cdr:tsne	largevis	cdr:largevis
caltech101-caffenet	**0.35**	**0.27**	**0.36**	**0.27**
caltech256-caffenet	**0.22**	**0.10**	**0.22**	**0.11**
cifar10-cnn	**0.15**	**0.07**	**0.14**	**0.07**
cmupie	**0.39**	**0.30**	**0.40**	**0.30**
coil20	**0.29**	**0.27**	**0.27**	**0.28**
fma-echonest	**0.38**	**0.22**	**0.38**	**0.24**
fma-rosa	**0.32**	**0.18**	**0.28**	**0.16**
hiva	**0.41**	**0.18**	**0.42**	**0.18**
mnist	**0.32**	**0.17**	**0.29**	**0.18**
norb	**0.33**	**0.22**	**0.30**	**0.23**
sylva	**0.11**	**0.02**	**0.11**	**0.02**
usps	**0.49**	**0.35**	**0.49**	**0.38**
20newsgroup	**0.25**	**0.11**	**0.26**	**0.11**
nova	**0.20**	**0.10**	**0.21**	**0.10**

Table 4.4 – **Embedding NNP.** This table reports the NNP scores for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

ACN	tsne	largevis	cdr:tsne	cdr:largevis
caltech101-caffenet	**0.20**	**0.32**	**0.21**	**0.33**
caltech256-caffenet	**0.36**	**0.49**	**0.37**	**0.50**
cifar10-cnn	**0.24**	**0.30**	**0.23**	**0.29**
cmupie	**0.36**	**0.47**	**0.36**	**0.46**
coil20	**0.04**	**0.07**	**0.02**	**0.07**
fma-echonest	**0.37**	**0.38**	**0.38**	**0.38**
fma-rosa	**0.41**	**0.43**	**0.41**	**0.42**
hiva	**0.31**	**0.24**	**0.26**	**0.24**
mnist	**0.07**	**0.20**	**0.06**	**0.18**
norb	**0.34**	**0.32**	**0.30**	**0.31**
sylva	**0.02**	**0.20**	**0.02**	**0.21**
usps	**0.06**	**0.16**	**0.06**	**0.16**
20newsgroup	**0.30**	**0.34**	**0.26**	**0.34**
nova	**0.32**	**0.17**	**0.06**	**0.18**

Table 4.5 – **Embedding ACN.** This table reports the ACN scores for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

ACC	tsne	cdr:tsne	largevis	cdr:largevis
caltech101-caffenet	**4.61**	**2.05**	**2.87**	**3.24**
caltech256-caffenet	**6.80**	**3.84**	**4.72**	**5.93**
cifar10-cnn	**12.28**	**5.40**	**9.89**	**6.85**
cmupie	**30.77**	**15.17**	**24.93**	**17.28**
coil20	**3.97**	**0.54**	**14.50**	**1.89**
fma-echonest	**34.97**	**14.60**	**26.62**	**21.26**
fma-rosa	**23.55**	**13.52**	**22.39**	**16.12**
hiva	**30.29**	**15.17**	**25.94**	**14.81**
mnist	**13.08**	**6.02**	**10.47**	**7.05**
norb	**26.93**	**23.48**	**34.66**	**41.80**
sylva	**21.08**	**9.05**	**28.99**	**10.41**
usps	**13.67**	**3.92**	**10.75**	**4.47**
20newsgroup	**15.15**	**3.08**	**8.77**	**2.59**
nova	**29.75**	**13.18**	**19.75**	**12.79**

Table 4.6 – **Embedding ACC.** This table reports the ACC scores for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

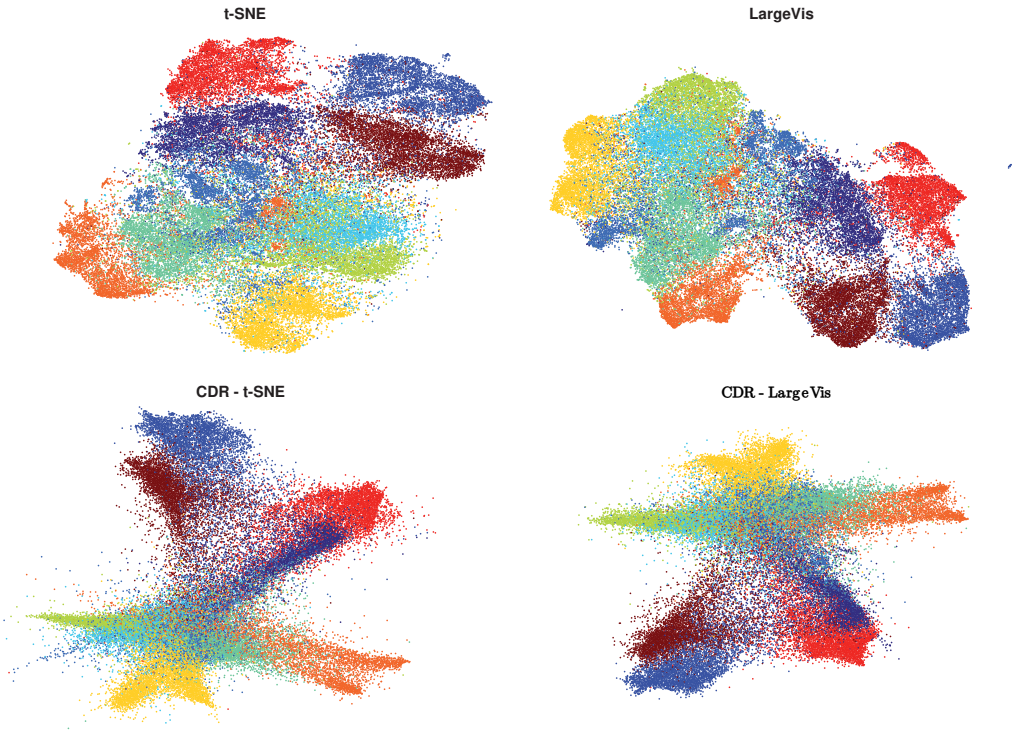

Figure 4.3 – Visualization of the Cifar10 dataset using t-SNE and LargeVis and their compressive versions. The colour map corresponds to the labels.

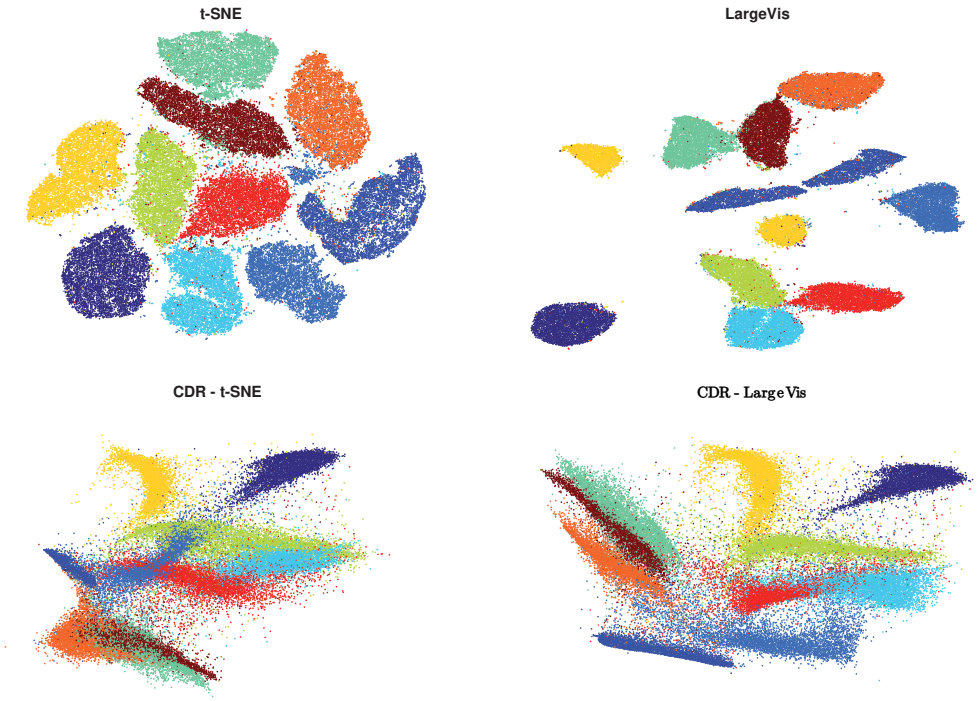

Figure 4.4 – Visualization of the MNIST dataset using t-SNE and LargeVis and their compressive versions. The colour map corresponds to the labels.

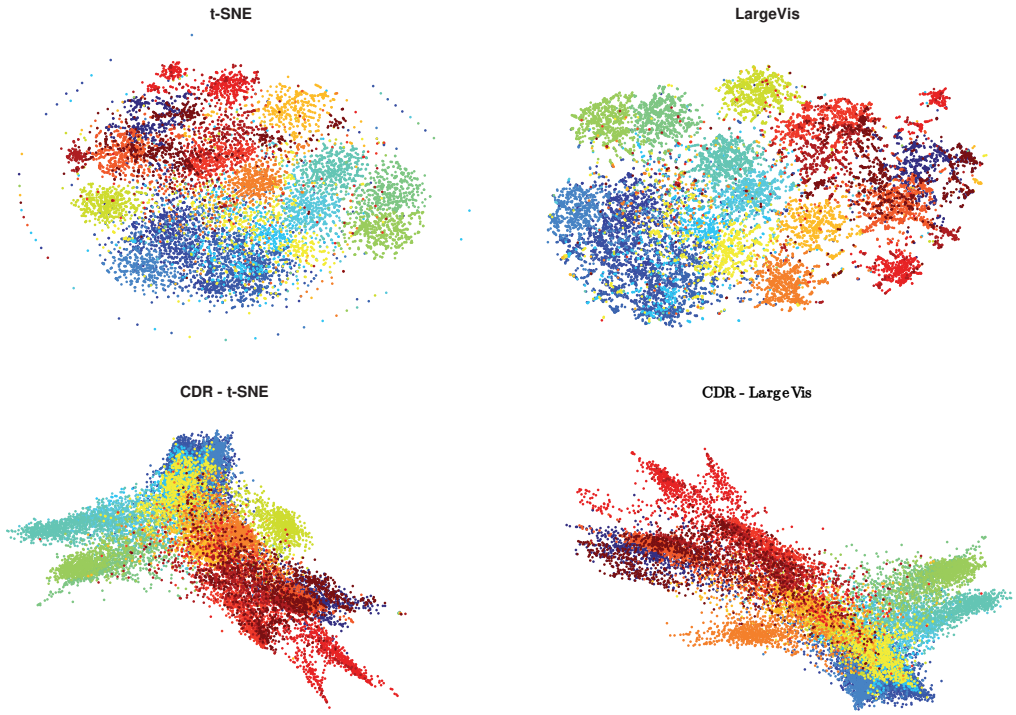

Figure 4.5 – Visualization of the 20Newsgroup dataset using t-SNE and LargeVis and their compressive versions. The colour map corresponds to the labels.

Heard melodies are sweet, but those unheard, are sweeter
— John Keats

5 Hierarchical Graph Structures

In this chapter, we will consider both dimensions of the feature space for the dimensionality reduction task. Indeed, high-dimensional data points are prone to contain too much information, which originally drives the need to apply dimensionality reduction techniques in the first place. This very fine-grained variability in the feature space can lead to an undesirable effect which we will refer to as over-discrimination.

This problem arises when the complete set of datapoints is considered to be equally important and, as a result, extremely similar points are still considered as individual and call for a mapping preserving the fact that they are distinct. In other words, assuming that data is sampled from a smooth manifold, it can be seen as an issue with the density of the samples: either the sampling density is imbalanced, or it is simply too high everywhere.

A very simple example of this effect can be made by considering audio features extracted from spectral histograms. Because of their good precision, they will discriminate data points that could semantically be aggregated as one feature-point, i.e. the same key pressed on a piano twice with approximately the same force and velocity should ideally be represented with a unique feature, while it is most probable that they will in practice be close but distinct in the feature-space.

Another undesirable effect of over-discriminative features is that they lead to very large spaces representing redundant data points. Let us take the audio example to the next level to illustrate it. Imagine that we compute the audio features for little time-windows of an entire discography. With thousands of data points per song, we can easily create a huge set of unique features, while it could be much more pertinent to aggregate the most similar ones to get a reduced dataset of better semantic representation.

In this last contribution, we present a new method to control the level of precision (i.e. discrimination) of the feature space using a hierarchical clustering of the data associated with a multi-resolution graph tree. We will use an efficient state-of-the-art algorithm to create the feature-tree and use it directly to compute a hierarchy of graphs. We will show that this

hierarchical graph structure, which we call a Clustering Graph Tree (CGT), can be used to coarsen and refine graph signals from one level to another using efficient down-sampling and up-sampling operations.

We will also show that gathering features (or meta-features) from intermediate levels of the CGT can be used to avoid the problem of over-discrimination and show that the tree can be constructed by setting a desired target size for the feature set, providing a complete control over the features precision.

In addition, we propose two generic methods for dimensionality reduction in a framework similar to the one described in Chapter 4 by providing a meta-algorithm. The similarity will be to accelerate any dimensionality reduction routine by running it only on a set of data points much smaller than the input size. The major difference, however, will be to use meta-features provided by intermediate levels of the CGT instead of sampled input features. The first approach will be to consider this embedding (of lower size) directly, providing control over the over-discriminatory features. The second approach will be to use the down-sampling operator defined on the tree to extend the embedding to its usual input size.

Finally, we present a tree-coding technique providing a one-dimensional encoding of the feature space that preserves some of the CGT structure. We show that it provides a low-cost one-dimensional embedding, and that it can be used to compute a structure-driven colour mapping.

This chapter is organized as follows. Section 5.1 gives an overview of the related work on the subject. Then, in Section 5.2 we show how to construct the feature-tree. We follow in Section 5.3 with the construction of the Clustering Graph Tree, the definition of the up/down-sampling operators and the dimensionality reduction schemes. Next, in Section 5.4, we present our tree-coding scheme and associated 1-dimensional embedding. Section 5.5 addresses the computational aspects of the schemes presented in the previous sections. Finally, we provide various experiments on different tasks in Section 5.6 and close the chapter with a discussion in Section 5.7.

5.1 Related work

The idea of grouping objects in a tree structure by use of similarity is the core concept of hierarchical clustering, first analysed in [205, 89]. The method refers to agglomerative clustering which states that the hierarchy is computed bottom-up, starting from individual data points and grouping them into bigger clusters incrementally. Later, in [94], a reverse approach, generally called divisive clustering, has been proposed as a top-down technique, starting with one cluster and dividing it recursively into smaller and smaller clusters. Formulated in a generic way, hierarchical clustering works using any kind of similarity measure and various merging (respectively splitting) decision criteria. This last element is based on linkage measure between sets of points with the most common ones for agglomerative clustering being maxi-

mum distance or minimum distance between sets (called complete-linkage or single-linkage respectively) and Ward's criterion [205] considering the variance of the clusters undergoing a merge. Despite its many successful applications in fields such as gene sequencing [53, 9], astro-physics [132, 209] wireless networks [12] or text processing [179, 216], hierarchical clustering has a profound scaleability issue. Indeed, the original implementations have prohibitive time complexities: $\mathcal{O}(N^2\log(N))$ and $\mathcal{O}(2^N)$ for the agglomerative and divisive approaches respectively. While some optimized methods using special linkages such as SLINK [171], CLINK [55] or UPGMA [176] have made improvements in terms of scaleability, none reduces the complexity below $\mathcal{O}(N^2)$, which is still too high to process large datasets in a reasonable amount of time.

Relaxing the global optimality of the original formulation of divisive clustering, one can use any traditional clustering algorithm in a top-down recursive way to obtain a hierarchical structure. In this context, k-means [117, 68], one of the most famous cluster analysis algorithm, has been employed using this hierarchical scheme with some successes [40, 103, 122]. Note that the resulting complexity of the general procedure is always $\mathcal{O}(c(N)\log(N))$ with $c(N)$ being the number of operations of the clustering technique used.[1] Unfortunately, since k-means is proven to be superpolynomial in the worst case, this makes the overall complexity of its hierarchical version too high to handle large datasets. In order to circumvent this issue, we saw in Section 1.2.4 that approximating the k-means step can provide a great acceleration. Indeed, by only letting the algorithm run for a few iterations, instead of waiting for convergence, the FLANN implementation of the k-means trees constructs its index (i.e. the hierarchical k-means tree) in $\mathcal{O}(N\log(N))$ [128].

Multiresolution has also been considered directly at the graph level, in particular in a class of works that has considered the use of multi-level coarsening for graph partitioning [14, 80, 91, 92]. These methods follow a common pattern: they start by recursively coarsening a graph, compute partitions at the top level (using spectral techniques or graph cuts) and then un-coarsen the graph, by recursively refining the partition (e.g. using variants of Kernighan-Lin [96]). The hierarchical sequences of graphs created by this technique use edge-collapsing coarsening algorithms to merge vertices and generate smaller graphs. More precisely, one step of coarsening amounts to collapsing a large number of edges, none of which is adjacent to the same vertex, which translates to the problem of finding a maximal matching. Since finding a maximal matching can be too costly, randomized methods are generally preferred: non-matched vertices are visited randomly and the edges with largest weights are selected for the collapse (i.e. heavy weight matching). The merged vertices are connected to the adjacent vertices of both original vertices. Ideally, using this procedure cuts the number of vertices in half at each coarsening step. In practice, however, the number of edges collapsed diminishes as the graph gets smaller. This effect is caused by the fact that the maximal matching is approximated by a random method and, mainly, because the graphs get denser. The decrease in sparsity can be easily explained by the fact that the number of edges collapsed

[1] Indeed, assuming a branching factor K, each recursive step cost $Kc(\frac{N}{K}), K^2c(\frac{N}{K^2}),\ldots$ is $\mathcal{O}(c(N))$ if $c(N)=\Omega(N)$.

is approximately the same amount than the number of vertices merged, which is insufficient to keep an average degree constant (i.e. $|\mathcal{E}| - \frac{N}{2} > \frac{|\mathcal{E}|}{2}$ in most cases).

Another graph-based approach using multi-resolution paradigms has been proposed in [169]. It uses GSP principles to transfer the concept of Laplacian pyramids [34] to graphs. More precisely, following the frame formulation as [59], the authors propose a scheme composed of a pair of down-sampling (coarsening) and up-sampling (interpolation) operations using mostly graph filtering. The down-sampling is performed using Kron reduction [100] which provides good spectral guarantees, in particular, spectral interlacing, and maps the original Laplacian to its down-sampled version. The interpolation is done using filtering with the Green kernel of the Laplacian. Similarly to edge-collapsing techniques, Kron reduction produces denser graphs at each coarsening operation. The proposed solution to counter this undesirable effect is to apply an efficient graph sparsification routine [178] after each down-sampling step.

It is convenient to represent hierarchical structures as trees, which can be used to encode information. Tree-base coding (or simply tree coding) is used in information theory to generate code words from alphabets. The most famous examples such as Huffman [85] or Shannon-Fano [163, 64] coding create binary trees from the symbols probabilities (or occurrences) and simply assign a binary code to each child which are then concatenated while descending into the tree. Such schemes give prefix-free, variable-length codes which intrinsically map the information present in the tree. The basic principles of tree coding combining ordered one digit assignment to children and concatenation have been used in various other fields such as speech coding [107, 199] or image and video compression [58, 13, 129, 181, 185].

5.2 Feature-tree

In this chapter, we propose to construct a hierarchical graph structure from a feature-tree. Indeed, we saw in the previous section that graph coarsening techniques were prone to densification, and also, needed an access to the complete graph beforehand. Since hierarchical clustering has a prohibitive cost, we chose to build the feature-tree using fixed iterations recursive hierarchical k-means, similarly to [128].

The tree construction process is quite simple. Starting from the complete set of features, K clusters are formed using I iterations of the k-means algorithm. For each of the K clusters, the process is applied recursively provided there are more than K datapoints remaining in the cluster. This procedure implies a tree structure T: the K children of the root are the first cluster centres, then, recursively, children of a node are either cluster centres of the next recursion level or feature points. The leaves are necessarily feature points. The height $h(T)$ of the tree is at least $\log_K(N)$. We call meta-features the set of all nodes of a given level l of the tree (except the root and the leaves), written $\mathcal{Z}_l$. Because they are derived from intermediate nodes, meta-features are comprised either of cluster centres uniquely or may include data points.

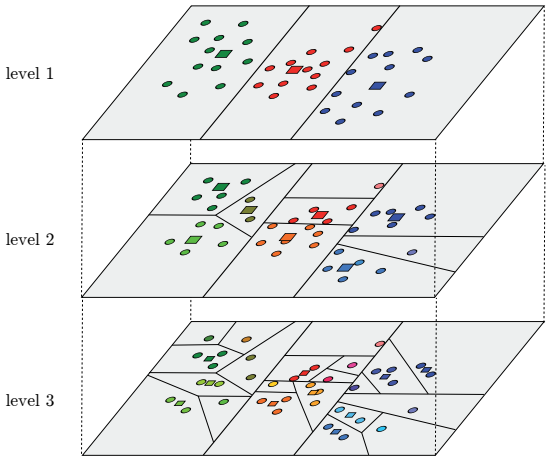

Figure 5.1 – **Hierarchical k-means (clustering).** In this figure, we show the different levels of recursion of the k-means process on a feature set with $K = 3$. Cluster centres are depicted with squares and feature points with circles. The colour scheme map the cluster hierarchy illustrating the principle described in Section 5.4

.

Also, the size of the meta-features sets $N_l = |\mathcal{Z}_l|$ are bounded by

$$K^{l-1} < N_l \leq K^l. \tag{5.1}$$

An illustration of the hierarchical k-means is shown in Figure 5.1 and its associated tree in Figure 5.2.

The only parameters for the feature-tree construction are the branching factor K and the number of iterations I. The latter influences the quality of the clustering, but it was shown experimentally to already provide sufficiently good results even for very small values, e.g. $I < 10$ (see [128, Figure 4]). The second parameter, the branching factor K, has an impact on the clustering topology (which is well depicted in [128, Figure 3]) and on the tree height. The branching also affects the size of meta-features sets. Assuming one targets a specific size s_l for a meta-feature set $\mathcal{Z}_l$, to get control of the features discriminability, the corresponding branching factor can be computed using Eq. (5.1):

$$\lfloor \sqrt[l]{s_l} \rfloor \leq K < \lceil \sqrt[l-1]{s_l} \rceil. \tag{5.2}$$

The flooring and ceiling operations are necessary because K has to be an integer. For low values of l, one may assume that the tree is balanced and safely choose the lower bound $K = \lfloor \sqrt[l]{s_l} \rfloor$ to get a target size s_l at level l. Unfortunately, the rounding may introduce an error, and K^l may not be equal to the desired size s_l. To solve this problem, one may want to use l as a parameter and search for a combination of l and K that gives the desired size s. Problematically l is also an integer number in a predefined range $[1, H_T]$ and thus it may not

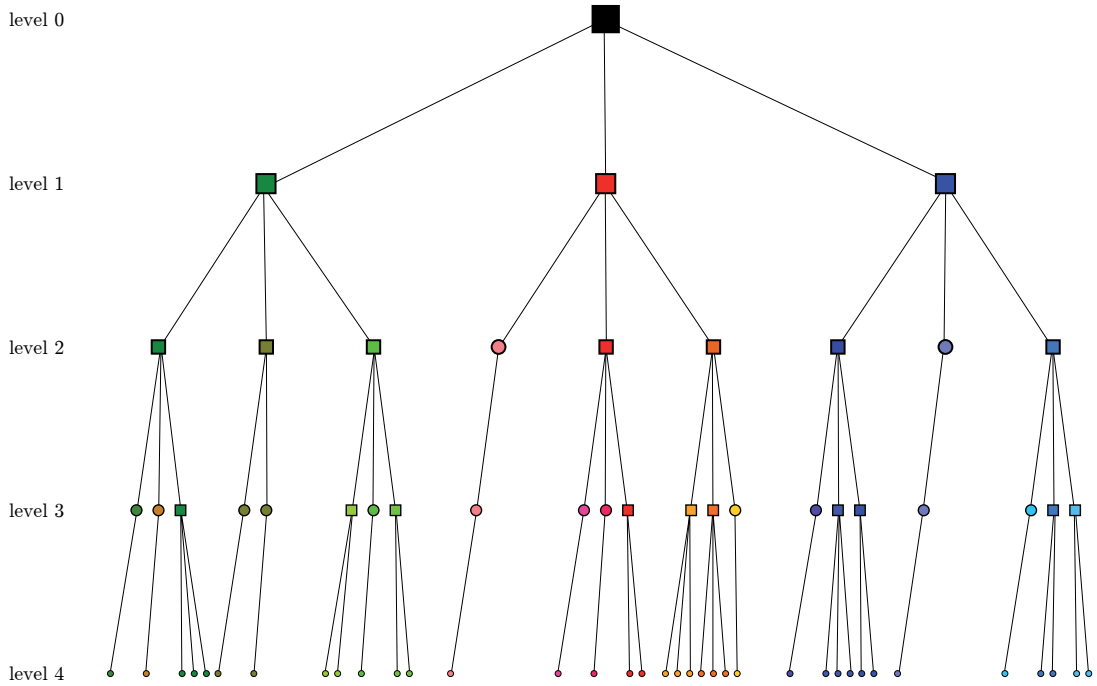

Figure 5.2 – **Hierarchical k-means (tree).** In this figure, we show feature-tree associated to the hierarchical k-means process depicted in Figure 5.1. Cluster centres are depicted with squares and feature points with circles. The colour scheme map the cluster hierarchy illustrating the principle described in Section 5.4

.

exist any combination of l and K which yields the desired size s. However, we can find the parameter set which provides the size closest to s by iterating on possible values of l. While searching for a value of the branching factor, we naturally want to have at least two levels in the tree and thus we need to ensure that

$$2 < K < N. \tag{5.3}$$

Indeed, we need $K > 2$ trivially, since otherwise the data does not get clustered, and $K < N$ because otherwise all data is in a unique cluster. Since we iterate over l, we need to translate the bounds from K to l, yielding the following relation

$$1 \leq l < \log_2(s) + 1, \tag{5.4}$$

which is derived in Appendix A.5.1. Note that we can adapt this relation, for example to $l \geq 2$ to avoid the trivial solution $K = S$ and also can set a lower bound $K_{\min}$ on the possible branching values by restricting to $l \leq \log_{K_{\min}}(s) + 1$. Following these rules, the method providing the best pair (K, l) from the desired size s is detailed in Algorithm 9.

Algorithm 9 Branching factor selection

Require: s, $K_{\min}$ (Optional, by default $K_{\min} = 2$)
$l_{\max} = \lfloor \log_{K_{\min}}(s) \rfloor + 1$
$err_m = +\infty$
for $2 \leq l \leq l_{\max}$ **do**
  $k = \lfloor \sqrt[l]{s} + \frac{1}{2} \rfloor$
  $err = |k^l - s|$
  if $err < err_m$ **then**
    $err_m = err$
    $K = k$
    $L = l$
  end if
end for
return (K, L)

5.3 Clustering Graph Tree

Having access to a feature-tree T providing a multi-resolution access to a set of features (or meta-features), i.e. a vertical structure, we want to compute graphs at all levels to provide a horizontal structure. In this section, we first show how to construct such a tree, which we call a Clustering Graph Tree (CGT), from the feature-tree. Then we evaluate what extra properties vertices possess in such a hierarchical construction. Then, we propose two methods to provide up-sampling and down-sampling operations of signals on the CGT.

5.3.1 Construction

The creation of the graph-tree from the feature-tree is done in a very simple way. For each level $l \in \{1, \ldots, H_T\}$ we construct a graph $\mathcal{G}_l$ from the meta-features set $\mathcal{Z}_l$ (with its associated data matrix $\mathbf{Z}_l$) using a kNN graph construction algorithm. Note that $\mathcal{G}_{H_T}$ is the graph computed on the complete set of input features.

In addition to their usual properties, vertices in the graphs of the CGT carry characteristics derived from the tree. More precisely, any node in a graph $\mathcal{G}_l$ has $1 < Nc < K$ children nodes in G_{l+1} (for $l < H$) and one parent node in $\mathcal{G}_{l-1}$ (for $l > 1$). Assuming we write $\mathcal{V}_l$ the vertex set of $\mathcal{G}_l$, we write $C(v)$, for $v \in \mathcal{V}_l$, the set of children of vertex v and $P(v)$ its parent node. An illustration of a graph-tree is given in Figure 5.3.

5.3.2 Up/Downsampling operations

Now that the graph-tree is defined, we want to look how to proceed with the two fundamental operations of such multiresolution structures: up-sampling (i.e. going from level l to $l+1$) and down-sampling (i.e. going from level l to $l-1$). Let us assume that we are at level l (with $1 < l < H_T$), the operations we wish to define will be applied on graph signals living on $\mathcal{V}_l$.

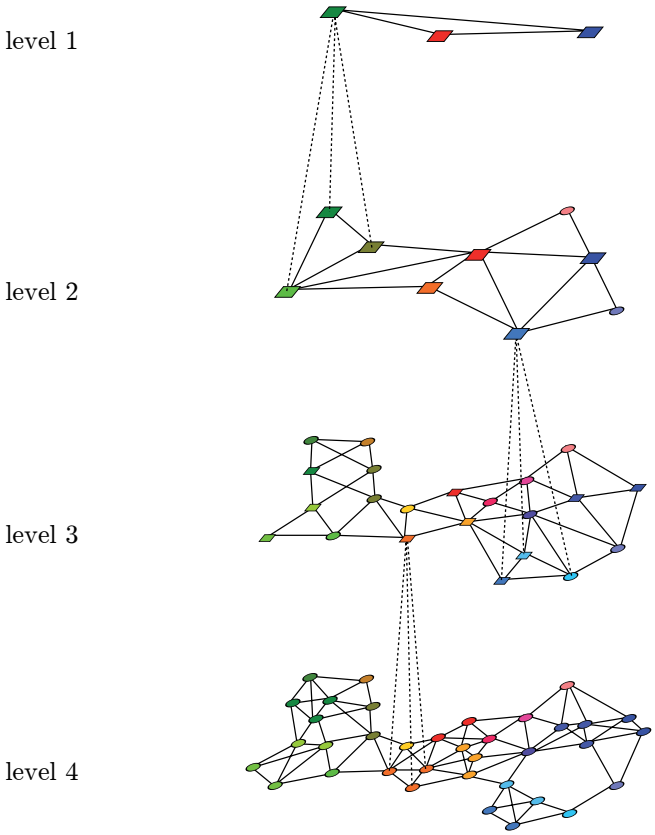

Figure 5.3 – **Clustering Graph Tree.** In this figure, we show the CGT created from the feature-tree depicted in Figures 5.1-5.2. Solid lines represent graph edges and dashed lines parent-child relations. To avoid clutter, only a few such vertical connections are drawn.

Note that we have $|\mathcal{V}_l| = N_l$ because the vertex set maps the meta-features set.

We define the down-sampling of a graph signal $\mathbf{x}_l$ as either a mean or a maximum-voting (depending on the signal having continuous or discrete values). More precisely, for all $v_i \in \mathcal{V}_{l-1}$, the down-sampled real-valued signal is

$$(\downarrow \mathbf{x}_l)[i] = \frac{1}{|C(v_i)|} \sum_{v_j \in C(v_i)} \mathbf{x}_l[j], \tag{5.5}$$

and the discrete-valued

$$(\downarrow \mathbf{x}_l)[i] = \arg\max_{c_k \in \mathcal{C}} \sum_{v_j \in C(v_i)} \mathbb{1}_{\{\mathbf{x}_l[j]=c_k\}}, \tag{5.6}$$

with $\mathcal{C}$ the set of possible categories.

For the up-sampling, we propose a two-step method: first, the signal is copied on all children

nodes and second the graph $\mathcal{G}_{l+1}$ is used for regularization. First, let us define the flat signal $\tilde{\mathbf{x}}_l$ for $v_i \in \mathcal{V}_{l+1}$ as:

$$\tilde{\mathbf{x}}_l[i] = P(v_i). \tag{5.7}$$

Then, we can enforce the smoothness by using Tikhonov regularization on $\mathcal{G}_{l+1}$, giving the up-sampled signal

$$(\uparrow \mathbf{x}_l) = \arg\min_{\mathbf{x}} \|\mathbf{x} - \tilde{\mathbf{x}}_l\|_2^2 + \gamma \|\nabla_G \mathbf{x}\|_2^2, \tag{5.8}$$

where γ is a simple regularization parameter. While we could use more advanced regularizers such as TV or data-driven smoothness constraints, we use Tikhonov for its efficient implementation via a simple graph filtering (as shown Section 1.2.3).

Note that both up and down-sampling operations can be made to work between two non-adjacent levels in the tree. In particular, it is sufficient to propagate the information in the subtrees between the two levels instead of stopping after a single step. To lower the computational load of up-sampling, the regularization can be computed on the target level instead of all the intermediate levels. The process of coarsening or refining between non-adjacent levels is made explicit in the notation as we write $\downarrow_{l_2}^{l_1}$ and $\uparrow_{l_1}^{l_2}$ for a down-sampling respectively up-sampling operation from level l_1 to level l_2.

5.3.3 Dimensionality reduction

Now that we have defined our multi-resolution structure, we can use it to tackle our main problem of interest: dimensionality reduction. As hinted in the introduction of this chapter, we propose a meta-algorithm using any dimensionality reduction method $\mathcal{A}_e$ as its inner routine. In short, we compute a CGT from input features in order to use meta-features from an intermediate level as a sketch and then apply $\mathcal{A}_e$ on the sketch.

The resulting embedding can either be used directly, if the goal is to reduce the input size (e.g. to compensate for over-discriminative features), or extended to the full size by upsampling the sketch embedding to the deepest level of the CGT. The first method is described in Algorithm 10 and the second in Algorithm 11. Note that the desired target size of the sketch can be achieved by using Algorithm 9 to determine the optimal branching factor yielding a level with the expected number of meta-features.

Algorithm 10 CGT Dimensionality Reduction (sketching)

Require: $\mathbf{X}, \mathcal{A}_e, s$

1: Compute K and $l^\star$ using Algo. 9 with parameter s
2: Construct the CGT $T(\mathbf{X})$ with parameter K, as described in Section 5.3.1.
3: Apply $\mathcal{A}_e$ to $\mathbf{Z}_{l^\star}$ to obtain an embedding $\mathbf{E}_{\mathbf{Z}_{l^\star}} = \mathcal{A}_e(\mathbf{Z}_{l^\star})$
4: return $\mathbf{E}_{\mathbf{Z}_{l^\star}}$

Algorithm 11 CGT Dimensionality Reduction (full)

Require: $\mathbf{X}, \mathcal{A}_e, s$

1: Compute K and $l^\star$ using Algo. 9 with parameter s
2: Construct the CGT $T(\mathbf{X})$ with parameter K, as described in Section 5.3.1.
3: Apply $\mathcal{A}_e$ to $\mathbf{Z}_{l^\star}$ to obtain an embedding $\mathbf{E}_{\mathbf{Z}_{l^\star}} = \mathcal{A}_e(\mathbf{Z}_{l^\star})$
4: Upsample the embedding to the input size $\mathbf{E} = \uparrow_{l^\star}^{H} (\mathbf{E}_{\mathbf{Z}_{l^\star}})$, as described in Section 5.3.2.
5: return $\mathbf{E}$

5.4 Feature coding

Both original features and meta-features are implicitly structured by the tree since it is constructed using clustering routines. Incidentally, nodes and leaves in a subtree are supposedly similar, while not necessarily the closest neighbours. Using this idea and inspired by tree-based coding, we propose a feature-coding scheme with the goal of providing a compact encoding of the features allowing, among other things, fast comparisons.

5.4.1 Encoding / Decoding

Since we want the codes to uniquely represent the features and contain the information from its path down the tree, the most compact representation is given using a big endian base K numbering on H_T levels. More precisely, assuming the cluster index (or tree branch index) of a feature $\mathbf{x}$ at level l is written $i_l(\mathbf{x})$, the feature code $c(\mathbf{x}) \in \mathbb{N}$ is computed as :

$$c(\mathbf{x}) = \sum_{l=1}^{H} i_l(\mathbf{x}) K^{H-l}. \tag{5.9}$$

The encoding procedure is given in Algorithm 12. By construction, we know that $0 \leq i_l(\mathbf{x}) < K$ since K is the maximum branching factor of the tree. Then, since every feature is on its own leaf, every path $\{i_1(\mathbf{x}), i_2(\mathbf{x}) \ldots, i_H(\mathbf{x})\}$ down the tree is unique. From those two facts, we can conclude that each feature code is unique.

Note that the encoding proposed in Eq. (5.9) yields a one-dimensional natural number with some interesting properties. First, the number is largely influenced by its first-level tree branches since they are assigned with the highest powers of K. In the same way, the deepest branches are weighted by lowest powers of K, having much less influence. In short, the structure revealed by the clustering is encoded in the numbering in base K. The largest the clustering scale, the most influence it has on the encoding.

Also, while being one-dimensional, $c(\mathbf{x})$ is sufficient to compute the set of cluster indices of $\mathbf{x}$, $\{i_1(\mathbf{x}), i_2(\mathbf{x}) \ldots, i_H(\mathbf{x})\}$, using only integer operations (modulo and integer division). The decoding procedure is given in Algorithm 13.

Algorithm 12 Feature encoding

Require: $\{i_1(\mathbf{x}), i_2(\mathbf{x}) \ldots, i_{H_T}(\mathbf{x})\}$, K

1: $c(\mathbf{x}) = 0$
2: **for** $1 \leq l \leq H_T$ **do**
3: $\quad c(\mathbf{x}) = c(\mathbf{x})K + i_l(\mathbf{x})$
4: **end for**
5: return $c(\mathbf{x})$

Algorithm 13 Feature decoding

Require: $c(\mathbf{x})$, K

1: $r = c(\mathbf{x})$
2: **for** $1 \leq l \leq H_T$ **do**
3: $\quad i_{H_T - l}(\mathbf{x}) = \mod(r, K)$
4: $\quad r = \lfloor r/K \rfloor$
5: **end for**
6: return $\{i_l(\mathbf{x})\}$, $l \in [1, \ldots, H_T]$

5.4.2 Colour mapping

One possible use for such an encoding is to provide a meaningful colour mapping for the data points. By meaningful, we only mean that it carries some of the structure contained in the tree and stored in the codes. To do so, we need similar codes to be assigned similar colours.

For our proposed colour mapping scheme we will use the Hue Saturation Value (HSV) colour model. Since colour spaces form a vast subject, we will only briefly justify this choice: HSV is an intuitive cylindrical model whose primaries are easy to grasp perceptually and it has a very efficient conversion to the traditional RGB colour space. While more advanced colour spaces such as CIELAB would have a better perceptual uniformity, their primaries are difficult to manipulate and they need more complicated conversions to the RGB space.

The HSV primaries control respectively the colour wheel (Hue), chroma (Saturation) and brightness (Value). Their ranges are usually $[0, 360]$ for the H channel and $[0, 1]$ for S and V. To avoid lavish or very dark colours, we will restrict our possible ranges for S and V to $[0.5, 1]$. Since our encoding is a base K number, we have to decide what importance each channel will have so that they are evaluated in order. In the HSV model, the Hue is clearly the most important perceptual parameter, so we will attribute the coefficients in the natural HSV order. The method to generate an HSV-triplet from a code $c(\mathbf{x})$ is given in Algorithm 14. As we see, the code is first decomposed into his tree branching coefficients, which are then used in ascending order of the levels to generate the HSV coefficients $[c_H, c_S, c_V]$. The heaviest digits are associated to the Hue, then middle digits to the Saturation and finally low-weight digits to the Value. The number of branching levels used per channel is given through the N_H, N_S, N_V parameters, whose sum needs to be lower than the tree height. To emphasize the importance of the H channel, we can attribute it more precision using a non-uniform set of parameters, e.g. $N_H = 2N_S = 2N_V$.

Algorithm 14 HSV colour mapping

Require: $c(\mathbf{x}), N_H, N_S, N_V, K$
Ensure: $N_H + N_S + N_V \leq H_T$
Compute $\{i_l(\mathbf{x})\}$, $l \in [1, H_T]$ using Algo. 13.
$c_H = 0$
for $1 \leq l \leq N_H$ **do**
    $c_H = c_H \times K + i_l(\mathbf{x})$
end for
$c_H = 360(c_H/(K^{N_H}))$
$c_S = 0$
for $N_H < l \leq (N_H + N_S)$ **do**
    $c_S = c_S \times K + i_l(\mathbf{x})$
end for
$c_S = c_S/(K^{N_S})$
$c_V = 0$
for $(N_H + N_S) < l \leq (N_H + N_S + N_V)$ **do**
    $c_V = c_V \times K + i_l(\mathbf{x})$
end for
$c_V = c_V/(K^{N_V})$
return $[c_H, c_S, c_V]$

5.4.3 Distances

The encoding we propose in this section is, in fact, a dimensionality reduction operation, going from D-dimensional features to 1-dimensional embeddings. We will now check if some geometrical isometries are preserved by the embedding. In particular, we will look at distances between codes.

To start, let us see what is the form of the difference between codes corresponding to two features $\mathbf{x}$ and $\mathbf{y}$:

$$c(\mathbf{x}) - c(\mathbf{y}) = \sum_{l=1}^{H_T} i_l(\mathbf{x}) K^{H_T - l} - \sum_{l=1}^{H_T} i_l(\mathbf{y}) K^{H_T - l} \tag{5.10}$$

$$= \sum_{l=1}^{H_T} (i_l(\mathbf{x}) - i_l(\mathbf{y})) K^{H_T - l} \tag{5.11}$$

From here, we assume that $i_l(\mathbf{x}) = i_l(\mathbf{y})$ for $1 \leq l \leq m$ and $i_{m+1}(\mathbf{x}) \neq i_{m+1}(\mathbf{y})$, i.e. $\mathbf{x}$ and $\mathbf{y}$ have a similar path down the tree until level m. With this hypothesis, the value of the difference is

bounded

$$
\begin{aligned}
c(\mathbf{x}) - c(\mathbf{y}) &= \sum_{l=1}^{m} (i_l(\mathbf{x}) - i_l(\mathbf{y}))K^{H_T-l} + \sum_{l=m+1}^{H_T} (i_l(\mathbf{x}) - i_l(\mathbf{y}))K^{H_T-l} && (5.12)\\
&= \sum_{l=m+1}^{H_T} (i_l(\mathbf{x}) - i_l(\mathbf{y}))K^{H_T-l} && (5.13)\\
&< K^{H_T-m} && (5.14)
\end{aligned}
$$

with the last inequality derived in Appendix A.5.2. Also, since $c(\mathbf{x})$ and $c(\mathbf{y})$ are positive, we have

$$-\max(c(\mathbf{x}), c(\mathbf{y})) \leq (c(\mathbf{x}) - c(\mathbf{y})) \leq \max(c(\mathbf{x}), c(\mathbf{y})) \tag{5.15}$$

and since $c(\mathbf{x}), c(\mathbf{y}) < K^{H_T-m}$, it becomes :

$$-K^{H_T-m} < (c(\mathbf{x}) - c(\mathbf{y})) < K^{H_T-m} \tag{5.16}$$

which means, using Eq. (5.14), that

$$|c(\mathbf{x}) - c(\mathbf{y})| < K^{H_T-m}. \tag{5.17}$$

This result, while quite simple, shows that the absolute value between codes is bounded above by their last level in common. This means that high distances imply dissimilarity in the tree branching coefficients. Unfortunately, it is not possible to find a simple lower-bound bigger than 1,[2] which means that a low distance does not necessarily imply similarity.

To get a meaningful tree dissimilarity, one needs to decode the features into their coefficients. Let us write $\mathbf{i}(\mathbf{x}) = [i_1(\mathbf{x}), i_2(\mathbf{x}) \ldots, i_{H_T}(\mathbf{x})]$ and $\mathbf{i}(\mathbf{y}) = [i_1(\mathbf{y}), i_2(\mathbf{y}) \ldots, i_{H_T}(\mathbf{y})]$ the coefficients extracted from $c(\mathbf{x})$ and $c(\mathbf{y})$ respectively, and consider a distance between these H_T-dimensional integers vectors that we call the Tree Branching Distance (TBD):

$$\mathrm{TBD}\left(\mathbf{i}(\mathbf{x}), \mathbf{i}(\mathbf{y})\right) = \sum_{l=1}^{H_T} \left| i_l(\mathbf{x}) - i_l(\mathbf{y}) \right| K^{H_T-l} \tag{5.18}$$

Lemma 12. *Assume that $\boldsymbol{i}(\mathbf{x})$ and $\boldsymbol{i}(\mathbf{y})$ have a level-m branching dissimilarity, i.e. $i_l(\mathbf{x}) = i_l(\mathbf{y})$ for $1 \leq l \leq m$ and $i_{m+1}(\mathbf{x}) \neq i_{m+1}(\mathbf{y})$. Then, the Tree Branching Distance is such that*

$$K^{H_T-(m+1)} \leq \mathrm{TBD}(\mathbf{i}(\mathbf{x}), \mathbf{i}(\mathbf{y})) < K^{H_T-m} \tag{5.19}$$

The proof of this Lemma is given in Appendix A.5.3. This result is interesting because we can see that the TBD is directly related to the tree-branching dissimilarity between two features.

[2] Indeed, if $K = 10$, $H_T = 5$, $c(\mathbf{x}) = 00200$ and $c(\mathbf{y}) = 00199$, then $m = 2$, which means that $|c(\mathbf{x}) - c(\mathbf{y})| < 10^3$ but the value is actually much lower $|c(\mathbf{x}) - c(\mathbf{y})| = 1$.

Finding the level m where the tree paths separate is actually quite easy. Indeed, taking the log on the bounds of Lemma 12 gives :

$$H_T - (m+1) \leq \log_K(\text{TBD}(\mathbf{i}(\mathbf{x}), \mathbf{i}(\mathbf{y}))) < (H_T - m), \tag{5.20}$$

which means that

$$m = H_T - (\lfloor \log_K(\text{TBD}(\mathbf{i}(\mathbf{x}), \mathbf{i}(\mathbf{y}))) \rfloor + 1). \tag{5.21}$$

So as we see, the TBD is sufficient to capture the precise level at which two features are placed in different clusters. And this information can be extracted from the one dimensional codes.

5.5 Computational aspects

As for all other chapters, we address the computational aspects with a focus on scaleability. We start by evaluating the graph tree construction, then the up/down-sampling operations and finally the feature-coding.

The complete graph-tree is computed using a modified version of the FLANN library [128]. The procedure consists of two operations: first, constructing the hierarchical k-means tree (i.e. the feature-tree), which amounts to constructing the FLANN index only, and second, compute a kNN graph per tree level using the FLANN index. Due to the limited number of iterations in the k-means clustering calls, the index construction has a $\mathcal{O}(N\log(N))$ complexity and is fast in practice. In consequence, constructing the graphs has a similar complexity. In fact, constructing only the full graph from the features has a very similar cost since the FLANN index is necessary to compute ANN searches and the full set of features corresponds to the last tree level which contains as many elements as all the levels above it combined.[3]

Then, performing a down-sampling of a signal on the graph-tree is very efficient as it costs only $\mathcal{O}(N_l)$ with l corresponding to the level from which the signal is coarsened. Indeed, both the mean and maximum voting are linear with the number of elements.

Similarly, the first phase of the up-sampling is $\mathcal{O}(N_l)$ with l corresponding to the level to which the signal is up-sampled. The second step can be accomplished using a single filtering, due to the special property of Tikhonov regularization with an ℓ_2 fidelity term (see Section 1.2.3). If the set of edges of $\mathcal{G}_l$ is written $\mathcal{E}_l$, then the regularization has a $\mathcal{O}(m|\mathcal{E}_l|)$ complexity, with m the order of the polynomial approximation for the fast filtering. Since we control the sparsity level of $\mathcal{G}_l$ this means that we have $|\mathcal{E}_l| = \mathcal{O}(N_l)$, lowering the complete up-sampling complexity to only $\mathcal{O}(N_l)$.

The dimensionality reduction algorithms presented above combine the cost of the tree construction, the application of the inner embedding algorithm routine $\mathcal{A}_e$ and optionally an

[3] Indeed, the number of elements in a tree follows a geometric series, thus $\sum_{l=0}^{H_T-1} K^l = \frac{1-K^{H_T}}{1-K} < K^{H_T}$ for $K \geq 2$.

up-sampling operation. This means that the complete cost is $O(N\log(N) + c(\mathcal{A}_e(N_{l^\star})) + N)$, with $c(\mathcal{A}_e(n))$ the complexity function of $\mathcal{A}_e$ and $l^\star$ the level on which $\mathcal{A}_e$ is applied. Also, for optimization, one does not need to compute all graphs in the CGT when using it only for dimensionality reduction, since only two (i.e. $\mathcal{G}_{l^\star}$ and $\mathcal{G}_H$) are sufficient.

Finally, the tree coding procedure is very efficient as it can be done in a single DFS tree traversal which costs $\mathcal{O}(N\log_K(N))$, i.e. the number of branches in the tree. Both encoding, and decoding, from, respectively to, the branching coefficients can be done very fast, costing only $\mathcal{O}(H_T)$ integers operations.

5.6 Experiments

In this section, we present different experiments covering the different topics introduced in this chapter. We start by an examination of meta-features, in order to see if they convey meaningful information for dimensionality reduction. The second experiment presents the different results obtained by up-sampling signals from intermediate levels to the leaves. We then explore the encoding scheme presented in Section 5.4 in two colour mapping tasks and used directly as one-dimensional embedding. We conclude this section by comparing the CGT driven dimensionality algorithms to our other meta-algorithm of compressive dimensionality reduction and to the original implementations on 2D visualization tasks.

5.6.1 Meta-features

In this first experiment, we look at the actual representation power of the meta-features by using Algorithm 10 with t-SNE as the inner algorithm on different CGT levels. In order to be able to visualize the labels in the intermediate levels, we use our down-sampling scheme on the label signal with maximum-voting.

The embeddings produced from four consecutive levels of the CGT are shown in Figure 5.4. First, we can observe that the meta-features seem to be meaningful at all levels since all embeddings provide a good class separation and clustering of the data. Apart from the first level depicted which does not really contain sufficiently many points for a decent visualization, all levels present very satisfactory visualization with only a few outliers. Overall, this tends to validate the representational power of the CGT meta-features and at the same time the effectiveness of the down-sampling procedure. Finally, note that the fifth level has a smaller number of features than its theoretical value[4] due to the CGT not being balanced.

[4] $28404 < 8^5 = 32768$

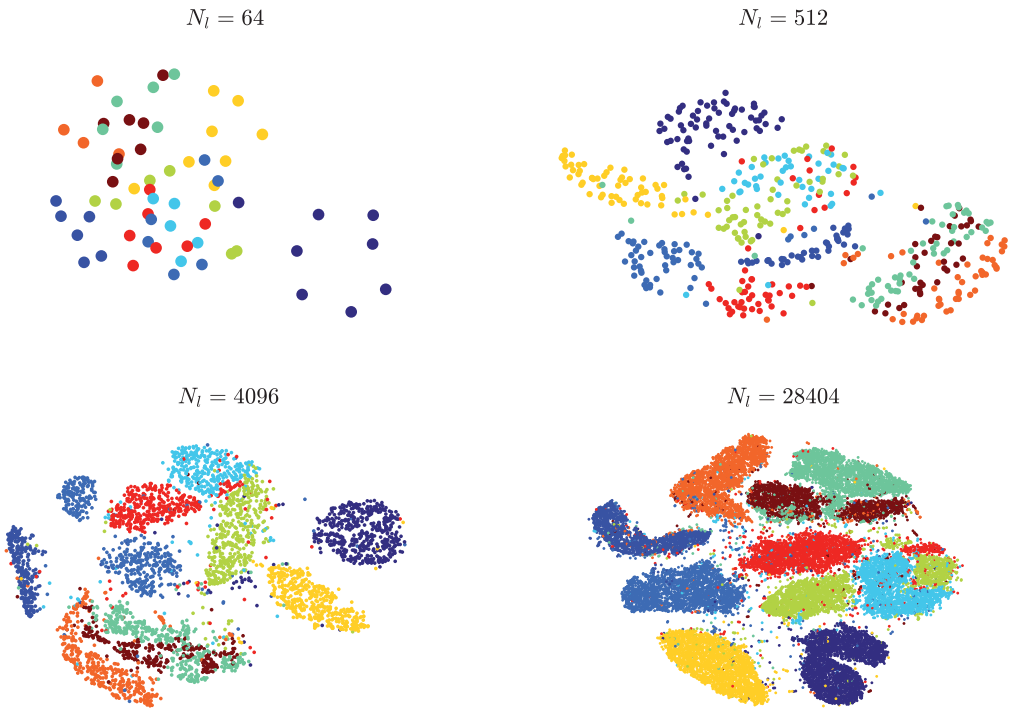

Figure 5.4 – **Meta-features visualization.** This figure shows 2D embeddings of meta-features extracted from a CGT constructed on the MNIST dataset. The branching factor was set to $K = 8$ resulting in a tree of height $H_T = 7$. The meta-features from levels 2 to 5 embedded with the t-SNE algorithm are shown in left-to-right, top-to-bottom order. The number of meta-features is indicated with N_l. The colours correspond to the label signal down-sampled to the four different levels.

5.6.2 Upsampling

In this experiment, we test the validity of our up-sampling process on a visualization task performed with Algorithm 11. Again, we use t-SNE as the inner algorithm so that we can relate to the results of the previous experiment. In fact, we reproduce much of the meta-features steps only going one step further by up-sampling the embeddings instead of down-sampling the labels.

The resulting embeddings obtained with this method are shown in Figure 5.5. First, we can see that the outcome is very satisfactory even with very coarse levels. The class separation and clustering are not perfect for the levels 2 and 3, but very good for the subsequent ones. From level 5, except some sparse noise, the cluster formations and overlap is very similar to the one produced by the algorithm on the full dataset. From this example, the up-sampling process and complete dimensionality reduction algorithm based on the CGT seem very promising.

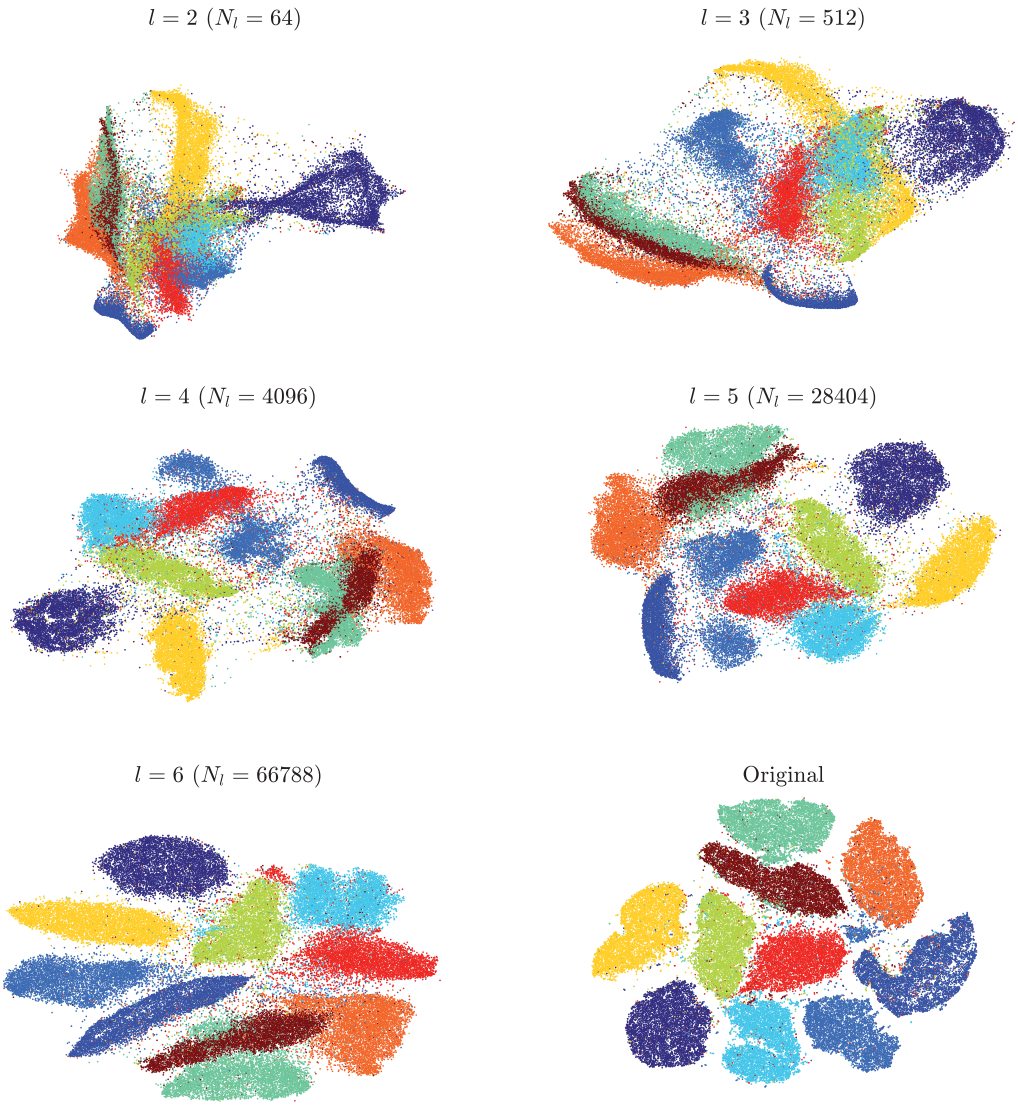

Figure 5.5 – **Up-sampled dimensionality reduction.** This figure shows 2D embeddings of meta-features extracted from a CGT constructed on the MNIST dataset. The branching factor was set to $K = 8$ resulting in a tree of height $H_T = 7$. The meta-features from levels 2 to 6 embedded with the t-SNE algorithm and up-sampled to the input size are shown in left-to-right, top-to-bottom order. And finally, the t-SNE algorithm run on the original data (i.e. the last level of the CGT) is displayed in bottom-right position. The colours correspond to the original labels.

5.6.3 Feature coding

Colour coding

In this experiment, we use the encoding scheme and colour mapping algorithm presented in Section 5.4 on 2D embeddings. The goal is to assess if the clustering made on high-dimensional

features can be made visible on 2D embeddings using the colour coding as a proxy to highlight the hierarchical clustering process. The setup of this experiment is the following: first we build two CGT with different branching factors on the MNIST dataset, second, we encode the features and generate a colour map using Algorithm 14, and finally we embed the features using both the CGT-accelerated and original t-SNE algorithm, and use the colour mapping to visualize the clustering structure.

The resulting visualizations are depicted in Figure 5.6. The first observation we can make is that the first level clustering is quite visible with both branching factors and embedding algorithms from the Hue. In addition, the different classes which we know to be well mapped to the different data clusters are comprised of well-defined colours with a well-defined spatial consistency. We also see that some colours that span across different clusters are nonetheless within well-defined regions. Moving on to the finer levels, i.e. to Saturation and Value variability, spatial coherence is not clear on the embeddings produced from the original t-SNE implementation, and data points of similar Hue, Saturation and Value seem to be scattered inside clusters of similar Hue. While not strikingly different, the visualizations produced from the CGT algorithm appear to yield small subclusters corresponding to similar Saturation and Value levels.

We propose here an explanation for these effects. As we have explained before, high-dimensional similarity cannot be perfectly represented by very low-dimensional mappings. Thus the spatial discrepancies between the different colour properties may come from the fact that dimensionality reduction algorithms favour high-level clustering to fine-grained differences between features. In other words, algorithms such as t-SNE relax fine coherence between datapoints in order to be able to preserve high-level similarity. The observation that low-level colour properties are best preserved in the CGT setting concurs with this hypothesis since embedding coordinates are first up-sampled following the tree structure before being scattered by the graph regularization.

One-dimensional embedding

In this experiment, we take the reverse postulate compared to the previous one: we use the codes directly for embedding and show the ground-truth labels on top. Although it might seem meaningless to look at one-dimensional embeddings, we will see that it provides interesting insights. For this experiment, we construct CGTs with different branching factors, visualize the one-dimensional embeddings and measure their clusterability.

The results of this experiment are shown in Figure 5.7. The first observation we can make is that the label spatial coherence is quite high. While there is clearly not a one-to-one mapping between the different one-dimensional clusters and the classes, there are only about three times more clusters than classes, quite consistently for all branching levels. Interestingly, the labels that usually mix in 2D embeddings are also intertwined in the one-dimensional embedding (e.g. brown and turquoise).

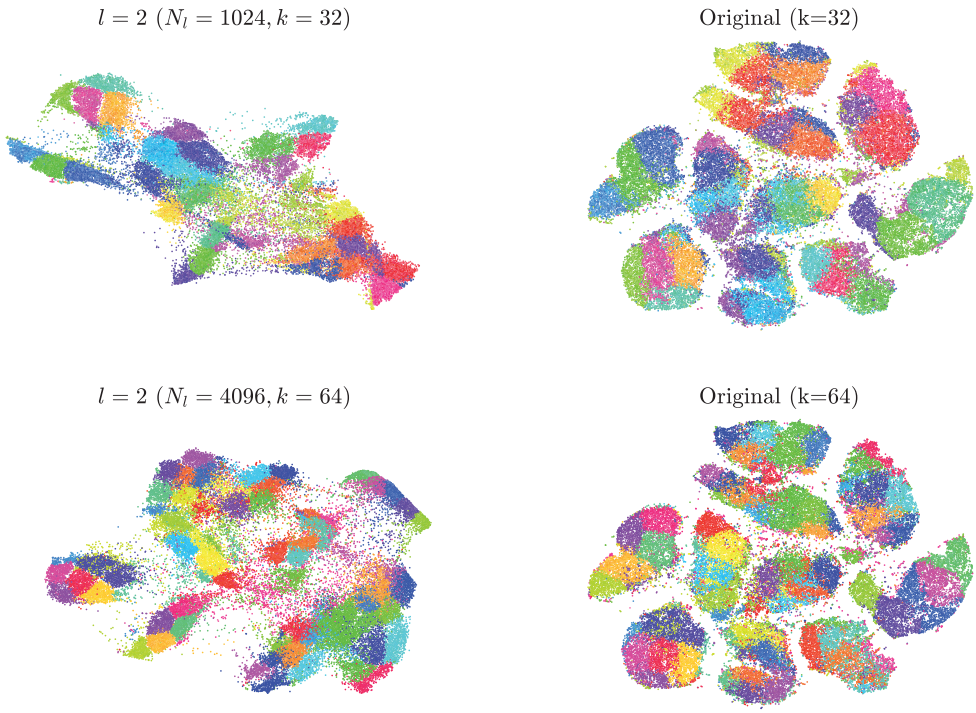

Figure 5.6 – **HSV colour-coded visualization.** In this figure 2D embeddings of the MNIST dataset produced by t-SNE on levels 2 (left) and H (right) of two Clustering Graph Trees with branching levels $K = 32$ (top) and $K = 64$ (bottom) are shown. The colours corresponds to the colour mapping produced by Algorithm 14 from the encoded features (see Section 5.4). An HSV to RGB conversion was done for rendering purposes.

The results provided by this experiment combined with the previous one seem to show that the encoding scheme proposed in Section 5.4 does carry structural information.

Audio feature visualization

In this last experiment on colour coding, we present a practical example for music visualization. Our goal is to provide a one-dimensional colour coded timeline of different songs using a CGT constructed on audio features. In order to have interesting potential interpretations, we chose three solo piano pieces each interpreted by two different pianists. Doing so introduce a few challenges because the same songs interpreted by different artists may have different characteristics (musical nuances, track duration, recording quality, etc.). Nevertheless, we extract spectral audio features in ERB scale [174] for small time windows in each track and group them in a large feature space. We then construct a CGT from this feature space and use the encoding of the features described in Section 5.4.

In Figure 5.8, we show the one-dimensional colour coded timelines for each song. At first

Figure 5.7 – **One-dimensional code embedding.** In this figure, four one-dimensional embeddings constructed using codes extracted from CGTs with branching factors $K = 4, 8, 16, 32$ are depicted. In addition, ACI scores are reported for all embeddings. The dataset on which the CGTs are constructed is MNIST and the colours correspond to the original 10-class labels.

glance, the provided visualizations do not seem to be satisfactory, since the different pairs of interpretations do not match very well. However, by looking more closely, we see that, taking into account the possible offsets of the pieces not played at the same speed (and not started at the same time), a few local patterns match very well between the different pairs of timelines.

One possible problem of these visualizations is that the order of the tree branching coefficient is random and does not take any similarity of the cluster centres into account. In order to compensate for this effect, we can use the cluster centres or their embedding to provide a meaningful ordering of the tree branching coefficients, thus providing a more structured encoding and colour mapping scheme.

The result of the visualization using a reordering of the branching coefficients is shown in Figure 5.8. With this modified scheme, the tracks can be paired instantly, and a closer look shows very distinctive patterns common to the different interpretations of the same piece. For example, we see that version a) of the nocturne starts after a longer period of silence than version b), but then the first sequence of chords, repeated two times, is clearly visible in both timelines.

But the most fascinating interpretation is that in the middle segment of Scriabin's Etude, the colours become very close to the ones displayed in Chopin's nocture. Now, this middle passage in the Etude is written in the key of E major, whose relative minor is C sharp minor, which is the key of the Nocturne. This seems to indicate that the colour mapping is related to the harmonic content of the pieces, which is an interesting achievement.

5.6.4 Natural datasets visualization

In this last experiment we compare both CGT algorithms to our reference dimensionality reduction methods t-SNE and LargeVis, as well as the compressive meta-algorithm presented in Chapter 4, on visualization tasks. Similarly to what was done in the previous chapters, we provide quantitative quality measures in Tables 5.1-5.6 and show a few selected 2D embeddings in Figures 5.10-5.12. Note that, due to the fact that the hierarchical k-means procedure

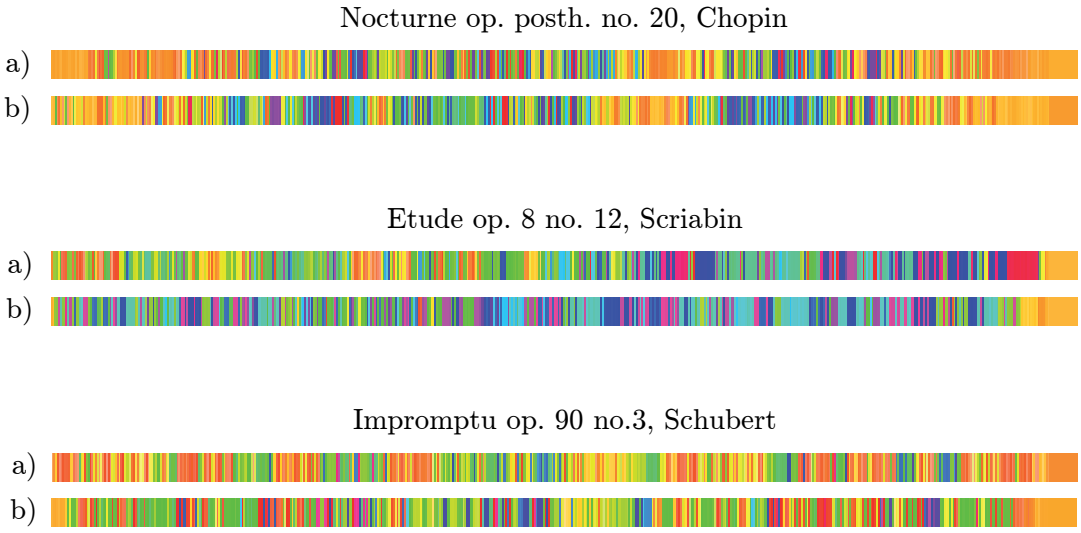

Figure 5.8 – **Audio colour coding (raw).** This figure shows the colour-coded audio features computed from different songs. The features are displayed from the start to the end of the track, from left to right. The colour scheme is derived by constructing a CGT on the set of all ERB features from all tracks and by using the colour mapping described in Algorithm 14. The different songs are three solo piano pieces each played by different pianists. The first one is the Nocturne op. posthume no. 20 composed by Frederic Chopin and interpreted by Paul Barton (a) and Vladimir Ashkenazy (b). The second piece is the Etude op. 8 no. 12 composed by Alexander Scriabin and interpreted by Vladimir Horowitz (a) and Harvey Van Cliburn (b). The last one is the Impromptu op.90 no.3 composed by Franz Schubert and interpreted by Vladimir Horowitz (a) and Paul Barton (b).

is unable to use the cosine distance, the CGT was not applicable on the two sparse datasets.

First, the timing reported in Table 5.1 show that the fastest methods are compressive t-SNE and CGT t-SNE, with a slight edge for the latter. On large datasets, CGT t-SNE is two orders of magnitude faster than the original implementation. All LargeVis variants have similar speed, with CGT LargeVis being the fastest of the three. Overall, the acceleration provided by Algorithm 11 is substantial.

Second, the clusterability measures and generalization error provided in Tables 5.2 and 5.3 give the original t-SNE and LargeVis algorithms as the best ones and the CDR versions as the worst. The embeddings provided by CGT algorithms score between these two extremes, better than CDR but worse than the original methods. This analysis is confirmed by the noise level and NN precision reported in Table **??** and 5.5 respectively, which, however, levels a bit the gap between the different methods.

Finally, the cluster split values are quite similar for all algorithms with CDR t-SNE scoring consistently best and t-SNE worst. Both CGT accelerated methods have intermediate values. Similarly to what was hypothesized in the previous chapter, the cause for these similar ACC

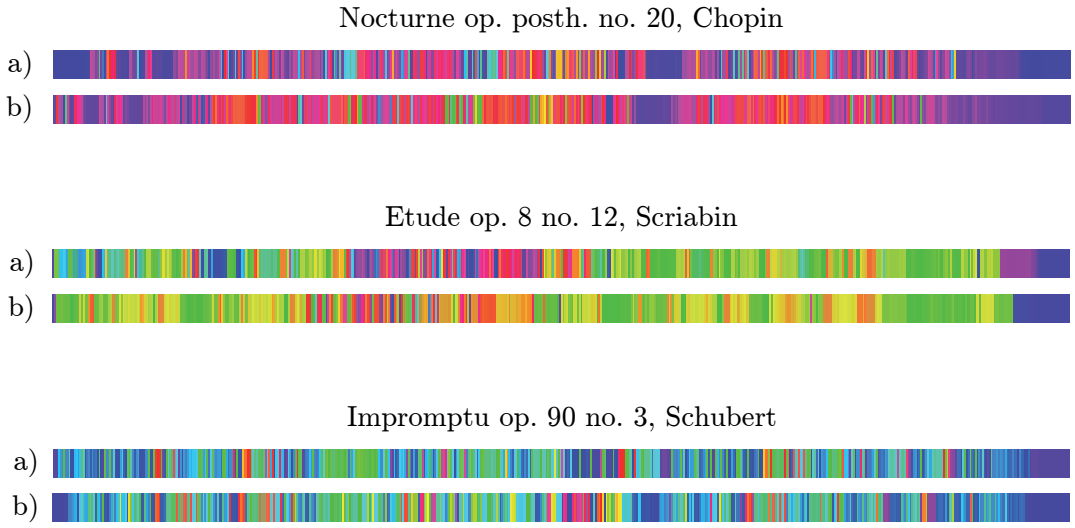

Figure 5.9 – **Audio colour coding (reordered).** This figure shows the colour-coded audio features computed from different songs. The features are displayed from the start to the end of the track, from left to right. The colour scheme is derived by constructing a CGT on the set of all ERB features from all tracks and by using the colour mapping described in Algorithm 14 using reordering of the codes indices from the embedding. The different songs are three solo piano pieces each played by different pianists. The first one is the Nocturne op. posthume no. 20 composed by Frederic Chopin and interpreted by Paul Barton (a) and Vladimir Ashkenazy (b). The second piece is the Etude op. 8 no. 12 composed by Alexander Scriabin and interpreted by Vladimir Horowitz (a) and Harvey Van Cliburn (b). The last one is the Impromptu op.90 no.3 composed by Franz Schubert and interpreted by Vladimir Horowitz (a) and Paul Barton (b).

scores is probably that the inner routines are essentially the same ones. The lowest ACC happens to be measured for the less clustered data since a class can be located in unique continuous clusters. For embeddings which are well defined spatially, discontinuity in classes tends to immediately generate cluster splits, which yields high ACC values.

Figures 5.10-5.12 globally confirm the results from the quality measures. We see that the CGT variant provides very good embeddings with good class separation and decent cluster formation. The main differences with the original implementation are the presence of more sparse noise and not very sharp cluster boundaries.

5.7 Conclusion

In this chapter, we have presented a new multi-resolution structure called the Clustering Graph Tree which has interesting characteristics. First, it gives an access to different discriminatory levels through meta-features. Second, up-sampling and down-sampling operators allow the design of a very efficient meta-algorithm for dimensionality reduction. Finally, we have proposed a one-dimensional encoding of the feature set using a tree traversal which can be used to create structured colour mappings or one-dimensional embeddings. All methods were

Time [s]	tsne	cdr:tsne	cgt:tsne	largevis	cdr:largevis	cgt:largevis
caltech101-caffenet	89.52	**27.31**	**58.46**	219.00	**232.24**	**1044.33**
caltech256-caffenet	**469.09**	**118.71**	**127.27**	256.87	**324.24**	292.46
cifar10-cnn	**1410.91**	346.35	**75.19**	315.21	**552.65**	**251.90**
cmupie	130.22	**35.18**	**36.74**	**219.71**	**243.88**	219.29
coil20	10.82	**9.02**	**9.45**	**216.03**	**215.84**	208.26
fma-echonest	139.97	**41.60**	**34.40**	**220.09**	**249.65**	212.68
fma-rosa	**2475.82**	542.23	**65.75**	413.09	**791.51**	**246.73**
hiva	**949.41**	**197.30**	**220.38**	279.98	402.83	**489.12**
mnist	**1271.24**	346.64	**59.02**	336.87	**549.16**	**249.22**
norb	**884.97**	**190.00**	318.24	**291.14**	394.10	**507.08**
sylva	**3336.83**	810.26	**48.93**	490.75	**1024.58**	**238.32**
usps	94.30	**27.41**	**25.99**	**215.55**	**236.33**	213.57

Table 5.1 – **Embedding time.** This table reports the embedding time (in seconds) for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

ACI	tsne	cdr:tsne	cgt:tsne	largevis	cdr:largevis	cgt:tsne
caltech101-caffenet	**0.14**	**0.49**	0.41	**0.13**	**0.49**	0.41
caltech256-caffenet	**0.39**	**0.83**	0.76	**0.44**	**0.82**	0.76
cifar10-cnn	**0.28**	**0.48**	**0.47**	**0.27**	0.47	0.47
cmupie	**0.42**	**0.90**	0.86	**0.65**	**0.90**	0.86
coil20	**0.06**	0.03	**0.02**	**0.05**	0.03	**0.02**
fma-echonest	**0.87**	**0.93**	0.91	**0.88**	**0.94**	0.91
fma-rosa	**0.92**	**0.95**	0.93	**0.93**	**0.95**	0.93
hiva	**0.62**	**0.71**	0.66	**0.65**	**0.71**	0.68
mnist	**0.05**	**0.30**	**0.17**	**0.06**	0.14	0.17
norb	**0.52**	**0.74**	0.70	**0.60**	**0.73**	0.70
sylva	**0.35**	**0.96**	0.90	**0.33**	**0.94**	0.90
usps	**0.03**	**0.07**	0.06	**0.04**	**0.07**	0.06

Table 5.2 – **Embedding ACI.** This table reports the ACI score for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

tested in various experiments demonstrating the validity of these contributions in practice.

For future directions, we think it would be worthwhile to study the theoretical relations linking the graphs from the different levels of the CGT, using both clustering theory and

1NN	tsne	cdr:tsne	cgt:tsne	largevis	cdr:largevis	cgt:tsne
caltech101-caffenet	**0.18**	**0.47**	0.39	**0.19**	**0.49**	0.39
caltech256-caffenet	**0.47**	**0.80**	0.74	**0.53**	**0.81**	0.74
cifar10-cnn	**0.28**	**0.44**	0.41	**0.29**	**0.44**	0.41
cmupie	**0.26**	**0.83**	0.78	**0.55**	**0.83**	0.78
coil20	**0.05**	**0.03**	0.03	**0.04**	**0.03**	0.04
fma-echonest	**0.70**	**0.74**	0.74	**0.71**	**0.74**	0.74
fma-rosa	**0.77**	**0.80**	**0.79**	0.79	**0.79**	0.79
hiva	**0.04**	**0.05**	0.04	**0.05**	**0.05**	0.05
mnist	**0.05**	**0.28**	**0.19**	**0.06**	0.17	0.19
norb	**0.38**	**0.59**	0.55	**0.46**	**0.58**	0.55
sylva	**0.05**	**0.10**	0.10	**0.04**	**0.10**	0.10
usps	**0.04**	**0.11**	0.08	**0.05**	**0.11**	0.08

Table 5.3 – **Embedding 1NN.** This table reports the 1NN scores for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

NNP	tsne	cdr:tsne	cgt:tsne	largevis	cdr:largevis	cgt:largevis
caltech101-caffenet	**0.35**	**0.27**	0.29	**0.36**	**0.27**	0.31
caltech256-caffenet	**0.22**	**0.10**	0.13	**0.22**	**0.11**	0.13
cifar10-cnn	**0.15**	**0.07**	0.08	**0.14**	**0.07**	0.09
cmupie	**0.39**	**0.30**	0.36	**0.40**	**0.30**	0.34
coil20	**0.29**	**0.27**	**0.28**	**0.27**	0.28	0.27
fma-echonest	**0.38**	**0.22**	0.31	**0.38**	**0.24**	0.31
fma-rosa	**0.32**	**0.18**	0.21	**0.28**	**0.16**	0.21
hiva	**0.41**	**0.18**	0.29	**0.42**	**0.18**	0.28
mnist	**0.32**	**0.17**	0.21	**0.29**	**0.18**	0.21
norb	**0.33**	**0.22**	0.28	**0.30**	**0.23**	0.28
sylva	**0.11**	**0.02**	0.04	**0.11**	**0.02**	0.04
usps	**0.49**	**0.35**	0.46	**0.49**	**0.38**	0.46

Table 5.4 – **Embedding NNP.** This table reports the NNP scores for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

spectral GSP analysis. In addition, it could be interesting to consider both the hierarchical clustering procedure and the graph construction simultaneously, potentially accelerating the process. Also, despite its good performance and cheap cost, one could explore the use of other regularization processes for the up-sampling operator. Finally, we think that the tree-based

ACN	tsne	cdr:tsne	cgt:tsne	largevis	cdr:largevis	cgt:tsne
caltech101-caffenet	**0.20**	**0.32**	0.30	**0.21**	**0.33**	0.30
caltech256-caffenet	**0.36**	**0.49**	0.48	**0.37**	**0.50**	0.48
cifar10-cnn	**0.24**	**0.30**	0.29	**0.23**	**0.29**	0.29
cmupie	**0.36**	**0.47**	0.43	**0.36**	**0.46**	0.43
coil20	**0.04**	**0.07**	0.06	**0.02**	**0.07**	0.06
fma-echonest	**0.37**	**0.38**	**0.38**	0.38	**0.38**	0.38
fma-rosa	**0.41**	**0.43**	0.41	**0.41**	**0.42**	0.41
hiva	**0.31**	0.24	**0.21**	**0.26**	0.24	**0.20**
mnist	**0.07**	**0.20**	0.16	**0.06**	**0.18**	0.16
norb	**0.34**	**0.32**	**0.30**	0.30	0.31	**0.30**
sylva	**0.02**	0.20	**0.23**	**0.02**	0.21	**0.23**
usps	**0.06**	**0.16**	0.12	**0.06**	**0.16**	0.12

Table 5.5 – **Embedding ACN.** This table reports the ACN scores for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

ACC	tsne	cdr:tsne	cgt:tsne	largevis	cdr:largevis	cgt:tsne
caltech101-caffenet	**4.61**	**2.05**	**2.51**	2.87	**3.24**	2.51
caltech256-caffenet	**6.80**	**3.84**	**4.68**	4.72	**5.93**	4.68
cifar10-cnn	**12.28**	**5.40**	8.61	**9.89**	**6.85**	8.61
cmupie	**30.77**	**15.17**	**26.27**	24.93	**17.28**	26.27
coil20	**3.97**	**0.54**	1.95	**14.50**	**1.89**	3.27
fma-echonest	**34.97**	**14.60**	24.90	**26.62**	**21.26**	24.90
fma-rosa	**23.55**	**13.52**	19.24	**22.39**	**16.12**	19.24
hiva	**30.29**	**15.17**	21.45	**25.94**	**14.81**	23.18
mnist	**13.08**	**6.02**	8.67	**10.47**	**7.05**	8.56
norb	**26.93**	**23.48**	24.01	**34.66**	**41.80**	28.41
sylva	**21.08**	**9.05**	12.89	**28.99**	**10.41**	14.25
usps	**13.67**	**3.92**	5.62	**10.75**	**4.47**	6.10

Table 5.6 – **Embedding ACC.** This table reports the ACC scores for different algorithms (columns) on different datasets (lines). Green indicate the best value on the line, red the worst, light green the second best and orange the second worst. Non-termination status are reported with the following acronyms : timeout (TO), runtime error (RE), out of memory (OM) and non-applicable (NA).

encoding could be enhanced by ordering the branches according to the similarity matrix of their cluster centres directly when constructing the tree, since the experiment in Section 5.6.3 already gives promising results. This way, the mapping, distances and colour schemes would be more tightly related to the data.

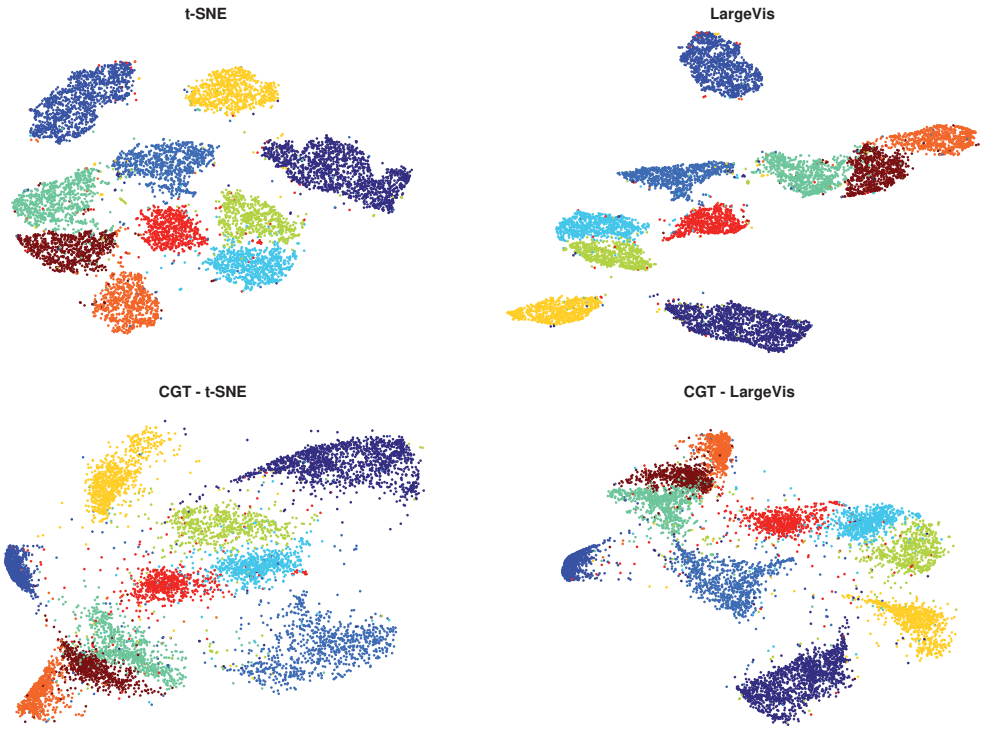

Figure 5.10 – Visualization of the USPS dataset using t-SNE and LargeVis and their CGT equivalent. The colour map corresponds to the labels.

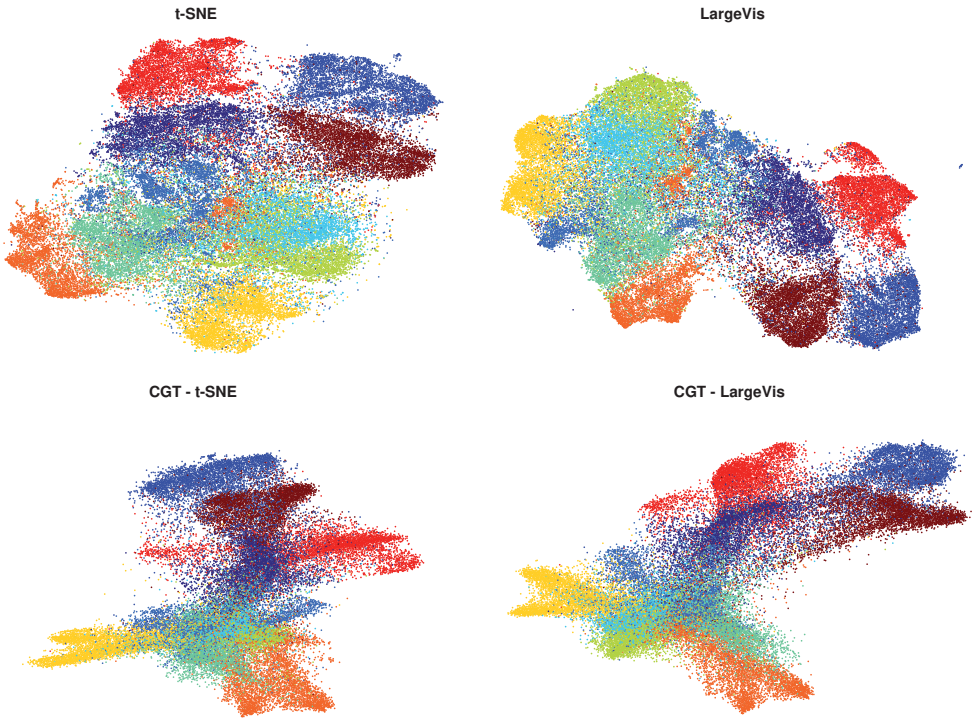

Figure 5.11 – Visualization of the Cifar10 dataset using t-SNE and LargeVis and their CGT equivalent. The colour map corresponds to the labels.

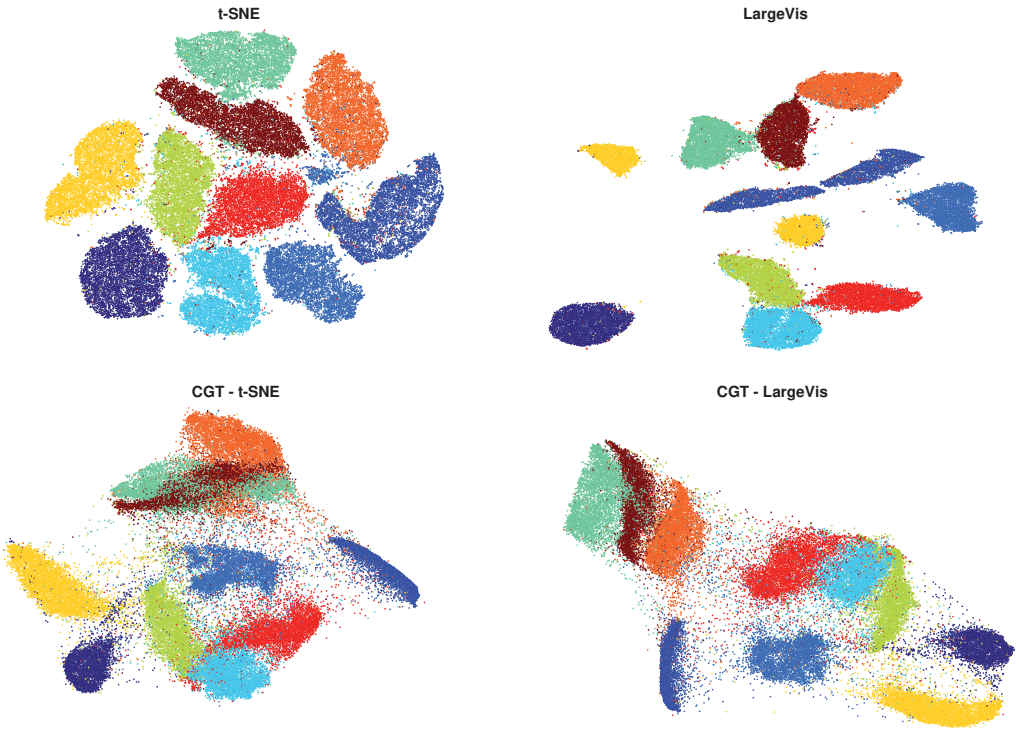

Figure 5.12 – Visualization of the MNIST dataset using t-SNE and LargeVis and their CGT equivalent. The colour map corresponds to the labels.

The only way of discovering the limits of the possible
is to venture a little way past them into the impossible.
— Arthur C. Clarke

6 Discussion

Having completed our journey in the world of dimensionality reduction viewed from the Graph Signal Processing perspective, we have seen that graphs are tightly intertwined with the structure of data, as it was hypothesized in the introduction, but it is time to take a step back. At the beginning of this work, we have listed open issues in the field. Let us review them now, to put our contributions in perspective.

- First, we claimed that methods to measure fine characteristics of dimensionality reduction algorithms lacked in the set of existing formal measures. We thus proposed three new supervised methods in Chapter 2, based on graphs constructed on the embeddings, to assess different quality properties: clusterability, noise level and cluster concentration (or split). We have seen that, while behaving as expected on synthetic datasets and providing useful insights on natural data, the ACC needs to be carefully interpreted to be really useful.

  We think that we contributed a useful general approach and tools for the problem of quantifying the quality of embeddings, but other directions might be worth exploring. In particular, it should be interesting to generalize the different graph-based measures we introduced to the unsupervised setting. In a different direction, it would be valuable to design formal measures that are not simple averages but either give other statistical indicators or return multidimensional data. While being the most difficult measure to interpret, we believe that measuring the cluster split (or more precisely, the class-cluster mapping) is key to a good evaluation of dimensionality reduction algorithms. Finally, we also observed that visual inspection is a much-needed step to put quality scores into context and that this task cannot be easily discarded.
- Second, we claimed that dimensionality reduction was often tackled in a generic way without specializing for tasks such as either clustering or visualization. Although none of the techniques we contributed was specialized by design, it is reasonable to say that FEARS (Chapter 3), is much more useful to generate features for clustering than for visualization tasks. The two meta-algorithms via CDR (Chapter 4) and CGT (Chap-

ter 5) are not, by essence, specialized for any particular task a priori, and inherit the potential specialization of their inner embedding algorithm. In general, we have mostly considered visualization tasks because they allow for a direct inspection and an easy validation of our different working hypotheses. On this topic, we think that visualization of high-dimensional data should not be constrained to fixed 2D embeddings. Indeed, we think that data should be displayed dynamically and that visualization should provide possibilities for interaction. This general objective could inspire new approaches in visualization.

- Third, we stated that scaleability was one of the most important issues of dimensionality reduction today. This specific problem has been the main focus of all our contributions throughout this thesis and we have shown that GSP is a key enabler of data-adapted accelerated techniques. Nevertheless, GSP alone was not sufficient to tackle this problem and was associated with concepts from other fields. We used random matrix theory in FEARS (Section 3.2.1), compressive sampling in CDR (Section 4.3), and hierarchical clustering and tree-coding in CGT (Section 5.2 and 5.4).

  From all tools of the GSP framework, the two most powerful ones were the fast-filtering via polynomials and the localization operator. The former providing the main acceleration and the latter an access to small neighbourhoods with traditional graph filters. In each of our contributions, the acceleration was provided through approximation, creating a need for a good balance between lost accuracy and speedup. Due to the size of the datasets we can encounter today, we sought for scaleability improvements measured in order of magnitudes, and most of our contributions achieved this goal.

  We actually reached the scale for which even very scalable techniques are hindered by hardware constraints rather than implementation efficiency. Indeed, the common assumption in this work was that it was possible to store the datasets in RAM, which is not possible for most dataset above the millions of datapoints mark. Therefore, we think that future works focusing on the scaleability of dimensionality reduction methods should first address the current hardware limits through data streaming and distributed processing. In fact, extending our contributions could be achieved without changing fundamentally the design principles. Indeed, for example, graph filtering can be efficiently distributed using message passing algorithms. This direction is probably the next most needed step for GSP applied to large scale problems.

- Fourth, we claimed that over-discrimination of the feature-space is an issue, creating unnecessary large datasets and artificially expanding the semantic information on repeated features. We have addressed this problem using the CGT (Section 5.2) and its ability to generate meta-features sets of almost any desired size. While our contributions seem promising, this domain has not been explored much and could lead to interesting extensions. An obvious choice with today's trends would be to train deep learning algorithms to create feature sets of a desired precision, or mapping a desired objective set.

- Lastly, we wrote in the introduction that the feature-extraction process usually discarded

the context, or the natural embedding of the features. This issue was in the end not addressed by our different contributions and is left for future work. An idea would be to regularize the dimensionality reduction process by using both the original structure of the features and their usual kNN graph in the feature-space. Leveraging multi-graphs seems to fit well in this context and we think it would benefit from further research.

This completes our review of the issues that motivated this work and how they were addressed. Another problem remains, however, which was not mentioned until then, and which is rarely ever spoken about in the dimensionality reduction literature: obtaining good data is, in itself, hard. More precisely, a great deal of work is done to preprocess the data prior to its use as input of an algorithm of interest. More often than not, this preprocessing is a scarcely documented manual operation that is considered as completely independent of the "real" research. This is the root of an important problem: the preprocessing is not considered in any evaluation metric while it may be key to either a faster processing time or better quality results. To solve this issue, we think that the preprocessing step should either be included in the evaluation process or standardized in libraries or repositories of freely accessible, open, pre-processed data. In an attempt to address this specific issue, all preprocessing steps applied to the datasets used in this thesis are documented in Appendix B. While we would like to provide all this standardized data for further research and benchmarking purposes, the sharing of most of the data we use is prohibited due to licensing issues.

On this topic, we firmly believe in open and reproducible research. We think that it is the only way to accomplish work while respecting the scientific method. For this reason, all the algorithms presented in this work, along with the code to perform all the examples and experiments are freely available under open source licences (see Appendix B).

We conclude by thanking the reader for his interest and attention.

A Proofs

A.1 Spectral Graph Theory

A.1.1 Combinatorial Laplacian

The combinatorial graph Laplacian $\mathbf{L} : \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}^{|\mathcal{V}|}$ is given by :

$$\begin{aligned}
(\mathbf{Lx})[i] &= ((\mathbf{D}-\mathbf{W})\mathbf{x})[i] \\
&= \mathbf{x}_i \mathbf{D}(i,i) - \sum_j \mathbf{W}(i,j)\mathbf{x}_j \\
&= \mathbf{x}_i \sum_j \mathbf{W}(i,j) - \sum_j \mathbf{W}(i,j)\mathbf{x}_j \\
&= \sum_j \mathbf{W}(i,j)\left(\mathbf{x}_i - \mathbf{x}_j\right) \\
&= \frac{1}{2}\left(\sum_j \mathbf{W}(i,j)\left(\mathbf{x}_i - \mathbf{x}_j\right) + \sum_j \mathbf{W}(i,j)\left(\mathbf{x}_i - \mathbf{x}_j\right)\right) \\
&= -\frac{1}{2}\left(\sum_{j \text{ s. t. } (v_j,v_i)\in\mathcal{E}} \mathbf{W}(i,j)\left(\mathbf{x}_j - \mathbf{x}_i\right) - \sum_{i \text{ s. t. } (v_i,v_j)\in\mathcal{E}} \mathbf{W}(i,j))\left(\mathbf{x}_i - \mathbf{x}_j\right)\right) \\
&= -\frac{1}{2}\left(\sum_{j \text{ s. t. } (v_j,v_i)\in\mathcal{E}} \sqrt{\mathbf{W}(i,j)}\sqrt{\mathbf{W}(i,j)}\left(\mathbf{x}_j - \mathbf{x}_i\right) - \sum_{i \text{ s. t. } (v_i,v_j)\in\mathcal{E}} \sqrt{\mathbf{W}(i,j)}\sqrt{\mathbf{W}(j,i)}\left(\mathbf{x}_i - \mathbf{x}_j\right)\right) \\
&= (-\mathrm{div}_{\mathcal{G}}\nabla_{\mathcal{G}}\mathbf{x})[i].
\end{aligned}$$

Thus, in matrix form :

$$\mathbf{Lx} = (\mathbf{D}-\mathbf{W})\mathbf{x} = -\mathrm{div}_{\mathcal{G}}\nabla_{\mathcal{G}}\mathbf{x}. \tag{A.1}$$

A.1.2 Localization operator and kernels

Proof of Lemma 3

Proof. Let us prove the properties in order.

- For the first property we have :

$$\begin{aligned} \mathcal{T}_i g[j] &= \sum_\ell g(\boldsymbol{\lambda}_\ell) \boldsymbol{u}_\ell^*[i] \boldsymbol{u}_\ell[j] \\ &= \sum_\ell g(\boldsymbol{\lambda}_\ell) \boldsymbol{u}_\ell^*[j] \boldsymbol{u}_\ell[i] \\ &= \mathcal{T}_j g[i] \end{aligned}$$

  with the chain of equalities working because $\boldsymbol{u}_\ell \in \mathbb{R}^N$.

- We derive the second property in the Fourier domain, using Parseval (Eq. (1.29)) :

$$\begin{aligned} \langle \mathcal{T}_i g, \mathcal{T}_j g \rangle &= \langle \hat{\mathcal{T}_i} g, \hat{\mathcal{T}_j} g \rangle \\ &= \sum_\ell \hat{\mathcal{T}_i} g[\ell] \hat{\mathcal{T}_j} g^*[\ell] \\ &= \sum_\ell g(\boldsymbol{\lambda}_\ell)^2 \boldsymbol{u}_\ell[i] \boldsymbol{u}_\ell^*[j] \\ &= \sum_\ell g(\boldsymbol{\lambda}_\ell)^2 \boldsymbol{u}_\ell^*[i] \boldsymbol{u}_\ell[j] \\ &= \mathcal{T}_i g^2[j] \end{aligned}$$

  with the chain of equalities working because $\boldsymbol{u}_\ell \in \mathbb{R}^N$.

- For the last property we have :

$$\begin{aligned} \mathcal{T}_i g^2[j] &= \langle \mathcal{T}_i g, \mathcal{T}_j g \rangle \\ &= \|\mathcal{T}_i g\| \, \|\mathcal{T}_j g\| . \end{aligned}$$

  where the first equality comes from the second property and the last one from Cauchy-Schwartz.

□

Proof of Lemma 6

Proof. Let us first write the localization operator as :

$$\begin{aligned} \mathcal{T}_i g_h[i] &= \sum_{\ell=0}^{N-1} g_h(\boldsymbol{\lambda}_\ell) \boldsymbol{u}_\ell^*[i] \boldsymbol{u}_\ell[j] \\ &= g(\mathbf{L}) \delta_{i,j} \\ &= \gamma e^{-\tau \mathbf{L}} \delta_{i,j} \end{aligned}$$

Then, knowing that $\mathbf{L} = \mathbf{D} - \mathbf{W}$, we get :

$$\gamma e^{-\tau \mathbf{L}} = \gamma e^{-\tau(\mathbf{D}-\mathbf{W})} = \gamma e^{-\tau \mathbf{D}} e^{\tau \mathbf{W}}$$

using the Glauber formula for the last equality since $\mathbf{D}$ and $\mathbf{W}$ commute (i.e. $\mathbf{DW} = \mathbf{WD}$). Now, all elements of $e^{-\tau \mathbf{D}}$ are positive since $\mathbf{D}$ is a diagonal matrix of positive elements. Finally, since $\mathbf{W}$ contains only positive (or null) elements, we have that $e^{\tau W} = \sum_{k=0}^{\infty} \frac{1}{k!} (\tau W)^k \geq 0$, which concludes the proof. □

A.2 Localized distances

A.2.1 Kernelized Diffusion Distance

KDD definitions equivalence

The equivalence between Eq. (2.19) and Eq. (2.20) is derived as follows :

$$\begin{aligned}
\mathrm{KDD}(i,j)^2 &= \|\mathcal{T}_i g - \mathcal{T}_j g\| \\
&= \sum_n \left(\sum_\ell g(\boldsymbol{\lambda}_\ell) \boldsymbol{u}_\ell^*[i] \boldsymbol{u}_\ell[n] - \sum_\ell g(\boldsymbol{\lambda}_\ell) \boldsymbol{u}_\ell^*[j] \boldsymbol{u}_\ell[n] \right)^2 \\
&= \sum_n \left(\sum_\ell g(\boldsymbol{\lambda}_\ell) (\boldsymbol{u}_\ell^*[i] - \boldsymbol{u}_\ell^*[j]) \boldsymbol{u}_\ell[n] \right)^2 \\
&= \sum_n \sum_\ell g(\boldsymbol{\lambda}_\ell)^2 (\boldsymbol{u}_\ell^*[i] - \boldsymbol{u}_\ell^*[j])^2 \boldsymbol{u}_\ell^2[n] \\
&= \sum_\ell g(\boldsymbol{\lambda}_\ell)^2 (\boldsymbol{u}_\ell^*[i] - \boldsymbol{u}_\ell^*[j])^2 \sum_n \boldsymbol{u}_\ell^2[n] \\
&= \sum_\ell g(\boldsymbol{\lambda}_\ell)^2 (\boldsymbol{u}_\ell^*[i] - \boldsymbol{u}_\ell^*[j])^2
\end{aligned}$$

which proves the equivalence by taking the square root on both sides.

Proof of Theorem 1

Proof. Let us verify the properties in order :

1. This property holds trivially due to the positivity of the ℓ_2-norm.

2. We have

$$\begin{aligned}\mathrm{KDD}(x,y) &= \|\mathcal{T}_x g - \mathcal{T}_y g\| \\ &= \sqrt{\sum_\ell g(\boldsymbol{\lambda}_\ell)^2 (\boldsymbol{u}_\ell^*[x] - \boldsymbol{u}_\ell^*[y])^2} \\ &= \sqrt{\sum_\ell g(\boldsymbol{\lambda}_\ell)^2 (\boldsymbol{u}_\ell^*[y] - \boldsymbol{u}_\ell^*[x])^2} \\ &= \|\mathcal{T}_y g - \mathcal{T}_x g\| \\ &= \mathrm{KDD}(y,x)\end{aligned}$$

3. We have

$$\begin{aligned}\mathrm{KDD}(x,z) &= \|\mathcal{T}_x g - \mathcal{T}_z g\| \\ &= \|\mathcal{T}_x g - \mathcal{T}_y g + \mathcal{T}_y g - \mathcal{T}_z g\| \\ &\leq \|\mathcal{T}_x g - \mathcal{T}_y g\| + \|\mathcal{T}_y g - \mathcal{T}_z g\| \\ &= \mathrm{KDD}(x,y) + \mathrm{KDD}(y,z)\end{aligned}$$

which holds using the triangle inequality for vectors.

□

Proof of Theorem 2

Proof. Properties 1-3 are still valid as stated in Theorem 1.

Now let us check Property 4.

- We first prove $x = y \Rightarrow \mathrm{KDD}(x,y) = 0$:

$$\begin{aligned}\mathrm{KDD}(x,y) &= \mathrm{KDD}(x,x) \\ &= \|\mathcal{T}_x g - \mathcal{T}_x g\| \\ &= 0\end{aligned}$$

- Now let us check that $\mathrm{KDD}(x,y) = 0 \Rightarrow x = y$. We do it by contradiction and thus want to find any pair x, y, with $x \neq y$ for which $\mathrm{KDD}(x,y) = 0$.

  In particular we need that :

$$\mathrm{KDD}(x,y) = \sqrt{\sum_\ell g(\boldsymbol{\lambda}_\ell)^2 (\boldsymbol{u}_\ell^*[x] - \boldsymbol{u}_\ell^*[y])^2} = 0 \tag{A.2}$$

  with $x \neq y$. Since g is full rank then $g(\boldsymbol{\lambda}_\ell) > 0$, $\forall \ell$ and thus the only way for (A.2) to hold is if $\boldsymbol{u}_\ell^*[x] = \boldsymbol{u}_\ell^*[y]$, $\forall \ell$. In other words, it would imply that the lines x and y of $\mathbf{U}$ are identical. Since $\mathbf{U}$ is a basis, it implies that all its lines are orthonormal, which means there exist no pair x, y such as (A.2) hold, and thus the contradiction is established, which concludes the proof.

□

A.2.2 Localized Kernel Distance

Proof of Theorem 7

Proof. Now let us verify the properties one by one :

1. We have using (4.28) :

$$\begin{aligned} \mathrm{LKD}(x,y) &= 1 - \frac{\langle \mathcal{T}_x g, \mathcal{T}_y g \rangle}{\|\mathcal{T}_x g\| \|\mathcal{T}_y g\|} \\ &\geq 0 \end{aligned}$$

   where the last inequality comes from the third property of Lemma 3.

2. Let us verify that $x = y \Rightarrow \mathrm{LKD}(x,y) = 0$:

$$\begin{aligned} \mathrm{LKD}(x,y) &= \mathrm{LKD}(x,x) \\ &= 1 - \frac{\mathcal{T}_x g^2[x]}{\|\mathcal{T}_x g\| \|\mathcal{T}_x g\|} \\ &= 1 - \frac{\sum_\ell g(\boldsymbol{\lambda}_\ell)^2 \boldsymbol{u}_\ell^*[x] \boldsymbol{u}_\ell[x]}{\|\mathcal{T}_x g\|^2} \\ &= 1 - \frac{\sum_\ell (g(\boldsymbol{\lambda}_\ell) \boldsymbol{u}_\ell[x])^2 \sum_n \boldsymbol{u}_\ell[n]^2}{\|\mathcal{T}_x g\|^2} \\ &= 1 - \frac{\sum_n \sum_\ell (g(\boldsymbol{\lambda}_\ell) \boldsymbol{u}_\ell[x] \boldsymbol{u}_\ell[n])^2}{\|\mathcal{T}_x g\|^2} \\ &= 1 - \frac{\|\mathcal{T}_x g\|^2}{\|\mathcal{T}_x g\|^2} \\ &= 0 \end{aligned}$$

3. Finally, we have

$$\begin{aligned} \mathrm{LKD}(x,y) &= 1 - \frac{\mathcal{T}_x g^2[y]}{\|\mathcal{T}_x g\| \|\mathcal{T}_y g\|} \\ &= 1 - \frac{\sum_\ell g(\boldsymbol{\lambda}_\ell)^2 \boldsymbol{u}_\ell^*[x] \boldsymbol{u}_\ell[y]}{\|\mathcal{T}_x g\| \|\mathcal{T}_y g\|} \\ &= 1 - \frac{\sum_\ell g(\boldsymbol{\lambda}_\ell)^2 \boldsymbol{u}_\ell^*[y] \boldsymbol{u}_\ell[x]}{\|\mathcal{T}_x g\| \|\mathcal{T}_y g\|} \\ &= 1 - \frac{\mathcal{T}_y g^2[x]}{\|\mathcal{T}_x g\| \|\mathcal{T}_y g\|} \\ &= \mathrm{LKD}(y,x) \end{aligned}$$

□

Proof of Theorem 8

Proof. Properties 1 and 3, as well as the backward implication are still valid as stated in Theorem 7.

Now let us check that $\mathrm{LKD}(x,y)=0 \Rightarrow x=y$.

We want to do it by contradiction and thus search any $x, y,\ x \neq y$ for which $\mathrm{LKD}(x,y)=0$, implying :

$$\langle \mathcal{T}_x g, \mathcal{T}_y g \rangle = \|\mathcal{T}_x g\| \|\mathcal{T}_y g\| \tag{A.3}$$

From Cauchy-Schwartz, we know that the above equality holds only if $\mathcal{T}_x g$ is linearly dependant of $\mathcal{T}_y g$, i.e. that $g(\mathbf{L})\delta_x$ is linearly dependant of $g(\mathbf{L})\delta_y$. Now, given the hypothesis that $g(\mathbf{L})$ is full rank, this statement is impossible to realize for $x \neq y$.

□

A.3 Random matrices and eigenspace approximation

A.3.1 Proof of Lemma 7

Proof. Let us start by showing a few characteristics of projected Gaussians.

Let $\mathbf{U} \in \mathbb{R}^{N\times N}$ describe a basis of N orthonormal vectors and $\mathbf{R} \in \mathbb{R}^{N\times d}$ be a Gaussian random matrix with i.i.d. entries $\sim \mathcal{N}(0,\sigma^2)$.

Mathematically,

$$\forall i,j : (\mathbf{UR})_{i,j} = \langle u_{i-1}, r_j \rangle = \sum_{\ell=1}^{N} u_{i-1}(\ell) r_{\ell,j}, \tag{A.4}$$

is a linear transformation of the elements of $\mathbf{R}$ (i.e. are Gaussian distributed). Moreover, we already knew that the size of the product is a $N \times d$ matrix. Next, we will evaluate the two first moments of those entries.

$$\mathbb{E}\left[\sum_{\ell=1}^{N} u_{i-1}(\ell) r_{\ell,j}\right] = \sum_{\ell=1}^{N} u_{i-1}(\ell)\mathbb{E}[r_{\ell,j}] = 0 \tag{A.5}$$

$$\begin{aligned}\mathrm{Var}\left(\sum_{\ell=1}^{N} u_{i-1}(\ell) r_{\ell,j}\right) &= \sum_{\ell=1}^{N} u_{i-1}^2(\ell)\,\mathrm{Var}(r_{\ell,j}) \\ &= \sigma^2 \sum_{\ell=1}^{N} u_{i-1}^2(\ell) = \sigma^2\end{aligned} \tag{A.6}$$

This shows that all entries of $\mathbf{UR}$ are identically distributed. Then we can compute the

covariance between any two entries $\left((\mathbf{UR})_{i,j}$ and $(\mathbf{UR})_{n,m}\right)$ to ensure independence:

$$\begin{aligned}
\mathrm{Cov}(\mathbf{UR}) &= \mathbb{E}\left[\sum_{\ell=1}^{N} u_{i-1}(\ell) r_{\ell,j} \sum_{\ell'=1}^{N} u_{n-1}(\ell') r_{\ell',m}\right] \\
&= \sum_{\ell=1}^{N}\sum_{\ell'=1}^{N} u_{i-1}(\ell) u_{n-1}(\ell') \mathbb{E}[r_{\ell,j} r_{\ell',m}] \\
&= \mathbb{1}_{\{m=j\}} \sum_{\ell=1}^{N} u_{i-1}(\ell) u_{n-1}(\ell) \mathbb{E}[r_{\ell,m}^2] \\
&= \sigma^2 \mathbb{1}_{\{m=j\}} \langle u_{i-1}, u_{n-1}\rangle \\
&= \sigma^2 \mathbb{1}_{\{m=j\}} \mathbb{1}_{\{n=i\}},
\end{aligned} \tag{A.7}$$

which shows that any two entries in $\mathbf{UR}$ are independent. Combining the last two shows that the entries of $\mathbf{UR}$ are i.i.d. Gaussian random samples with pdf $\sim \mathcal{N}(0,\sigma^2)$ similarly to $\mathbf{R}$.

Knowing that the multiplication of a Gaussian random matrix by a basis such as $\mathbf{U}$ preserves all the properties of the initial random matrix (Gaussian, entry-wise independence, identical mean, variance, and size), we need to evaluate the effect of the row selection process.

Selecting any subset of the rows of $\mathbf{U}$ changes the size but conserves the orthonormal properties over the rows. Indeed, without loss of generality on the row selection, we have

$$\begin{pmatrix} I_k \\ 0 \end{pmatrix} \mathbf{UR} = \begin{pmatrix} I_k \\ 0 \end{pmatrix} \mathbf{R}' = \mathbf{R}_k \tag{A.8}$$

Thus, only the size will be altered compared to a multiplication by the full matrix $\mathbf{U}$. This concludes the proof. □

A.3.2 Proof of Lemma 8

Proof. Let us consider the limit case $d = k$. In this case we have to show that the square $(k \times k)$ matrix $\mathbf{R}_k$ is non-singular. Indeed, the set of singular Gaussian random matrices $\mathcal{R}_s = \{\mathbf{R}_k : \det(\mathbf{R}_k) = 0\}$ is of dimension $k-1$ since it is generated by the zeros of a polynomial of order k. Moreover, since the complete set $\mathcal{R} = \{\mathbf{R}_k\}$ has dimension k, the codimension of $\mathcal{R}_s$ is 1. Thus, the set $\mathcal{R}_s$ is a null set, which means that picking a matrix at random from the set $\mathcal{R}$ returns a matrix from $\mathcal{R}_s$ with probability 0. Hence, $\mathbf{R}_k$ is non-singular with probability 1.

If we consider $d > k$, any square matrix formed of k of the columns of $\mathbf{R}_k$ has rank k following the proof above for the square case. Now, adding columns to this matrix can not change the rank since it can not reduce it and the matrix is full rank already. □

A.3.3 Proof of Theorem 3

Proof. From $\mathbf{F}$, we can find a set of k orthonormal vectors $\mathbf{B} = \{\mathbf{b_1}|\mathbf{b_2}|\dots|\mathbf{b_k}\}$, e.g., by applying an SVD. We obtain a decomposition such as $\mathbf{F} = \mathbf{B}\Sigma\mathbf{V}^\top$, with Σ a diagonal matrix and $\mathbf{V}$ an orthogonal matrix. This gives the following equality:

$$\mathbf{U}_k\mathbf{R}_k = \mathbf{F} = \mathbf{B}\Sigma\mathbf{V}^\top \tag{A.9}$$

and thus $\mathbf{U}_k$ and $\mathbf{B}$ have the same column space by definition. But since $\mathbf{B}$ and $\mathbf{U}_k$ also have the same shape and orthonormal columns, they necessarily relate to each other as $\mathbf{B} = \mathbf{U}_k\mathbf{Q}$, for some rotation matrix $\mathbf{Q} \in \mathbb{R}^{k\times k}$. □

A.4 Sampling theorems

Let us first recall two important lemmas necessary for the proofs. The first one is a generalization of the Bernstein inequality for matrices.

Lemma 13 (Matrix Bernstein: Bounded Case)**.** *[193, Theorem 6.1] Consider a finite sequence $\mathbf{X}_m$ of independent, random, self-adjoint matrices with dimension d. Assume that*

$$\mathbb{E}[\mathbf{X}_m] = 0 \quad \textit{and} \quad \sigma_{\max}(\mathbf{X}_m) \le R \quad \textit{almost surely.}$$

Compute the norm of the total variance,

$$A^2 = \left\| \sum_m \mathbb{E}[\mathbf{X}_m^2 \right\|]_{op}$$

Then the following chain of inequalities holds for all $\delta \ge 0$.

$$\begin{aligned}
\mathbb{P}(\lambda_{\max}\left(\sum_m \boldsymbol{X}_m\right) \ge \delta) &\le d\cdot\exp\left(-\frac{A^2}{R^2}\cdot h\left(\frac{R\delta}{A^2}\right)\right) \\
&\le d\cdot\exp\left(\frac{-\delta^2/2}{A^2+R\delta/3}\right) \\
&\le \begin{cases} d\cdot\exp(-3\delta^2/8A^2) & \textit{for } \delta \le A^2/R; \\ d\cdot\exp(-3\delta/8R) & \textit{for } \delta \ge A^2/R. \end{cases}
\end{aligned}$$

where the function h is defined as $h(u) = (1+u)\log(1+u) - u$ for $u \ge 0$.

The second lemma is a generalization of the triangular inequality for the norm of the localization operator.

Lemma 14. *Given any continuous kernel g and g', the norm of the localization operator*

satisfies:

$$\left\|\mathcal{T}_i g'\right\|_2^2 - \left\|\mathcal{T}_i\left(|g'|-|g|\right)\right\|_2^2 \quad\leq\quad \left\|\mathcal{T}_i g\right\|_2^2 \quad\leq\quad \left\|\mathcal{T}_i g'\right\|_2^2 + \left\|\mathcal{T}_i\left(|g'|-|g|\right)\right\|_2^2 \tag{A.10}$$

Proof. From the definition of the localization operator, we have:

$$\begin{aligned}
\left\|\mathcal{T}_i g\right\|_2^2 &= \sum_{\ell=0}^{N-1} g^2(\lambda_\ell)\boldsymbol{u}_\ell^2[i] \\
&= \sum_{\ell=0}^{N-1}\left(g^2(\lambda_\ell)-g'^2(\lambda_\ell)\right)\boldsymbol{u}_\ell^2[i] + \sum_{\ell=0}^{N-1} g'^2(\lambda_\ell)\boldsymbol{u}_\ell^2[i] \\
&\geq \sum_{\ell=0}^{N-1}\left(g(\lambda_\ell)-g'(\lambda_\ell)\right)^2\boldsymbol{u}_\ell^2[i] + \sum_{\ell=0}^{N-1} g'^2(\lambda_\ell)\boldsymbol{u}_\ell^2[i] \\
&= \left\|\mathcal{T}_i g'\right\|_2 + \left\|\mathcal{T}_i\left(|g'|-|g|\right)\right\|_2 .
\end{aligned} \tag{A.11}$$

A simple change of variable concludes the proof. The inequality A.11 comes from the following assertion. For all λ_ℓ such that $|g(\lambda_\ell)| \leq |g'(\lambda_\ell)|$, we have

$$\begin{aligned}
g^2(\lambda_\ell) &= g^2(\lambda_\ell) - g'^2(\lambda_\ell) + g'^2(\lambda_\ell) \\
&= \left(|g(\lambda_\ell)| - |g'(\lambda_\ell)|\right)\left(|g(\lambda_\ell)| + |g'(\lambda_\ell)|\right) + g'^2(\lambda_\ell) \\
&\geq -\left(|g'(\lambda_\ell)| - |g(\lambda_\ell)|\right)\left(|g'(\lambda_\ell)| - |g(\lambda_\ell)|\right) + g'^2(\lambda_\ell) \\
&= g'^2(\lambda_\ell) - \left(|g(\lambda_\ell)| - |g'(\lambda_\ell)|\right)^2 .
\end{aligned}$$

For the λ_ℓ such that $|g'(\lambda_\ell)| \leq |g(\lambda_\ell)|$, the inequality $g^2(\lambda_\ell) \geq g'^2(\lambda_\ell) - \left(|g(\lambda_\ell)| - |g'(\lambda_\ell)|\right)^2$ is trivially satisfied. □

A.4.1 Proof of Theorem 4

The proof of Theorem 4 is inspired by [1, Theorem 2] but contains some subtleties.

Proof. Les us define $\boldsymbol{\alpha} = \mathbf{U}_k^* \mathbf{x}$. We first notice that

$$g(\mathbf{L})\mathbf{x} = \mathbf{U}_k \mathbf{U}_k^* \mathbf{x} = \mathbf{U}_k g(\boldsymbol{\Lambda}_k)\boldsymbol{\alpha}$$

The quantity of interest is then rewritten as

$$\begin{aligned}
& \frac{1}{N_s}\left\|\mathbf{M}\mathbf{P}^{-\frac{1}{2}}g(\mathbf{L})\mathbf{x}\right\|_2^2 - \left\|g(\mathbf{L})\mathbf{x}\right\|_2^2 \\
= \; & \frac{1}{N_s}\left\|\mathbf{M}\mathbf{P}^{-\frac{1}{2}}\mathbf{U}_k g(\boldsymbol{\Lambda}_k)\boldsymbol{\alpha}\right\|_2^2 - \left\|\mathbf{U}_k g(\boldsymbol{\Lambda}_k)\boldsymbol{\alpha}\right\|_2^2 \\
= \; & \boldsymbol{\alpha}^*\left(\frac{1}{N_s}g(\boldsymbol{\Lambda}_k)\mathbf{U}_k^*\mathbf{P}^{-\frac{1}{2}}\mathbf{M}^*\mathbf{M}\mathbf{P}^{-\frac{1}{2}}\mathbf{U}_k g(\boldsymbol{\Lambda}_k) - g(\boldsymbol{\Lambda}_k)g(\boldsymbol{\Lambda}_k)\right)\boldsymbol{\alpha} \\
= \; & \boldsymbol{\alpha}^*\boldsymbol{Y}\boldsymbol{\alpha}
\end{aligned}$$

where $\boldsymbol{Y} = \frac{1}{N_s}g(\boldsymbol{\Lambda}_k)\mathbf{U}_k^*\mathbf{P}^{-\frac{1}{2}}\mathbf{M}^*\mathbf{M}\mathbf{P}^{-\frac{1}{2}}\mathbf{U}_k g(\boldsymbol{\Lambda}_k) - g(\boldsymbol{\Lambda}_k)g(\boldsymbol{\Lambda}_k)$. The remaining of the proof characterizes the maximum and minimum eigenvalue of $\boldsymbol{Y}$. To do so, we decompose $\boldsymbol{Y}$ into a sum of N_s independent, random, self-adjoint matrices $\mathbf{X}_i$ in order to apply Lemma 13.

Let us define

$$\mathbf{X}_i = \frac{1}{N_s}\left(g(\boldsymbol{\Lambda}_k)\mathbf{U}_k^*\left(\frac{\boldsymbol{\delta}_{\omega_i}\boldsymbol{\delta}_{\omega_i}^*}{\boldsymbol{p}_{\omega_i}} - \boldsymbol{I}\right)\mathbf{U}_k g(\boldsymbol{\Lambda}_k)\right). \tag{A.12}$$

It can be verified that

$$\boldsymbol{Y} = \sum_{i=1}^{N_s}\mathbf{X}_i = \sum_{i=1}^{N_s}\left(\frac{1}{N_s}g(\boldsymbol{\Lambda}_k)\mathbf{U}_k^*\left(\frac{\boldsymbol{\delta}_{\omega_i}\boldsymbol{\delta}_{\omega_i}^*}{\boldsymbol{p}_{\omega_i}} - \boldsymbol{I}\right)\mathbf{U}_k g(\boldsymbol{\Lambda}_k)\right). \tag{A.13}$$

By construction, the matrices $\mathbf{X}_i$ inherit independence from the random variables $\boldsymbol{\delta}_{\omega_i}$. Furthermore, we have

$$\begin{aligned}
\mathbb{E}[\mathbf{X}_i] &= \sum_{n=1}^{N} p_n \frac{1}{N_s}\left(g(\boldsymbol{\Lambda}_k)\mathbf{U}_k^*\left(\frac{\boldsymbol{\delta}_n\boldsymbol{\delta}_n^*}{\boldsymbol{p}_n} - \boldsymbol{I}\right)\mathbf{U}_k g(\boldsymbol{\Lambda}_k)\right) \\
&= \frac{1}{N_s}\left(g(\boldsymbol{\Lambda}_k)\mathbf{U}_k^*\left(\sum_{n=1}^{N}\boldsymbol{\delta}_n\boldsymbol{\delta}_n^* - \boldsymbol{I}\right)\mathbf{U}_k g(\boldsymbol{\Lambda}_k)\right) \\
&= \mathbf{0} \\
&= \mathbb{E}[-\mathbf{X}_i]
\end{aligned}$$

To apply Lemma 13 we need the maximum eigenvalue of $\mathbf{X}_i$ and $-\mathbf{X}_i$.

$$\begin{aligned}
\sigma_{\max}(\mathbf{X}_i) &= \sigma_{\max}\left(\frac{1}{N_s} g(\mathbf{\Lambda}_k)\mathbf{U}_k^* \left(\frac{\boldsymbol{\delta}_{\omega_i}\boldsymbol{\delta}_{\omega_i}^*}{\boldsymbol{p}_{\omega_i}} - \boldsymbol{I}\right)\mathbf{U}_k g(\mathbf{\Lambda}_k)\right) \\
&\leq \frac{1}{N_s}\sigma_{\max}\left(\frac{1}{\boldsymbol{p}_{\omega_i}} g(\mathbf{\Lambda}_k)\mathbf{U}_k^*\boldsymbol{\delta}_{\omega_i}\boldsymbol{\delta}_{\omega_i}^*\mathbf{U}_k g(\mathbf{\Lambda}_k)\right) \\
&= \frac{1}{N_s}\sigma_{\max}\left(\frac{1}{\boldsymbol{p}_{\omega_i}} \boldsymbol{\delta}_{\omega_i}^*\mathbf{U}_k g(\mathbf{\Lambda}_k) g(\mathbf{\Lambda}_k)\mathbf{U}_k^*\boldsymbol{\delta}_{\omega_i}\right) \\
&= \frac{1}{N_s}\max_i \frac{1}{\boldsymbol{p}_i}\boldsymbol{\delta}_i^*\mathbf{U}_k g(\mathbf{\Lambda}_k) g(\mathbf{\Lambda}_k)\mathbf{U}_k^*\boldsymbol{\delta}_i \\
&= \frac{1}{N_s}\max_i \frac{\left\|\mathcal{T}_i g\right\|_2^2}{\boldsymbol{p}_i}
\end{aligned}$$

$$\begin{aligned}
\sigma_{\max}(-\mathbf{X}_i) &= \sigma_{\max}\left(\frac{1}{N_s} g(\mathbf{\Lambda}_k)\mathbf{U}_k^* \left(\boldsymbol{I} - \frac{\boldsymbol{\delta}_{\omega_i}\boldsymbol{\delta}_{\omega_i}^*}{\boldsymbol{p}_{\omega_i}}\right)\mathbf{U}_k g(\mathbf{\Lambda}_k)\right) \\
&\leq \frac{1}{N_s}\sigma_{\max}\left(g^2(\mathbf{\Lambda})\right) = \frac{1}{N_s}\left\|g(\boldsymbol{\lambda})\right\|_\infty^2
\end{aligned}$$

Finally, before we need to compute one last quantity before applying the lemma

$$\begin{aligned}
A^2 &= \sigma_{\max}\left(\mathbb{E}\left[\sum_{i=1}^{N_s}\mathbf{X}_i^2\right]\right) \\
&= \sigma_{\max}\left(\mathbb{E}\left[\frac{1}{N_s^2}\sum_{i=1}^{N_s} g(\mathbf{\Lambda}_k)\mathbf{U}_k^*\left(\boldsymbol{I} - \frac{\boldsymbol{\delta}_{\omega_i}\boldsymbol{\delta}_{\omega_i}^*}{\boldsymbol{p}_{\omega_i}}\right)\mathbf{U}_k g(\mathbf{\Lambda}_k) g(\mathbf{\Lambda}_k)\mathbf{U}_k^*\left(\boldsymbol{I} - \frac{\boldsymbol{\delta}_{\omega_i}\boldsymbol{\delta}_{\omega_i}^*}{\boldsymbol{p}_{\omega_i}}\right)\mathbf{U}_k g(\mathbf{\Lambda}_k)\right]\right) \\
&= \frac{1}{N_s}\sigma_{\max}\left(g(\mathbf{\Lambda}_k)\mathbf{U}_k^*\mathbb{E}\left[\left(\boldsymbol{I} - \frac{\boldsymbol{\delta}_{\omega_i}\boldsymbol{\delta}_{\omega_i}^*}{\boldsymbol{p}_{\omega_i}}\right)\mathbf{U}_k g(\mathbf{\Lambda}_k) g(\mathbf{\Lambda}_k)\mathbf{U}_k^*\left(\boldsymbol{I} - \frac{\boldsymbol{\delta}_{\omega_i}\boldsymbol{\delta}_{\omega_i}^*}{\boldsymbol{p}_{\omega_i}}\right)\right]\mathbf{U}_k g(\mathbf{\Lambda}_k)\right) \\
&\leq \frac{1}{N_s}\sigma_{\max}\left(g(\mathbf{\Lambda}_k)\mathbf{U}_k^*\left(\sum_{i=1}^{N}\frac{\left\|\mathcal{T}_i g\right\|_2^2}{\boldsymbol{p}_i}\boldsymbol{\delta}_i\boldsymbol{\delta}_i^*\right)\mathbf{U}_k g(\mathbf{\Lambda}_k)\right) \\
&\leq \frac{1}{N_s}\left\|g(\lambda)\right\|_\infty^2 \max_i \frac{\left\|\mathcal{T}_i g\right\|_2^2}{\boldsymbol{p}_i},
\end{aligned}$$

since

$$\begin{aligned}
\mathbb{E}\left[\left(\boldsymbol{I}-\frac{\boldsymbol{\delta}_{\omega_i}\boldsymbol{\delta}_{\omega_i}^*}{\boldsymbol{p}_{\omega_i}}\right)\mathbf{U}_k g(\boldsymbol{\Lambda}_k) g(\boldsymbol{\Lambda}_k)\mathbf{U}_k^*\left(\boldsymbol{I}-\frac{\boldsymbol{\delta}_{\omega_i}\boldsymbol{\delta}_{\omega_i}^*}{\boldsymbol{p}_{\omega_i}}\right)\right] &= \sum_{i=1}^{N}\boldsymbol{p}_i\left(\boldsymbol{I}-\frac{\boldsymbol{\delta}_i\boldsymbol{\delta}_i^*}{\boldsymbol{p}_i}\right)\mathbf{U}_k g^2(\boldsymbol{\Lambda}_k)\mathbf{U}_k^*\left(\boldsymbol{I}-\frac{\boldsymbol{\delta}_i\boldsymbol{\delta}_i^*}{\boldsymbol{p}_i}\right)\\
&= \sum_{i=1}^{N}\boldsymbol{p}_i\boldsymbol{\delta}_i\boldsymbol{\delta}_i^* g^2(\mathbf{L})\boldsymbol{\delta}_i\boldsymbol{\delta}_i^* - g^2(\mathbf{L})\\
&= \sum_{i=1}^{N}\frac{\left\|\mathcal{T}_i g\right\|_2^2}{\boldsymbol{p}_i}\boldsymbol{\delta}_i\boldsymbol{\delta}_i^* - g^2(L)\\
&\preceq \sum_{i=1}^{N}\frac{\left\|\mathcal{T}_i g\right\|_2^2}{\boldsymbol{p}_i}\delta_i\delta_i^T
\end{aligned}$$

Let us denote $\max_i \frac{\left\|\mathcal{T}_i g\right\|_2^2}{\boldsymbol{p}_i} = \alpha$. We now apply Lemma 13 to the $\boldsymbol{Y} = \sum_{i=1}^{N_s}\mathbf{X}_i$ and we find

$$\mathbb{P}\left(\frac{1}{N_s}\left\|\mathbf{M}\mathbf{P}^{-\frac{1}{2}} g(\mathbf{L})\mathbf{x}\right\|_2^2 - \left\|g(\mathbf{L})\mathbf{x}\right\|_2^2 \geq \delta\left\|\boldsymbol{\alpha}\right\|_2^2\right) \leq k\exp\left(-\frac{N_s\frac{\delta^2}{2}}{\alpha\left(\left\|g(\lambda)\right\|_\infty^2 + \frac{\delta}{3}\right)}\right). \tag{A.14}$$

Similarly for $-\boldsymbol{Y} = \sum_{i=1}^{N_s} -\mathbf{X}_i$, we find

$$\mathbb{P}\left(\left\|g(\mathbf{L})\mathbf{x}\right\|_2^2 - \frac{1}{N_s}\left\|\mathbf{M}\mathbf{P}^{-\frac{1}{2}} g(\mathbf{L})\mathbf{x}\right\|_2^2 \geq \delta\left\|\boldsymbol{\alpha}\right\|_2^2\right) \leq k\exp\left(-\frac{N_s\frac{\delta^2}{2}}{\left\|g(\boldsymbol{\lambda})\right\|_\infty^2\left(\alpha + \frac{\delta}{3}\right)}\right). \tag{A.15}$$

In order to optimize the bound, we need to minimize α. Thus we choose $p_i = \frac{\left\|\mathcal{T}_i g\right\|_2^2}{\left\|g(\boldsymbol{\lambda})\right\|_2^2}$ and we get $\alpha = \left\|g(\boldsymbol{\lambda})\right\|_2^2$. The two previous inequalities become

$$\mathbb{P}\left(\frac{1}{M}\left\|\mathbf{M}\mathbf{P}^{-\frac{1}{2}} g(\mathbf{L})\mathbf{x}\right\|_2^2 - \left\|g(\mathbf{L})\mathbf{x}\right\|_2^2 \geq \delta\left\|\mathbf{U}_k^*\mathbf{x}\right\|_2^2\right) \leq k\exp\left(-\frac{N_s\delta^2}{2\left\|g(\boldsymbol{\lambda})\right\|_2^2\left(\left\|g(\lambda)\right\|_\infty^2 + \frac{\delta}{3}\right)}\right)$$

$$\mathbb{P}\left(\left\|g(\mathbf{L})\mathbf{x}\right\|_2^2 - \frac{1}{M}\left\|\mathbf{M}\mathbf{P}^{-\frac{1}{2}} g(\mathbf{L})\mathbf{x}\right\|_2^2 \geq \delta\left\|\mathbf{U}_k^*\mathbf{x}\right\|_2^2\right) \leq k\exp\left(-\frac{N_s\delta^2}{2\left\|g(\boldsymbol{\lambda})\right\|_\infty^2\left(\left\|g(\boldsymbol{\lambda})\right\|_2^2 + \frac{\delta}{3}\right)}\right)$$

We make the change of variables $\delta' = \frac{\delta}{\left\|g(\boldsymbol{\lambda})\right\|_\infty^2}$

$$\mathbb{P}\left(\frac{\frac{1}{N_s}\left\|\mathbf{M}\mathbf{P}^{-\frac{1}{2}} g(\mathbf{L})\mathbf{x}\right\|_2^2 - \left\|g(\mathbf{L})\mathbf{x}\right\|_2^2}{\left\|g(\boldsymbol{\lambda})\right\|_\infty^2} \geq \delta'\left\|\mathbf{U}_k^*\mathbf{x}\right\|_2^2\right) \leq k\exp\left(-\frac{1}{2}\frac{\left\|g(\boldsymbol{\lambda})\right\|_\infty^2}{\left\|g(\boldsymbol{\lambda})\right\|_2^2}\frac{N_s\delta'^2}{\left(1+\frac{\delta'}{3}\right)}\right)$$

$$\mathbb{P}\left(\frac{\left\|g(\mathbf{L})\mathbf{x}\right\|_2^2 - \frac{1}{N_s}\left\|\mathbf{M}\mathbf{P}^{-\frac{1}{2}}g(\mathbf{L})\mathbf{x}\right\|_2^2}{\left\|g(\boldsymbol{\lambda})\right\|_\infty^2} \geq \delta\left\|\mathbf{U}_k^*\mathbf{x}\right\|_2^2\right) \leq k\exp\left(-\frac{1}{2}\frac{N_s\delta'^2}{\left(\frac{\|g(\boldsymbol{\lambda})\|_2^2}{\|g(\boldsymbol{\lambda})\|_\infty^2}+\frac{\delta'}{3}\right)}\right)$$
$$\leq k\exp\left(-\frac{1}{2}\frac{\left\|g(\boldsymbol{\lambda})\right\|_\infty^2}{\left\|g(\boldsymbol{\lambda})\right\|_2^2}\frac{N_s\delta'^2}{\left(1+\frac{\delta'}{3}\right)}\right) \tag{A.16}$$

Finally, we substitute δ for δ'. We set the success probability of the event

$$\left|\frac{\frac{1}{N_s}\left\|\mathbf{M}\mathbf{P}^{-\frac{1}{2}}\mathbf{U}g(\boldsymbol{\Lambda})\mathbf{x}\right\|_2^2 - \left\|\mathbf{U}g(\boldsymbol{\Lambda})\mathbf{x}\right\|_2^2}{\left\|g(\boldsymbol{\lambda})\right\|_\infty^2}\right| \geq \delta\|\mathbf{x}\|_2^2. \tag{A.17}$$

to $1-\epsilon$. Since both sides of the bound have to be taken into account, we need

$$\frac{\epsilon}{2} \geq k\exp\left(-\frac{1}{2}\frac{\left\|g(\lambda)\right\|_\infty^2}{\left\|g(\boldsymbol{\lambda})\right\|_2^2}\frac{N_s t^2}{\left(1+\frac{\delta}{3}\right)}\right), \tag{A.18}$$

which is equivalent to impose on N_s

$$N_s \geq 2\frac{1}{\delta^2}\frac{\left\|g(\boldsymbol{\lambda})\right\|_2^2}{\left\|g(\boldsymbol{\lambda})\right\|_\infty^2}\left(1+\frac{\delta}{3}\right)\log\left(\frac{2k}{\epsilon}\right)$$

□

A.4.2 Proof of Theorem 5

Proof. Given $N_s \geq 2\frac{1}{\delta^2}\frac{\|g(\boldsymbol{\lambda})\|_2^2}{\|g(\boldsymbol{\lambda})\|_\infty^2}\left(1+\frac{\delta}{3}\right)\log\left(\frac{k}{\epsilon}\right)$, we use (A.16) and set $\mathbf{x}=\boldsymbol{\delta}_i$. Then with a probability ϵ, we have

$$\frac{\left\|\mathcal{T}_i g\right\|_2^2}{\left\|g(\boldsymbol{\lambda})\right\|_\infty^2} - \frac{\frac{1}{N_s}\left\|\mathbf{M}\mathbf{P}^{-\frac{1}{2}}\mathcal{T}_i g\right\|_2^2}{\left\|g(\boldsymbol{\lambda})\right\|_\infty^2} \geq \delta\left\|\mathbf{U}_k^*\boldsymbol{\delta}_i\right\|_2^2. \tag{A.19}$$

As a result, with a probability $1-\epsilon$, we have

$$\frac{\frac{1}{N_s}\left\|\mathbf{M}\mathbf{P}^{-\frac{1}{2}}\mathcal{T}_i g\right\|_2^2}{\left\|\mathcal{T}_i g\right\|_2^2} \geq 1-\delta\frac{\left\|g(\boldsymbol{\lambda})\right\|_\infty^2\left\|\mathbf{U}_k^*\boldsymbol{\delta}_i\right\|_2^2}{\left\|\mathcal{T}_i g\right\|_2^2}. \tag{A.20}$$

The change of variable $\delta' = \delta\frac{\|g(\boldsymbol{\lambda})\|_\infty^2\|\mathbf{U}_k^*\boldsymbol{\delta}_i\|_2^2}{\|\mathcal{T}_i g\|_2^2}$ concludes the proof. For the factor $\frac{\delta}{3}$, we use the fact that $\frac{\|g(\boldsymbol{\lambda})\|_\infty^2\|\mathbf{U}_k^*\boldsymbol{\delta}_i\|_2^2}{\|\mathcal{T}_i g\|_2^2} \geq 1$. □

A.4.3 Proof of Theorem 6

Proof. We first use the fact that $\left\|\boldsymbol{A}\mathcal{T}_i g'\right\|_2 \geq \left\|\boldsymbol{A}\mathcal{T}_i g\right\|_2$ for any linear operator $\boldsymbol{A}$. This comes from the fact that $\mathcal{T}_i g$ for a fixed i can be written as $\boldsymbol{T}_i g(\boldsymbol{\lambda})$ where $\boldsymbol{T}_i$ is a linear operator. We successively apply Theorem 5 and in a similar way to the proof of Theorem 5, Equation A.16 to obtain

$$\begin{aligned}\frac{1}{N_s}\left\|\mathbf{M}\mathbf{P}^{\frac{1}{2}}\mathcal{T}_i g\right\|_2^2 &\geq \frac{1}{N_s}\left\|\mathbf{M}\mathbf{P}^{\frac{1}{2}}\mathcal{T}_i g'\right\|_2^2 \\ &\geq \left\|\mathcal{T}_i g'\right\|_2^2 - \delta\left\|g'(\boldsymbol{\lambda})\right\|_\infty^2\left\|\mathbf{U}_k^*\boldsymbol{\delta}_i\right\|_2^2 \\ &\geq \left\|\mathcal{T}_i g\right\|_2^2 - \left\|\mathcal{T}_i\left(|g'|-|g|\right)\right\|_2^2 - \delta\left\|g'(\boldsymbol{\lambda})\right\|_\infty^2\left\|\mathbf{U}_k^*\boldsymbol{\delta}_i\right\|_2^2,\end{aligned}$$

for a number of samples

$$N_s \geq 2\frac{1}{\delta^2}\frac{\left\|g'(\boldsymbol{\lambda})\right\|_2^2\left\|g'(\boldsymbol{\lambda})\right\|_\infty^2\left\|\mathbf{U}_k^*\boldsymbol{\delta}_i\right\|_2^4}{\left\|\mathcal{T}_i g'\right\|_2^4}\left(1+\frac{\delta}{3}\right)\log\left(\frac{k}{\epsilon}\right).$$

The change of variable $\delta' = \delta\frac{\left\|g'(\boldsymbol{\lambda})\right\|_\infty^2\left\|\mathbf{U}_k^*\boldsymbol{\delta}_i\right\|_2^2}{\left\|\mathcal{T}_i g\right\|_2^2}$ and the division by $\left\|\mathcal{T}_i g\right\|_2^2$ conclude the proof. For the factor $\frac{\delta}{3}$, we use the fact that $\frac{\left\|g'(\boldsymbol{\lambda})\right\|_\infty^2\left\|\mathbf{U}_k^*\boldsymbol{\delta}_i\right\|_2^2}{\left\|\mathcal{T}_i g\right\|_2^2} \geq 1$. □

A.5 Clustering Graph Trees

A.5.1 Optimal branching bound

Using Eq. (5.1) and (5.3) we can state an upper bound on l, using the equivalences :

$$\begin{aligned}K > 2 &\iff \sqrt[l-1]{s} > 2 \\ &\iff \frac{1}{l-1}\log(s) > \log(2) \\ &\iff l < \log_2(s) + 1.\end{aligned}$$

And, similarly, derive a lower bound

$$\begin{aligned}K < N &\iff \sqrt[l]{s} < N \\ &\iff \frac{1}{l}\log(s) < \log(N) \\ &\iff l > \frac{\log(s)}{\log(N)}.\end{aligned}$$

Since it is given that $s < N$, this means that $0 < \frac{\log(s)}{\log(N)} < 1$, implies that the minimum integer value for l is 1. This gives the final bound for l:

$$1 \leq l < \log_2(s). \tag{A.21}$$

A.5.2 Upper bound on code difference

We know that $i_l(\mathbf{x})$ and $i_l(\mathbf{y})$ are integers in the range $[0, K[$, which means that $-(K-1) \leq (i_l(\mathbf{x}) - i_l(\mathbf{y})) \leq K-1$. Thus, starting with the assumption that $i_l(\mathbf{x}) = i_l(\mathbf{y})$ for $1 \leq m$ and $i_{m+1}(\mathbf{x}) \neq i_{m+1}(\mathbf{y})$, we have

$$\begin{aligned}
c(\mathbf{x}) - c(\mathbf{y}) &= \sum_{l=m+1}^{H_T} (i_l(\mathbf{x}) - i_l(\mathbf{y}))K^{H_T-l} \\
&\leq \sum_{l=m+1}^{H_T} (K-1)K^{H_T-l} \\
&= (K-1)\sum_{n=0}^{H_T-(m+1)} K^n \\
&= (K-1)\frac{1-K^{H_T-m}}{1-K} \\
&= K^{H_T-m} - 1 \\
&< K^{H_T-m}
\end{aligned}$$

which is the relation we want.

A.5.3 Proof of Lemma 12

Proof. We know that $i_l(\mathbf{x})$ and $i_l(\mathbf{y})$ are integers in the range $[0, K[$, which means that $-(K-1) \leq (i_l(\mathbf{x}) - i_l(\mathbf{y})) \leq K-1$ and thus $0 \leq |i_l(\mathbf{x}) - i_l(\mathbf{y})| \leq K-1$. Thus, for the upper bound we have

$$\begin{aligned}
\mathrm{TBD}\left(\mathbf{i}(\mathbf{x}), \mathbf{i}(\mathbf{y})\right) &= \sum_{l=1}^{H_T} |i_l(\mathbf{x}) - i_l(\mathbf{y})| K^{H_T - l} \\
&= \sum_{l=m+1}^{H_T} |i_l(\mathbf{x}) - i_l(\mathbf{y})| K^{H_T - l} \\
&\leq \sum_{l=m+1}^{H_T} (K-1) K^{H_T - l} \\
&= (K-1) \sum_{n=0}^{H_T-(m+1)} K^n \\
&= (K-1) \frac{1 - K^{H_T - m}}{1-K} \\
&= K^{H_T - m} - 1 \\
&< K^{H_T - m}.
\end{aligned}$$

We use the finite sum of geometric series, i.e. the fact that $\sum_{l=0}^{H_T-1} K^l = \frac{1-K^{H_T}}{1-K}$ for $K \geq 1$.

Now, using the fact that $0 \leq |i_l(\mathbf{x}) - i_l(\mathbf{y})| \leq K-1$ and $i_{m+1}(\mathbf{x}) \neq i_{m+1}(\mathbf{y})$, we can state a lower bound

$$\begin{aligned}
\mathrm{TBD}\left(\mathbf{i}(\mathbf{x}), \mathbf{i}(\mathbf{y})\right) &= \sum_{l=1}^{H_T} |i_l(\mathbf{x}) - i_l(\mathbf{y})| K^{H_T - l} \\
&= \sum_{l=m+1}^{H_T} |i_l(\mathbf{x}) - i_l(\mathbf{y})| K^{H_T - l} \\
&= |i_{m+1}(\mathbf{x}) - i_{m+1}(\mathbf{y})| K^{H_T-(m+1)} + \sum_{l=m+2}^{H_T} |i_l(\mathbf{x}) - i_l(\mathbf{y})| K^{H_T - l} \\
&\geq |i_{m+1}(\mathbf{x}) - i_{m+1}(\mathbf{y})| K^{H_T-(m+1)} \\
&\geq K^{H_T-(m+1)}
\end{aligned}$$

Putting both results together, we get

$$K^{H_T-(m+1)} \leq \mathrm{TBD}(\mathbf{i}(\mathbf{x}), \mathbf{i}(\mathbf{y})) < K^{H_T-m} \tag{A.22}$$

which is the result we wanted. □

B Experiments

B.1 Reproducible research

As we deem that reproducible research principles are essential to any kind of scientific research, all the examples, experiments, algorithms and any code needed to reproduce all the results in this work is open and freely available.[1] Whenever applicable, the code is licensed under the GNU GPLv3.[2]

Since some of the methods presented in this work use (pseudo) random numbers, it is expected that the results shall be slightly different from the ones presented in the details, but overall consistent.

B.2 Libraries and Algorithms

The main part of the experiments are written in Matlab/Octave and have two open source toolboxes as dependencies: the UnLocBox [142], a convex optimization toolbox and the Graph Signal Processing Toolbox (GSPBox) [141].

B.2.1 ANN search

The default algorithm we used for ANN computations (including kNN graphs construction) is the modified version of the randomized k-d tree provided in the FLANN library [128] with an automatic parameters estimation constraining an approximately constant cell size over all trees.

More precisely, the number of checks and number of trees called n_c and n_t respectively are computed using the following formulas, with N and D respectively the input size and

[1] Available at https://github.com/jparatte/thesis
[2] https://www.gnu.org/copyleft/gpl.html

dimension:

$$n_c = \max(\gamma_D 2^{\lceil \log_2(N)\mu_c \rceil}, 256) \quad \text{(B.1)}$$

$$n_t = \max(\rho_D \lceil \log_2(N)\mu_t \rceil, 4) \quad \text{(B.2)}$$

with

$$\gamma_D = \max(\lceil \log_2(D) - \mu_D \rceil, 1) \quad \text{(B.3)}$$

$$\rho_D = \max(\lceil \log 2(\gamma_D) \rceil, 1) \quad \text{(B.4)}$$

and μ_c, μ_t and μ_D the parameters controlling the precision of the search. Empirically, the values $\mu_c = 0.5$, $\mu_t = 0.5$ and $\mu_D = 5$ have shown to give precision consistently above 95% for a wide range of values for N and D.

As mentioned in Section 1.2.4, different ANN methods are used originally for specific algorithms: vantage-point trees and quadtrees in BH t-SNE, variants of RP trees and NN-descent in LargeVis, etc. In order to level the importance of the ANN search precision, the original implementations of the different algorithms mentioned afterwards have been adapted to accept input similarities computed with the randomized k-d tree method with automatic parameters selection presented in this section.

This choice is not mandatory and as long as a unified implementation is chosen, any other efficient algorithm could be used for this task. Interesting alternatives are included in these recent works : ANN Benchmarks,[3] NMSLib,[4] FALCONN.[5]

B.2.2 Dimensionality reduction algorithms

In this section, we describe the implementation and parameters used in all experiments.

- PCA: Matlab implementation using the 'eig' eigensolver. The original function handle is derived from the DRToolbox[6]
- Locally Linear Embedding: Matlab implementation adapted from the 'lle' function in the DRToolbox. The main change is an efficient kNN search using the ANN search described above. The eigendecomposition is performed with the 'eigs' eigensolver which uses the IRLM algorithm [35] implemented in ARPACK [112]. The only parameter is the number of neighbours k set to 12.
- Laplacian Eigenmaps: Matlab implementation adapted from the 'laplacian_eigen' func-

[3] https://github.com/erikbern/ann-benchmarks
[4] https://github.com/searchivarius/nmslib
[5] https://github.com/FALCONN-LIB/FALCONN
[6] https://lvdmaaten.github.io/drtoolbox/

tion in the DRToolbox. The main change is the use of a GSPBox kNN graphs as input, computed using the ANN search described above. The only parameter is the number of neighbours k set to 150.

- t-SNE : Matlab wrapper and C++ implementation adapted from the Barnes-Hut t-SNE implementation.[7] The changes were the addition of pre-computed similarity graphs as input in addition to raw features. The parameters were left to the default ones, except for the number of neighbours k set to 150. The weight matrix of the graph was computed using the ANN search described above and the Gaussian weights computed using a fixed perplexity.
- LargeVis : Matlab wrapper created from scratch and original C++ implementation.[8] All parameters were set to default and the interface used with the weight matrices as input.
- FEARS: Matlab implementation of Algorithm 6 as originally implemented in the GSPBox. The order of the Jackson-Chebyshev polynomials m was set to 100 and the number of neighbours k was set to 150 to be consistent with other methods.
- CDR : Matlab implementation of Algorithm 8. The sampling strategy was set to $\mathcal{T}_i g$ adapted sampling with a number of samples $N_s = 50\log(N)$.
- CGT : Matlab implementation of Algorithms 10 and Algorithm 11. The target sketch size was set to 4096, i.e. $K = 64$ and $l = 2$.

B.3 Datasets

In this section, we introduce the different datasets used for the experiments throughout this work. The goal while selecting them was to cover a wide variety of domains, applications, dimensionality, and size. Many are traditional in their fields, and in the machine learning community. As it was mentioned before, since the feature extraction phase is not of particular interest in this work, the data is always preprocessed, when necessary, so that the dimensionality is constant over the data samples of a set. Nevertheless, the data is kept as raw as possible and the preprocessing steps are always detailed. The main characteristics of the datasets is summarized in Table B.1

Core data :

- Caltech101 : collection of natural images without normalization belonging to 101 categories, first described in [65]. The features used are extracted from the 'fc7' feature blob of an AlexNet CNN trained on the ILSVRC12 dataset [99] implemented in Caffe,[9] also called Caffenet.
  Raw data :http://www.vision.caltech.edu/Image_Datasets/Caltech101/
- Caltech256: extended collection of the Caltech101 datasets consisting of natural images

[7] https://github.com/ninjin/barnes-hut-sne
[8] https://github.com/lferry007/LargeVis
[9] http://caffe.berkeleyvision.org

without normalization belonging to 256 categories, first described in [75]. The features used are extracted from the 'fc7' feature blob of an AlexNet CNN trained on the ILSVRC12 dataset [99] implemented in Caffe, also called Caffenet.
Raw data :http://www.vision.caltech.edu/Image_Datasets/Caltech256/

- CIFAR-10: collection of 60000 32x32 color images belonging to 10 different classes, first described in [98]. The features used are extracted from the last pooling layer 'pool3' of a CNN trained on the dataset using Caffe with a basic CNN architecture.[10]
Raw data : https://www.cs.toronto.edu/~kriz/cifar.html
- CMUPIE: collection of 11554 32x32 grayscale images of human faces belonging to 68 different people, first described in [173].
Raw data : http://www.cs.cmu.edu/afs/cs/project/PIE/MultiPie/Multi-Pie/Home.html
- Coil20 : collection of 1440 128x128 grayscale images belonging to five object categories, first described in [133].
Raw data : http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php
- FMA (Rosa) : a collection of 106574 songs from the Free Music Archive, described in [25]. The features are common audio features extracted with librosa.[11]
Raw data : https://github.com/mdeff/fma/
- FMA (echonest): a subset of the FMA dataset containing 13129 songs for which Echonest features have been extracted.
Raw data : https://github.com/mdeff/fma/
- HIVA: chemoinformatics dataset containing molecular structure related to compounds activity for AIDS HIV detection.
Raw data : http://www.causality.inf.ethz.ch/al_data/HIVA.html
- MNIST : collection of 70000 28x28 grayscale images of handwritten digits, described in [105].
Raw data : http://yann.lecun.com/exdb/mnist/
- NORB: collection of 48600 96x96 grayscale images of small toys belonging to 5 different categories, described in [106].
Raw data : http://www.cs.nyu.edu/~ylclab/data/norb-v1.0-small/.
- SYLVA: dataset of 216 ecology variables measured on 145252 30 by 30 metres cells related to forest cover types.
Raw data : http://www.causality.inf.ethz.ch//al_data/SYLVA.html
- USPS : collection of 9298 16x16 grayscale images of handwritten digits, first described in [86].
Raw data : http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#usps

Sparse (very high-dimensional) data :

[10] http://caffe.berkeleyvision.org/gathered/examples/cifar10.html
[11] https://librosa.github.io/librosa/

Datasets - properties	N	D	N_c
caltech101-caffenet	9145	4096	102
caltech256-caffenet	30607	4096	257
cifar10-cnn	60000	1024	10
cmupie	11554	1024	68
coil20	1440	16384	21
fma-echonest	13129	232	12
fma-rosa	106574	518	17
hiva	42678	1617	2
mnist	70000	784	10
norb	48600	9216	5
sylva	145252	216	2
usps	9298	256	10
20newsgroup	19996	33546	20
nova	19466	16969	2
livejournal	3997962	-	38648

Table B.1 – **Datasets summary**

- 20Newsgroup : collection of 19996 newsgroup documents, belonging to 20 different newsgroups. The features used are 33546 TF-IDF (Time-Frequency Inverse Document Frequency) computed using
  Raw data : http://qwone.com/~jason/20Newsgroups/
- NOVA: subset of the 20Newsgroup dataset using bag-of-word features from two classes.
  Raw data : http://www.causality.inf.ethz.ch//al_data/NOVA.html

Large-scale data ($N > 10^6$) :

- LiveJournal: a dataset from the LiveJournal social network. It contains no raw features but only a graph composed of the largest connected component which has 3 997 962 nodes.
  Raw data : http://snap.stanford.edu/data/com-LiveJournal.html

List of publications

Below is the list of publications I authored or co-authored during my Thesis.

1. J. Paratte, N. Perraudin, and P. Vandergheynst. Compressive embedding and visualization using graphs. *arXiv preprint arXiv:1702.05815*, 2017

2. J. Paratte and L. Martin. Fast eigenspace approximation using random signals. *arXiv preprint arXiv:1611.00938*, 2016

3. Y. Schoenenberger, J. Paratte, and P. Vandergheynst. Graph-based denoising for time-varying point clouds. In *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2015*, pages 1–4. IEEE, 2015

4. N. Perraudin, J. Paratte, D. Shuman, V. Kalofolias, P. Vandergheynst, and D. K. Hammond. GSPBOX: A toolbox for signal processing on graphs. *ArXiv e-prints*, 2014

Bibliography

[1] A. E. Alaoui and M. W. Mahoney. Fast randomized kernel ridge regression with statistical guarantees. pages 775–783, 2015. URL http://dl.acm.org/citation.cfm?id=2969239.2969326.

[2] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, et al. *LAPACK Users' guide*. SIAM, 1999.

[3] A. Anis, A. Gadde, and A. Ortega. Towards a sampling theorem for signals on arbitrary graphs. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3864–3868. IEEE, 2014.

[4] A. Anis, A. El Gamal, S. Avestimehr, and A. Ortega. Asymptotic justification of bandlimited interpolation of graph signals for semi-supervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5461–5465. IEEE, 2015.

[5] A. Anis, A. Gadde, and A. Ortega. Efficient sampling set selection for bandlimited graph signals using graph spectral proxies. *IEEE Transactions on Signal Processing*, 64(14):3775–3789, 2016.

[6] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of applied mathematics*, 9(1):17–29, 1951.

[7] S. Arya and D. M. Mount. Algorithms for fast vector quantization. In *Data Compression Conference, 1993. DCC'93.*, pages 381–390. IEEE, 1993.

[8] M. Aupetit. Visualizing distortions and recovering topology in continuous projection techniques. *Neurocomputing*, 70(7):1304–1330, 2007.

[9] E. C. Baechler, F. M. Batliwalla, G. Karypis, P. M. Gaffney, W. A. Ortmann, K. J. Espe, K. B. Shark, W. J. Grande, K. M. Hughes, V. Kapur, et al. Interferon-inducible gene expression signature in peripheral blood cells of patients with severe lupus. *Proceedings of the National Academy of Sciences*, 100(5):2610–2615, 2003.

[10] Y. Bai. High performance parallel approximate eigensolver for real symmetric matrices. 2005.

[11] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the solution of algebraic eigenvalue problems: a practical guide*, volume 11. Siam, 2000.

[12] S. Bandyopadhyay and E. J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1713–1723. IEEE, 2003.

[13] S. Banerjee and D. Chakroborty. Fast near-lossless image compression with tree coding having predictable output compression size. *Information Technology and Mobile Communication*, pages 39–44, 2011.

[14] S. T. Barnard and H. D. Simon. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency and computation: Practice and Experience*, 6(2):101–117, 1994.

[15] J. Barnes and P. Hut. A hierarchical o (n log n) force-calculation algorithm. *nature*, 324 (6096):446–449, 1986.

[16] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1000–1006. IEEE, 1997.

[17] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591, 2001.

[18] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[19] M. Belkin and P. Niyogi. Semi-supervised learning on riemannian manifolds. *Machine learning*, 56(1-3):209–239, 2004.

[20] M. Belkin and P. Niyogi. Towards a theoretical foundation for laplacian-based manifold methods. In *International Conference on Computational Learning Theory*, pages 486–500. Springer, 2005.

[21] M. Belkin and P. Niyogi. Convergence of laplacian eigenmaps. In *NIPS*, pages 129–136, 2006.

[22] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *Learning theory*, pages 624–638. Springer, 2004.

[23] R. E. Bellman. *Adaptive control processes: a guided tour*. Princeton university press, 2015.

[24] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[25] K. Benzi, M. Defferrard, P. Vandergheynst, and X. Bresson. Fma: A dataset for music analysis. *arXiv preprint arXiv:1612.01840*, 2016.

[26] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *International conference on database theory*, pages 217–235. Springer, 1999.

[27] G. Biau and D. M. Mason. High-dimensional p-norms. In *Mathematical Statistics and Limit Theorems*, pages 21–40. Springer, 2015.

[28] C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near-optimal column-based matrix reconstruction. *SIAM Journal on Computing*, 43(2):687–717, 2014.

[29] C. Boutsidis, A. Gittens, and P. Kambadur. Spectral clustering via the power method-provably. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2015.

[30] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[31] R. A. Brown. Building a balanced kd tree in o (kn log n) time. *arXiv preprint arXiv:1410.5420*, 2014.

[32] A. Brun, H.-J. Park, H. Knutsson, and C.-F. Westin. Coloring of dt-mri fiber traces using laplacian eigenmaps. In *International conference on computer aided systems theory*, pages 518–529. Springer, 2003.

[33] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE, 2005.

[34] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on communications*, 31(4):532–540, 1983.

[35] D. Calvetti, L. Reichel, and D. C. Sorensen. An implicitly restarted lanczos method for large symmetric eigenvalue problems. *Electronic Transactions on Numerical Analysis*, 2 (1):21, 1994.

[36] E. J. Candès and M. B. Wakin. An introduction to compressive sampling. *IEEE signal processing magazine*, 25(2):21–30, 2008.

[37] Y. H. Chang and C. Tomlin. Data-driven graph reconstruction using compressive sensing. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 1035–1040. IEEE, 2012.

[38] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[39] J. Cheeger. A lower bound for the smallest eigenvalue of the laplacian. 1969.

[40] B. Chen, P. C. Tai, R. Harrison, and Y. Pan. Novel hybrid hierarchical-k-means clustering method (hk-means) for microarray analysis. In *Computational Systems Bioinformatics Conference, 2005. Workshops and Poster Abstracts. IEEE*, pages 105–108. IEEE, 2005.

[41] L. Chen and A. Buja. Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the American Statistical Association*, 104(485):209–219, 2009.

[42] S. Chen, A. Sandryhaila, and J. Kovačević. Sampling theory for graph signals. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 3392–3396. IEEE, 2015.

[43] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević. Discrete signal processing on graphs: Sampling theory. *IEEE transactions on signal processing*, 63(24):6510–6523, 2015.

[44] S. Chen, R. Varma, A. Singh, and J. Kovacević. Signal recovery on graphs: Random versus experimentally designed sampling. In *Sampling Theory and Applications (SampTA), 2015 International Conference on*, pages 337–341. IEEE, 2015.

[45] Y. Chi. Compressive graph clustering from random sketches. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5466–5469. IEEE, 2015.

[46] F. Chung. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 9(1):1–19, 2005.

[47] F. Chung. The diameter and laplacian eigenvalues of directed graphs. *Electronic Journal of Combinatorics*, 13(4), 2006.

[48] F. R. Chung. *Spectral graph theory*, volume 92. AMS Bookstore, 1997.

[49] B. Claude. Théorie des graphes et ses applications, 1966.

[50] R. R. Coifman and S. Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[51] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

[52] J. Cook, I. Sutskever, A. Mnih, and G. Hinton. Visualizing similarity data with a mixture of maps. In *Artificial Intelligence and Statistics*, pages 67–74, 2007.

[53] F. Corpet. Multiple sequence alignment with hierarchical clustering. *Nucleic acids research*, 16(22):10881–10890, 1988.

[54] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 537–546. ACM, 2008.

[55] D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20 (4):364–366, 1977.

[56] E. Di Napoli, E. Polizzi, and Y. Saad. Efficient estimation of eigenvalue counts in an interval. *Numerical Linear Algebra with Applications*, 2016.

[57] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[58] R. Distasi, M. Nappi, and S. Vitulano. Image compression by b-tree triangular coding. *IEEE Transactions on Communications*, 45(9):1095–1100, 1997.

[59] M. N. Do and M. Vetterli. Framing pyramids. *IEEE Transactions on Signal Processing*, 51 (9):2329–2342, 2003.

[60] W. Dong, C. Moses, and K. Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*, pages 577–586. ACM, 2011.

[61] S. Ekins, K. V. Balakin, N. Savchuk, and Y. Ivanenkov. Insights for human ether-a-go-go-related gene potassium channel inhibition using recursive partitioning and kohonen and sammon mapping techniques. *Journal of medicinal chemistry*, 49(17):5059–5071, 2006.

[62] A. Elmoataz, O. Lezoray, and S. Bougleux. Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing. *IEEE transactions on Image Processing*, 17(7):1047–1060, 2008.

[63] T. Erlebach, K. Jansen, and E. Seidel. Polynomial-time approximation schemes for geometric graphs. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 671–679. Society for Industrial and Applied Mathematics, 2001.

[64] R. M. Fano. *The transmission of information*. Massachusetts Institute of Technology, Research Laboratory of Electronics Cambridge, Mass, USA, 1949.

[65] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 106(1):59–70, 2007.

[66] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4(1):1–9, 1974.

[67] R. W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[68] E. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability models. *Biometrics*, 61(3):768–769, 1965.

[69] S. Foucart and H. Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Birkhäuser Basel, 2013.

[70] E. Gallopoulos and Y. Saad. Efficient solution of parabolic equations by krylov approximation methods. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1236–1264, 1992.

[71] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[72] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[73] A. N. Gorban and A. Zinovyev. Principal manifolds and graphs in practice: from molecular biology to dynamical systems. *International journal of neural systems*, 20(03): 219–232, 2010.

[74] L. J. Grady and J. Polimeni. *Discrete calculus: Applied analysis on graphs for computational science*. Springer, 2010.

[75] G. H. Griffin and A. Perona. P. the caltech-256. *Caltech Technical Report, Tech. Rep.*, 2012.

[76] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[77] D. K. Hammond, Y. Gur, and C. R. Johnson. Graph diffusion distance: A difference measure for weighted graphs based on the graph laplacian exponential kernel. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 419–422. IEEE, 2013.

[78] V. Harikumar, P. P. Gajjar, M. V. Joshi, and M. S. Raval. Multiresolution image fusion: Use of compressive sensing and graph cuts. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(5):1771–1780, 2014.

[79] C. Hegde, A. C. Sankaranarayanan, and R. G. Baraniuk. Learning manifolds in the wild. *Preprint, July*, 1(2):4, 2012.

[80] B. Hendrickson and R. W. Leland. A multi-level algorithm for partitioning graphs. *SC*, 95 (28), 1995.

[81] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What is the nearest neighbor in high dimensional spaces? In *26th Internat. Conference on Very Large Databases*, pages 506–515, 2000.

[82] G. Hinton and S. Roweis. Stochastic neighbor embedding. In *NIPS*, volume 15, pages 833–840, 2002.

[83] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[84] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[85] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[86] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.

[87] P. Indyk, R. Motwani, P. Raghavan, and S. Vempala. Locality-preserving hashing in multidimensional spaces. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 618–625. ACM, 1997.

[88] R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural networks*, 1(4):295–307, 1988.

[89] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[90] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst. Matrix completion on graphs. *unpublished, arXiv:1408.1717*, 2014.

[91] G. Karypis and V. Kumar. Analysis of multilevel graph partitioning. In *Proceedings of the 1995 ACM/IEEE conference on Supercomputing*, page 29. ACM, 1995.

[92] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[93] S. Kaski, J. Nikkilä, M. Oja, J. Venna, P. Törönen, and E. Castrén. Trustworthiness and metrics in visualizing similarity of gene expression. *BMC bioinformatics*, 4(1):48, 2003.

[94] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[95] B. Kégl. Intrinsic dimension estimation using packing numbers. In *NIPS*, pages 681–688, 2002.

[96] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.

[97] A. Krishnamuthy, J. Sharpnack, and A. Singh. Recovering graph-structured activations using adaptive compressive measurements. In *Signals, Systems and Computers, 2013 Asilomar Conference on*, pages 765–769. IEEE, 2013.

[98] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[99] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[100] G. Kron. *Tensor analysis of networks*. J. Wiley & Sons, 1939.

[101] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[102] S. Lafon. *Diffusion maps and geometric harmonics*. PhD thesis, PhD thesis, Yale University, 2004.

[103] S. Lamrous and M. Taileb. Divisive hierarchical k-means. In *Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, pages 18–18. IEEE, 2006.

[104] C. Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.

[105] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[106] Y. LeCun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–104. IEEE, 2004.

[107] I. Lee and J. D. Gibson. Tree coding combined with tdhs for speech coding at 6.4 and 4.8 kbps. *Speech communication*, 29(1):23–37, 1999.

[108] J. Lee. Multiscale estimation of intrinsic dimensionality of point cloud data and multiscale analysis of dynamic graphs. 2010.

[109] J. A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[110] J. A. Lee and M. Verleysen. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing*, 72(7):1431–1443, 2009.

[111] J. A. Lee, M. Verleysen, et al. Rank-based quality assessment of nonlinear dimensionality reduction. In *ESANN*, pages 49–54, 2008.

[112] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.

[113] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[114] I. S. Lim, P. de Heras Ciechomski, S. Sarni, and D. Thalmann. Planar arrangement of high-dimensional biomedical data sets by isomap coordinates. In *Computer-Based Medical Systems, 2003. Proceedings. 16th IEEE Symposium*, pages 50–55. IEEE, 2003.

[115] A. V. Little, J. Lee, Y.-M. Jung, and M. Maggioni. Estimation of intrinsic dimensionality of samples from noisy low-dimensional manifolds in high dimensions with multiscale svd. In *Statistical Signal Processing, 2009. SSP'09. IEEE/SP 15th Workshop on*, pages 85–88. IEEE, 2009.

[116] A. V. Little, M. Maggioni, and L. Rosasco. Multiscale geometric methods for estimating intrinsic dimension. *Proc. SampTA*, 4(2), 2011.

[117] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[118] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[119] F. Mahmood, N. Shahid, U. Skoglund, and P. Vandergheynst. Compressed sensing and adaptive graph total variation for tomographic reconstructions. *CoRR*, abs/1610.00893, 2016. URL http://arxiv.org/abs/1610.00893.

[120] M. W. Mahoney and L. Orecchia. Implementing regularization implicitly via approximate eigenvector computation. *arXiv preprint arXiv:1010.0703*, 2010.

[121] Y. A. Malkov and D. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *arXiv preprint arXiv:1603.09320*, 2016.

[122] G. Mantena and X. Anguera. Speed improvements to information retrieval-based dynamic time warping using hierarchical k-means clustering. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8515–8519. IEEE, 2013.

[123] V. A. Marchenko and L. A. Pastur. Distribution of eigenvalues for some sets of random matrices. *Matematicheskii Sbornik*, 114(4):507–536, 1967.

[124] 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.

[125] D. J. Meagher. *Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer*. Electrical and Systems Engineering Department Rensseiaer Polytechnic Institute Image Processing Laboratory, 1980.

[126] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[127] B. Mokbel, W. Lueks, A. Gisbrecht, and B. Hammer. Visualizing the quality of dimensionality reduction. *Neurocomputing*, 112:109–123, 2013.

[128] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36, 2014.

[129] A. Munteanu, J. Cornelis, G. Van der Auwera, and P. Cristea. Wavelet image compression-the quadtree coding approach. *IEEE Transactions on Information Technology in Biomedicine*, 3(3):176–185, 1999.

[130] B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. *arXiv preprint math/0506090*, 2005.

[131] S. K. Narang and A. Ortega. Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs. *IEEE transactions on signal processing*, 61(19):4673–4685, 2013.

[132] J. F. Navarro, C. S. Frenk, and S. D. White. A universal density profile from hierarchical clustering. *The Astrophysical Journal*, 490(2):493, 1997.

[133] S. A. Nene, S. K. Nayar, H. Murase, et al. Columbia object image library (coil-20). 1996.

[134] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[135] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[136] M. Niskanen and O. Silvén. Comparison of dimensionality reduction methods for wood surface inspection. In *Quality Control by Artificial Vision*, pages 178–188. International Society for Optics and Photonics, 2003.

[137] J. Paratte and L. Martin. Fast eigenspace approximation using random signals. *arXiv preprint arXiv:1611.00938*, 2016.

[138] J. Paratte, N. Perraudin, and P. Vandergheynst. Compressive embedding and visualization using graphs. *arXiv preprint arXiv:1702.05815*, 2017.

[139] K. Peason. On lines and planes of closest fit to systems of point in space. *Philosophical Magazine*, 2(11):559–572, 1901.

[140] N. Perraudin. *Graph-based structures in data science: fundamental limits and applications to machine learning*. PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2017.

[141] N. Perraudin, J. Paratte, D. Shuman, V. Kalofolias, P. Vandergheynst, and D. K. Hammond. GSPBOX: A toolbox for signal processing on graphs. *ArXiv e-prints*, 2014.

[142] N. Perraudin, D. Shuman, G. Puy, and P. Vandergheynst. UNLocBoX A matlab convex optimization toolbox using proximal splitting methods. *ArXiv e-prints*, 2014.

[143] N. Perraudin, B. Ricaud, D. Shuman, and P. Vandergheynst. Global and local uncertainty principles for signals on graphs. *arXiv preprint arXiv:1603.03030*, 2016.

[144] I. Pesenson. Sampling in paley-wiener spaces on combinatorial graphs. *Transactions of the American Mathematical Society*, 360(10):5603–5627, 2008.

[145] I. Z. Pesenson and M. Z. Pesenson. Sampling, filtering and sparse approximations on combinatorial graphs. *Journal of Fourier Analysis and Applications*, 16(6):921–942, 2010.

[146] P. T. Peter Tang and E. Polizzi. Feast as a subspace iteration eigensolver accelerated by approximate spectral projection. *SIAM Journal on Matrix Analysis and Applications*, 35 (2):354–390, 2014.

[147] G. Peyré, S. Bougleux, and L. Cohen. Non-local regularization of inverse problems. In *Computer Vision–ECCV 2008*, pages 57–68. Springer, 2008.

[148] G. M. Phillips. *Interpolation and approximation by polynomials*, volume 14. Springer, 2003.

[149] H. C. Purchase, R. F. Cohen, and M. I. James. An experimental study of the basis for graph drawing algorithms. *Journal of Experimental Algorithmics (JEA)*, 2:4, 1997.

[150] G. Puy, N. Tremblay, R. Gribonval, and P. Vandergheynst. Random sampling of bandlimited signals on graphs. *Applied and Computational Harmonic Analysis*, 2016.

[151] D. Ramasamy and U. Madhow. Compressive spectral embedding: sidestepping the svd. In *Advances in Neural Information Processing Systems*, pages 550–558, 2015.

[152] L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 515–524. ACM, 2013.

[153] D. Romero, M. Ma, and G. B. Giannakis. Kernel-based reconstruction of graph signals. *IEEE Transactions on Signal Processing*, 65(3):764–778, 2017.

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

[155] J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 100(5):401–409, 1969.

[156] A. Sandryhaila and J. Moura. Discrete signal processing on graphs: Frequency analysis. 2014.

[157] A. Sandryhaila and J. M. Moura. Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61:1644–1656, 2013.

[158] G. Sanguinetti. Dimensionality reduction of clustered data sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):535–540, 2008.

[159] Y. Schoenenberger, J. Paratte, and P. Vandergheynst. Graph-based denoising for time-varying point clouds. In *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2015*, pages 1–4. IEEE, 2015.

[160] D. W. Scott and J. R. Thompson. Probability density estimation in higher dimensions. In *Computer Science and Statistics: Proceedings of the fifteenth symposium on the interface*, volume 528, pages 173–179. North-Holland, Amsterdam, 1983.

[161] N. Shahid, N. Perraudin, V. Kalofolias, G. Puy, and P. Vandergheynst. Fast robust pca on graphs. *Journal of Selected Topics in Signal Processing*, 10(4):740–756, 2016.

[162] G. Shakhnarovich, P. Indyk, and T. Darrell. *Nearest-neighbor methods in learning and vision: theory and practice*. 2006.

[163] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[164] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[165] D. I. Shuman, B. Ricaud, and P. Vandergheynst. A windowed graph Fourier transform. *Statistical Signal Processing Workshop (SSP), 2012 IEEE*, pages 133–136, 2012.

[166] D. I. Shuman, M. J. Faraji, and P. Vandergheynst. A framework for multiscale transforms on graphs. *arXiv preprint arXiv:1308.4942*, 2013.

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

[168] D. I. Shuman, B. Ricaud, and P. Vandergheynst. Vertex-frequency analysis on graphs. *arXiv preprint arXiv:1307.5708*, 2013.

[169] D. I. Shuman, M. J. Faraji, and P. Vandergheynst. A multiscale pyramid transform for graph signals. *IEEE Transactions on Signal Processing*, 64(8):2119–2134, 2016.

[170] D. I. Shuman, B. Ricaud, and P. Vandergheynst. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291, 2016.

[171] R. Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34, 1973.

[172] C. Silpa-Anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[173] T. Sim, S. Baker, and M. Bsat. The cmu pose, illumination, and expression (pie) database. In *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pages 53–58. IEEE, 2002.

[174] J. O. Smith and J. S. Abel. Bark and erb bilinear transforms. *IEEE Transactions on speech and Audio Processing*, 7(6):697–708, 1999.

[175] A. J. Smola and R. Kondor. Kernels and regularization on graphs. In *Learning theory and kernel machines*, pages 144–158. Springer, 2003.

[176] R. R. Sokal. A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, 38:1409–1438, 1958.

[177] D. C. Sorensen. Implicit application of polynomial filters in ak-step arnoldi method. *Siam journal on matrix analysis and applications*, 13(1):357–385, 1992.

[178] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[179] M. Steinbach, G. Karypis, V. Kumar, et al. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Boston, 2000.

[180] G. Strang. The discrete cosine transform. *SIAM review*, 41(1):135–147, 1999.

[181] G. J. Sullivan and R. L. Baker. Efficient quadtree coding of images and video. *IEEE Transactions on image processing*, 3(3):327–331, 1994.

[182] J. Sun, C. Fyfe, and M. Crowe. Extending sammon mapping with bregman divergences. *Information Sciences*, 187:72–92, 2012.

[183] A. Susnjara, N. Perraudin, D. Kressner, and P. Vandergheynst. Accelerated filtering on graphs using lanczos method. *arXiv preprint arXiv:1509.04537*, 2015.

[184] M. Takatsuka. An application of the self-organizing map and interactive 3-d visualization to geospatial data. In *Proceedings of the 6th International Conference on GeoComputation*, pages 24–26, 2001.

[185] Y. H. Tan, C. Yeo, H. L. Tan, and Z. Li. On residual quad-tree coding in hevc. In *Multimedia Signal Processing (MMSP), 2011 IEEE 13th International Workshop on*, pages 1–4. IEEE, 2011.

[186] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. ACM, 2015.

[187] J. Tang, J. Liu, M. Zhang, and Q. Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web*, pages 287–297. International World Wide Web Conferences Steering Committee, 2016.

[188] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[189] L. Teng, H. Li, X. Fu, W. Chen, and I.-F. Shen. Dimension reduction of microarray data based on local tangent space alignment. In *Cognitive Informatics, 2005.(ICCI 2005). Fourth IEEE Conference on*, pages 154–159. IEEE, 2005.

[190] W. S. Torgerson. Multidimensional scaling of similarity. *Psychometrika*, 30(4):379–393, 1965.

[191] N. Tremblay and P. Borgnat. Subgraph-based filterbanks for graph signals. *IEEE Transactions on Signal Processing*, 64(15):3827–3840, 2016.

[192] N. Tremblay, G. Puy, R. Gribonval, and P. Vandergheynst. Compressive spectral clustering. In *33rd International Conference on Machine Learning*, 2016.

[193] J. A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.

[194] Y.-H. Tsai and M.-H. Yang. Locality preserving hashing. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 2988–2992. IEEE, 2014.

[195] J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information processing letters*, 40(4):175–179, 1991.

[196] A. Ultsch. Kohonen's self organizing feature maps for exploratory data analysis. In *Proceedings INNC'90, International Neural Network Conference, 1990*, pages 305–308. Kluwer, 1990.

[197] L. Van Der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245, 2014.

[198] L. Van Der Maaten, E. Postma, and J. Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10:66–71, 2009.

[199] D. Veeneman and B. Mazor. Stochastic tree coding: voice coding for wireless channels. In *Personal, Indoor and Mobile Radio Communications, 1992. Proceedings, PIMRC'92., Third IEEE International Symposium on*, pages 675–679. IEEE, 1992.

[200] J. Venna and S. Kaski. Local multidimensional scaling with controlled tradeoff between trustworthiness and continuity. In *Proceedings of WSOM*, volume 5, pages 695–702. Citeseer, 2005.

[201] J. Venna and S. Kaski. Visualizing gene interaction graphs with local multidimensional scaling. In *ESANN*, volume 6, pages 557–562, 2006.

[202] J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *Journal of Machine Learning Research*, 11(Feb):451–490, 2010.

[203] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

[204] J. Wang. *Geometric structure of high-dimensional data and dimensionality reduction*. Springer, 2011.

[205] J. H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[206] C. Ware, H. Purchase, L. Colpoys, and M. McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002.

[207] S. Warshall. A theorem on boolean matrices. *Journal of the ACM (JACM)*, 9(1):11–12, 1962.

[208] D. B. West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[209] S. D. White and C. S. Frenk. Galaxy formation through hierarchical clustering. *The Astrophysical Journal*, 379:52–79, 1991.

[210] R. J. Wilson. *An introduction to graph theory*. Pearson Education India, 1970.

[211] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 15(11):1101–1113, 1993.

[212] R. Xu, S. Damelin, and D. C. Wunsch. Applications of diffusion maps in gene expression data-based cancer diagnosis analysis. In *Engineering in medicine and biology society, 2007. EMBS 2007. 29th annual international conference of the IEEE*, pages 4613–4616. IEEE, 2007.

[213] W. Xu, E. Mallada, and A. Tang. Compressive sensing over graphs. In *INFOCOM, 2011 Proceedings IEEE*, pages 2087–2095. IEEE, 2011.

[214] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

[215] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA*, volume 93, pages 311–21, 1993.

[216] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 515–524. ACM, 2002.

[217] D. Zhou and B. Schölkopf. A regularization framework for learning from graph data. 2004.

[218] D. Zhou, J. Huang, and B. Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *the 22nd international conference*, pages 1036–1043, New York, New York, USA, 2005. ACM Press.

[219] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in neural information processing systems*, pages 1601–1608, 2006.

[220] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.

Johan Paratte
Avenue du 1er mai 4
1020 Renens VD
+41 78 863 80 77
johan.paratte@epfl.ch

EPFL Engineer in Communication Systems, specialist in Computer Vision, Machine Learning and Graph Signal Processing

Education

2013 – 2017	PhD in Signal Processing, EPFL-LTS2
2010 – 2013	Master in Communication Systems, EPFL With specialization in Signals, Images and Interfaces.
2006 – 2010	Bachelor in Communication Systems, EPFL
2003 – 2006	Maturity with specific option maths/physics, Lycée Cantonal de Porrentruy.

Professional experiences

from 2016	**Arcanite Solutions Ltd** (co-founder) : company executive management and project management.
2014 - 2015	**Theoriz Ltd** (mandate) : generic detection and tracking pipeline in distributed multi-camera network.
2012	**Visiosafe SA** (mandate) : large-scale pedestrian flow analytics using depth camera networks.
2011-2012	**EPFL-LTS2** (research assistant) : multi-camera system and face recognition algorithms for embedded systems.
2011	**CERN** (internship) : monitoring micro-services and sensors in the CERN Computer Center.
2009 – 2010	**CEP Technique SA** (mandate) : visualization software for Theatre automation.

Technical Skills

Artificial Intelligence and Machine Learning : Intelligent agents, Inference Engines, Supervised and Non-Supervised Learning, Deep Learning, Graph-based methods.
Computer Vision and Image Processing : Augmented reality, face/eye detection, hands postures detection, super resolution, optical flow, object recognition, stereo 3D reconstruction, 3D Point Cloud processing.
Programming : C++, Python, Perl, Matlab, Qt, OpenCV, Intel TBB, OpenMesh, Django.
Other computer skills : Networking (large scale LAN design and monitoring, DNS filtering and zone configuration, traffic shaping, CISCO switches administration), GNU/Linux (debian, ubuntu) system administration.

Theses and Selected Publications

PhD thesis (2013-2017) : « Graph-based methods for Visualization and Clustering »

Master thesis (2012-2013) : « Sparse Binary Features for Image Classification »

Paratte, J., Perraudin, N., Vandergheynst, P. (2017) **Compressive Embedding and Visualization using Graphs.** ArXiv preprint arXiv:1702.05815

Paratte, J., Martin, L. (2016) **Fast Eigenspace Approximation using Random Signals.** ArXiv preprint arXiv:1611.00938

Schoenenberger, Y., Paratte, J., Vandergheynst, P. (2015) **Graph-based denoising for time-varying point clouds.** 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2015

Perraudin, N., Paratte, J., Shuman, D., Kalofolias, V., Vandergheynst, P., & Hammond, D. K. (2014). **GSPBOX: A toolbox for signal processing on graphs.** arXiv preprint arXiv:1408.5781.

D'Angelo, E., Paratte, J., Puy, G., Vandergheynst, P. (2011) **Fast TV-L1 Optical Flow for Interactivity** IEEE International Conference on Image Processing (ICIP) 2011, Brussels, Belgium, IEEE International Conference on Image Processing ICIP, 2011.

Languages

French Mother tongue
English Fluent (C2)

Extra-curricular activities

Piano (certifcate)
Orchestral direction (diploma)

Personnal Information

29, single, Swiss and French citizenships