THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à l'Université Paris-Dauphine

Contributions to unsupervised learning from massive high-dimensional data streams: structuring, hashing and clustering

École doctorale de Dauphine – ED 543

Spécialité INFORMATIQUE

COMPOSITION DU JURY :

M. Rémi GRIBONVAL
INRIA
Président du jury

M. Albert BIFET
Télécom ParisTech
Rapporteur

M. Liva RALAIVOLA
Université Aix-Marseille, Critéo
Rapporteur

M. Jamal ATIF
Université Paris-Dauphine
Directeur de thèse

M. Cédric GOUY-PAILLER
CEA
Co-encadrant de thèse

Mme Florence d'ALCHE-BUC
Télécom ParisTech
Examinateur

M. Krzysztof CHOROMANSKI
Google Brain Robotics
Membre du jury

Soutenue par **Anne MORVAN**
le **12.11.2018**

Dirigée par **Jamal ATIF**

DAUPHINE UNIVERSITÉ PARIS | PSL RESEARCH UNIVERSITY PARIS

Université Paris-Dauphine, PSL Research University

Doctoral Thesis

Contributions to unsupervised learning from massive high-dimensional data streams: structuring, hashing and clustering

---

Author:
Anne Morvan

Supervisors:
Pr. Jamal Atif
Dr. Cédric Gouy-Pailler

Examiners:
Dr. Rémi Gribonval
Pr. Albert Bifet
Pr. Liva Ralaivola
Pr. Florence d'Alché-Buc
Dr. Krzysztof Choromanski

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

between

the Data Science Team, LAMSADE & the LADIS Lab, CEA-LIST

November 12, 2018

"*Pour ce qui est de l'avenir, il ne s'agit pas de le prévoir mais de le rendre possible.*"

Antoine de Saint-Exupéry

Université Paris-Dauphine, PSL Research University

Abstract

Ecole doctorale de l'Université Paris-Dauphine
LAMSADE & LADIS, CEA-LIST

Doctor of Philosophy

Contributions to unsupervised learning from massive high-dimensional data streams: structuring, hashing and clustering

by Anne Morvan

This thesis focuses on how to perform efficiently unsupervised machine learning such as the fundamentally linked nearest neighbor search and clustering task, under time and space constraints for high-dimensional datasets. These constraints require to store a very limited amount of information of the dataset, to potentially handle the data points as a stream and to increase their processing rate of flow. This is realized by designing compact data representations preserving approximatively the data points distance and structure. The engendered loss of information generates approximative results but this approximation should be controlled and compensated by an easier data streams processing. Therefore, the compact structures building is a parametrized trade-off between memory usage, rate of flow and precision of the result. After introduction to the context and review for the nearest neighbor search and clustering of some state-of-the-art exact and approximation algorithms which can be classified into three types: *data-independent*, *data-dependent* and *graph-based*, three chapters describe a contribution to the field for each one of these categories.

First, a new theoretical framework named *Structured spinners* is proposed to reduce the space cost and to increase the rate of flow of *data-independent* Cross-polytope LSH, a state-of-the-art algorithm for the nearest neighbor search. The idea is to replace the random Gaussian matrices in the hash functions by pseudo-random structured ones such as the classical circulant, Toeplitz, Hankel, Hadamard etc. matrices. Theoretical guarantees on the accuracy result support their use, in particular for the state-of-the-art matrix built from three randomized Hadamard blocks for which the efficiency was assessed only experimentally until this work. Experiments not only on the approximate nearest neighbors search, but also on a wide range of applications such as kernel approximation, convex optimization via Newton sketches or neural networks confirm the usefulness of the *Structured spinners* which represent almost no loss of accuracy in comparison with the unstructured counterparts.

Second, a novel streaming *data-dependent* method is designed to learn compact binary codes from high-dimensional data points in only one pass. Besides some theoretical guarantees, the quality of the obtained embeddings is evaluated on the approximate nearest neighbors search task.

Finally, a recent *graph*-sketching technique is used to conceive a space-efficient parameter-free clustering algorithm from the recovery of an approximate minimum spanning tree of the compressed data dissimilarity graph. An application to the context of privacy-preserving clustering is then demonstrated.

Open questions and perspectives for future research conclude this work.

Université Paris-Dauphine, PSL Research University

Résumé

Ecole doctorale de l'Université Paris-Dauphine
LAMSADE & LADIS, CEA-LIST

Thèse de doctorat

Contributions à l'apprentissage non supervisé à partir de flux de données massives en grande dimension : structuration, hashing et clustering

par Anne Morvan

Cette thèse étudie deux tâches d'apprentissage non supervisé fondamentalement liées que sont la recherche des plus proches voisins et le clustering sur des données en grande dimension sous d'importantes contraintes de temps et d'espace. Ces dernières imposent de ne stocker qu'un nombre très limité d'information, possiblement de traiter les données en flux et d'augmenter le débit de traitement. Ceci est permis par la conception de représentations de données compactes préservant approximativement la distance et la structure des données. La perte d'information engendrée génère des résultats approximatifs mais cette approximation doit être contrôlée et compensée par un traitement du flux de données facilité. Ainsi, la construction des structures compactes est un compromis entre la mémoire utilisée, le débit et la précision du résultat. Après présentation du contexte et revue de l'état de l'art des algorithmes exacts et approchés pour la recherche des plus proches voisins et le clustering, classés en trois types: *indépendants* ou *dépendants* des données et fondés sur les *graphes*, trois chapitres décrivent une contribution au domaine dans chacune des catégories.

Tout d'abord, un nouveau cadre théorique nommé *Pivoteurs structurés* est proposé pour réduire le coût spatial et augmenter le débit du Cross-polytope LSH, une méthode état de l'art *indépendante des données* pour la recherche du plus proche voisin. L'idée est de remplacer les matrices aléatoires Gaussiennes dans les fonctions de hachage par des matrices pseudo-aléatoires structurées classiques. Des garanties théoriques sur la précision du résultat supportent leur usage, en particulier celui de la matrice état de l'art construite à partir de trois blocs de Hadamard aléatoirisés pour laquelle l'efficacité n'était jusqu'à présent prouvée qu'expérimentalement. Des expériences ne se restreignant pas à la recherche des plus proches voisins approchés confirment l'utilité des *Pivoteurs structurés* qui n'impliquent presque aucune perte de précision en comparaison avec leurs homologues non structurés.

Ensuite, une méthode originale *adaptée aux données* est conçue pour apprendre en une seule passe sur le flux des codes compacts binaires de données en grande dimension. En plus de certaines garanties théoriques, la qualité des représentations obtenues est mesurée dans le cadre de la recherche des plus proches voisins approchés.

Puis, un algorithme de clustering sans paramètre et efficace en termes de coût de stockage est développé en s'appuyant sur l'extraction d'un arbre couvrant minimal approché du graphe de dissimilarité *compressé*. Une application au clustering préservant la vie privée différentielle est démontrée.

Des questions restées ouvertes et des perspectives de recherche future concluent enfin ce travail.

Acknowledgements

Many people deserve my thanks. First, I am immensely grateful to my PhD advisors, Pr. Jamal Atif from Université Paris-Dauphine, PSL Research University and Dr. Cédric Gouy-Pailler from the Commissariat à l'Energie Atomique et aux Energies alternatives (CEA). Both of them have been great role models as researchers and mentors. In particular, I admire Jamal deeply for his dedication and impressive depth of knowledge, while Cédric has been of great help for precising my research work always in a pragmatic way and for overcoming many technical issues.

I would like to thank the CEA for the complete financial support of this thesis (*bourse CFR*), Dr. Véronique Serfaty from the Direction Générale de l'Armement (DGA) for the half contribution to the funding, Dr. Anthony Larue and then Dr. Lorène Allano for their welcoming at LADIS lab in CEA, Pr. Alexis Tsoukiàs and then Pr. Daniela Grigori at LAMSADE lab and Rida Laraki from the Doctoral school in Université Paris-Dauphine.

I thank Dr. Rémi Gribonval from INRIA for presiding the jury, Pr. Albert Bifet from Télécom ParisTech and Liva Ralaivola from Université Aix-Marseille and Critéo for having accepted to be *rapporteurs* of this thesis and Pr. Florence d'Alché-Buc from Télécom ParisTech for being a member of my jury.

I thank all my coauthors. In particular, I am greatly indebted with Dr. Krzysztof Choromanski from Google Brain Robotics, Dr. Tamás Sarlós from Google and Dr. Antoine Souloumiac from CEA for the pleasure of collaborating with them, for their help and for encouraging me. I also thank Rafaël Pinot, PhD student between Université Paris-Dauphine and CEA and his advisor Florian Yger, associate Professor at Université Paris-Dauphine for their help and fruitful collaboration.

I want to thank also my former officemates at LAMSADE and CEA for providing me with the ideal environment to work, for their enthusiasm and their valuable and enjoyable time, in particular: Alexis, Fabien, Arnaud, Marisnel, Ismail, Youen, Anaëlle, Maud, Maxime, Noëlie, Marine, Krystyna, Jean-Pierre, Pierre, Jérôme, Aurore, Flore, Paul, Satya, Marcel, Thomas, Olivier, Michel.

Most of all, I am grateful to my family and friends for their unconditional support and patience in difficult times, and especially to my love Thibaut: always motivating me for going further and further; without him, this PhD thesis could not have been possible.

Contents

List of Figures

List of Tables

List of Abbreviations

cdf	**c**umulative **d**istribution **f**unction
DFT	**D**iscrete **F**ourier **T**ransform
e.g.	***e**xempli **g**ratia*
Eq.	**Eq**uation
EVD	**E**igen**V**alue **D**ecomposition
FHT	**F**ast **H**adamard **T**ransform
FFT	**F**ast **F**ourier **T**ransform
FJLT	**F**ast **J**ohnson-**L**indenstrauss **T**ransform
i.e	***i**d **e**st*
i.i.d	**i**dentically and **i**ndependently **d**istributed
IFFT	**I**nverse **F**ast **F**ourier **T**ransform
JL	**J**ohnson-**L**indenstrauss
JLT	**J**ohnson-**L**indenstrauss **T**ransform
MSF	**M**inimum **S**panning **F**orest
MST	**M**inimum **S**panning **T**ree
PCA	**P**rincipal Component **A**nalysis
PSD	**P**ositive **S**emi-**D**efinite
Q.E.D	***q**uod **e**rat **d**emonstrandum*
resp.	**resp**ectively
s.t.	**s**uch **t**hat
std	**st**andard **d**eviation
SVD	**S**ingular **V**alue **D**ecomposition

List of Symbols

$\mathbb{C}$	Set of complex numbers	
$\mathbb{R}$	Set of real numbers	
$\mathbb{N}$	Set of natural integers	
$\mathbb{Z}$	Set of relative integers	
$\mathbb{Z}^+$	Set of positive integers	
$\mathbb{Z}/a\mathbb{Z}$	The ring of integers modulo a	$\mathbb{Z}/a\mathbb{Z} = \{0, \dots, a-1\}$
$\lceil x \rceil$	Ceiling function of x	$\lceil x \rceil = \min\{a \in \mathbb{Z} \mid a \geq x\}$
N	Number of instances in a dataset	
d	Dimension of one data point	
$\mathbb{R}^d$	Set of d-dimensional real-valued vectors	
$\mathbb{R}^{d \times N}$	Set of $d \times N$ real-valued matrices	
$\mathcal{M}_{d \times N}$	Set of matrices of dimensions $d \times N$	
$\mathbf{x}$	An arbitrary vector	
$\mathbf{M}$	An arbitrary matrix	
$\mathbf{I}_d$	$d \times d$ Identity matrix	
$\mathbf{0}_{d \times N}$	$d \times N$ Matrix with null coefficients	
$\mathcal{X}$	Input space	
$\mathbf{X}$	Dataset	$\mathbf{X} \in \mathbb{R}^{d \times N}$
$\mathbf{M}^T$	Transpose of $\mathbf{M}$	
$\mathbf{\Sigma_M}$	Covariance matrix of matrix $\mathbf{M} \in \mathbb{R}^{d \times N}$	$\mathbf{\Sigma_M} = \frac{1}{N-1}\mathbf{M}\mathbf{M}^T$
$\mathrm{Tr}(\mathbf{M})$	Trace function of $\mathbf{M}$	
$\mathbf{M}^\dagger$	The Moore Penrose / pseudo inverse of $\mathbf{M}$	
$\mathbf{M}^*$	The adjoint operator of $\mathbf{M}$	$\mathbf{M}^* = \overline{\mathbf{M}}^T = \overline{\mathbf{M}^T}$
$\|\|\mathbf{M}\|\|_2$	Spectral norm of $\mathbf{M}$	$\|\mathbf{M}\|_2 = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{M}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$
$\|\mathbf{M}\|_F$	Frobenius norm of $\mathbf{M}$	$\|\mathbf{M}\|_F = (\mathrm{Tr}(\mathbf{M}\mathbf{M}^*))^{1/2} = (\mathrm{Tr}(\mathbf{M}^*\mathbf{M}))^{1/2}$
$\mathbf{M} \succeq 0$	$\mathbf{M}$ is Positive Semi-Definite	
$[a]$	Set of integers between 1 and a	$[a] = \{1, \dots, a\}$
$\|\mathbf{v}\|_2$	L_2-norm of $\mathbf{v} \in \mathbb{R}^d$	$\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^d \mathbf{v}_i^2}$
$\|\mathbf{v}\|_\infty$	Infinite norm of $\mathbf{v} \in \mathbb{R}^d$	$\|\mathbf{v}\|_\infty = \max_{i \in [d]}(\|\mathbf{v}_i\|)$
$\mathrm{diag}(\mathbf{v})$	Returns a diagonal matrix with diagonal $\mathbf{v}$	
$\mathrm{diag}(\mathbf{M})$	Returns the diagonal of a matrix $\mathbf{M}$	
$\mathrm{sign}(.)$	Sign function applied pointwise	
$\mathbf{u} .* \mathbf{v}$	the element-wise vector-vector product	
$\Re(a)$	Returns the real part of $a \in \mathbb{C}$	
$\mathbb{S}_+^d$	Cone of symmetric PSD $\mathbf{M} \in \mathbb{R}^{d \times d}$	
$\mathcal{O}(d)$	Set of all orthogonal matrices in $\mathbb{R}^{d \times d}$	
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution: mean μ, std σ	

To my family,
To my better half, Thibaut.

Chapter 1

Introduction

Contents

1.1 Context and motivation

1.1.1 *Big data* era and the need of adapted processing

Today, every electronic device is conceived to collect an overwhelming volume of data in the hope to transform it into valuable information and insightful decisions through machine learning applications. A lot of emerging applications involve massive datasets such as the observations made by sensors from the Internet of Things (IoT), the customer click streams (cookies), the phone calls records, the large sets of web pages, multimedia data, financial transactions, etc. These amassed data serve as a colossal source of revenue for companies when analyzed. Therefore, tremendous quantity of data should be handled. In practice, because of this huge volume of data, some critical issues can arise and make some classical analysis approaches fail:

1. There is a high number of instances - also called records - in the database. Then, two cases can be distinguished.
   - On the one hand, these data may not be loaded entirely into memory.
   - On the other hand, even if the data have been successfully centralized, this can make some algorithms intractable. For instance, algorithms with time complexity equal to $O(N^2)$ where N is the number of instances are infeasible above some thousands or tens of thousands points.

2. These data are potentially *high dimensional*. This means that each record is described by a large set of attributes. This can be problematic for several reasons. First, this increases the complexity cost of the analysis algorithms. In some high-dimensional space, every data points are also far from each other, which makes very difficult to find similarities or differences between them. Furthermore, the algorithms could suffer from the *curse of dimensionality*. See below in the next Section for a definition.

3. The data may be also intrinsically a (potentially infinite) *stream*. This means that the data are caught on the fly as they are measured - for instance by the Internet of things' sensors - and should be processed directly. Besides, even if the data are not by nature a stream, as previously written, they can be stored on distributed sources in such amount that it is impossible to load the whole data into a centralized memory.

   For both cases, an *online* algorithm[1] is required to process the data in a limited number of passes (ideally in only one pass), either because of the intrinsic streaming nature of data or because of the memory limits constraining the data to be handled as a stream.

   Let us consider a data stream as the sequence of data points $x_1, x_2, \ldots, x_t, \ldots, x_N$ from a dataset $\mathcal{X}$ that come one at a time. A particularity of the streaming setting is that the output of a decision for data point x_t should be done before x_{t+1} is read, without access to the next and previous data points. However, the corresponding decisions can be made with an available memory of previous seen data points bounded by a function of the input size, mainly a sublinear function.

4. Finally, the vast quantity of available data makes them tedious to label. Therefore, most of the time, analysis should be performed in an *unsupervised manner*.

[1] In the sequel, we use without difference the terms *streaming* and *online*.

1.1.2 Unsupervised machine learning problems

The main paradigms that can be drawn in *unsupervised* learning are *dimensionality reduction*, *clustering* and *density estimation* [Bishop, 2006].

> In this thesis, we focus in particular on *clustering* and the *nearest neighbor search* (in the sense of similarity search) problems. Dimensionality reduction is seen as a tool to tackle the *curse of dimensionality* in both of these problems.

Dimensionality reduction and the curse of dimensionality

The *dimensionality reduction* is a transformation which maps data points from a high-dimensional space onto a lower-dimensional one while preserving the distance between the points. As classical methods, one can cite Principal Component Analysis (PCA) [Pearson, 1901, Hotelling, 1933], Independent Component Analysis (ICA) [Herault and Ans, 1984], the nonlinear ISOMAP [Tenenbaum et al., 2000], Locally Linear Embedding (LLE) [Roweis and Saul, 2000] and kernel PCA [Schölkopf et al., 1998]. One can also mention the approach of *feature selection* [Guyon and Elisseeff, 2003] which chooses a reduced number of existing attributes.

The *curse of dimensionality* term expresses the fact that it is very challenging to learn in a high-dimensional space since much more data are needed to prevent overfitting. Indeed, the curse of dimensionality arises when the number of instances required to obtain good model generalization performance is exponential in the dimension d of the data. As an example, for learning in an unsupervised manner Lipschitz-continuous functions in $\mathbb{R}^d$, at least $\Omega(\epsilon^{\max(d,2)})$ data points are required to learn a function with estimation error[2] $\epsilon > 0$ [Luxburg and Bousquet, 2004].

Why nearest neighbors and clustering in particular? Our choice to consider in particular the nearest neighbors search and clustering is motivated by the fact they are two fundamental *exploratory* analysis tools. They have in common the ability to segment data points into groups with *similar* properties by applying some (approximate) *distance - or structure - preserving transformations* on the data. Besides, nearest neighbors and clustering algorithms are as a matter of fact the most basic machine learning classification rules. In practice, they can help at identifying clients with similar behavior for market segmentation, exhibiting communities in social networks, detecting recurrent patterns in financial transactions, or from a sequenced genome of a new organism, finding the genes that are similar to those already present in the databases, just to name a few concrete applications. Moreover, they are significantly useful for the highly frequent case when *data labeling* is too expensive. Actually, from a few manually labeled data, these unsupervised learning algorithms enable to extend the properties of some point to all other ones contained in the same cluster or neighborhood. Then, manual labeling can be easier if it is performed directly on some pre-selected formed clusters or neighboring points. Now let us introduce more precisely the *nearest neighbors search* and *clustering* problems with some classical approaches. Figure 1.1 sums up the context of this thesis.

Nearest neighbors search

We give here the definition of the *nearest neighbor* problem for simplicity.

[2] The *estimation error* defines a loss function on the set of possible decision functions for an algorithm outputting a predictor [Hastie et al., 2001].

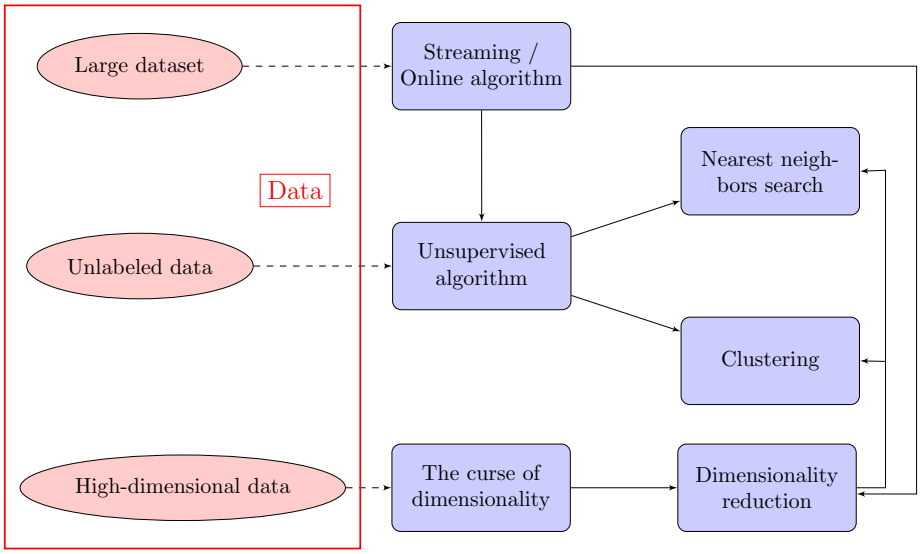


FIGURE 1.1: The thesis subject positioning. On large high-dimensional unlabeled data, unsupervised learning such as the nearest neighbors search and clustering can be applied, ideally in a streaming fashion. To overcome the curse of dimensionality, (online) dimensionality reduction is made on the data and suitable algorithms working on these compressed data - possibly in the form of a stream - are designed for the NN search and clustering.

Definition 1.1.1 (Nearest neighbor search)**.** *Given a set of instances* $X = \{x_1, x_2, \ldots, x_N\}$ *in a metric space defined by a set* $\mathcal{X}$ *with distance function dist, the* nearest neighbor search *consists in building a data structure that given any query point* $y \in \mathcal{X}$ *returns the* nearest neighbor *defined as* $\operatorname{argmin}_{x \in X} dist(x, y)$.

Notice that several instances of the nearest neighbor problem lead to solutions for the more general nearest neighbors one when all points within a radius r to the query or k nearest neighbors should be retrieved. The nearest neighbor problem can be defined for any *distance* function (see Appendix A for examples of distances).

After having defined the *nearest neighbor* problem, let us see classical approaches to solve it.

Naive approach The naive approach for retrieving the nearest neighbor from a query point y begins with the building of a data structure with all points from X. Then, a *linear scan* over this structure is performed. This means that all distances between y and the points of the data structure are computed. Finally the point with the minimal distance from y is returned. The time and space complexity is linear in N, the number of points in the dataset and this is clearly prohibitive for large datasets (some applications can have easily 10^9 data points).

To save the N pairwise distance computations, one can for $\mathcal{X} = \{0,1\}^d$ precompute and save all possible 2^d corresponding pairwise Hamming distance. The time cost is significantly reduced: only one memory lookup is required. However for high-dimensional data, the $O(2^d)$ space cost is inefficient.

Space partitioning suffers from the curse of dimensionality Therefore other data structures have been investigated looking for a better balance between the necessary time to answer a query which is called the *query time* and the amount of required memory or *space cost*. From the family of space partitioning, the Voronoi diagram [Aurenhammer, 1991], kd-trees [Bentley, 1975] and variants have space and time costs unacceptable for $d \geq 20$. This is another effect of the *dimensionality curse*: when the optimized methods do not longer bring any advantage over the naive linear scan.

Considered solution: approximation algorithms and dimensionality reduction via hashing For overcoming the running time and memory requirements bottlenecks, *approximation* methods have been designed.

> Approximative Nearest Neighbor (ANN) algorithms simply give a point whose distance from the query is at most a times the distance from the query to its exact nearest point. $a > 1$ is called the approximation factor.

The most known method from this category is *Locality-Sensitive Hashing* (LSH) [Gionis et al., 1999, Indyk and Motwani, 1998]. See more details on LSH in Section 2.2.2.

> The principle of approximative nearest neighbor algorithms is to change the feature representation by reducing the data dimensionality of the dataset and the query point, for instance with *hashing*.

This dimensionality reduction via hashing can be *data-independent* with random projection (see Section 2.2) or *data-dependent* (see Section 2.3) and is obtained from an optimization or machine learning scheme. Finally, the similarity search is done on the feature space. It is expected that some data points have the same smaller feature representation. Thus, the linear scan is done only on the set of data points with the same feature code or neighboring ones (looking for neighboring codes is called *multi-probing* [Andoni et al., 2015a]). That is why optimally, considered features are binary in order to use the cheap-to-compute Hamming distance. Reducing the number of distance computations clearly diminishes the query time.

Data-independent approach and introduction to the context of our first contribution The two popular methods of this family are *Hyperplane LSH* [Charikar, 2002] and *Cross-polytope LSH* [Terasawa and Tanaka, 2007]. In the streaming setting, the fact that the feature transformation is data-independent avoids the necessity of storing information on previously seen data points to compute it. It is enough to perform the linear scan as data are seen in a streaming fashion and to maintain the current nearest neighbor.

> In that case, this is critical to compute the dimensionality reduction efficiently and to reduce the cost of the distance computations.

The latter is guaranteed by working on short binary codes as the new feature representation and to apply the Hamming distance on them. Regarding the cost of the dimensionality reduction, let us denote c, $c \ll d$, the targeted dimension of the new feature vectors while d is the initial dimension of the input vectors. The computation cost of the projection of one date point for Hyperplane LSH and Cross-polytope LSH is respectively $O(cd)$ and $O(cd^2)$. The space costs are the same. Please see

Section 2.2.2 for evidence. State-of-the-art approaches, taking their inspiration from the Fast Johnson-Lindenstrauss Transform (FJLT) [Ailon and Chazelle, 2006], reduce the computation cost to respectively $O(c \log d)$ and $O(cd \log d)$ with a space cost - depending on the chosen transform - decreased to subquadratic, usually at most linear, or sometimes even constant (in d). This is enabled by heavy use of structured random projections instead of fully random ones. However, until this work they lack of a general theoretical framework comprising all existing structured transformations. In particular, the state-of-the-art structured matrix proposed by Andoni et al. [2015a] for Cross-polytope LSH $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ - where $\mathbf{H}$ is the normalized Hadamard transform (see Definition 2.2.3 p.25) and the $\mathbf{D}_i$ are random diagonal matrices with ± 1 entries for $i \in [3]$ - had no proof until this work that it performs similarly to a random rotation, despite the experimental evidence. Therefore:

> In machine learning applications approximatively preserving the distance and the structure of data, how to prove the effectiveness of the structured approach for random projections in general and $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ in particular with $O(c \log d)$ or $O(cd \log d)$ time cost and subquadratic, usually at most linear, or sometimes even constant (in d) space cost?

Data-dependent approach and introduction to the context of our second contribution Even if *data-independent* methods have already a good precision for the nearest neighbor search, sometimes it is not enough. In practice, *data-dependent* methods still have a better accuracy than LSH based on random projections. One method ticks almost all of the constraints checkboxes. This is *Online Sketching Hashing* (OSH) [Leng et al., 2015a]. It is an *online data-dependent unsupervised* hashing method from the Hypercubic hashing family inspired by the original offline state-of-the-art ITerative Quantization (ITQ) [Gong et al., 2013] algorithm. In brief, the principle is to project the data points onto the first principal components, apply a suitable learned rotation and finally the sign function to obtain binary codes. OSH uses a recent sketching technique to perform efficiently the Principal Component Analysis (PCA) in order to avoid the storage of the entire dataset. However, by default, it takes a random rotation instead of learning it carefully as for ITQ. Besides, OSH, which will be more detailed in Section 2.3.2, p. 30, is rather mini-batch than fully online. Then, an open question is:

> How to perform accurate *nearest neighbor* search by learning the smaller binary feature representations in a *streaming* fashion with *minimal space cost*? More precisely, in the frame of the state-of-the-art Hypercubic hashing family, how to propose an online version of ITQ [Gong et al., 2013], the popular state-of-the-art algoritm representing this family? How to perform the online estimation of the PCA? How to learn in an online fashion the relevant rotation after PCA-projection of the data?

Framed questions are the open problems which are partially addressed in this thesis. Before digging into more details on how the challenges are handled, let us see now the clustering problem definition and associated classical approaches.

Clustering

The clustering problem can be defined as the following.

Definition 1.1.2 (Clustering problem)**.** *Given a set of instances* $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ *and a pairwise distance* $dist(.,.)$ *or similarity function* $sim(.,.)$*, the goal of the* Clustering problem *is to find a partition* $\Pi = \{C_1, \ldots, C_K\}$ *of* $\mathcal{X}$ *such that:* $\forall i \in [K],\ \forall \mathbf{x}_u, \mathbf{x}_v \in C_i,\ \forall \mathbf{x}_w \in C_j$ *with* $j \in [K] \backslash \{i\}$*,*

$$sim(\mathbf{x}_u, \mathbf{x}_v) > sim(\mathbf{x}_u, \mathbf{x}_w) \tag{1.1}$$

and

$$sim(\mathbf{x}_u, \mathbf{x}_v) > sim(\mathbf{x}_v, \mathbf{x}_w). \tag{1.2}$$

The clustering problem consists in finding and optimizing a good objective function measuring the quality of the partition. The latter heavily depends on the choice of the similarity function. The time cost is, as for the nearest neighbors approach, subject to the number of required pairwise similarity computations.

The purpose of this thesis is not to review all clustering algorithms. We focus rather on very classical approaches. As the concept of *Cluster* is not easy to define, there are as many methods as ways to define it:

Centroid-based clustering In this model, clusters are summarized by a *central* vector which is not necessary in the dataset. Methods like k-means [Lloyd, 1982], k-medians [Jain and Dubes, 1988], k-medoids [Kaufman and Rousseeuw, 1987] and variants belong to this family model. Given an expected number of clusters k, the goal is to find k centers and to assign the data points to the nearest cluster center while minimizing the squared distance to this center. The main drawbacks of this model are:

- The number k should be specified in advance.
- The resulting clustering partition is very sensitive to the initialization of the k centers.
- It is not able to detect non-convex shaped clusters.

Distribution-based clustering This model defines a cluster as a group of points belonging most likely to the same distribution. Methods from this model have a strong theoretical foundation but the choice of an approximate distribution is difficult. The most popular model of this kind of approaches is known as Gaussian mixture models and is solved with the Expectation-maximization algorithm [Dempster et al., 1977].

Connectivity-based clustering also named as hierarchical clustering Within this family of methods, two approaches are possible: *agglomerative* or *divisive*. In the agglomerative version, all points are initially individual clusters. Then for the next steps, the two nearest clusters are merged until there remains only one cluster or K. In the divisive version, all points are contained in one cluster and the clusters with the largest "variance" are split until K clusters are reached or the clusters contain one point. For both cases, the key question is: how to compute the similarity or the distance between two clusters? Popular choices of distance between two clusters are: the minimum distance between the points of two different clusters (single-linkage clustering [Sibson, 1973]), the maximum (complete linkage clustering [Defays, 1977]) or the mean (average linkage clustering [Day and Edelsbrunner, 1984]). A particularity of these methods is that they do not produce a unique partitioning of the dataset, but a *hierarchy*. Some drawbacks that can be mentioned:

- They are sensitive to outliers which can cause additional clusters or unwilling clusters merging.
- The complexity is $O(N^3)$ for N data points with agglomerative clustering and $O(2^{N-1})$ for the divisive one. This is prohibitive for large datasets.

Density-based clustering For density-based clustering, clusters are modelled as areas of higher density separated by sparse ones. The points in sparse areas are considered as noise or border points. One prominent method is known as DBSCAN [Ester et al., 1996]. The building of the clusters is based on "density-reachability". Inspired by the linkage-based clustering, DBSCAN connects points within a given distance threshold if and only if there is a minimum number of other points within this radius. In this model, a cluster contains all density-connected points plus all the ones that are within these points' range. The advantages of DBSCAN are:

- The concept of density-reachability enables the recovering of clusters with arbitrary shapes.
- In terms of time complexity, DBSCAN requires only a linear number of range queries on the database.

Nevertheless, DBSCAN needs two parameters to tune: the radius for the reachability distance and the minimum number of points to consider a group of points as a cluster. So the following question remains open:

> How to recover a partition of non-convex clusters without any parameter with time and space complexities linear in the number of data points?

To this end, in Section 2.3.4, state-of-the-art space-efficient clustering algorithms will be reviewed before introducing our model in Chapter 5.

1.1.3 Nearest neighbor search and clustering of high-dimensional massive data through compression

Dimensionality reduction, hashing and sampling

In Section 1.1.2, we motivated the choice to consider classical unsupervised learning problems such as nearest neighbor search (and dimensionality reduction as a tool for it) and clustering. The very brief overview of classical approaches for these distance/structure preserving algorithms stresses that they do not necessary take into account all the major constraints[3] we have to face though in real-world applications. It is recalled that these constraints require analysis of data in an unsupervised manner for large-scale datasets with high-dimensional data points potentially in a streaming fashion while providing a model with acceptable precision. Some state-of-the-art methods take into account one or several of these constraints to preserve approximately the distance/structure. Other ones should be adapted. How to realize this adaptation for handling these massive high-dimensional data?

> A natural idea to handle massive high-dimensional data is to apply the learning task on a *compressed* representation of the dataset which eases the process by allowing space savings and some increase of the data processing rate of flow.

[3] Enumerated in Section 1.1.1.

To respect this constraint, we impose that this compression, if *learned*, should be done in an unsupervised manner. We already mentioned *dimensionality reduction* as a compression tool for fighting against the curse of dimensionality. Coupled with *hashing*, it reduces time and space complexities of the algorithms by compressing each data point individually but without reducing the size of the dataset. On the contrary, *sampling* [Vitter, 1985] is another "compression" tool which selects a reduced number of data points to keep from the dataset without touching the number of features. Finally, the technique of *sketching* borrows the best to all these three concepts for data compression. We briefly expose the advantages of sketching and how it is well-adapted to stream data in the two following Sections.

Sketching

Sketching [Cormode et al., 2012] compacts the whole database into a single vector, named *sketch* but still approximates it well by enabling to infer its relevant properties. Most of considered sketches are *linear*. This means that when a data point is added to the database, updating the corresponding compact data structure is simply adding the individual sketch of this single item. This linearity implies also that the database can be divided in different chunks for which local sketches are computed in parallel and then, the final sketch of the entire database is simply the concatenation of the local ones. This is very interesting for distributed computing and stream data. Sketching is indeed closely linked with the development of streaming algorithms that have in common to heavily rely on these compact probabilistic data structures in order to handle data streams while observing the strong space constraints.

The streaming model was described in works from Munro and Paterson [1980], Flajolet and Martin [1985], Henzinger et al. [1999] and real interest in streaming algorithms follows the work from Alon et al. [1996] where lower and upper bounds for the space complexity are provided to approximate frequency moments of sequence of data points. Early work focused indeed on processing numerical data for estimating basic statics such as quantiles [Agrawal and Swami, 1995, Cormode and Muthukrishnan, 2005, Ma et al., 2013], heavy hitters [Boyer and Moore, 1991, Cormode and Muthukrishnan, 2005], or the number of distinct elements [Flajolet and Martin, 1985, Durand and Flajolet, 2003, Flajolet et al., 2007] in a stream. Finally, regarding the streaming algorithms field, three great surveys should be stressed out. Work from Muthukrishnan [2005] introduces general techniques while the survey from Cormode and Muthukrishnan [2012] highlights applications of one of the most well known sketch approach, namely *Count-Min Sketch*. More recently, McGregor [2014] discusses streaming algorithms applied to graphs.

The price of compression: approximation and looking for a trade-off

> Data compression due to the lossy information, either by dimensionality reduction, sampling or sketching comes with a price. This is the result of the processing algorithm, namely the model or the answer which is approximated.

Hopefully, an approximative result is enough in many applications. For instance, while wanting to compute the number of hits of each Wikipedia corpus webpage (about 35 millions of Wikipedia pages are stored), a simple magnitude of order is already meaningful. Hence, in this thesis while wanting to construct compact data representations in order to address some distance and structure preserving unsupervised problems, we should keep in mind this *trade-off* game between:

1. the space cost,
2. the rate of flow and
3. the accuracy of the result.

They should be parametrized according to the user's needs. The principle is summarized in Figure 1.2. To end this introduction, let us enumerate the different contributions described in this thesis.

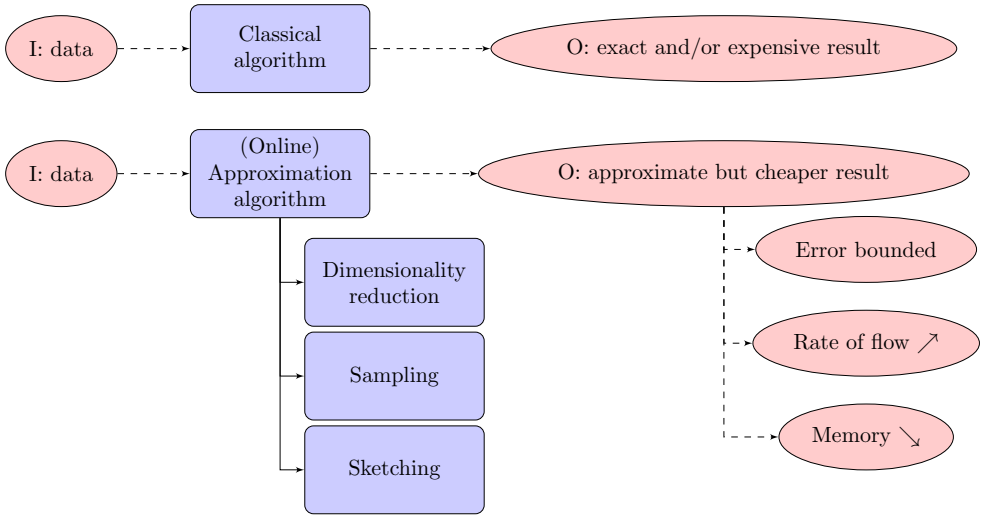


FIGURE 1.2: How to handle the constraints. I: Input, O: Output. (up) The classical algorithm vs (bottom) an (online) approximation algorithm applied on compressed data. The compression can be performed with dimensionality reduction, sampling or sketching techniques. The result of an approximation algorithm is approximate but has been obtained at a cheaper cost: the rate of flow has been sped up while the memory requirements have been reduced.

1.2 Main contributions and outline of the thesis

In this thesis, three main contributions are addressing the previously stated problems. They are respectively *data-independent*, *data-dependent* (parameters are learned) and *graph-based* corresponding to the three principal approaches for preserving approximatively the structure and the distance of the data points. Driven by the needs pointed in Section 1.1.3, our contributions are approximative distance and structure-preserving algorithms for nearest neighbors search and clustering in unsupervised learning. These approximation algorithms rely on a certain compression of the input data. Our ways of compression for each contribution can be summarized in one word: *structuring*, *hashing* and *minimum spanning tree* (so one part of this thesis' title). Our contributions are now briefly summed up in the following sections and schematized in Figure 1.3.

1.2.1 Structured random matrices, an approach for fast and large-scale machine learning computations

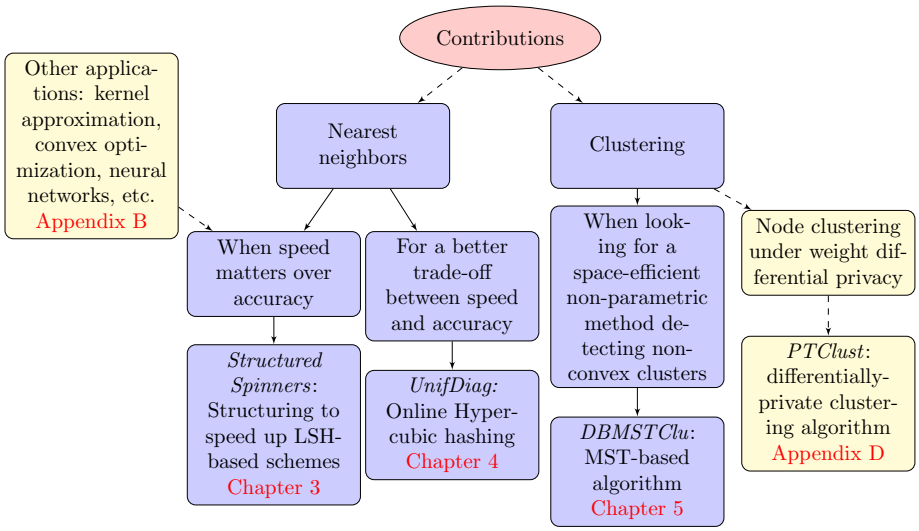


FIGURE 1.3: Overview of our contributions to unsupervised learning from massive high-dimensional data streams.

A structuring technique is explained to speed up the state-of-the-art *data-independent* LSH-based schemes for Approximate Nearest Neighbor (ANN) search.

We recall briefly, LSH-based schemes [Indyk and Motwani, 1998, Charikar, 2002, Terasawa and Tanaka, 2007, Sundaram et al., 2013, Andoni et al., 2015a] consist in performing dimensionality reduction of the high-dimensional input vectors with random projections to make the similarity search faster on the smaller obtained feature vectors. When the dimensions of the projection matrix and the vector are high, this product computation can be very expensive. To remedy this problem, it is suggested to replace the use of random matrices by structured ones chosen from the family of the introduced model *Structured Spinners*. Matrices of the *Structured Spinners'* family are formed as products of three structured matrix-blocks that incorporate rotations and follow some conditions. Using this kind of matrices leads both to space cost reduction and considerable speedups of the matrix-vector product thanks to fast Fourier or Hadamard transforms that are applied instead.

Theoretical guarantees characterizing the capacity of the structured model in reference to its unstructured counterpart are provided.

They rely essentially on concentration theorems such as the results for random vectors of the Berry-Esseen type Central Limit Theorem [Bentkus, 2003] or Azuma's inequality [Azuma, 1967]. They demonstrate that there is almost no loss of accuracy in the considered applications, as confirmed by the experimental part. As a by-product, the proposed technique is generalizable to lots of machine learning applications relying on matrice-vector products where the parameters of the matrix are random (typically, each entry of the matrix has been drawn from a Gaussian distribution) or learned, as shown in Appendix C. A few examples include vector quantization, kernel approximation via random feature maps, convex optimization with Newton sketches and deep neural networks.

1.2.2 Learning compact binary codes from massive data streams with hypercubic hashing for Nearest Neighbors search

When *data-independent* LSH techniques do not provide sufficient accuracy for similarity search involved in ANN and clustering, we propose a new method for *learning* compact similarity-preserving binary codes for massive high-dimensional data streams.

Given an expected code length c and high-dimensional input data points, the presented algorithm provides a binary embedding of c bits aiming at preserving the distance between the points from the original high-dimensional space. The method can be categorized into the family of *hypercubic hashing*. It combines:

1. the online estimation of the principal subspace,
2. the projection of the data points onto the c first estimated approximate principal components of the covariance matrix,
3. the rotation of the approximate PCA-projected data points where the rotation is learned so that to equalize the variance over the different approximate PCA directions,
4. and the application of the sign function pointwise (defined later by Equation 2.11, p.23) in order to obtain binary codes.

Some theoretical results in this thesis attempt to justify for the first time, to the best of our knowledge, the requirement of a rotation matrix after the PCA projection in hypercubic hashing. This has been only intuitively motivated until this work.

We denote d the dimension of the data in the initial high-dimensional space. This algorithm is particularly time and space efficient with complexities respectively in $O(dc + c^2)$ and $O(c^2)$ additional to the storage of the projection matrix and the rotation. This outperforms the state-of-the-art online hypercubic hashing method on this aspect.

The quality of the obtained binary codes is demonstrated through extensive experiments on real data for the nearest neighbors search task in the online setting. The algorithm provides a similar quality of the binary embeddings in comparison with the competing state-of-the-art online hypercubic hashing approaches.

1.2.3 A Minimum Spanning Tree-based approach for clustering massive data

For this last contribution, we propose a new clustering algorithm named DBMSTClu based on the *graph approach* for preserving the structure of the data.

The dataset can be represented as a dissimilarity graph but for large-scale applications, this graph is too big to be entirely loaded into memory. Therefore, the information on the dataset will be compressed into a Minimum Spanning Tree (MST)[4] of this data dissimilarity graph. Relying exclusively on a MST helps to respect the linear time and space complexity constraints. As a new space-efficient density-based non-parametric clustering algorithm, DBMSTClu, by performing suitable cuts on the MST, enables the recovery of arbitrary-shaped data clusters from massive datasets.

[4] See Definition 2.4.5 p.43.

- *Space-efficient:* It is applied only on a Minimum Spanning Tree (MST) $\mathcal{T}$ of the dissimilarity graph $\mathcal{G}$ between the N objects to cluster. For N objects, there are only $N-1$ edges to store.
- *Density-based:* Unlike k-means, k-medians or k-medoids algorithms, it does not fail at distinguishing clusters with particular (possibly embedded) structures.
- *Non-parametric:* No input parameter (such as the number of clusters or characteristics of the clusters) is needed contrarily to DBSCAN or the Spectral Clustering method.

Thanks to the property of the MST for expressing the underlying structure of a graph, the algorithm correctly detects the right number of non-convex clusters by cutting suitable edges on $\mathcal{T}$.

The optimality of the MST-based approach is indeed justified in this thesis, to the best of our knowledge, for the first time.

In DBMSTClu, the cut criterion to maximize, named DBCVI for Density-Based Clustering Validation Index, is based on the weighted sum of indices computed for each cluster. At the cluster level, one can compute this index by taking the normalized subtract of the *Separation* and *Dispersion* of the cluster. The *Separation* measures how the clusters are well-separated while low *Dispersion* expresses a dense cluster with less empty space. DBMSTClu basically performs cuts on the MST as long as the cut criterion can be improved by taking at each step the cut maximizing the DBCVI. Theoretical guarantees on the retrieval of the exact clustering partition are procured and DBMSTClu's advantage over the existing state-of-the-art is exhibited on several datasets. As a by-product, the algorithm accompanied by theoretical guarantees has been successfully applied to differentially-private clustering.

Nevertheless, storing a dissimilarity graph $\mathcal{G}$ costs theoretically $O(N^2)$ edges. So, an issue is:

How to compute space-efficiently a Minimum Spanning Tree of a weighted graph?

As a solution to retrieve efficiently an MST from it, $\mathcal{G}$ can be handled as a stream of edge weight updates and only a *limited number of linear measurements* of $\mathcal{G}$ is kept. $\mathcal{G}$ is sketched in one pass over the data into a compact structure requiring $O(N \operatorname{polylog}(N))$ space. Hence, one can say that the sketching phase follows the dynamic semi-streaming model. Then, an approximate MST is recovered from this graph sketch in $O(N \operatorname{polylog}(N))$ time.

Outline of the thesis

For presenting these contributions, the thesis is organized as the following. A first Chapter (Chapter 2) of this thesis is dedicated to enumerating state-of-the-art *data-independent*, *data-dependent* and *graph-based* approaches for the nearest neighbor search and clustering. Then, follow three Chapters (Chapters 3, 4, 5) devoted to the three main obtained results for which Sections 1.2.1, 1.2.2 and 1.2.3 gave a bird's-eye view. Finally, some perspectives described in the last Chapter (Chapter 6) conclude this work. Appendices C, D and E constitute some complements to respectively Chapter 3, 4 and 5. Appendix F enumerates the publications made during this PhD thesis.

Chapter 2

Related work

Contents

2.1 Introduction

Addressed problems in this thesis are from the field of *unsupervised* machine learning. Specifically, it is attempted to design efficient distance and structure preserving algorithms for essential exploratory tasks such as *nearest neighbors search* and *clustering*. They are constrained by bounded memory and the necessity to process massive high-dimensional datasets very fast, possibly in an online manner. Though, a certain approximation on the result, in comparison with the exact answer obtained with unlimited time and memory, is allowed. The challenge is in finding a good balance between time, memory usage and precision of the model.

It will be seen in this Chapter that approaches for solving this problem may be from three different types: *data-independent*, *data-dependent* and *graph-based*. See Figure 2.1 for an illustration. Some existing algorithms from each class of methods will be described. They do not always take into account all aforementioned constraints. The limits of these methods raise the open questions that we tend to address in this thesis:

> How to perform data-independent or data-dependent similarity search in an efficient manner?

> How to recover non-convex clusters without any parameter with time and space complexities linear in the number of data points?

More particularly, we began in Chapter 1 to draw the contours of the proposed models. Compression (hashing, sampling, dimensionality reduction, sketching) is the key tool to achieve this goal. We consider:

1. a *data-independent* dimensionality reduction method based on structured random projections,

2. a *data-dependent* hashing approach relying on an online estimation of the Principal Components Analysis (PCA) and

3. a *graph-based* technique for clustering which summarizes a dataset by the underlying Minimum Spanning Tree of the dissimilarity data graph.

We recap there all the corresponding open questions:

In machine learning applications approximatively preserving the distance and the structure of data, how to prove the effectiveness of the structured approach for random projections in general and $\mathbf{HD_3HD_2HD_1}$ in particular with $O(c \log d)$ or $O(cd \log d)$ time cost and subquadratic, usually at most linear, or sometimes even constant (in d) space cost?

How to perform accurate *nearest neighbor* search by learning the smaller binary feature representations in a *streaming* fashion with *minimal space cost*? More precisely, in the frame of the state-of-the-art Hypercubic hashing family, how to propose an online version of ITQ [Gong et al., 2013], the popular state-of-the-art algoritm representing this family? How to perform the online estimation of the PCA? How to learn in an online fashion the relevant rotation after PCA-projection of the data?

How to compute space-efficiently a Minimum Spanning Tree of a weighted graph?

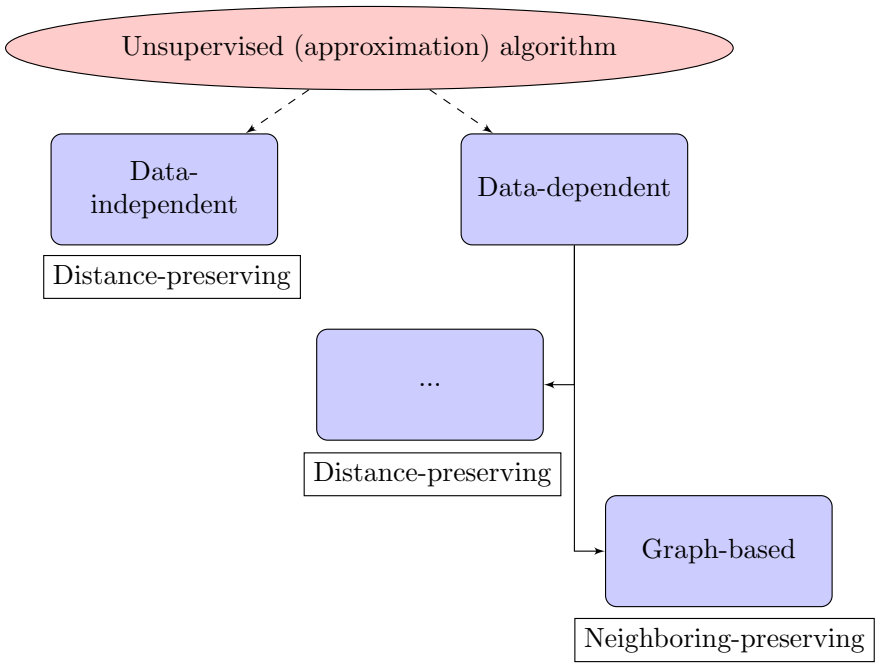


FIGURE 2.1: Typology of considered methods for unsupervised (approximation) algorithms. Graph-based methods are data-dependent but perform a relaxation of the distance-preserving constraint. They preserve the structure, the neighborhood of the points, which is looser than the distance.

Plan of the Chapter In the remaining part of this Chapter, we will present fundamental principles of state-of-the-art methods competing with our proposed models and the ones from which we have been inspired. We begin to answer partially to the previous raised framed questions by introducing preliminaries to our models. To that purpose, the description is made by following the distinctive scheme *data-independent* (Section 2.2) / *data-dependent* (Section 2.3) / *graph-based* (Section 2.4). More precisely, we will cover more or less briefly hashing (Section 2.2.1), Locality-Sensitive Hashing and variants (Section 2.2.2), (structured) random projection (Section 2.2.3), PCA, associated data-dependent hashing techniques and some efficient online estimations of PCA (Section 2.3.2), space-efficient (Section 2.3.4) and graph-based clustering methods (Section 2.4.1) and a graph-sketching technique for retrieving a Minimum Spanning Tree (Section 2.4.2).

2.2 Data-independent approaches

Among *data-independent* methods, we choose here to detail Locality-Sensitive Hashing (LSH) [Indyk and Motwani, 1998], the state-of-the-art for similarity search and for which we made some improvements. First, this is necessary to explain the concept of *hashing* behing LSH.

2.2.1 Hashing

Problems addressed in this thesis heavily rely on *hashing* to reduce the space cost of the handled objects. In particular, randomness involved in the compact data structures is due to random functions from a family of *hashing* or *hash* functions applied to the input data in order to make them have a certain independence. A family of functions $\mathcal{H}$ is defined as follows: $\mathcal{H} := \{h \mid h : S \to D\}$ where S (resp. D) is the source (resp. destination) set. The cardinality of S is potentially infinite while the cardinality of D is controlled. The properties of $\mathcal{H}$ are:

- Every function $h \in \mathcal{H}$ is easy to represent.
- For any $x \in S$, the evaluation of $h(x)$ is cheap to compute.
- If $|S| > |D|$, then,
  - there are collisions i.e. for any $h \in \mathcal{H}$, there are $x, y \in S, x \neq y$ such that $h(x) = h(y)$. In particular, if $|S|$ is small, hashed values of elements in S have smaller probability of collision.
  - every $h \in \mathcal{H}$ is irreversible.

Hash functions are sometimes required to be k-wise independent or pairwise independent. The definition of pairwise independent hash functions is given below in Definition 2.2.1, the case of k-wise independence can be easily derived.

Definition 2.2.1 (Pairwise independent hash functions)**.** *A family of functions* $\mathcal{H} = \{h \mid h : S \to D\}$ *is* pairwise independent *if for any* h *randomly drawn from* $\mathcal{H}$*, these two conditions hold:*

1. $\forall x \in S$*, the random variable* $h(x)$ *is uniformly distributed in* D*.*
2. $\forall x_1 \neq x_2 \in S$*, the random variables* $h(x_1)$ *and* $h(x_2)$ *are independent.*

This definition implies that over all random choices of a function $h \in \mathcal{H}$, for any $x_1 \neq x_2 \in S$ and $y_1, y_2 \in D$,

$$\mathbb{P}_{h\in\mathcal{H}}[(h(x_1) = y_1) \cap (h(x_2) = y_2)] = 1/|D|^2.$$

Typically, *hash functions* are used in *hash tables* with $|S| > |D|$. They enable to quickly access a data record from its *hashed* search key: the latter points out a *bucket* of the hash table where a set of records is stored. For conventional hash functions, such as those used in cryptography (ex: SHA-1 [National Security Agency, 1995]), the probability of collisions is supposed to be minimized. Figure 2.2 illustrates this principle.

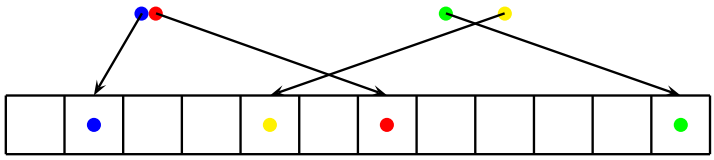


FIGURE 2.2: For points in 1 dimension, illustration of *hashing* to fill a hash table with 12 buckets. The hash function should be defined such that collisions should be avoided as much as possible.

Suppose there are n elements to place in k buckets and the considered hash function is ideal i.e. with a uniform distribution. Then Propositions 2.2.1, 2.2.2 and 2.2.3 hold. They are illustrative results for which the proof is easily re-demonstrable.

Proposition 2.2.1. *Considering the problem of placing n elements in k buckets of a hash table with a uniformly distributed hash function, the probability that at least two elements are in collision is approximatively*

$$1 - e^{\frac{-n(n-1)}{2k}}. \tag{2.1}$$

Proof. After placing the first item in a bucket, the probability that the second finds an empty bucket is $\frac{k-1}{k}$. By generalization, the probability that there is no collision is

$$p = \frac{k-1}{k} \times \frac{k-2}{k} \times \ldots \times \frac{k-(n-2)}{k} \times \frac{k-(n-1)}{k}. \tag{2.2}$$

The Taylor series expansion of the exponential function provides a first-order approximation for e^x with $x \ll 1$: $e^x \approx 1 + x$. Thus, $e^{-\frac{c}{k}} \approx 1 - \frac{c}{k} = \frac{k-c}{k}$ for $c \in \{1, 2, \ldots, n-2, n-1\}$. So,

$$p \approx e^{-\frac{1}{k}} \times e^{-\frac{2}{k}} \times \ldots \times e^{\frac{-(n-2)}{k}} \times e^{\frac{-(n-1)}{k}} = e^{\frac{-(1+2+\ldots+(n-2)+(n-1))}{k}} = e^{\frac{-n(n-1)}{2k}}. \tag{2.3}$$

The probability that at least two elements are in collision is $1-p$ which completes the proof. □

Proposition 2.2.2. *Considering the problem of placing n elements in k buckets of a hash table with a uniformly distributed hash function, the expected number of elements*

which will be in collision is

$$\begin{aligned}&\mathbb{E}[\textit{number of elements which collide with at least another one}]\\&= n\left(1-\left(1-\frac{1}{k}\right)^{n-1}\right).\end{aligned} \tag{2.4}$$

Proof. Given two elements x, y, the probability that x collides with y is $1/k$. So the probability that x does not collide with y is $1-1/k$. The probability that $n-1$ items do not collide with x is then $\left(1-\frac{1}{k}\right)^{n-1}$. By complementarity, the probability that at least one element collides with x is $1-\left(1-\frac{1}{k}\right)^{n-1}$. Therefore

$$\begin{aligned}&\mathbb{E}[\text{number of elements which collide with at least another one}]\\&= n\left(1-\left(1-\frac{1}{k}\right)^{n-1}\right).\end{aligned} \tag{2.5}$$

□

Proposition 2.2.3. *Considering the problem of placing n elements in k buckets of a hash table with a uniformly distributed hash function, the expected number of buckets containing collisions is*

$$\mathbb{E}[\textit{number of buckets with 2 or more items}] = k - k\left(1-\frac{1}{k}\right)^{n} - n\left(1-\frac{1}{k}\right)^{n-1}. \tag{2.6}$$

Proof. The expected number of buckets containing collisions is equal to

$$\begin{aligned}&\mathbb{E}[\text{number of buckets with 2 or more items}]\\&\quad= k - \mathbb{E}[\text{number of empty buckets}]\\&\qquad- \mathbb{E}[\text{number of buckets with only one element}].\end{aligned} \tag{2.7}$$

- The probability that any bucket is chosen for one given element is $1/k$. So the probability that it is not chosen is $1-1/k$. After n elements, the probability that any bucket is still empty is $\left(1-\frac{1}{k}\right)^{n}$. Therefore

$$\mathbb{E}[\text{number of empty buckets}] = k\left(1-\frac{1}{k}\right)^{n}. \tag{2.8}$$

- Given two elements x, y, the probability that x collides with y is $1/k$. So the probability that x does not collide with y is $1-1/k$. The probability that $n-1$ items do not collide with x is then $\left(1-\frac{1}{k}\right)^{n-1}$. Therefore

$$\mathbb{E}[\text{number of buckets with only one element}] = n\left(1-\frac{1}{k}\right)^{n-1}. \tag{2.9}$$

This completes the proof. □

2.2.2 Locality-Sensitive Hashing (LSH)

For a family of hashing functions $\mathcal{H} = \{h \mid h : S \to D\}$ where S (resp. D) is the source (resp. destination) set, when the cardinality of S is such that some collisions are not avoidable, it could be useful to *exploit the collision probabilities* by defining hash functions preserving a relationship for instance, the *distance* between the input data points. This is the goal of *Locality-Sensitive Hashing* (LSH) [Indyk and Motwani, 1998] in order to perform Approximate Nearest Neighbor (ANN) search. While Figure 2.2 shows general hashing, Figure 2.3 illustrates LSH.

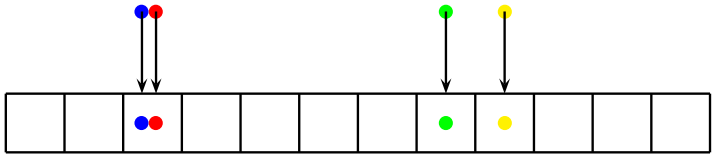


FIGURE 2.3: For points in 1 dimension, illustration of *Locality-Sensitive Hashing* to fill a hash table with 12 buckets. The hash function should be defined such that near points in the initial space have a high probability to collide. On the contrary, far points have a low probability to collide.

Locality-Sensitive Hashing (LSH) [Indyk and Motwani, 1998] is a method for approximative search in high-dimensional spaces and constitutes a solution to the curse of dimensionality problem. A classic application is the *Approximative Nearest Neighbors* (ANN) search.

LSH uses a family of hash functions to hash input data into several hash tables so that similar data points are mapped to the same buckets in different hash tables with high probability, the number of buckets being obviously much smaller than the universe of possible input data. Let $dist(x, y)$ denote any distance function between points x and y (some examples of distance functions are given in Appendix A). Formally, a family $\mathcal{H}$ is called (R, aR, p_1, p_2)-*sensitive* with R and a some strictly positive thresholds, $p_1 \gg p_2$, if $\mathcal{H} = \{h \mid h : S \to D\}$ satisfies the following conditions for any two points $x, y \in S$ and any $h \in \mathcal{H}$:

- If $dist(x, y) \leq R$, then $\mathbb{P}_{h \in \mathcal{H}}[h(x) = h(y)] \geq p_1$.
- If $dist(x, y) \geq aR$, then $\mathbb{P}_{h \in \mathcal{H}}[h(x) = h(y)] \leq p_2$.

The constant a quantifies the gap between what it is meant for a distance to be "near" or "far". The quality of a hash function depends on the two parameters: p_1, the collision probability for nearby points, and p_2 the collision probability for points that are far apart. The gap between p_1 and p_2 expresses how "sensitive" the hash function is to variations in the distance. It is summed up by the quantity $\rho = \frac{\log(1/p_1)}{\log(1/p_2)}$ which is usually written as a function of the distance gap a.

The LSH algorithm for ANN search works in two steps, with parameter c denoting the number of hash functions to concatenate for constructing a hash value and L, the number of hash tables:

1. *Building of the data structure:* The algorithm defines L families of hash functions $\mathcal{H}_1, \ldots, \mathcal{H}_L$ for each table $i \in [L]$. For a given point x and a hash table $i \in [L]$, the hash value $f_i(x)$ is the concatenation of c hash functions $h_1^{\mathcal{H}_i}, \ldots, h_c^{\mathcal{H}_i}$ chosen uniformly at random in $\mathcal{H}_i$. It means that for any input point x, for all $i \in [L]$,
$$f_i(x) = [h_1^{\mathcal{H}_i}(x) \ldots h_c^{\mathcal{H}_i}(x)]. \tag{2.10}$$

Then L hash tables are built where the i-th hash table contains all the points from S, hashed into buckets according to function f_i. If $|S| = N$, then the space required for LSH is $O(NL)$.

2. *Nearest neighbors query:* For a given query point x, the algorithm iterates on the L functions to return the points located in the same bucket than x for each hash table. Then the brute-force is applied to the union of all returned points to find the nearest neighbors: the distances between all points are computed and the nearest neighbors are retrieved. Note that the nearest two points are, the higher the probability that these two points collide together into the same bucket among all the hash tables.

   Also note that some hash functions from the same family can have an intrinsic distance. In this case, the search can be enlarged to the nearest bucket(s) defined by the one(s) pointed by the hash functions for each hash table. This is *multiprobing* [Lv et al., 2007, Andoni et al., 2015a]. It is necessary though, to define carefully what are the neighboring buckets according to the used hash functions.

Figure 2.4 illustrates the whole algorithm. Determining good Locality-Sensitive Hash

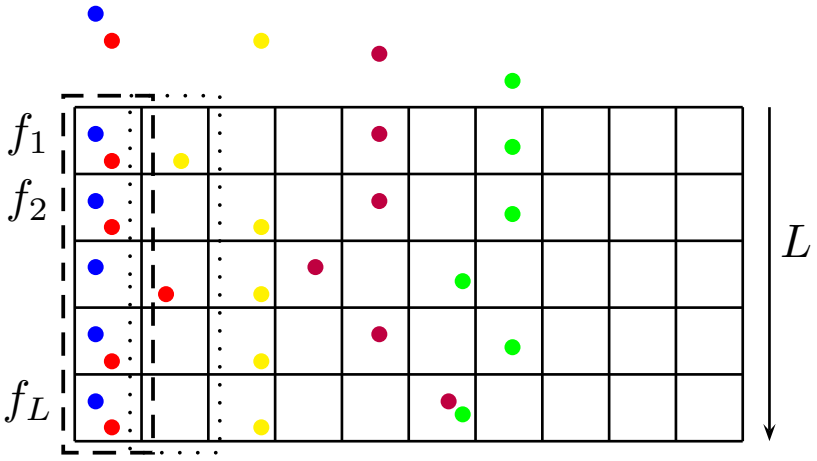


FIGURE 2.4: For points in 2 dimensions, illustration of *Locality-Sensitive Hashing* to fill a hash table with 12 buckets. The hash function should be defined such that near points in the initial space have a high probability to collide. On the contrary, far points have a low probability to collide. If the blue point is a query point to find its nearest neighbors, find them among the points from the same buckets (union represented by the dash line) among the hash tables. Thus, the red point is returned. The search can be enlarged to the nearest bucket for each hash table (delimited by the dotted line) and then the yellow point is considered.

functions and thus, designing LSH-based efficient nearest neighbor search algorithms are problems that have attracted much interest over the last few years [Andoni and Indyk, 2008, Raginsky and Lazebnik, 2009, Grauman and Kulis, 2011]. Two variants of LSH are dominant, *Hyperplane* [Charikar, 2002] and *Cross-polytope* LSH [Terasawa and Tanaka, 2007]. In the next Sections, they are explained but restricted to the case of only one hash table for writing simplicity.

Hyperplane LSH

Given a set of data points in the unit sphere $S^{d-1} \subset \mathbb{R}^d$, the family of hash functions $\mathcal{H}$ in Hyperplane LSH [Charikar, 2002] is defined as follows. Taking a random vector $\mathbf{r} \in \mathbb{R}^d$ such that each entry of $\mathbf{r}$ is drawn from the normalized Gaussian distribution noted $\mathcal{N}(0,1)$, we defined a hash function $h_{\mathbf{r}}$ corresponding to this vector such that for any data point $\mathbf{x} \in S^{d-1}$,

$$h_{\mathbf{r}}(\mathbf{x}) := \text{sign}(\mathbf{r} \cdot \mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{r} \cdot \mathbf{x} \geq 0 \\ -1 & \text{if } \mathbf{r} \cdot \mathbf{x} < 0 \end{cases} \tag{2.11}$$

Denoting $dist_{ang}$ the angular distance (see Definition A.0.8 p.128), one has the following locality sensitive hashing scheme:

$$\mathbb{P}[h_{\mathbf{r}}(\mathbf{x}) = h_{\mathbf{r}}(\mathbf{y})] = 1 - dist_{ang}(\mathbf{x}, \mathbf{y}). \tag{2.12}$$

The data point $\mathbf{x} \in S^{d-1}$ is then represented by the hash vector

$$f(\mathbf{x}) = [h_{\mathbf{r}_1}(\mathbf{x}) \dots h_{\mathbf{r}_c}(\mathbf{x})] = \text{sign}(\mathbf{R}\mathbf{x}) \tag{2.13}$$

where the sign function is applied pointwise and $\mathbf{R} \in \mathbb{R}^{c \times d}$ is the matrix formed by $\mathbf{r}_1, \dots, \mathbf{r}_c$ as rows. In practice, the corresponding implementations on real data decrease the query time complexity by multiple orders of magnitude in comparison with a linear scan [Lv et al., 2007, Sundaram et al., 2013]. Nevertheless, Hyperplane LSH has worse theoretical guarantees than Cross-polytope LSH [Terasawa and Tanaka, 2007].

Cross-polytope LSH

Cross-polytope has been proposed by Terasawa and Tanaka [2007] and its performance has been recently further analyzed by Andoni et al. [2015a].

Given a set of data points in the unit sphere $S^{d-1} \subset \mathbb{R}^d$, the family of hash functions $\mathcal{H}$ in Cross-polytope LSH is defined as follows. Taking a random matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ such that each entry of $\mathbf{A}$ is drawn from the normalized Gaussian distribution[1], we defined a hash function $h_{\mathbf{A}}$ corresponding to this matrix such that for any data point $\mathbf{x} \in S^{d-1}$,

$$h_{\mathbf{A}}(\mathbf{x}) := \eta\left(\frac{\mathbf{A}\mathbf{x}}{\|\mathbf{A}\mathbf{x}\|_2}\right), \tag{2.14}$$

where $\eta(.)$ returns the point closest to $\frac{\mathbf{A}\mathbf{x}}{\|\mathbf{A}\mathbf{x}\|_2}$ from the set $\{\pm\mathbf{e}_i\}_{1 \leq i \leq d}$, with $\mathbf{e}_i$ being the i-th standard basis vector of $\mathbb{R}^d$. The data point $\mathbf{x} \in S^{d-1}$ is then represented by the hash vector $f(\mathbf{x}) = [h_{\mathbf{A}_1}(\mathbf{x}) \dots h_{\mathbf{A}_c}(\mathbf{x})]$.

Theorem 2.2.1 from Andoni et al. [2015a] (Theorem 1 in the paper) bounds the collision probability for two points under the above family $\mathcal{H}$. We let the reader refer to the corresponding paper for the proof.

Theorem 2.2.1 (Andoni et al. [2015a]). *Suppose that $\mathbf{x}, \mathbf{y} \in S^{d-1}$ such that $\|\mathbf{x} - \mathbf{y}\|_2 = \tau$, where $0 < \tau < 2$. Then,*

$$\ln \frac{1}{\mathbb{P}_{h \sim \mathcal{H}}[h(\mathbf{x}) = h(\mathbf{y})]} = \frac{\tau^2}{4 - \tau^2} \cdot \ln d + O(\ln \ln d). \tag{2.15}$$

[1] When d is large, $\mathbf{A}$ can be considered as a rotation.

By comparing both Hyperplane and Cross-polytope LSH, it becomes obvious that the Cross-polytope LSH is not quite practical. The main bottleneck is generating and storing the $\mathbf{A}_i$ matrices, and finally multiply them with the data points. For each $\mathbf{A}_i$, this costs $O(d^2)$ time and space which is infeasible for large d. That is why for any matrix $\mathbf{A}$, Andoni et al. [2015a] proposed instead to multiply the input vector with matrix $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ where $\mathbf{H}$ is the normalized Hadamard transform, and $\mathbf{D}_i$ for $i \in \{1, 2, 3\}$ is a random diagonal ± 1-matrix. $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ is an orthogonal transformation which requires only $O(d)$ memory space. Its product with any input vector can be computed in $O(d \log d)$ time using the Fast Hadamard Transform. Andoni et al. [2015a] noted experimentally that three applications of $\mathbf{HD}_i$ are exactly equivalent to applying a true random rotation when d tends to infinity, while for instance, only two applications of $\mathbf{HD}_i$ are not sufficient. But they are not giving a theoretical explanation.

How to prove that for Cross-polytope LSH, $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ is equivalent to a random Gaussian matrix in terms of accuracy of the similarity search?

We provide an answer to this question in Chapter 3. In the meantime, as the theory governing the LSH-based approach comes from the field of random projections, commonly used for instance in dimensionality reduction, let us see the corresponding fundamentals.

2.2.3 Random Projection

Random projection is a cost-efficient alternative to Principal Component Analysis (PCA) (see Section 2.3.2, p.27) for dimensionality reduction since it is *data-independent*. The theoretical foundations are given by the Lemma from Johnson and Lindenstrauss [1984]:

Lemma 2.2.1 (Johnson and Lindenstrauss [1984]). *Let* $\epsilon \in]0, 1[$, $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^d$. *Let* $c \in \mathbb{N}$, *s.t.* $c \geq C\epsilon^{-2}\log(N)$ *for some constant* C. *Then there exists a linear map* $\mathbf{\Phi} : \mathbb{R}^d \to \mathbb{R}^c$ *such that :*

$$\forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X},\ (1-\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq \|\mathbf{\Phi}\mathbf{x}_i - \mathbf{\Phi}\mathbf{x}_j\|_2 \leq (1+\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2. \tag{2.16}$$

This Lemma states that N data points from an Euclidean space can be projected onto a lower dimensional space with a dimension equal to $O(1/\epsilon^2 \log(N))$ while incurring a distorsion bounded by $1 \pm \epsilon$ in the pairwise distances of the points. To achieve this with high probability, the matrix $\mathbf{\Phi}$ to choose is required to be dense and can be a near orthonormal random matrix i.e. $\mathbf{\Phi}$ is drawn uniformly from all the near orthonormal $c \times d$ matrices. As a result, when comparing with the original space, there is a near isometric embedding in the target space which preserves pairwise distances and angles.

Dense distribution

Since Johnson-Lindenstrauss (JL) Lemma, some progress has been made concerning the required dense distribution of $\mathbf{\Phi}$. Instead of drawing uniformly at random a matrix, it is enough to draw each entry from an identically and independently distributed (i.i.d.) Gaussian distribution [Frankl and Maehara, 1987]:

$$\mathbf{\Phi}_{i,j} \sim \mathcal{N}(0, (\frac{1}{\sqrt{c}})^2) \tag{2.17}$$

where c is the lower targeted dimension. Then Indyk and Motwani [1998] and Dasgupta and Gupta [1999] simplified the JL Lemma's proof. Soon after, more compact forms were proposed: $\mathbf{\Phi}_{i,j}$ follows a uniformly distribution from the set $\{-1, 1\}$ [Achlioptas, 2003] or any sub-Gaussian distribution [Matoušek, 2008].

Definition 2.2.2 (sub-Gaussian variable)**.** *The probability distribution of a random variable X is called* sub-Gaussian *if there exist $C, a > 0$ such that for every $t > 0$,*

$$\mathbb{P}[|X| > t] \leq Ce^{-at^2}. \tag{2.18}$$

More simply, this means that the tails of a sub-Gaussian distribution are dominated by the tails of a Gaussian one or, said differently, that the tails of a sub-Gaussian distribution decay at least as fast as the tails of a Gaussian one.

So at this point, more efficient ways to compute the projection matrix $\mathbf{\Phi}$ under the dense distribution have been developed. But the storage of $\mathbf{\Phi}$ still requires a dense $c \times d$ matrix and the cost to map all the data onto the lower space is also in $O(c \times d)$. This is very expensive when dealing with high-dimensional datasets, so sparse forms have been also investigated.

Sparse distribution

Sparsification of the matrix $\mathbf{\Phi}$ is not obvious. Kane and Nelson [2010] showed indeed that the number of nonzero entries in $\mathbf{\Phi}$ should be in $O(d \log(N)/\epsilon)$ while Ailon and Chazelle [2006] argued that the matrix cannot get too sparse because in case of an already sparse input vector, the projection will distort it too much. In that event, the resulting projection would be poorly concentrated. The authors get around this problem by showing that a randomized Fast Fourier Transform (FFT) can be used as a first step. This way, a sparse input vector would be densified while an already dense vector would not be sparsified. This is the role of $\mathbf{H}$ and $\mathbf{D}$ in the following proposed matrix $\mathbf{\Phi}$ called the Fast Johnson-Lindenstrauss Transform (FJLT) [Ailon and Chazelle, 2006]:

$$\mathbf{\Phi} = \mathbf{PHD} \tag{2.19}$$

where for some constant q,

$$\mathbf{P}_{i,j} = \begin{cases} \sim \mathcal{N}(0, (\frac{1}{\sqrt{q}})^2) & \text{with probability } q \\ 0 & \text{with probability } 1 - q \end{cases} \tag{2.20}$$

and $\mathbf{H}$ stands for the normalized Hadamard matrix (cf. Definition 2.2.3 below) and $\mathbf{D}$ for a diagonal matrix with independent Rademacher (± 1) entries.

Definition 2.2.3 (normalized Hadamard matrix)**.** *The normalized Hadamard matrix is defined recursively for an even dimension $d = 2^i$:*

$$\mathbf{H}_0 = 1 \tag{2.21}$$

$$\mathbf{H}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.22}$$

$$\mathbf{H}_i = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{H}_{i-1} & \mathbf{H}_{i-1} \\ \mathbf{H}_{i-1} & -\mathbf{H}_{i-1} \end{pmatrix} \tag{2.23}$$

The role of the block $\mathbf{HD}$ is to densify a possibly sparse input vector. Another possibility for sparsifying $\mathbf{\Phi}$ is described by Matoušek [2008]: for some constants η

and $q \in O(\eta^2 c) \leq 1$:

$$\mathbf{P}_{i,j} = \begin{cases} \frac{1}{\sqrt{q}} & \text{with probability } \frac{q}{2} \\ 0 & \text{with probability } 1-q \\ \frac{-1}{\sqrt{q}} & \text{with probability } \frac{q}{2} \end{cases} \tag{2.24}$$

for $\mathbf{x}$ such that $\|\mathbf{x}\|_\infty / \|\mathbf{x}\|_2 \leq \eta$. The latter condition means that $\mathbf{x}$ should not be sparse, though.

This Section showed that sparsification of the projection matrix is enabled by combination of classical structured matrices. Previous examples include the Hadamard transform, diagonal and permutation matrices. Further structured matrices are detailed in Appendix B.

When the precision of the result in preserving the distance between the points is not enough with data-independent methods, data-dependent ones can be involved. Lots of work have been made in this field. This would be vain in this thesis to review them all. Instead, the choice has been made to give in Section 2.3 the most important work related to ours (from Chapters 4 and 5) and the algorithms we are building a new online method on (in Chapter 4).

2.3 Data-dependent approaches

2.3.1 Some data-dependent hashing techniques

For hashing functions introduced in Section 2.2.1, the *data-dependent* approaches, where the binary embeddings are learned from the training set, are known to preserve better the distance between points. There exists a plethora of them. This is out of the scope of this thesis to review them all. Among this branch of methods, *unsupervised* methods - which are of interest here - [Weiss et al., 2008, Liu et al., 2011, Gong et al., 2013, Kong and Li, 2012, Lee, 2012, Liu et al., 2014, Yu et al., 2014, Raziperchikolaei and Carreira-Perpiñán, 2016] design hash codes preserving distances in the original space while (semi-)supervised ones attempt to keep label similarity [Wang et al., 2012, Liu et al., 2012]. Wang et al. [2018] propose an extensive survey of these methods and make further distinctions to finally show that quantization-based techniques are superior in terms of search accuracy. We can cite: ITerative Quantization (ITQ) [Gong et al., 2013], Isotropic Hashing (IsoHash) [Kong and Li, 2012], Cartesian K-means [Norouzi and Fleet, 2013] and some deep-learning-based methods [Lai et al., 2015, Liong et al., 2015, Do et al., 2016].

Specifically, when the dataset is too large to fit into memory, distributed hashing can be used [Leng et al., 2015b] or online hashing techniques [Huang et al., 2013, Leng et al., 2015a, Cakir and Sclaroff, 2015, Cakir et al., 2017] can process the data in only one pass, as a continuous stream, and compute binary hash codes as a new data point is seen. This latter area has attracted lots of interest in the past few years. Online Hashing (OKH) [Huang et al., 2013] learns the hash functions from a stream of similarity-labeled pair of data with a "Passive-Aggressive" method [Crammer et al., 2006]. Supervised MIHash algorithm [Cakir et al., 2017] also uses similarity labels between pairs of data and considers Mutual Information to compute the binary embeddings. On the *unsupervised* side, in Online Sketching Hashing (OSH) [Leng et al., 2015a], the binary embeddings are learned from a maintained sketch of the dataset with a smaller size which preserves the property of interest.

From the family of quantization-based hashing methods, there is the category of *Hypercubic hashing* which at some point relies on the Principal Component Analysis (PCA). Before introducing prominent hashing functions from this type, let us first describe the PCA in Section 2.3.2.

2.3.2 Principal Component Analysis-based approaches (PCA)

PCA Principle

Let us represent an original dataset by a data matrix $\mathbf{X} \in \mathbb{R}^{d \times N}$ where N is the number of observations and d the dimensionality of data. One observation $\mathbf{x}_j = \mathbf{X}_{*j}$ is a column of $\mathbf{X}$ for $j \in [N]$. Data are then zero-centered, i.e. the following operation is performed, where $\overline{\mathbf{X}}_i = \frac{1}{N} \sum_{j=1}^{N} \mathbf{X}_{ij}$:

$$\forall (i,j) \in [d] \times [N],\ \mathbf{X}_{ij} \leftarrow \mathbf{X}_{ij} - \overline{\mathbf{X}}_i. \tag{2.25}$$

Principal Component Analysis [Pearson, 1901, Hotelling, 1933] *linearly* transforms $\mathbf{X}$ into another matrix $\mathbf{Y} \in \mathbb{R}^{d \times N}$ so that for some $d \times d$ *orthogonal* matrix $\mathbf{P}$,

$$\mathbf{Y} = \mathbf{P}\mathbf{X}. \tag{2.26}$$

This change of basis, without any dimensionality reduction, returns a set of *linearly* uncorrelated attributes named *principal components* from a dataset of possibly correlated ones and aims at reexpressing the data so that to optimally discriminate linearly separable data. Let us rewrite the equation as the following, where $\{\mathbf{P}_{i*}\}$ denotes the set of rows of $\mathbf{P}$, the *principal components*, and $\{\mathbf{x}_j\}$ denotes the set of columns of $\mathbf{X}$ i.e. input data:

$$\mathbf{P}\mathbf{X} = \begin{pmatrix} \mathbf{P}_{1*} \\ \mathbf{P}_{2*} \\ \vdots \\ \mathbf{P}_{d*} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N \end{pmatrix} = \begin{pmatrix} \mathbf{P}_{1*}.\mathbf{x}_1 & \mathbf{P}_{1*}.\mathbf{x}_2 & \dots & \mathbf{P}_{1*}.\mathbf{x}_N \\ \mathbf{P}_{2*}.\mathbf{x}_1 & \mathbf{P}_{2*}.\mathbf{x}_2 & \dots & \mathbf{P}_{2*}.\mathbf{x}_N \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{d*}.\mathbf{x}_1 & \mathbf{P}_{d*}.\mathbf{x}_2 & \dots & \mathbf{P}_{d*}.\mathbf{x}_N \end{pmatrix}. \tag{2.27}$$

This shows clearly that $\mathbf{X}$ is being *projected* onto the rows of $\mathbf{P}$ which are the new basis for describing the original data points of $\mathbf{X}$. In the PCA decomposition, this new basis is defined by the directions which enable to decorrelate the original data, i.e. the directions in which the variance is maximized.

Let us consider the covariance matrix of $\mathbf{X}$ defined as $\mathbf{\Sigma}_\mathbf{X} = \frac{1}{N-1}\mathbf{X}\mathbf{X}^T \in \mathbb{R}^{d \times d}$.

$$\mathbf{\Sigma}_\mathbf{X} = \frac{1}{N-1} \begin{pmatrix} \mathbf{X}_{1*}.\mathbf{X}_{1*}^T & \mathbf{X}_{1*}.\mathbf{X}_{2*}^T & \dots & \mathbf{X}_{1*}.\mathbf{X}_{d*}^T \\ \mathbf{X}_{2*}.\mathbf{X}_{1*}^T & \mathbf{X}_{2*}.\mathbf{X}_{2*}^T & \dots & \mathbf{X}_{2*}.\mathbf{X}_{d*}^T \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{X}_{d*}.\mathbf{X}_{1*}^T & \mathbf{X}_{d*}.\mathbf{X}_{2*}^T & \dots & \mathbf{X}_{d*}.\mathbf{X}_{d*}^T \end{pmatrix} \tag{2.28}$$

$\mathbf{\Sigma}_\mathbf{X}$ sums up all the possible covariance pairs between the d variables or attributes. In particular, the diagonal entries correspond to the actual variance of each dimension. Similarly, $\mathbf{\Sigma}_\mathbf{Y}$ can be defined for the transformed matrix $\mathbf{Y}$. According to the PCA objective, the covariances of different variables (non-diagonal entries) in the matrix $\mathbf{\Sigma}_\mathbf{Y}$ should be minimized i.e. being as close to zero as possible while the variances (diagonal coefficients) should be maximized. This is equivalent to say that the targeted

$\mathbf{\Sigma_Y}$ should be diagonal. The following holds:

$$\mathbf{\Sigma_Y} = \frac{1}{N-1}\mathbf{Y}\mathbf{Y}^T = \frac{1}{N-1}(\mathbf{P}\mathbf{X})(\mathbf{P}\mathbf{X})^T = \frac{1}{N-1}\mathbf{P}(\mathbf{X}\mathbf{X}^T)\mathbf{P}^T. \tag{2.29}$$

Therefore, the problem goal can also be expressed as maximizing the following objective function J, where Tr(.) stands for the Trace application:

$$J(\mathbf{P}) = \mathrm{Tr}(\mathbf{P}\mathbf{X}\mathbf{X}^T\mathbf{P}^T),\ \mathbf{P}\mathbf{P}^T = \mathbf{I}_d. \tag{2.30}$$

Moreover, $(\mathbf{X}\mathbf{X}^T)^T = (\mathbf{X}^T)^T\mathbf{X}^T = \mathbf{X}\mathbf{X}^T$. Hence, $\mathbf{X}\mathbf{X}^T$ is a $d \times d$ symmetric matrix and orthogonally (in particular orthonormally) diagonalizable. So there exists an orthonormal $d \times d$ matrix $\mathbf{U}$ whose columns are the orthonormal eigenvectors of $\mathbf{X}\mathbf{X}^T$ so that:

$$\mathbf{X}\mathbf{X}^T = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \tag{2.31}$$

where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d)$ is a diagonal matrix formed with the set of eigenvalues of $\mathbf{X}\mathbf{X}^T$ $\{\lambda_i\}$ for $i \in [d]$ such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$. If the rank r of $\mathbf{X}\mathbf{X}^T$, i.e. the number of nonzero eigenvalues or the number of orthonormal eigenvectors, is less than d, the $d - r$ corresponding eigenvectors in $\mathbf{U}$ are random orthonormal vectors. So $\mathbf{\Sigma_Y}$ can be rewritten:

$$\mathbf{\Sigma_Y} = \frac{1}{N-1}\mathbf{P}(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T)\mathbf{P}^T. \tag{2.32}$$

If one takes $\mathbf{P} = \mathbf{U}^T$, the equation becomes:

$$\mathbf{\Sigma_Y} = \frac{1}{N-1}(\mathbf{U}^T\mathbf{U})\mathbf{\Lambda}(\mathbf{U}^T\mathbf{U}) = \frac{1}{N-1}\mathbf{\Lambda} \tag{2.33}$$

since $\mathbf{U}$ is orthonormal, i.e. $\mathbf{U}^T\mathbf{U} = \mathbf{I}_d$. Finally, the principal components, i.e. the rows of $\mathbf{P}$ are chosen to be the d eigenvectors of $\mathbf{X}\mathbf{X}^T$ sorted in the decreasing order of corresponding eigenvalues such that the first principal component has the highest variance possible.

Note that PCA can be performed through EigenValue Decomposition (EVD) of the covariance matrix as previously shown but also with Singular Value Decomposition (SVD) of the data matrix $\mathbf{X}$.

To perform *dimensionality reduction* i.e. to obtain $\mathbf{V} \in \mathbb{R}^{c \times N}$ given a targeted lower dimension c with $c \ll d$, it suffices to keep the c first principal components to build the $c \times d$ matrix $\mathbf{W}$. $\mathbf{V}$ is then defined as follows:

$$\mathbf{V} = \mathbf{W}\mathbf{X} \in \mathbb{R}^{c \times N}. \tag{2.34}$$

Hypercubic quantization hashing

In *Hypercubic quantization hashing* methods [Jégou et al., 2010, Gong et al., 2013, Kong and Li, 2012, Leng et al., 2015a, Chen et al., 2017], after having obtained the PCA-projected dataset $\mathbf{V} = \mathbf{W}\mathbf{X} \in \mathbb{R}^{c \times N}$, $\mathbf{V}$ is rotated with the orthogonal matrix $\mathbf{R} \in \mathbb{R}^{c \times c}$ s.t. $\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I}_c$:

$$\mathbf{Y} = \mathbf{R}\mathbf{V} \in \mathbb{R}^{c \times N}. \tag{2.35}$$

In the sequel, we denote $\tilde{\mathbf{W}} = \mathbf{RW}$. Then, to have finally binary codes, the sign function is applied pointwise:

$$\mathbf{B} = \text{sign}(\mathbf{Y}) \in \{-1, 1\}^{c \times N}. \tag{2.36}$$

Recall that for any real x, $\text{sign}(x) = 1$ if $x \geq 0$ and -1 otherwise. We also use this notation for the same function applied component-wise on coefficients of vectors. Figure 2.5 summarizes the general scheme of Hypercubic hashing methods.

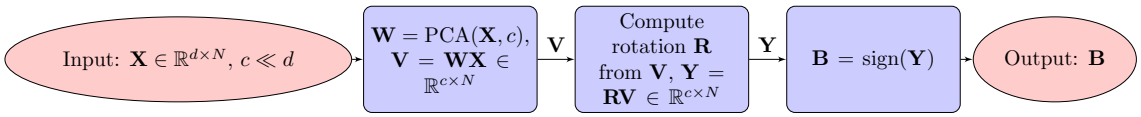


FIGURE 2.5: General (offline) scheme for Hypercubic hashing methods. The first step consists in computing the $c \times d$-projection matrix with PCA. The second rotates the PCA-projected data. Finally, the sign function is applied pointwise to obtain binary coefficients.

We describe now more in details two main figures of this family: ITerative Quantization (ITQ) [Gong et al., 2013] and Isotropic Hashing (IsoHash) [Kong and Li, 2012].

ITerative Quantization (ITQ) For ITQ [Gong et al., 2013], $\mathbf{R}$ is the solution of an orthogonal Procustes problem which consists in minimizing the quantization error $Q(\mathbf{B}, \mathbf{R})$ of mapping the resulting data to the vertices of the 2^c hypercube:

$$Q(\mathbf{B}, \mathbf{R}) = \|\mathbf{B} - \tilde{\mathbf{W}}\mathbf{X}\|_F^2 = \|\mathbf{B} - \mathbf{RWX}\|_F^2 = \|\mathbf{B} - \mathbf{RV}\|_F^2, \tag{2.37}$$

where $\|.\|_F$ denotes the Froebenius norm.

Algorithm 1 ITQ algorithm [Gong et al., 2013]

Inputs : data : $\mathbf{V} = \mathbf{WX}$, $\mathbf{V} \in \mathbb{R}^{c \times N}$ PCA-projected $\mathbf{X}$; code length: c; number of iterations: K
$\mathbf{R} \leftarrow randomOrthogonalMatrix(c, c)$ // *Initialization*
for i in $[K]$ **do**
  $\mathbf{B} = \text{sign}(\mathbf{RV})$ // *Fix* $\mathbf{R}$ *and update* $\mathbf{B}$
  $\mathbf{S}, \mathbf{\Omega}, \tilde{\mathbf{S}}^T = \text{SVD}(\mathbf{V}^T\mathbf{B})$ // *Fix* $\mathbf{B}$ *and update* $\mathbf{R}$
  $\mathbf{R} = \tilde{\mathbf{S}}\mathbf{S}^T$
return $\mathbf{R}$

Algorithm 1 describes ITQ. $\mathbf{R}$ is initially some random orthogonal matrix: $randomOrthogonalMatric(c, c)$ in Algorithm 1 constructs a $c \times c$ random orthogonal matrix. Then, iteratively, ITQ alternatively computes $\mathbf{B} = \text{sign}(\mathbf{RWX})$ after freezing $\mathbf{R}$, and optimizes $\mathbf{R}$ according to $\mathbf{B}$. In the latter step, $\mathbf{R} = \tilde{\mathbf{S}}\mathbf{S}^T$ by defining $\mathbf{S}, \mathbf{\Omega}, \tilde{\mathbf{S}}^T = \text{SVD}(\mathbf{VB}^T)$ the Singular Values Decomposition (SVD) of $\mathbf{VB}^T$. Hence, ITQ's goal is to map the values of the projected data to their component-wise sign. This is currently the best method among Hypercubic quantization techniques. Its drawback is that it is a full offline process and there is no convergence guarantees for obtaining $\mathbf{R}$.

Isotropic Hashing (IsoHash) Let us begin with some notation. For any real a, $\text{diag}_c(a)$ returns a $c \times c$-diagonal matrix whose diagonal coefficients are all equal to a, while for any matrix $\mathbf{M}$, $\text{diag}(\mathbf{M})$ returns a diagonal matrix with the same diagonal as $\mathbf{M}$. For any matrix $\mathbf{M}$, we denote $\mathbf{\Sigma_M} = \mathbf{M}\mathbf{M}^T$[2]. $\mathcal{O}(c)$ is the set of all orthogonal matrices in $\mathbb{R}^{c\times c}$, i.e. $\mathcal{O}(c) = \{\mathbf{Q} \mid \mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}_c\}$.

Let $\sigma_1^2, ..., \sigma_c^2$ be the diagonal coefficients of $\mathbf{\Sigma_V}$, hence $\sigma_1^2 \geq ... \geq \sigma_c^2$. IsoHash [Kong and Li, 2012] (but also UnifDiag, see Section 4.4) looks for a matrix $\mathbf{R}$ balancing the variance over the c directions, i.e. equalizing the diagonal coefficients of $\mathbf{\Sigma_Y}$ to the same value:

$$\tau = \text{Tr}(\mathbf{\Sigma_V})/c. \tag{2.38}$$

Let us define, for any real a, $\mathcal{T}(a)$, the set of all $c\times c$ matrices with diagonal coefficients equal to a:

$$\mathcal{T}(a) = \{\mathbf{T} \in \mathbb{R}^{c\times c} \mid \text{diag}(\mathbf{T}) = \text{diag}_c(a)\} \tag{2.39}$$

and

$$\mathcal{M}(\mathbf{\Sigma_V}) = \{\mathbf{Q}\mathbf{\Sigma_V}\mathbf{Q}^T \mid \mathbf{Q} \in \mathcal{O}(c)\}. \tag{2.40}$$

Then, IsoHash determines $\mathbf{R}$ by solving this optimization problem:

$$\mathbf{R} \in \underset{\mathbf{Q}:\ \mathbf{T}\in\mathcal{T}(\tau),\ \mathbf{Z}\in\mathcal{M}(\mathbf{\Sigma_V})}{\text{argmin}} \|\mathbf{T} - \mathbf{Z}\|_F. \tag{2.41}$$

One of the proposed methods by IsoHash to solve this problem (*Gradient Flow*) is to reformulate it as:

$$\mathbf{R} \in \underset{\mathbf{Q}\in\mathcal{O}(c)}{\text{argmin}} \frac{1}{2}\| \text{diag}(\mathbf{Q}\mathbf{\Sigma_V}\mathbf{Q}^T) - \text{diag}(\tau)\|_F^2. \tag{2.42}$$

Then, $\mathbf{R}$ results from a gradient descent converging to the intersection between the set of orthogonal matrices and the set of transfer matrices making $\mathbf{\Sigma_Y}$ diagonal.

The burden of PCA for ITQ and IsoHash PCA described as above requires the storage of the entire dataset which costs $O(Nd)$. Besides, in practice, the time complexity of the *eigenvalue decomposition* of $\mathbf{X}\mathbf{X}^T$is $O(d^3)$ while the complexity of the SVD of $\mathbf{X}$ is $O(\min(dN^2, Nd^2))$ [Jolliffe, 1986]. Therefore, this becomes infeasible for high-dimensional datasets. Efficient approximation of PCA, especially in an online fashion, has led to numerous works. It is out the scope of this thesis to review them all. Nevertheless, in the following Sections two approaches will be described:

1. one based on the sketching of the dataset,
2. another one based on the tracking in an online fashion of the principal subspace.

Streaming matrix sketching for PCA and Online Sketching Hashing (OSH)

First, an algorithm named *Frequent-Directions* [Liberty, 2013] is presented for sketching a matrix.

Streaming matrix sketching with *Frequent-Directions* Let consider $\mathbf{A} \in \mathbb{R}^{n\times m}$ the large matrix to sketch where n is the dimension to reduce. In the streaming model, the sketching algorithm takes as input the sequence of n rows of $\mathbf{A}$ and

[2]This is a slight abuse of notation as the normalization factor is discarded.

returns a sketch matrix $\mathbf{B} \in \mathbb{R}^{l \times m}$ such that $l \ll n$, l is even and $l \ll d$. This is done using only $O(ml)$ space, which is actually the required space to store $\mathbf{B}$. The goal is to guarantee that $\mathbf{A}^T\mathbf{A} \approx \mathbf{B}^T\mathbf{B}$ and more precisely:

$$\forall \mathbf{x} \in \mathbb{R}^m,\ \|\mathbf{x}\|_2 = 1,\ 0 \leq \|\mathbf{A}\mathbf{x}\|_2^2 - \|\mathbf{B}\mathbf{x}\|_2^2 \leq 2\|\mathbf{A}\|_F^2/l \tag{2.43}$$

or

$$\mathbf{B}^T\mathbf{B} \prec \mathbf{A}^T\mathbf{A} \text{ and } \|\mathbf{A}^T\mathbf{A} - \mathbf{B}^T\mathbf{B}\|_2 \leq 2\|\mathbf{A}\|_F^2/l \tag{2.44}$$

where any matrix positive semidefinite (resp. positive definite) $\mathbf{M}$ is denoted as $\mathbf{M} \succeq 0$ (resp. $\mathbf{M} \succ 0$). So $\mathbf{A}^T\mathbf{A} - \mathbf{B}^T\mathbf{B}$ is positive definite. Each sketch update which corresponds to one processed row of $\mathbf{A}$ costs $O(ml)$ operations.

The principle of the *Frequent-Directions* algorithm is the following: $\mathbf{B}$ is initialized to $\mathbf{0}_{l \times m}$ which is the $l \times m$ matrix with all coefficients set to zero. While there are zero-valued rows in $\mathbf{B}$, they are filled with rows from $\mathbf{A}$. If there are still rows from $\mathbf{A}$ to process but no zero-valued rows, half the rows of $\mathbf{B}$ are nullified by a two-stage process:

1. $\mathbf{B}$ is rotated from the left using its SVD such that its rows are orthogonal and in descending magnitude of order. In this setting, $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, $\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}_l$, $\mathbf{\Sigma} = \text{diag}((\sigma_1, \ldots, \sigma_l))$, $\sigma_1 \geq \ldots \geq \sigma_l \geq 0$.

2. The squared median singular value ($\sigma_{l/2}^2$, with l supposed to be an even integer) is used to nullify all singular values below this threshold: it leads to a new $\mathbf{B}$ with at least half rows that are zero vectors.

The method is described in Algorithm 2. Let us analyze the time complexity of the

Algorithm 2 *Frequent-Directions* algorithm [Liberty, 2013] - row version

Inputs : $\mathbf{A} \in \mathbb{R}^{n \times m}$, l
$\mathbf{B} \leftarrow \mathbf{0}_{l \times m}$ *// Initialization*
for j in $1, \ldots, n$ **do**
  Insert $\mathbf{A}_{j*}$ into a zero-valued row of $\mathbf{B}$
  if $\mathbf{B}$ has no zero-valued rows **then**
    $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] \leftarrow \text{SVD}(\mathbf{B})$
    $\delta \leftarrow \sigma_{l/2}^2$
    $\hat{\mathbf{\Sigma}} \leftarrow \sqrt{\max\left(\mathbf{\Sigma} - \mathbf{I}_l\delta, 0\right)}$
    $\mathbf{B} \leftarrow \hat{\mathbf{\Sigma}}\mathbf{V}^T$ *// At least half the rows of* $\mathbf{B}$ *are all zero.*
return $\mathbf{B}$

algorithm. The worst case for one update is when SVD of $\mathbf{B}$ is performed and it costs $O(ml^2)$. It is required every $l/2$ processed rows of $\mathbf{A}$ since at each SVD, at least $l/2$ rows of $\mathbf{B}$ are nullified. Otherwise, the update time is $O(m)$ when no SVD is performed since the cost is only to add a vector with size m in $\mathbf{B}$. So the total running time for n rows of $\mathbf{A}$ is

$$O\left(\underbrace{n/(l/2) \times ml^2}_{\text{with SVD}} + \underbrace{(n - \frac{n}{l/2})m}_{\text{without SVD}}\right) = O(nml) \tag{2.45}$$

because the cost is dominated by the SVD computation.

Finally, remark that this sketch has the interesting (and desired) property to enable the partitioning of matrix $\mathbf{A}$ into chunks of rows less than n for which local sketches can be computed and then combined in an arbitrary order. This is particularly relevant for distributed computations.

Online Sketching Hashing (OSH) The *Frequent-Directions* technique is used for instance in Online Sketching Hashing (OSH) algorithm from Leng et al. [2015a] to sketch online the dataset $\mathbf{X} \in \mathbb{R}^{d \times N}$ in order to perform an efficient version of the PCA. The dataset $\mathbf{X} \in \mathbb{R}^{d \times N}$ is split into different chunks $\mathbf{X}_1, \mathbf{X}_2, \ldots \in \mathbb{R}^{d \times n}$, $n < N$ and the *Frequent-Directions* algorithm is applied to obtain a compact updated representation of the dataset $\mathbf{X'} \in \mathbb{R}^{d \times l}$ as new chunks are seen, with $l \ll n$ and $l \ll d$. Then, classical PCA is performed on $\mathbf{X'} \in \mathbb{R}^{d \times l}$ as an approximate PCA of $\mathbf{X} \in \mathbb{R}^{d \times N}$. In this case, PCA has been obtained efficiently with RSVD [Golub and van der Vorst, 2000] for the specific case $d \gg l$, with complexity $O(dl^2 + l^3)$, which is much more efficient than the original $O(d^3)$. Algorithm 3 presents the used *column* version of *Frequent-Directions* algorithm.

Algorithm 3 *Frequent-Directions* algorithm [Liberty, 2013] - column version

Inputs : $\mathbf{A} \in \mathbb{R}^{n \times m}$, l
$\mathbf{B} \leftarrow \mathbf{0}_{n \times l}$ // *Initialization*
for j in $1, \ldots, m$ **do**
  Insert $\mathbf{A}_{*j}$ into a zero-valued column of $\mathbf{B}$
  if $\mathbf{B}$ has no zero-valued columns **then**
    $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] \leftarrow \text{SVD}(\mathbf{B})$
    $\delta \leftarrow \sigma_{l/2}^2$
    $\hat{\mathbf{\Sigma}} \leftarrow \sqrt{\max\left(\mathbf{\Sigma} - \mathbf{I}_l \delta, 0\right)}$
    $\mathbf{B} \leftarrow \mathbf{U}\hat{\mathbf{\Sigma}}$ // *At least half the columns of* $\mathbf{B}$ *are all zero.*
return $\mathbf{B}$

However one can argue that PCA is not exactly made online, rather in *mini-batches*. Recent works from Boutsidis et al. [2015], Karnin and Liberty [2015] present online algorithms for PCA projection of $\mathbf{X} \in \mathbb{R}^{d \times N}$ (by handling the columns of the matrix as a stream) with respectively bounds on the norm of the residual matrices $\|\mathbf{X} - \mathbf{W}^T\mathbf{Y}\|_F^2$ and $\|\mathbf{X} - (\mathbf{X}\mathbf{Y}^\dagger)\mathbf{Y}\|_F^2$ where $\mathbf{M}^\dagger$ stands for the Moore Penrose inverse or pseudo-inverse of $\mathbf{M}$, $\mathbf{Y} \in \mathbb{R}^{c \times N}$ is the projected dataset of $\mathbf{X} \in \mathbb{R}^{d \times N}$ onto the approximate c first principal components and $\mathbf{W} \in \mathbb{R}^{c \times d}$ is the PCA projection matrix.

Nevertheless, in practice, we found easier to use OPAST algorithm [Abed-Meraim et al., 2000] which does not require repeated Singular Value Decompositions (SVD), a task that is very time consuming.

Fast Orthonormal Projection Approximation and Subspace Tracking (OPAST)

OPAST algorithm [Abed-Meraim et al., 2000] is explicited in Algorithm 4. We let the reader refer to the paper's proof in order to justify each step of the algorithm. In brief, the projection matrix $\mathbf{W}$ is initialized to a random orthonormal $c \times d$ matrix. The $c \times c$-matrix $\mathbf{Z}$ maintains the information relatively to the covariance matrix. The forgetting factor $\alpha \in (0, 1]$ diminishes the weight of the old seen points in order to deal with the *non-stationary* data points. Then, each input vector of the stream is projected on the current updated $\mathbf{W}$ before updating $\mathbf{W}$ with suitable calculus.

Algorithm 4 OPAST algorithm [Abed-Meraim et al., 2000]

Inputs : stream : $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathbb{R}^d$; dimension of the principal subspace c; forgetting factor $\alpha \in (0, 1]$
// Initialization:
$\mathbf{W} \leftarrow randomOrthonormalMatrix(c, d)$
$\mathbf{Z} \leftarrow \mathbf{I}_c$
for $\mathbf{x}$ in stream **do**
  $\mathbf{y} := \mathbf{W}\mathbf{x}$
  $\mathbf{q} := \frac{1}{\alpha}\mathbf{Z}\mathbf{y}$
  $\gamma := \frac{1}{1+\mathbf{y}^T\mathbf{q}}$
  $\mathbf{p} := \gamma(\mathbf{x} - \mathbf{W}^T\mathbf{y})$
  $\mathbf{Z} \leftarrow \frac{1}{\alpha}\mathbf{Z} - \gamma\mathbf{q}\mathbf{q}^T$
  $\tau := \frac{1}{\|\mathbf{q}\|^2}\left(\frac{1}{\sqrt{1+\|\mathbf{p}\|^2\|\mathbf{q}\|^2}} - 1\right)$
  $\mathbf{p}' := \tau\mathbf{W}^T\mathbf{q} + (1 + \tau\|\mathbf{q}\|^2)\mathbf{p}$
  $\mathbf{W} \leftarrow \mathbf{W} + \mathbf{q}\mathbf{p}'^T$
return $\mathbf{W}$

2.3.3 Similarity / Metric learning

Locality Sensitive Hashing (LSH) [Indyk and Motwani, 1998] described in Section 2.2.2 maps with high probability *similar* items to the same buckets in the hash table representing the memory. This approach can be seen as one *data-independent* paradigm for *similarity learning*.

The goal of similarity learning is to learn from data a *similarity function* that measures how close two objects are. Other approaches of this field will be data-dependent and *supervised*. That is why we won't review extensively this field as we rather focus on unsupervised methods. Similarity learning is strongly related to *metric learning* (see survey from Bellet et al. [2013]) where a *distance metric* is learned and should satisfies the four conditions enumerated in Definition A.0.1 p. 127 for a distance.

Besides LSH, other principal approaches in similarity learning are task-dependent:

- Regression: The dataset is a set of tuples (x_i, y_i, s_i) where x_i, y_i are a couple of points from a dataset $\mathcal{X}$ labeled by a similarity $s_i \in \mathbb{R}$. The objective is to learn a function f such that $f(x_i, y_i) \approx s_i$ for each tuple of the dataset. This can help to predict the similarity given an unseen couple with unknown similarity.

- Classification: The dataset is the same as for the regression but $s_i \in \{-1, 1\}$ such that if two points are similar, $s_i = 1$ or $s_i = -1$ otherwise. One can define the sets:
$$\mathcal{S} := \{\ (x_i, y_i) \mid s_i = 1\} \tag{2.46}$$
and
$$\mathcal{D} := \{\ (x_i, y_i) \mid s_i = -1\}. \tag{2.47}$$
The goal is to learn a classifier that settles if a new pair of points is similar or not. Some selected works are [Huang et al., 2013, Cakir and Sclaroff, 2015, Cakir et al., 2017].

- Ranking: The dataset is a set of triplets of points (x_i, x_i^+, x_i^-) which means that x_i is more similar to x_i^+ than to x_i^-. One can define the set:

$$\mathcal{R} := \{ (x_i, x_i^+, x_i^-) \mid x_i \text{ is more similar to } x_i^+ \text{ than } x_i^- \}. \tag{2.48}$$

  The purpose is to learn a function f such that for any new triplet of points (x, x^+, x^-), it holds:

$$f(x, x^+) > f(x, x^-), \tag{2.49}$$

  corresponding to a ranking function. Please note that this supervision is weaker than in regression. Indeed, the exact measure of similarity is not given, only the relative order of similarity instead. Hence, ranking-based similarity learning is more scalable. Some selected work is [Lai et al., 2015].

Metric learning has been proposed as a preprocessing step for many approaches in clustering or nearest neighbors algorithms [Xing et al., 2002]. Metric learning aims at learning the parameters of some pairwise real-valued metric function according to the application. The most prominent one is the *Mahalanobis distance*.

Definition 2.3.1 (Mahalanobis distance)**.** *For* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$*, the* Mahalanobis distance *between* $\mathbf{x}$ *and* $\mathbf{y}$ *is defined as:*

$$dist_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{y})} \tag{2.50}$$

where $\mathbf{\Sigma}$ *is the covariance matrix corresponding to the distribution of* $\mathbf{x}$ *and* $\mathbf{y}$*.*

Due to a slight common writing abuse, the Mahalanobis distance refers rather to the following definition:

Definition 2.3.2 ((general) Mahalanobis distance)**.** *For* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$*, the* (general) Mahalanobis distance *between* $\mathbf{x}$ *and* $\mathbf{y}$ *is defined as:*

$$dist_{\mathbf{M}}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{M} (\mathbf{x} - \mathbf{y})} \tag{2.51}$$

where $\mathbf{M} \in \mathbb{S}_+^d$ *and* $\mathbb{S}_+^d$ *is the cone of symmetric Positive Semi-Definite (PSD)* $d \times d$ *real-values matrices. The role of* $\mathbf{M}$ *is to guarantee that* $dist_{\mathbf{M}}$ *obeys to the properties of a pseudo-distance: nonnegativity, symmetry and triangle inequality are preserved but there is only* $dist_{\mathbf{M}}(\mathbf{x}, \mathbf{x}) = 0$ *instead of the identity of indiscernibles' property.*

The Mahalanobis distance is often considered for the following interesting property. Please note that if $\mathbf{M} = \mathbf{I}$, the Mahalanobis distance is equivalent to the Euclidean distance. For any other $\mathbf{M}$, by the fact that $\mathbf{M}$ is PSD, $\mathbf{M}$ can be factorized by $\mathbf{L} \in \mathbf{R}^{k \times d}$ such that $\mathbf{M} = \mathbf{L}^T \mathbf{L}$ where k is the rank of $\mathbf{M}$. Then for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,

$$\begin{aligned} dist_{\mathbf{M}}(\mathbf{x}, \mathbf{y}) &= \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{M} (\mathbf{x} - \mathbf{y})} && (2.52) \\ &= \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{L}^T \mathbf{L} (\mathbf{x} - \mathbf{y})} && (2.53) \\ &= \sqrt{(\mathbf{L}\mathbf{x} - \mathbf{L}\mathbf{y})^T (\mathbf{L}\mathbf{x} - \mathbf{L}\mathbf{y})}. && (2.54) \\ & && (2.55) \end{aligned}$$

This can be interpreted as the Euclidean distance *after the linear projection* of the data by the transformation $\mathbf{L}$. If $\mathbf{M}$ is taken low-rank with $rank(\mathbf{M}) = r < d$, the linear projection is a dimensionality reduction technique.

Finding the parameters of the metric, i.e here the coefficients of $\mathbf{M}$ depends on the sets of $\mathcal{S}$, $\mathcal{D}$ and the sets of constraints $\mathcal{R}$. This can be typically solved by optimizing the following general form of objective function:

$$\min_{\mathbf{M}} \ell(\mathbf{M}, \mathcal{S}, \mathcal{D}, \mathcal{R}) + \lambda R(\mathbf{M}) \tag{2.56}$$

where $\ell(\mathbf{M}, \mathcal{S}, \mathcal{D}, \mathcal{R})$ is a loss function which penalizes the violation of the constraints and $R(\mathbf{M})$ is a regularizer and $\lambda \geq 0$ the regularization parameter. The different state-of-the art approaches will differ in the choice of the metric, constraints, loss function and regularizer. Even if the Mahalanobis distance has a less expressive power due to its linear definition,

- It is easier to optimize because it generally leads to convex objective functions and hence to global optimal solutions.
- It is less prone to overfitting.

Please refer to the survey from [Bellet et al., 2013] for an extensive study.

In the next part, we review briefly space-efficient clustering approaches which are *unsupervised data-dependent* methods preserving and highlighting the structure of a dataset.

2.3.4 Space-efficient clustering approaches

In Chapter 5, we propose a new parameter-free clustering algorithm recovering arbitrary-shaped clusters working in $O(N)$ for the time and space costs. Hence, we expose here some state-of-the-art space-efficient and/or online approaches for an overview of what has been done.

First, lots of work have been pursued to propose a streaming version of k-means algorithm [Lloyd, 1982]. Let us recall the corresponding problem definition.

The k-means problem

Definition 2.3.3 (k-means clustering problem)**.** *Given a set $\mathbf{X} \in \mathbb{R}^{d \times N}$ and an associated weight function $w : \mathbf{X} \to \mathbb{Z}^+$, find a clustering partition $\Pi = \{C_1, \ldots, C_k\}$, with for all $i \in [k]$, $C_i \subset \mathbb{R}^d$ that minimizes the objective function*

$$\phi_\Pi(\mathbf{X}) = \sum_{\mathbf{x} \in \mathbf{X}} \min_{i \in [k]} (w(\mathbf{x}) \cdot \|\mathbf{x} - \mathbf{m}_i\|_2^2) \tag{2.57}$$

where $\mathbf{m}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$ encodes the centroid of the cluster C_i. ϕ_Π is called the potential function *corresponding to the clustering partition.*

A point can have a positive integral weight associated with it. By default, this weight is assumed to be equal to 1. The initial solving algorithm from Lloyd [1982] is:

1. Choose k random points from $\mathbf{X}$ (columns) to form the centers $\mathbf{m}_1, \ldots, \mathbf{m}_k$ of clusters $C_1, \ldots, C_k$.
2. Assign each point $\mathbf{x}$ from $\mathbf{X}$ to the nearest center $\mathbf{m}_i$.
3. Recompute the newly formed cluster centers $\mathbf{m}_i$ for $i \in [k]$.

4. Repetition of steps 2 and 3 until the set of centers is stable.

The space complexity of k-means is $O(N)$ while the time complexity is $O(Nkdt)$ where t is the number of iterations to reach the stability of the centers. As k-means is very sensitive to initialization, k-means++ [Arthur and Vassilvitskii, 2007] provides a smart way to choose the initial k centers. Previous step 1 is replaced by:

1. Choose an initial center $\mathbf{m}_1$ from $\mathbf{X}$.
2. For $i \in [k-1]$,
   - Choose the next center $\mathbf{m}_i$ as $\mathbf{x}' \in \mathbf{X}$, $\mathbf{x}' \neq \mathbf{m}_j$ for all $j \in [i-1]$, with probability $\frac{\min_{j\in[i-1]} \|\mathbf{x}'-\mathbf{m}_j\|_2^2}{\sum_{\mathbf{x}\in\mathbf{X}} \min_{j\in[i-1]} \|\mathbf{x}-\mathbf{m}_j\|_2^2}$.

Streaming k-means When all the data points can not be accessed at the same time and/or do not fit into memory, STREAM algorithm has been designed [Guha et al., 2003] to that purpose. It is based on a *divide-and-conquer* strategy. Before showing the principle, we need the definition of an (a, b)-approximation for the k-means problem.

Definition 2.3.4 ((a, b)-approximation)**.** *An algorithm $\mathcal{A}$ is called* (a, b)-approximation *for the k-means problem if it returns a clustering partition Π with ak centers and potential ϕ_Π such that $\frac{\phi_\Pi}{\phi_{OPT}} \leq b$ in the worst case, for $a > 1$, $b > 1$ and ϕ_{OPT} denoting the potential of the optimal clustering partition Π_{OPT}.*

The principle of STREAM algorithm [Guha et al., 2003] is then:

1. Divide the dataset $\mathbf{X}$ into n chunks $\mathbf{X}_1, \ldots, \mathbf{X}_n$.
2. For each $i \in [n]$,
   - Run an (a, b)-approximation algorithm to the k-means objective $\mathcal{A}$ on $\mathbf{X}_i$ to get at most ak centers $\mathbf{m}_{i_1}, \mathbf{m}_{i_2}, \ldots$ corresponding to clusters $\Pi_i = \{C_{i_1}, C_{i_2}, \ldots\}$.
3. Define the set $\mathbf{X}' = \cup_i \Pi_i$ by associating to each point $\mathbf{m}_{i_j}$ the weight $|C_{i_j}|$.
4. Run an (a', b')-approximation algorithm to the k-means objective $\mathcal{A}'$ on $\mathbf{X}'$ to return at most $a'k$ centers $\mathbf{m}'_1, \mathbf{m}'_2, \ldots$.
5. Return associated clustering partition Π'.

Ailon et al. [2009] proposed, accompanied by theoretical approximation guarantees, to take:

- $\mathcal{A}$ as the run of $3\log(N)$ independent k-means# (see definition below) on data and keeping the clustering partition with the smaller cost.
- $\mathcal{A}'$ as k-means++.

k-means# [Ailon et al., 2009] is an alternative to k-means++:

1. Choose $3\log(k)$ centers independently at random from $\mathbf{X}$.
2. For $i \in [k-1]$,

- Choose $3\log(k)$ centers independently and with probability $\frac{\min_j \|\mathbf{x}'-\mathbf{m}_j\|_2^2}{\sum_{\mathbf{x}\in\mathbf{X}} \min_j \|\mathbf{x}-\mathbf{m}_j\|_2^2}$.

Hence, in comparison with k-means++, the algorithm k-means#, at each round for choosing centers, stores $O(\log k)$ centers instead of just one. The running time of k-means# is $O(Ndk \log k)$. Besides, Ailon et al. [2009] state that the final proposed streaming algorithm (combining k-means#, k-means in STREAM from Guha et al. [2003]) costs N^α memory for some fixed $\alpha > 0$ and $O(Ndk^2 \log N \log k)$ time.

Streaming k-means [Ailon et al., 2009] is interesting since it delivers a one-pass streaming method for the k-means problem but it has essentially the same drawbacks as k-means: it still fails to detect clusters with non-convex shapes since only the centroid point of each cluster is stored.

This is not the case of CURE algorithm [Guha et al., 2001] presented in the next Section.

CURE algorithm

CURE [Guha et al., 2001] (CURE for Clustering Using REpresentatives) is a hierarchical agglomerative clustering algorithm taking as parameters the number of desired clusters k and a number n of scattered points chosen to represent each cluster. Initially, each point is a single cluster. In memory, each cluster is stored as a data structure containing:

- a mean point as the average of other ones contained in the same cluster,
- some sample of points representative of the cluster indexed by a k-d tree [Bentley, 1975],
- a heap with the nearest representative points of each other cluster.

Then, this information is used to merge the closest pair of clusters until they are only k. The sample of representative points for each cluster enables to recover arbitrary-shaped ones. After the merging of two clusters, the sample of representative points is reduced as the following: the first chosen point is the farthest one from the mean point and the $n-1$ other ones are the farthest from the previously selected scattered points.

To reduce the time and space costs, the authors propose also to perform the clustering on a random sample of the dataset [Vitter, 1985]. Then, remaining points which have not been used for the clustering are assigned to the cluster containing the representative point closest to them.

After some implementation tricks, the space cost of the algorithm is $O(N)$ while the time complexity of this offline method is $O(N^2 \log(N))$ which is not suitable for large datasets.

Therefore, let us see CluStream [Aggarwal et al., 2003] and DenStream [Cao et al., 2006] algorithms which are more time-efficient. The following presented algorithms have the advantage to take into account *evolution* in the data. This means that it can take few points to see before identifying one given point to be an outlier and vice versa. Hence, merging but also splitting clusters are enabled. Both algorithms create *micro-clusters* based on local densities in an online fashion and aggregate them later to build bigger clusters in offline steps.

CluStream

In Clustream [Aggarwal et al., 2003], the data structure to represent a micro-cluster is a temporal extension of the *cluster feature vector* from Zhang et al. [1996]. Let us consider a stream of data points $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ with associated time stamps $t_1, \ldots, t_N$. The corresponding statistics summary contains the tuple with size $(2d+3)$ $(\mathbf{CF2}^x, \mathbf{CF1}^x, CF2^t, CF1^t, n)$ where:

- $\mathbf{CF2}^x \in \mathbb{R}^d$ is the squared sum of the points in the cluster
- $\mathbf{CF1}^x \in \mathbb{R}^d$ is the linear sum of the points in the cluster,
- $CF2^t \in \mathbb{Z}^+$ is the sum of the squares of the time stamps for the points in the cluster,
- $CF1^t \in \mathbb{Z}^+$ is the sum of the time stamps for the points in the cluster,
- n is the number of contained points in the cluster.

The additivity property of the stored statistics makes the merging and splitting of micro-clusters easy.

During the process, CluStream maintains k micro-clusters $C_1, \ldots, C_k$. k is determined by the amount of main memory allocated to store the micro-clusters. This means that k is larger than the number of underlying clusters but still significantly smaller than N. The initial clusters are built by performing standard k-means on the first observed points of the stream. Then, when a new point is seen, two possibilities arise:

- The new point is assigned to a current micro-cluster which should be updated accordingly,
- or it builds a micro-cluster on it own.

The point is added to the cluster with the nearest centroid if the distance to this nearest centroid is below a defined maximum boundary. Otherwise, this point creates a new cluster. In this case, to maintain the same number of clusters k, one should be removed if identified as outlier or two nearest ones should be merged. To determine if a cluster is an outlier, the decision is made on the *relevance stamp* based on some approximation of the average stamps of the last points entered in the cluster. When the relevance stamp of any micro-cluster is below a certain threshold, the cluster can be deleted.

Some drawbacks of CluStream are:

1. CluStream works with a fixed number of micro-clusters. This would make it sensitive to noise because it could cause lots of clusters to merge for maintaining the same number of clusters.
2. CluStream produces mainly spherical clusters.

These are addressed by DenStream [Cao et al., 2006].

DenStream

With DenStream [Cao et al., 2006],

- There is no assumption on the number of clusters.

- Clusters recovered can be non-convex.
- DenStream can deal with outliers.

In their model, to handle the evolving data stream, the weight of each point is decreased exponentially with time T using the fading function $f(T) = 2^{-\lambda T}$ where $\lambda > 0$. The algorithm maintains in an online fashion three types of structure, the group of *core-micro-clusters*, *potential-micro-clusters* and *outlier-micro-clusters*. We define them now.

Definition 2.3.5 (core-micro-cluster [Cao et al., 2006])**.** *Given two parameters $\mu > 0$ and $\epsilon > 0$, a* core-micro-cluster *at time T is defined as a tuple of a weight $w \geq \mu$, a center $\mathbf{m} \in \mathbb{R}^d$ and a radius $r \leq \epsilon$ denoted as $cmc(w, \mathbf{m}, r)$ for a group of close points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ with time stamps $t_1, \ldots, t_n$.*

- $w := \sum_{i=1}^{n} f(T - t_i)$,
- $\mathbf{m} := \frac{\sum_{i=1}^{n} f(T-t_i)\mathbf{x}_i}{w}$,
- $r := \frac{\sum_{i=1}^{n} f(T-t_i)\|\mathbf{x}_i - \mathbf{m}\|_2}{w}$.

The constraints on the weight and the radius express that the core-micro-cluster should be dense. Let us denote N_c the number of core-micro-clusters. The constraint on the radius makes N_c larger than the number of true clusters but less than N. Moreover, each point is assigned to only one core-micro-cluster. Therefore, by denoting $T_c (T_c \to \infty)$ the current time, $N_c \leq \frac{W}{\mu}$ where $W := v \sum_{T=0}^{T_c} 2^{-\lambda T} = \frac{v}{1-2^{-\lambda}}$ is the overall weight of the data stream as a function of the speed v i.e. the number of points arriving in one unit time. Non-convex clusters will be described by several core-micro-clusters.

Definition 2.3.6 (potential core-micro-cluster [Cao et al., 2006])**.** *Given three parameters $\mu > 0$, $\epsilon > 0$, $0 < \beta \leq 1$, a* potential core-micro-cluster *at time T for a group of close points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ with time stamps $t_1, \ldots, t_n$, is defined as a tuple of $\mathbf{CF1} \in \mathbb{R}^d$ the weighted linear sum of the points, $\mathbf{CF2} \in \mathbb{R}^d$ the weighted squared sum of the points, a weight $w \geq \beta\mu$, a center $\mathbf{m} \in \mathbb{R}^d$ and a radius $r \leq \epsilon$.*

- $\mathbf{CF1} := \sum_{i=1}^{n} f(T - t_i)\mathbf{x}_i$
- $\mathbf{CF2} := \sum_{i=1}^{n} f(T - t_i)\mathbf{x}_i^2$
- $w := \sum_{i=1}^{n} f(T - t_i)$,
- $\mathbf{m} := \frac{\mathbf{CF1}}{w}$,
- $r := \sqrt{\frac{|\mathbf{CF2}|}{w} - (\frac{|\mathbf{CF1}|}{w})^2}$.

β parametrizes the threshold distinguishing potential core-micro-clusters of outlier micro-clusters.

Definition 2.3.7 (outlier-micro-cluster [Cao et al., 2006])**.** *Given three parameters $\mu > 0$, $\epsilon > 0$, $0 < \beta \leq 1$, a* potential core-micro-cluster *at time T for a group of close points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ with time stamps $t_1, \ldots, t_n$, is defined as a tuple of $\mathbf{CF1} \in \mathbf{R}^d$ the weighted linear sum of the points, $\mathbf{CF2} \in \mathbb{R}^d$ the weighted squared sum of the points, a weight $w < \beta\mu$, T_0 the creation time, a center $\mathbf{m} \in \mathbb{R}^d$ and a radius $r \leq \epsilon$. $\mathbf{CF1}$, $\mathbf{CF2}$, w, $\mathbf{m}$, r have the same expression as in Definition 2.3.6. $T_0 := t_1$ determines the life span of the outlier-micro-cluster.*

DenStream works then as the following:

1. Initialization: on some first data points from the stream, DBSCAN [Ester et al., 1996] is used to determine a group of potential micro-clusters.
2. When a new point $\mathbf{x}$ is seen,
   - If the radius of the nearest potential core-micro-cluster after merging with $\mathbf{x}$ is below or equal to ϵ, the merging is effective.
   - If the radius of the nearest outlier-micro-cluster after merging with $\mathbf{x}$ is below or equal to ϵ, the merging is effective. Compute the new weight w. If $w > \beta\mu$, this micro-cluster becomes a potential core-micro-cluster.
   - Otherwise, $\mathbf{x}$ creates a new outlier-micro-cluster.
3. For each other existing potential core-micro-cluster, the weight is decayed regularly. It the latter goes below $\beta\mu$, the corresponding micro-cluster becomes an outlier one.

To avoid too many outlier-micro-clusters, a pruning strategy is used: the ones with a weight below a lower limit of weight ξ are removed. See please the paper for how to define this quantity.

DenStream captures non-spherical clusters by applying in an offline process on the core-micro-clusters a DBSCAN-like [Ester et al., 1996] algorithm.

> This method, and so do the other ones described in this Section, requires parameters which are not easy to tune, so the need of a free-parameter clustering algorithm but still keeping the memory usage low and the ability to recover arbitrarily-shaped clusters. This is the contribution of Chapter 5.

Our method belongs to the *graph-based* approximative structure preserving algorithms family. Therefore, we will cover now some state-of-the art graph-based approaches generally for approximative distance / structure preserving algorithms.

2.4 Graph-based approaches

Graph data is known to be a useful representation for structured data in many fields, such as bioinformatics or social, computer, information network analysis where individuals and their interaction need to be expressed. More generally, a graph can always be built based on the dissimilarity (respectively similarity) of data where points of the dataset are the vertices and weighted edges stand for "distances" (respectively similarity) between these objects. In this thesis, we will work with simple undirected graphs. We begin with some fundamental definitions:

Definition 2.4.1 (unweighted graph)**.** *An unweighted graph $G = (V,\ E)$ consists in a set of vertices or nodes V and a set of edges $E \subseteq V \times V$. No attributes are assigned to nodes or edges. The graph is undirected and simple (no self and multiple loops).*

Definition 2.4.2 (weighted graph $\mathcal{G}$)**.** *Let $\mathcal{G} = (V, E, w)$ be a simple undirected weighted graph with a vertex set V, an edge set E, and a weight function $w := E \to \mathbb{R}$ but we assume if not stated otherwise that the weight function $w := E \to (0, 1]$ for simplicity. One will respectively call the edge set and the node set of a graph $\mathcal{G}$ using the applications $E(\mathcal{G})$ and $V(\mathcal{G})$. We call $G = (V, E)$ the topology of the graph.*

In the sequel, cursive letters are used to represent weighted graphs and straight letters refer to topological arguments. $|V|$ and $|E|$ stand respectively for the cardinality of sets V and E. In short, $|V| = N$ and $|E| = M$. For a dense graph,

$$|E| := M = \frac{N(N-1)}{2} = \binom{N}{2} \tag{2.58}$$

and $E := \{e_1, \ldots, e_M\}$. $E(G)$ is used to describe the set of edges of a graph G. Hence, when working with graph data, with N the number of nodes, the data dimension of the object to handle is the number of edges which is $O(N^2)$. If the graph is too large to be stored in the main memory of a single machine, specific data structures can be designed to represent it.

Our proposed model for space-efficient clustering works on a Minimum Spanning Tree (MST) of the dissimilarity graph and will be explained in Chapter 5. A challenge is to retrieve efficiently an MST of this dissimilarity graph when memory does not allow $O(N^2)$ space.

Hence, this Section is organized as follows. First, graph-based clustering algorithms such as graph clustering, Spectral clustering and MST-based approaches are presented in Section 2.4.1. The limits of the methods will be given to introduce our model from Chapter 5. Then, Section 2.4.2 details the preliminaries to our model i.e. how to retrieve efficiently an MST of the dissimilarity graph with a graph sketching technique [Ahn et al., 2012a].

2.4.1 Graph-based clustering

General graph clustering

The approach of representing data with a graph has led to an extensive literature over graph clustering related to graph partitioning [Schaeffer, 2007]. From the clustering methods point of view, DenGraph [Falkowski et al., 2007] proposes a graph version of DBSCAN which is able to deal with noise while work from Ailon et al. [2013] focuses on the problem of recovering clusters with considerably dissimilar sizes.

Recent works include also approaches from convex optimization using low-rank decomposition of the adjacency matrix [Oymak and Hassibi, 2011, Chen et al., 2012a,b, 2014a,b]. These methods bring theoretical guarantees about the exact recovery of the ground truth clustering partition for the Stochastic Block Model [Holland et al., 1983, Condon and Karp, 2001, Rohe et al., 2011] but demand to compute the eigendecomposition of a $N \times N$ matrix which leads respectively to $O(N^3)$ and $O(N^2)$ for time and space complexities. Moreover they are restricted to unweighted graphs. Indeed, weights in the work from Chen et al. [2014b] are about uncertainty of existence of an edge, not a distance between points.

Spectral clustering

Spectral clustering algorithms may be the most popular graph-based clustering family of methods [Luxburg, 2007].

Principle Given N points $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{X} \subset \mathbb{R}^d$ and a similarity function $s : \mathcal{X} \times \mathcal{X} \to \mathbb{Z}$, algorithms from the spectral clustering family:

1. Build a simple undirected weighted *similarity graph* $\mathcal{G} = (V, E, w)$ where V is the set of nodes constituted by the N points, E the set of edges linking

nodes with a positive similarity or a similarity above a certain threshold and $w : E \to \mathbb{Z}^+$ is the corresponding weight function.

2. Find a partition of $\mathcal{G}$ such that the edges between different groups have a very low weight (i.e. small similarity which means that points in different clusters are dissimilar from each other) and the edges within a group have high weight (i.e. high similarity which means that points within the same cluster are similar to each other).

The similarity graph Spectral clustering methods differ first in the way of computing the similarity graph. But for all, the weight on an existing edge is the similarity between the two connected vertices.

- The ϵ-neighborhood graph: All the points with pairwise distance smaller than some $\epsilon > 0$ are connected.
- The k-nearest neighbors graphs: In this model, a vertex is connected with all its k nearest neighbors. As this relationship is asymmetric, the resulting graph would be directed. To make the graph undirected, two strategies can be applied:
  - The directions of the edges are ignored: if vertex u is connected with vertex v, then v is also connected with u. This is the commonly known k-nearest neighbors graph.
  - To connect vertices u and v, v should be among the k-nearest neighbors of u and u should be one of the k-nearest neighbors of v. This is the *mutual* k-nearest neighbors graph.
- The fully connected graph: All points with a positive similarity are connected to each other.

After building the similarity graph $\mathcal{G}$, let us denote the matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ the *adjacency matrix* representing $\mathcal{G}$ where $\mathbf{W}_{ij}$ is the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. We define also $\mathbf{D}$ the diagonal *degree matrix* such that $\forall i \in [N],\ \mathbf{D}_{ii} = \sum_{j \neq i} \mathbf{W}_{ij}$.

The (Graph) Laplacian matrix A useful tool used in spectral clustering algorithms is the *Graph Laplacian* matrix or in short, the *Laplacian* matrix. Again, the definition of the Laplacian matrix depends on the algorithm. There are:

- The unnormalized Laplacian matrix:

$$\mathbf{L} := \mathbf{D} - \mathbf{W} \tag{2.59}$$

- The symmetric normalized Laplacian matrix:

$$\mathbf{L}_{sym} := \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I}_N - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} \tag{2.60}$$

- The random walk normalized Laplacian matrix:

$$\mathbf{L}_{rw} := \mathbf{D}^{-1}\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-1}\mathbf{W}. \tag{2.61}$$

Interesting properties of these matrices are :

- $\mathbf{L}$ is symmetric.

- $\mathbf{L}$, $\mathbf{L}_{sym}$, $\mathbf{L}_{rw}$ are positive semi-definite and have N non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N$. Be careful, here the eigenvalues are sorted in increasing order.
- The multiplicity k of the eigenvalue 0 of $\mathbf{L}$, $\mathbf{L}_{sym}$ and $\mathbf{L}_{rw}$ equals the number of connected components of the similarity graph $\mathcal{G}$.

From this stage, all spectral clustering algorithms share the similar following steps:

- Computing the k first eigenvectors of the Laplacian matrix, $\mathbf{U}_k := [\mathbf{u}_1, \ldots \mathbf{u}_k] \in \mathbb{R}^{N \times k}$.
- Running k-means on the rows of $\mathbf{U}_k$.

Shi and Malik [2000]'s version uses the unnormalized Laplacian matrix, while Ng et al. [2002] takes the normalized symmetric one (and the rows of $\mathbf{U}_k$ are normalized). Where k-means fails, spectral clustering manages to recover non-convex-shaped clusters. Before performing the standard k-means algorithm, a new feature representation of data is computed as the rows of $\mathbf{U}_k$. Nevertheless, k is required and the eigendecomposition of the Laplacian matrix with cost $O(N^3)$ is prohibitive for large datasets.

Our proposed graph-based clustering algorithm in Chapter 5 is only linear in N for the time and space cost. This is made possible by working on a very sparse graph, the Minimum Spanning Tree (MST). Clustering based on an MST is not a new thing. In next Section, we show some MST-based clustering methods.

MST-based clustering approaches

For decades since, MST-based clustering methods [Zahn, 1971, Asano et al., 1988, Xu et al., 2002, Grygorash et al., 2006] have been developed and can be classified into the group of *density*-based techniques since they do not only recover spherical clusters. The Minimum Spanning Tree (MST) is indeed known to help recognizing clusters with arbitrary shapes [Zahn, 1971]. Clustering algorithms from this family identify clusters by performing suitable cuts among the MST edges. The fact that the *connected* graph is a tree i.e. without any cycle, guarantees that each cut in the MST creates two new *connected components* which are assimilated as clusters.

In practice, MST-based clustering algorithms have been successfully applied in bioinformatics [Xu et al., 2002] and image color segmentation [Grygorash et al., 2006].

We recall the definition of a connected graph, of the connected components of a graph and of a minimum spanning tree before introducing some MST-based clustering approaches.

Definition 2.4.3 (connected graph)**.** *A* connected *graph $G = (V, E)$ is a graph where there exists a path between every pair of vertices.*

Definition 2.4.4 (connected component)**.** *A* connected component *of a graph $G = (V, E)$ is a maximal connected subgraph of G such that each vertex is linked to each other by paths, and vertices and edges belong only to this connected component.*

Definitions 2.4.3 and 2.4.4 are straightforwardly adaptable to a weighted graph. Let us now recall the definition of a minimum spanning tree.

Definition 2.4.5 (Minimum spanning tree)**.** *A* minimum spanning tree *$\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T}, w_\mathcal{T})$ of a connected weighted graph $\mathcal{G} = (V, E, w)$ is a connected acyclic subgraph of $\mathcal{G}$ minimizing the sum of edge weights such that $V_\mathcal{T} = V$. The definition*

is extended for a disconnect graph $\mathcal{G}$ as the union of minimum spanning trees of each of its connected components, building a spanning forest (MSF).

Regarding MST-based clustering approaches, let us begin with a simple naive algorithm, called Standard Euclidean MST (SEMST) [Asano et al., 1988, Xu et al., 2002].

Standard Euclidean MST (SEMST) Given a number of expected clusters K, SEMST [Asano et al., 1988, Xu et al., 2002] consists in deleting the $K-1$ heaviest edges from the Euclidean MST of the considered graph. Nevertheless, this completely fails when the intra-cluster distance is lower than the inter-clusters one.

Zahn Euclidean MST (ZEMST) ZEMST algorithm [Zahn, 1971] does not require the number of expected parameters. It simply removes edges that are considered *inconsistent* i.e. in brief the ones with a weight significantly larger than the average weight of the nearby edges in the tree.

Definition 2.4.6 (inconsistent edge [Zahn, 1971])**.** *Let us consider an edge e linking nodes v_1 and v_2. For some $l > 0$, let us define $\mathcal{N}_l(v_1)$ and $\mathcal{N}_l(v_2)$ the set of all edges that belong to the paths of length l originating from v_1, respectively v_2, except edge e. Let us denote $\bar{w}_{\mathcal{N}_l(v_1)}$ and $\bar{w}_{\mathcal{N}_l(v_2)}$ the average weight of edges in $\mathcal{N}_l(v_1)$ respectively in $\mathcal{N}_l(v_2)$. Similarly, $\sigma_{\mathcal{N}_l(v_1)}$ and $\sigma_{\mathcal{N}_l(v_2)}$ are the standard deviation of edges in $\mathcal{N}_l(v_1)$, respectively $\mathcal{N}_l(v_2)$.*

Edge e with weight w is said to be inconsistent *for some parameters $a, a > 0$ and hence will be removed by ZEMST algorithm if any of these equations holds:*

- $w > \bar{w}_{\mathcal{N}_l(v_1)} + a \times \sigma_{\mathcal{N}_l(v_1)}$,
- $w > \bar{w}_{\mathcal{N}_l(v_2)} + a \times \sigma_{\mathcal{N}_l(v_2)}$,
- $\frac{w}{a \max(\bar{w}_{\mathcal{N}_l(v_1)}, \bar{w}_{\mathcal{N}_l(v_2)})} > b$.

Maximum Standard Deviation Reduction (MSDR) MSDR algorithm [Grygorash et al., 2006] considers initially the whole MST as a cluster. It computes the initial standard deviation of the edge weights within this cluster. Then, at each iteration, it tests each cut by computing the standard deviation of the two resulting clusters. It chooses the cut maximizing the overall standard deviation reduction in comparison with the preceding clustering partition. The stopping criterion is when the standard deviation reduction becomes too small. A post-processing based on a polynomial regression of the standard deviation at each step may decide to get back at a previous clustering partition and replace some cut edges.

Some drawbacks of the method can be pointed out.

- MSDR encourages clusters with points far from each other as soon as they are equally "far".
- Moreover, it does not handle clusters with less than three points on which the standard deviation on edge weights is not defined.
- The time complexity is by default $O(N + (N-1) + (N-2) + \ldots 2) = O(N^2)$ since at each iteration, the standard deviation of the edge weights should recomputed.
- The post-processing phase to refine the number of clusters is purely heuristic.

A major bottleneck of MST-based clustering algorithms is the building of an MST which costs $O(|E| \log |V|)$ time, which is basically the time cost to sort the edge set. Even if algorithms for constructing MSTs have been developped to ensure a close to linear time complexity under different hypotheses [Gabow et al., 1986, Fredman and Willard, 1994, Karger et al., 1995], the space cost complexity remains a problem since the $|E|$ edges are assumed to be stored. In the next Section, a graph sketching technique is used to recover an *approximate* MST in $O(N \operatorname{polylog} N)$ time and space.

2.4.2 Graph-sketching approach

Specific data structures have been designed to deal with graphs of $O(N^2)$ edges when they do not fit into memory. Approximation algorithms in charge of analyzing them work directly on this built compact representation of the graph, named *sketch*. The sketch is usually built in a streaming fashion where the stream is a sequence of edges.

The survey from McGregor [2014] gives an overview of the existing literature on processing massive graphs in the streaming model, though lots of graph analyzing tasks involve a memory space in $O(N \operatorname{polylog} N)$. Thus, associated algorithms are rather considered *semi-streaming*.

Moreover, most of the state-of-the-art algorithms work on graphs defined by a sequence of inserted edges and do not consider deletions. This *insert-only streaming model* is a very restrictive one. Work from Ahn et al. [2012a] is the first to provide a *sketch* of a *dynamic* graph supporting insertions and deletions for which *semi-streaming* algorithms can be designed to recover different properties of graphs: connectivity, approximate weight of a minimum spanning tree, bipartiteness, etc. It has lead to numerous works since, with among others the ones from Ahn et al. [2012b, 2013], Bhattacharya et al. [2015], Huang and Peng [2016], Bandyopadhyay et al. [2016].

> In this Section, it is described how to sketch an undirected weighted graph, based on the work from Ahn et al. [2012a]. Please note that in this thesis, only *undirected* graphs are considered.

Let us begin with *unweighted* graphs.

Virtual representation of an unweighted graph

Stream description of an unweighted graph One edge is compactly determined by its position among the ordered set of all possible edges, i.e. the edges of the corresponding complete graph. As the graph is undirected, this set is ordered as the following:

$$E = \{(i, j),\ \forall i \in [N],\ \forall j \in [N] \mid i < j\}. \tag{2.62}$$

Definition 2.4.7 (stream of an unweighted graph)**.** *In the streaming model, an unweighted graph $G = (V,\ E)$ is described by a stream s of edge updates:*

$$s := \langle s_1, \ldots,\ s_j, \ldots \rangle \tag{2.63}$$

where s_j is the j-th update in the stream corresponding to the tuple $s_j = (i, w)$ with i denoting the index of the edge to update according to Equation 2.62 and $w = 1$ if the edge is added or $w = 0$ if it is removed from the graph.

Figure 2.6 represents a graph with 4 nodes. Hence, according to Equation 2.62, the edge set of the corresponding complete graph is, in this order:

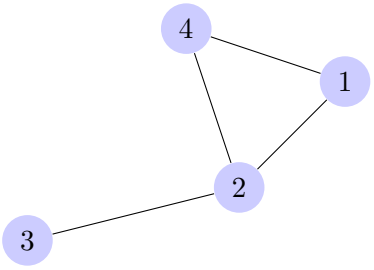


FIGURE 2.6: Graph with 4 nodes defined by the stream s used to illustrate the Graph sketch definition.

$(1,2)$, $(1,3)$, $(1,4)$, $(2,3)$, $(2,4)$, $(3,4)$ indexed from 1 to 6. So the graph can be described by the following stream, considering that only insertions have been made:

$$s = \langle (1,1), (3,1), (4,1), (5,1) \rangle . \tag{2.64}$$

Another possible stream, involving deletion is:

$$s = \langle (1,1), (2,1), (3,1), (4,1), (5,1), (2,0) \rangle \tag{2.65}$$

where the edge number 2 between nodes 1 and 3 is added and later removed from the graph.

Representation of the stream Now, for each node $v_i \in V$, define a vector $\mathbf{a}^{(i)} \in \{-1,0,1\}^M$, $M = |E|$, such that:

$$\mathbf{a}^{(i)}_{\{j,k\}} = \begin{cases} 1 & \text{if } i = j < k \text{ and } \{v_j, v_k\} \in E \\ -1 & \text{if } j < k = i \text{ and } \{v_j, v_k\} \in E \\ 0 & \text{otherwise.} \end{cases} \tag{2.66}$$

For instance, nodes in Figure 2.6 have the following vector representations:

$$\begin{aligned} \mathbf{a}^{(1)} &= (\ 1,\ \ 0,\ 1,\ 0,\ 0,\ 0) \\ \mathbf{a}^{(2)} &= (-1,\ 0,\ 0,\ 1,\ 1,\ 0) \\ \mathbf{a}^{(3)} &= (\ 0,\ \ 0,\ \ 0, -1,\ 0,\ 0) \\ \mathbf{a}^{(4)} &= (\ 0,\ 0, -1,\ 0, -1,\ 0) \end{aligned} \tag{2.67}$$

This representation allows the following Lemma:

Lemma 2.4.1 (Ahn et al. [2012a]). *For any subset of nodes $S \subset V$, the nonzero entries of the concatenated vector $\sum_{i \in S} \mathbf{a}^{(i)}$ correspond exactly to the edges resulting from the cut $(S, V \backslash S)$.*

Representation application for connectivity

Based on the previously explained representation of an unweighted graph G, here is an easy algorithm to recover its connected components in $O(\log N)$ time complexity.

At the beginning, each node is considered as a connected component. One connected component $C \subset V$ is described as a *supernode* the vector representation of which is $\sum_{i \in C} \mathbf{a}^{(i)}$ (cf. Lemma 2.4.1). This way, this supernode behaves like a node resulting from the collapse of the ones contained in C. The ± 1 coefficients trick

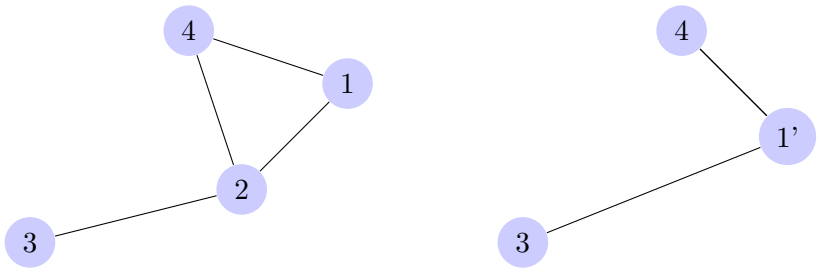


FIGURE 2.7: Supernode illustration with a graph of 4 nodes. (left) the original graph. (right) graph with collapsing nodes 1 and 2 into supernode $1'$.
(left) $\mathbf{a}^{(1)} = (\ 1,\ 0,\ 1,\ 0,\ 0,\ 0)$ and $\mathbf{a}^{(2)} = (-1,\ 0,\ 0,\ 1,\ 1,\ 0)$.
(right) $\mathbf{a}^{(1')} = \mathbf{a}^{(1)} + \mathbf{a}^{(2)} = (0, 0, 1, 1, 1, 0)$

makes edges from the same connected components disappear. Figure 2.7 illustrates the concept of supernodes.

A whole connected component has been recovered as soon as the vector representation of the corresponding supernode is equal to the zero vector. For instance, let us take the graph from Figure 2.6. It is composed from one connected component with vector representation $\sum_{i=1}^{4} \mathbf{a}^{(i)} = (0, \dots, 0)$ where $\mathbf{a}^{(i)}$ for $i = 1, \dots, 4$ are defined in Equation 2.67.

Finally, the algorithm is:

- In the first iteration, an arbitrary incident edge is selected for each node.
- Linked nodes are then collapsed into supernodes by merging the corresponding representation vectors.
- In the next iterations, while there exists an edge from every supernode to another one, they are collapsed into new supernodes making growing the detected connected components.

The number of iterations is bounded by $O(\log N)$ and a spanning forest of the graph is included in the set of selected edges during the process to connect supernodes.

Sketch of an unweighted graph with ℓ_0-sampling

The space cost $O(N^3)$ of the described representation is obviously prohibitive and hopefully not used explicitly in practice. A more space-efficient approach involves the ℓ_0-*sampling* technique [Cormode and Firmani, 2014]. The latter requires $O(\text{polylog}\, N)$ space for representing $\mathbf{a}^{(i)}$ for a given node i. Hence, the whole graph can be sketched into $O(N\, \text{polylog}\, N)$ space and the ℓ_0-*sampling* technique enables to return a random nonzero element from a given $\mathbf{a}^{(i)}$ which corresponds to the selection process of an incident edge from node i.

Definition 2.4.8 (ℓ_0-sampling)**.** *An* (ϵ, δ) ℓ_p-sampler *for a nonzero vector* $\mathbf{x} \in \mathbb{R}^d$ *fails with a probability at most* δ *or returns some* $i \in [d]$ *with probability*

$$(1 \pm \epsilon)\frac{|\mathbf{x}_i|^p}{\|\mathbf{x}\|_p^p} \tag{2.68}$$

where $\|\mathbf{x}\|_p^p = (\sum_{i\in[d]} |\mathbf{x}_i|^p)^{1/p}$ *is the p-norm of x. In particular, if* $p = 0$*, in case of no failure, it returns some i with probability*

$$(1 \pm \epsilon)\frac{1}{|\operatorname{supp}\mathbf{x}|} \tag{2.69}$$

where $\operatorname{supp}\mathbf{x} = \{i \in [d] \mid \mathbf{x}_i \neq 0\}$.

Here considered vectors are $\mathbf{a}^{(i)} \in \mathbb{N}^M$ for all $i \in [N]$. Given some i, $\mathbf{a}^{(i)}$ is described on L levels, $L = O(\log(N))$. The L levels represent virtual copies of the initial vector $\mathbf{a}^{(i)}$ which are more sparse as the number of levels increases. More precisely, for each level $l \in [L]$, the virtual copy is obtained from input $\mathbf{a}^{(i)}$ by zeroing out coordinates $\mathbf{a}_j^{(i)}$ for $j \in [M]$ when $h(j) \neq l$ for some random function h taken from a family of hash functions such that :

$$\begin{cases} h : [M] \to [L] \\ Pr[h(j) = l] = \frac{1}{2^l} \end{cases} \tag{2.70}$$

where $j \in [M]$ corresponds to the j-th coordinate of $\mathbf{a}^{(i)}$. For each level $l \in [L]$, the following linear sketches ϕ, ι, τ are computed as a representation of the virtual copies of the input vector $\mathbf{a}^{(i)}$:

- $\phi = \sum_j \mathbf{a}_j^{(i)}$: is the sum of all coordinates of $\mathbf{a}^{(i)}$,
- $\iota = \sum_j j\ \mathbf{a}_j^{(i)}$: is the sum of all coordinates of $\mathbf{a}^{(i)}$ weighted by the index of the coordinate itself,
- $\tau = \sum_j \mathbf{a}_j^{(i)}\ z^j \mod p$, with p a suitably large prime and $z \in \mathbb{Z}/p\mathbb{Z}$.

Hence, for a given node i, instead of storing the whole vector $\mathbf{a}^{(i)}$ with dimension M, $O(\log N)$ triplets of counters ϕ, ι and τ are stored. In order to apply the previous connectivity algorithm directly on these sketches, the edge selection process should be adapted. If one can detect a triplet representing a nonzero 1-*sparse* vector, the nonzero coordinate corresponds obviously to an incident edge to the (super)node i.

Definition 2.4.9 (k-sparsity)**.** *A vector* $\mathbf{x} \in \mathbb{R}^d$ *is* k-sparse *if and only if* $|\operatorname{supp}\mathbf{x}| \leq k$ *where* $\operatorname{supp}\mathbf{x} = \{i \in [d] \mid \mathbf{x}_i \neq 0\}$*. A vector* $\mathbf{x} \in \mathbb{R}^d$ *is exactly* k*-sparse if* $|\operatorname{supp}\mathbf{x}| = k$.

So a nonzero vector is 1-sparse if it has a unique nonzero coordinate. With only the triplet (ϕ, ι, τ) available, the following test enables to check whether a vector $\mathbf{x}$ is 1-*sparse*: if $\tau = \phi\ z^{\frac{\iota}{\phi}} \mod p$ then $\mathbf{x}$ is 1-sparse.

Lemma 2.4.2 (1-sparsity test)**.** *Given a vector* $\mathbf{x} \in \mathbb{R}^d$*, if* $\mathbf{x}$ *is* 1*-sparse, then the test always gives a positive answer. If* $\mathbf{x}$ *is* k*-sparse,* $1 < k \leq d$*, then the test gives a negative answer with probability at least* $1 - d/p$ *with* p *a suitable large prime taken from the expression of* τ.

Indeed, if $\mathbf{x}$ is 1-sparse for coordinate j, $\phi = 1$, $\iota = j$, $\tau = z^j \mod p$ and clearly $\tau = \phi\ z^{\frac{\iota}{\phi}} \mod p$. But if $\mathbf{x}$ is k-sparse, $k > 1$, there is a risk with probability less than d/p that the test fails, i.e. that the test states $\mathbf{x}$ is 1-sparse instead. So the necessity to take p large enough. It follows that: if 1-sparsity test gives a positive answer, $i = \phi/\iota$ gives the unique nonzero coordinate of $\mathbf{x}$.

Since the edge selection process is now explicit for a graph sketch built from $O(N \log N)$ triplets (ϕ, ι, τ), the final algorithm to compute the connected components of an *unweighted* graph is now described.

Sketch application for connectivity

Independent repetitions of the L levels When willing to draw an incident edge to node i, there is a risk that there is no 1-sparse vector among the L levels of $\mathbf{a}^{(i)}$ and thus, no edge can be selected. To decrease this risk probability, $O(\log N)$ independent repetitions (with independent hash functions) are stored for each level. So the size of the whole sketch becomes $O(N \log^2 N)$ and if no edge has been selected among the L levels of the first repetition, then it is tried to the next L levels of the second repetition, etc. It stops as soon as an edge is found or every repetitions have been investigated without success.

Independent rounds of the repetitions of the L levels Another important tricky point should be stressed about the graph sketches. Suppose that from the sketch of node i, $S(\mathbf{a}^{(i)})$, a neighbor has been queried and node j as been yield.

> The question is: is it allowed to continue to use $S(\mathbf{a}^{(i)})$ to return an additional neighbor of i?

Updating the sketch by deleting j from the the neighborhood and then trying to sample another edge does not work, otherwise by repeating the process, all neighbors of i could be returned from a sketch with size significantly less than $O(N^2)$! It is crucial that the data being sketched should not be adaptively updated based on the sketch itself. The issue can be remedied by growing the sketch to the size $O(N \log^3 N)$: to each node sketch with already size $O(N \log^2 N)$, add $O(\log N)$ rounds corresponding to the number of tries to sample a neighbor for each node i.

The final algorithm which returns the connected components of an unweighted graph based on its sketch with $(N \log^3 N)$ space cost is:

- Construct $t = \log(N)$ sketches S_1, ... S_t for each node i represented by $\mathbf{a}^{(i)}$, $i \in [N]$ with L levels and K repetitions for each node.
- Initialize the set of supernodes $\hat{V} = V$ where supernodes will be build from the collapse of nodes from the same connected component.
- For round $r = 1, \ldots, t$:
  - For each $s \in \hat{V}$, try to sample an inter-supernode edge using the sketch $\sum_{v_i \in s} S_r(\mathbf{a}^{(i)})$ using all levels and repetitions.
  - Update $\hat{V}$ by collapsing the connected supernodes.
- Return $|\hat{V}|$ connected components of G with the selected edges corresponding to a spanning forest.

In the next part, the method is extended to a weighted graph in order to retrieve an approximate MST.

Sketch extension for a weighted graph and application to the approximate (weight of a) MST

Work from Ahn et al. [2012a] gives the following lemma regarding the weight of an approximate MST recovered from the sketch of a graph $\mathcal{G}$. The given upper bound and the proof are corrected here.

Lemma 2.4.3. *Let $\mathcal{G} = (V, E, w)$ be a simple undirected weighted graph with a vertex set V, an edge set E, and a weight function $w := E \to [1, w_{max}]$ where $w_{max} = \text{poly}(|V|) = \text{poly}(N)$ is the maximal possible weight in $\mathcal{G}$. Suppose $\mathcal{G}$ is connected but the same applies if $\mathcal{G}$ is disconnected. Let $\mathcal{G}_i$ be the subgraph of $\mathcal{G}$ consisting of all edges whose weight is at most $w_i = (1+\epsilon)^i$ for $i = 0, \dots, r$ where $r = \lceil \log_{1+\epsilon}(w_{max}) \rceil$ for some small ϵ. Let $cc(\mathcal{H})$ denote the number of connected components of a graph $\mathcal{H}$. Let $\mathcal{T}$ be a minimum spanning tree of $\mathcal{G}$, then it holds:*

$$w(\mathcal{T}) \leq N - (1+\epsilon)^{r+1}\ cc(\mathcal{G}_r) + \sum_{i=0}^{r} \lambda_i\ cc(\mathcal{G}_i) \leq (1+\epsilon)\ w(\mathcal{T}) \tag{2.71}$$

with $w(\mathcal{T})$ denoting the weight of $\mathcal{T}$ i.e. the sum of all edge weights and $\lambda_i = (1+\epsilon)^{i+1} - (1+\epsilon)^i$.

Proof. Consider the graph $\mathcal{G}'$ formed by rounding each edge weight of $\mathcal{G}$ up to the nearest power of $(1+\epsilon)$. It holds clearly:

$$w(\mathcal{T}) \leq w(\mathcal{T}') \leq (1+\epsilon)\ w(\mathcal{T}) \tag{2.72}$$

where $\mathcal{T}'$ is a MST of $\mathcal{G}'$. The idea is now to determine $w(\mathcal{T}')$ which constitutes a lower and upper bound of $w(\mathcal{T})$. $\mathcal{G}'$ has N nodes and K connected components with respectively N_i nodes in each connected component cc_i such that $\sum_{i=1}^{K} N_i = N$. It is clear that for all $i \in [K]$, each connected component cc_i has $N_i - 1$ edges. Indeed, more would imply a cycle which is not allowed in a tree. There is also no edge between the other connected components by Definition 2.4.4.

$$|E(\mathcal{G}')| = \sum_{i=1}^{K}(N_i - 1) = \sum_{i=1}^{K} N_i - K = N - K \tag{2.73}$$

Similarly, where $\mathcal{G}'_i$ is the subgraph of $\mathcal{G}'$ consisting of all edges whose weight is at most $w_i = (1+\epsilon)^i$,

$$\forall i = 0, \dots, r,\ |E(\mathcal{G}'_i)| = N - cc(\mathcal{G}'_i) \tag{2.74}$$

To build $\mathcal{T}'$, let us consider the classical Kruskal algorithm. First, $N - cc(\mathcal{G}'_0)$ edges of weight 1 corresponding to $\mathcal{G}'_0$ are added. Then, edges of weight exactly $1+\epsilon$. Their number is the number of edges in $\mathcal{G}'_1$ minus the number of edges of weight 1 i.e. $N - cc(\mathcal{G}'_1) - (N - cc(\mathcal{G}'_0)) = cc(\mathcal{G}'_0) - cc(\mathcal{G}'_1)$. For the following edges to add, it can be easily shown that the number of edges with weight $(1+\epsilon)^{i+1}$ is equal to $cc(\mathcal{G}'_i) - cc(\mathcal{G}'_{i+1})$. Finally, $w(\mathcal{T}')$ is a weighted sum of the number of edges of each weight corresponding exactly to the Kruskal algorithm:

$$w(\mathcal{T}') = \underbrace{1}_{weight} \times \underbrace{(N - cc(\mathcal{G}'_0))}_{number\ of\ edges} + \sum_{i=0}^{r-1} [\ \underbrace{(1+\epsilon^{i+1})^{i+1}}_{weight}\ \underbrace{(cc(\mathcal{G}'_i) - cc(\mathcal{G}'_{i+1}))}_{number\ of\ edges}\ ] \tag{2.75}$$

$$w(\mathcal{T}') = N - cc(\mathcal{G}_0) + \sum_{i=0}^{r-1} [\ (1+\epsilon^{i+1})^{i+1}\ (cc(\mathcal{G}_i) - cc(\mathcal{G}_{i+1}))\ ] \tag{2.76}$$

by remarking that for all $i \in \{0, \dots, r\}$, $cc(\mathcal{G}'_i) = cc(\mathcal{G}_i)$, where $\mathcal{G}_i$ is the subgraph of $\mathcal{G}$ with the same topology as $\mathcal{G}'_i$. □

Lemma 2.4.3 gives a formula to compute the approximate weight of an MST of a given graph $\mathcal{G}$. To this end, $r+1 = \lceil \log_{1+\epsilon}(w_{max}) \rceil + 1$ subgraphs $\mathcal{G}'_i$ (defined in the Lemma's proof) associated to $\mathcal{G}$ should be sketched with the method previously described for an unweighted graph. Then, the connected components of each subgraphs should be retrieved in order to compute the approximate formula. We easily extend the process to recover a whole approximate MST or MSF (and not just the approximate weight) by storing the selected edges during the procedure to compute the connected components of each subgraph.

As a conclusion, work from Ahn et al. [2012a] has been slightly modified to compute approximately in a single pass an approximate MST (or MSF) by appropriate samplings from the graph sketch. The characteristics of the sketch are the following:

- The spatial complexity of the sketch is more precisely $O(\epsilon^{-1} N \log^3 N)$ with ϵ defined in Lemma 2.4.3 (Theorem 3.4 p.8 from Ahn et al. [2012a]).
- The MST recovery time from the sketch is $O(N \operatorname{polylog}(N))$.
- The time for each update of the sketch is $\operatorname{polylog}(N)$.

Since the algorithm needs only one pass over the data, in this context, the number of nodes is known while the edges, whose weights can be increased or decreased (but have always to stay positive) are summarized into these sketches.

Lastly, it should be noted that the *weighted* graphs on which our node clustering algorithm is applied (Chapter 5) are defined such that all edge weights are a *real number between* 0 *and* 1 but the sketching phase here outputs an MST with *integer* weights and some *known maximal value*. It suffices to scale the weights to go from one type to another.

2.5 Position of the contributions regarding the state of the art

In the previous Sections some current advances have been presented concerning approximatively-distance and structure-preserving algorithms for unsupervised machine learning applications such as nearest neighbor search and clustering.

Some limitations of the state-of-art are addressed in this thesis.

- First, the *data-independent* state-of-the-art Cross-polytope LSH [Terasawa and Tanaka, 2007] for efficient similarity search working with random rotations are not very practical when dealing with very large and high-dimensional datasets. The random matrices can be indeed too large to fit into memory and the multiplication cost with the input vectors prohibitive. To remedy this problem, this thesis describes in Chapter 3 a family of *structured* matrices, named *Structured Spinners* which can be used instead of the random ones. This comes with theoretical guarantees on the accuracy results which are confirmed by the experimental part. In particular, the state-of-the-art structured matrix $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ proposed by Andoni et al. [2015a] where $\mathbf{H}$ denotes the Hadamard transform and $\mathbf{D}_i$ for $i \in \{1,2,3\}$ random diagonal ± 1-matrices lacked of theoretical foundations. This work rectifies this.
- Second, even if the dimensionality reduction techniques based on random projections (like Hyperplane LSH [Charikar, 2002]) have the advantage to constitute a cheaper alternative in terms of space and cost, *data-dependent* (even unsupervised) methods are known to better preserve the distance and structure of

the data and in particular, to provide better accuracy of the result in similarity search via hashing [Wang et al., 2016]. However, state-of-the-art unsupervised hashing techniques *adapted* to the data are rarely compatible with a construction in an *online* fashion. In Chapter 4, a new method is proposed to *learn* small binary codes from a data *stream* in order to retrieve the nearest neighbors of data points. It achieves better accuracy than the competing unsupervised online hashing methods.

- Finally, regarding the state-of-the-art clustering algorithms, almost none of them enable the recover of some arbitrary-shaped clusters space-efficiently without any parameter. Chapter 5 exposes the development of a solution to this effect with DBMSTClu algorithm. DBMSTClu preserves the structure of the dataset by working on only a minimum spanning tree of the underlying dissimilarity graph which ensures a time and space complexity linear in the number of points.

As bonuses, Appendix C shows further applications with theoretical guarantees of the *Structured Spinners* while Appendix E describes how DBMSTClu can be applied for differentially-private clustering.

Chapter 3

Structured random matrices, an approach for fast and large-scale machine learning computations

Contents

This Chapter concerns a collaboration with Krzysztof Choromanski[1], Tamas Sarlos[1], Anna Choromanska[2], Mariusz Bojarski[3], Francois Fagan[4] and Nourhan Sakr[4] published at the International Conference on Artificial Intelligence and Statistics (AISTATS) 2017 under the title "Structured adaptive and random spinners for fast machine learning computations". Anne was responsible of the experimental results except those of Newton sketches and Neural Network accuracy results conducted respectively by Cédric and Mariusz.

[1]Google Brain Robotics
[2]NYU Tandon School of Engineering, ECE
[3]NVIDIA
[4]Columbia University

3.1 Introduction

3.1.1 Projections and classical framework of machine learning algorithms

A striking majority of machine learning algorithms performs *projections* of input data $\mathbf{x}_t \in \mathbb{R}^n$ via some matrices of parameters $\mathbf{A} \in \mathbb{R}^{m \times n}$, where the obtained projections are often passed to a possibly highly *nonlinear* function f: $f(\mathbf{A}\mathbf{x}_t)$. In the case of *randomized* machine learning algorithms, the projection matrix is typically Gaussian with i.i.d. entries taken from $\mathcal{N}(0,1)$. Otherwise, it is learned through an optimization scheme. In the randomized setting, a few examples include:

- the variants of the Johnson-Lindenstrauss Transform (JLT) applying some random projections to reduce the data dimensionality of the input data while approximately preserving the Euclidean distance [Ailon and Chazelle, 2006, Liberty et al., 2008, Ailon and Liberty, 2011],
- the LSH-based schemes [Har-Peled et al., 2012, Charikar, 2002, Terasawa and Tanaka, 2007], including the fastest known variant of the Cross-polytope LSH [Andoni et al., 2015b].

For both cases, $m \ll n$.

> In the frame of our work on *unsupervised* large-scale machine learning, these are the applications considered in this Chapter. For further applications in the randomized and adaptive settings, with potentially $m \gg n$, please refer to Appendix C.

3.1.2 Classical projection cost

The computation of projections takes $\Theta(mn|\mathcal{X}|)$ time, where $m \times n$ is the size of the projection matrix and $|\mathcal{X}|$ denotes the number of data samples from a dataset $\mathcal{X}$. In case of high-dimensional data,

1. this comprises a significant fraction of the overall computational time,
2. while storing the projection matrix frequently becomes a bottleneck in terms of space complexity.

3.1.3 Structuring for time and space savings

In this Chapter, a remedy for both problems is proposed, which relies on replacing the aforementioned algorithms by their "structured variants". As mentioned in Chapter 2, structured matrices were previously explored in the literature mostly in the context of the Johnson-Lindenstrauss Transform (JLT) [Johnson and Lindenstrauss, 1984], where the high-dimensional data are linearly transformed and embedded into a much lower dimensional space while approximately preserving the Euclidean distance between the data points. Section 2.2.3 showed that several extensions of JLT have been proposed, e.g. [Liberty et al., 2008, Ailon and Liberty, 2011, Ailon and Chazelle, 2006, Vybíral, 2011]. Most of these structured constructions involve sparse [Ailon and Chazelle, 2006, Dasgupta et al., 2010] or circulant matrices [Vybíral, 2011, Hinrichs and Vybíral, 2011] providing computational speedups and space compression.

Section 2.2.2 stated that linear projections are used in the LSH setting to construct codes for some given data points which speed up such tasks as approximate

nearest neighbor search. As a recall, a notable set of methods are the so-called Cross-polytope techniques introduced in work from Terasawa and Tanaka [2007] and their aforementioned discrete structured variants proposed in work from Andoni et al. [2015b] that are based on the Walsh-Hadamard transform. Before this work, they were only experimentally verified to produce good quality codes.

Hence, in this Chapter, we propose that the projection is performed by applying a structured matrix from the family that we introduce here as the *Structured spinners*. Depending on the setting,

- the structured matrix is either learned (please see Appendix C for an example with neural networks),
- or its parameters are taken from a random distribution (either continuous or discrete if further compression is required). This is the case considered in this Chapter.

Each structured spinner is a product of three matrix-blocks that incorporate rotations.

A notable member of this family is a matrix of the form $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$, where $\mathbf{D}_i$s are either random diagonal ± 1-matrices or adaptive diagonal matrices and $\mathbf{H}$ is the Hadamard matrix described in Definition 2.2.3.

This matrix is used in the fastest known Cross-polytope LSH method introduced in work from Andoni et al. [2015b] such that the $\mathbf{D}_i$s are random diagonal ± 1-matrices.

In the structured case, the computational speedups are significant. Indeed, the projections can be calculated often in

$$O(\ \max(n, m)\ \log n\ )\ \text{time} \tag{3.1}$$

if the Fast Fourier Transform techniques are applied. In the context considered in this Chapter for Cross-polytope LSH, the cost is more precisely $O(n \log n)$. At the same time, using matrices from the family of *Structured spinners* leads to the reduction of the space complexity to subquadratic, usually at most linear, or sometimes even constant.

3.1.4 Contributions

The key contributions of this Chapter are:

1. The description of the *Structured spinners* family providing a highly parametrized class of structured methods with applications in various randomized or adaptive settings such as:
   - **dimensionality reduction algorithms**,
   - **new fast Cross-polytope LSH techniques**,
   - quantization with random projection trees,
   - kernel approximations via random feature maps,
   - deep learning,
   - convex optimization algorithms via Newton sketches, and more.
2. A comprehensive theoretical explanation of the effectiveness of the structured approach based on the *Structured spinners*. Such analysis was provided in the literature before for a strict subclass of a very general family of structured matrices that are considered in this Chapter. This means that:

The proposed family of the *Structured spinners* contains all previously considered structured matrices as special cases.

This is including the recently introduced P-model [Choromanski and Sindhwani, 2016].

In this Chapter, the first theoretical guarantees for a wide range of discrete structured transforms are provided, in particular for the fastest known Cross-polytope LSH method [Andoni et al., 2015b] based on $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ discrete matrices.

The proposed theoretical methods in the random setting apply the relatively new Berry-Esseen type Central Limit Theorem results for random vectors. The theoretical findings are supported by empirical evidence regarding the accuracy and efficiency of the *Structured spinners* in a wide range of different applications. Not only do the *Structured spinners* cover all already existing structured transforms as special instances, but also many other structured matrices that can be applied in all aforementioned applications.

Plan of the Chapter The model of the *Structured spinners* is explained in Section 3.2. Theoretical guarantees for the random setting are given in Section 3.3 while the reader should refer to Appendix C for the adaptive setting. Finally, experiments are conducted in Section 3.4.

3.2 The family of *Structured spinners*

Before introducing the family of *Structured spinners*, some notations are explained. If not specified otherwise, matrix $\mathbf{D}$ is a random diagonal matrix with diagonal entries taken independently at random from $\{-1, +1\}$. $\mathbf{D}_{t_1,\ldots,t_n}$ denotes the diagonal matrix with diagonal equal to $(t_1, \ldots, t_n)$.

For a matrix $\mathbf{A} = \{a_{i,j}\}_{i,j=1,\ldots,n} \in \mathbb{R}^{n\times n}$, $\|\mathbf{A}\|_F$ denotes its Frobenius norm, i.e.

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i,j\in\{1,\ldots,n\}} a_{i,j}^2}, \tag{3.2}$$

and by $\|\mathbf{A}\|_2$ its spectral norm, i.e.

$$\|\mathbf{A}\|_2 = \sup_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}. \tag{3.3}$$

$\mathbf{H}$ stands for the L_2-normalized Hadamard matrix from Definition 2.2.3. $\mathbf{r}$ is said to be a random Rademacher vector if every element of $\mathbf{r}$ is chosen independently at random from $\{-1, +1\}$. For a vector $\mathbf{r} \in \mathbb{R}^k$ and $n > 0$, let $\mathbf{C}(\mathbf{r}, n) \in \mathbb{R}^{n\times nk}$ be a matrix, where the first row is of the form $(\mathbf{r}^T, 0, \ldots, 0)$ and each subsequent row is obtained from the previous one by right-shifting in a circulant manner the previous one by k. For a sequence of matrices $\mathbf{W}^1, \ldots, \mathbf{W}^n \in \mathbb{R}^{k\times n}$, $\mathrm{V}(\mathbf{W}^1, \ldots, \mathbf{W}^n) \in \mathbb{R}^{nk\times n}$ represents a matrix obtained by vertically stacking matrices: $\mathbf{W}^1, \ldots, \mathbf{W}^n$. Each structured matrix $\mathbf{G}_{struct} \in \mathbb{R}^{n\times n}$ from the family of *Structured spinners* is a product of three main structured components/blocks, i.e.:

$$\mathbf{G}_{struct} = \mathbf{M}_3\mathbf{M}_2\mathbf{M}_1, \tag{3.4}$$

where matrices $\mathbf{M}_1, \mathbf{M}_2$ and $\mathbf{M}_3$ satisfy Conditions 3.2.1, 3.2.2 and 3.2.3.

Condition 3.2.1. *Matrices* $\mathbf{M}_1$ *and* $\mathbf{M}_2\mathbf{M}_1$ *are* $(\delta(n), p(n))$*-balanced isometries.*

Condition 3.2.2. $\mathbf{M}_2 = \mathrm{V}(\mathbf{W}^1, \ldots, \mathbf{W}^n)\mathbf{D}_{\rho_1,\ldots,\rho_n}$ *for some* (Δ_F, Δ_2)*-smooth set such that* $\mathbf{W}^1, \ldots, \mathbf{W}^n \in \mathbb{R}^{k\times n}$ *and* $\rho_1, \ldots, \rho_n$ *are some i.i.d sub-Gaussian random variables with sub-Gaussian norm* K*.*

Condition 3.2.3. $\mathbf{M}_3 = \mathbf{C}(\mathbf{r}, n)$ *for* $\mathbf{r} \in \mathbb{R}^k$*, where* $\mathbf{r}$ *is random Rademacher/Gaussian in the random setting or is learned in the adaptive setting.*

Hence, matrix $\mathbf{G}_{struct}$ is a structured spinner with parameters: $\delta(n), p(n), K, \Lambda_F$ and Λ_2. The introduced conditions are explained below with Definition 3.2.1 of a $(\delta(n), p(n))$-balanced matrix, Definition 3.2.2 for a sub-Gaussian norm (see Definition 2.2.2 p.25 for a sub-Gaussian variable) and Definition 3.2.3 of a (Λ_F, Λ_2)-smooth set of matrices.

Definition 3.2.1 ($(\delta(n), p(n))$-balanced matrix)**.** *A randomized matrix* $\mathbf{M} \in \mathbb{R}^{n\times m}$ *is* $(\delta(n), p(n))$-balanced *if for every* $\mathbf{x} \in \mathbb{R}^m$ *with* $\|\mathbf{x}\|_2 = 1$ *the following holds:*

$$\mathbb{P}[\|\mathbf{M}\mathbf{x}\|_\infty > \frac{\delta(n)}{\sqrt{n}}] \leq p(n). \tag{3.5}$$

Remark 3.2.1. *One can take as* $\mathbf{M}_1$ *a matrix* $\mathbf{HD}_1$ *since matrix* $\mathbf{HD}_1$ *is* $(\log(n), 2ne^{-\frac{\log^2(n)}{8}})$*-balanced.*

This result first appeared in work from Ailon and Chazelle [2006]. The following proof was given in work from Choromanski and Sindhwani [2016], it is repeated it here for completeness. In this proof, the standard concentration result named Azuma's Inequality is used [Azuma, 1967].

Lemma 3.2.1. *(Azuma's Inequality) Let* $X_1, \ldots, X_n$ *be a martingale and assume that* $-\alpha_i \leq X_i \leq \beta_i$ *for some positive constants* $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$*. Denote* $X = \sum_{i=1}^n X_i$*. Then the following is true:*

$$\mathbb{P}[|X - \mathbb{E}[X]| > a] \leq 2e^{-\frac{a^2}{2\sum_{i=1}^n (\alpha_i+\beta_i)^2}} \tag{3.6}$$

Now let us proof Remark 3.2.1.

Proof. Denote by $\tilde{\mathbf{x}}^j$ an image of $\mathbf{x}^j$ under transformation $\mathbf{HD}$. Note that the i^{th} dimension of $\tilde{\mathbf{x}}^j$ is given by the formula: $\tilde{x}_i^j = h_{i,1}x_1^j + \ldots + h_{i,n}x_n^j$, where $h_{l,u}$ stands for the l^{th} element of the u^{th} column of the randomized Hadamard matrix $\mathbf{HD}$. First, Azuma's Inequality is used to find an upper bound on the probability such that $|\tilde{x}_i^j| > a$, where $a = \frac{\log(n)}{\sqrt{n}}$. By Azuma's Inequality, there is:

$$\mathbb{P}[|h_{i,1}x_1^j + \ldots + h_{i,n}x_n^j| \geq a] \leq 2e^{-\frac{\log^2(n)}{8}}. \tag{3.7}$$

The following is used: $\alpha_i = \beta_i = \frac{1}{\sqrt{n}}$. Now the union bound over all n dimensions is taken and the proof is completed. □

Definition 3.2.2 (sub-Gaussian norm)**.** *For a random sub-Gaussian variable* X*, the sub-Gaussian norm of* X *is defined as:*

$$\|X\|_{\psi_2} = \inf\{t > 0 \mid \mathbb{E}[e^{\frac{X^2}{t^2}}] \leq 2\} \tag{3.8}$$

Definition 3.2.3 ((Λ_F, Λ_2)-smooth set)**.** *A deterministic set of matrices* $\mathbf{W}^1, \ldots, \mathbf{W}^n \in \mathbb{R}^{k \times n}$ *is* (Λ_F, Λ_2)-smooth *if:*

- $\|\mathbf{W}_1^i\|_2 = \ldots = \|\mathbf{W}_n^i\|_2$ *for* $i = 1, \ldots, n$*, where* $\mathbf{W}_j^i$ *stands for the* j^{th} *column of* $\mathbf{W}^i$*,*
- *for* $i \neq j$ *and* $l = 1, \ldots, n$ *it holds:* $(\mathbf{W}_l^i)^T \cdot \mathbf{W}_l^j = 0$*,*
- $\max_{i,j} \|(\mathbf{W}^j)^T \mathbf{W}^i\|_F \leq \Lambda_F$ *and* $\max_{i,j} \|(\mathbf{W}^j)^T \mathbf{W}^i\|_2 \leq \Lambda_2$.

Remark 3.2.2. *If the unstructured matrix* $\mathbf{G}$ *has rows taken from the general multivariate Gaussian distribution with diagonal covariance matrix* $\mathbf{\Sigma} \neq \mathbf{I}$ *then one needs to rescale vectors* $\mathbf{r}$ *accordingly. For clarity, it is assumed here that* $\mathbf{\Sigma} = \mathbf{I}$ *and theoretical results are presented for this setting.*

All structured matrices previously considered are special cases of a wider family of *Structured spinners* (for clarity, it will be explicitly shown for some important special cases). It holds:

Lemma 3.2.2. *The following matrices:* $\mathbf{G}_{circ}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$*,* $\sqrt{n}\mathbf{H}\mathbf{D}_3\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$ *and* $\sqrt{n}\mathbf{H}\mathbf{D}_{g_1,\ldots,g_n}\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$*, where* $\mathbf{G}_{circ}$ *is Gaussian circulant, are valid structured spinners for* $\delta(n) = \log(n)$*,* $p(n) = 2ne^{-\frac{\log^2(n)}{8}}$*,* $K = 1$*,* $\Lambda_F = O(\sqrt{n})$ *and* $\Lambda_2 = O(1)$*. The same is true if one replaces* $\mathbf{G}_{circ}$ *by a Gaussian Hankel or Toeplitz matrix.*

Proof. See proof below in Section 3.3.3 after introduction to new tools. □

3.2.1 The role of the three blocks $\mathbf{M_1}$, $\mathbf{M_2}$, and $\mathbf{M_3}$

The role of the blocks $\mathbf{M}_1$, $\mathbf{M}_2$, $\mathbf{M}_3$ can be intuitively explained. Matrix $\mathbf{M}_1$ makes vectors "balanced", so that there is no dimension that carries too much of the L_2-norm of the vector. The balanceness property was already applied in the structured setting [Ailon and Chazelle, 2006].

The role of $\mathbf{M}_2$ is more subtle and differs between adaptive and random settings. In the random setting, the cost of applying the structured mechanism is the loss of independence. For instance, the dot products of the rows of a circulant Gaussian matrix with a given vector $\mathbf{x}$ are no longer independent, as it is the case in the fully random setup. Those dot products can be expressed as a dot product of a fixed Gaussian row with different vectors $\mathbf{v}$. Matrix $\mathbf{M}_2$ makes these vectors close to orthogonal[5].

Finally, matrix $\mathbf{M}_3$ defines the capacity of the entire structured transform by providing a vector of parameters (either random or to be learned). The near-independence of the aforementioned dot products in the random setting is now implied by the near-orthogonality property achieved by $\mathbf{M}_2$ and the fact that the projections of the Gaussian vector or the random Rademacher vector onto "almost orthogonal directions" are "close to independent".

The role of the three matrices is described pictorially in Figure 3.1.

[5]In the adaptive setup, the "close to orthogonality" property is replaced by the independence property.

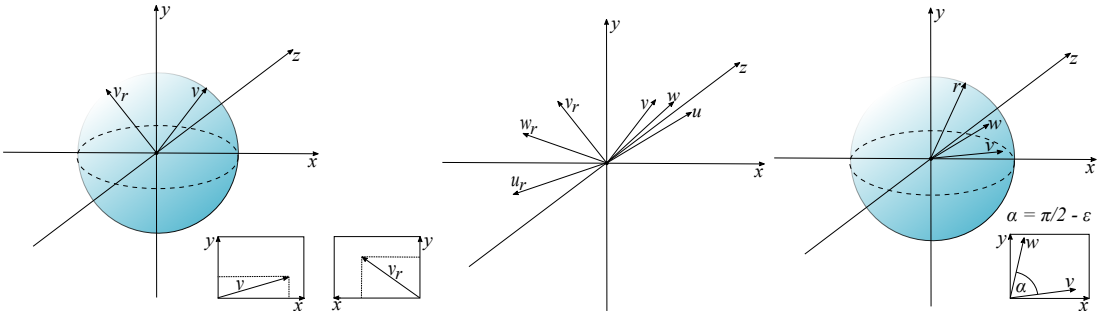


FIGURE 3.1: Pictorial explanation of the role of the three matrix-blocks in the construction of the structured spinner. (left) $\mathbf{M}_1$ rotates vector $\mathbf{v}$ such that the rotated version $\mathbf{v}_r$ is balanced. (middle) $\mathbf{M}_2$ transforms vectors $\mathbf{v}, \mathbf{w}, \mathbf{u}$ such that their images $\mathbf{v}_r, \mathbf{w}_r, \mathbf{u}_r$ are near-orthogonal. (right) The projections of the random vector $\mathbf{r}$ onto such two near-orthogonal vectors $\mathbf{v}$, $\mathbf{w}$ are near-independent.

3.2.2 Stacking together *Structured spinners*

The *Structured spinners* are described as square matrices, but in practice it is not restricted to those. One can construct an $m \times n$ structured spinner for $m \leq n$ from the square $n \times n$ structured spinner by taking its first m rows(see Appendix B). One can then stack vertically these independently constructed $m \times n$ matrices to obtain an $k \times n$ matrix for both: $k \leq n$ and $k > n$.

The constant m should be seen as another parameter of the model that tunes the "structuredness" level, i.e. larger values of m indicate a more structured approach while smaller values lead to more random matrices (the case $m = 1$ is the fully unstructured one).

Now that the model of the *Structured spinners* is fully defined, let us see the theoretical guarantees that accompany it. The main purpose of them is to prove Lemma 3.2.2 and the accuracy of the *Structured Spinners* in the randomized setting, in particular for the Cross-polytope LSH with $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$[6].

3.3 Theoretical results

3.3.1 What will be shown

It is shown now that *Structured spinners* can replace their unstructured counterparts in many machine learning algorithms (not only for dimensionality reduction techniques and LSH-based schemes) with minimal loss of accuracy.

Let $\mathcal{A}_\mathcal{G}$ be a machine learning algorithm applied to a fixed dataset $\mathcal{X} \subseteq \mathbb{R}^n$ and parametrized by a set $\mathcal{G}$ of matrices $\mathbf{G} \in \mathbb{R}^{m \times n}$, where each $\mathbf{G}$ is either learned or Gaussian with independent entries taken from $\mathcal{N}(0, 1)$. Assume furthermore, that $\mathcal{A}_\mathcal{G}$ consists of functions $f_1, \ldots, f_s$, where each f_i applies a certain matrix $\mathbf{G}_i$ from $\mathcal{G}$ to vectors from some linear space $\mathcal{L}_i$ of dimensionality at most d. Note that for a fixed dataset $\mathcal{X}$ function f_i is a function of a random vector

$$\mathbf{q}_{f_i} = ((\mathbf{G}_i\mathbf{x}^1)^T, \ldots, (\mathbf{G}_i\mathbf{x}^{d_i})^T)^T \in \mathbb{R}^{d_i \cdot m}, \tag{3.9}$$

[6]Some proofs in the adaptive setting can be found in Appendix C.

where $dim(\mathcal{L}_i) = d_i \leq d$ and $\mathbf{x}^1, \ldots, \mathbf{x}^{d_i}$ stands for some fixed basis of $\mathcal{L}_i$.

Denote by f_i' the structured counterpart of f_i, where $\mathbf{G}_i$ is replaced by the structured spinner (for which vector $\mathbf{r}$ is either learned or random).

It will be shown that $f_i's$ "resemble" f_is distribution-wise; and this, surprisingly, under very weak conditions regarding f_is. In particular, they can be nondifferentiable, even non-continuous.

To that purpose, an equivalent definition of the *Structured spinners*' model is introduced. It is more technical, yet more convenient to work with in the proofs. Besides, this new model enables to give the proof of Lemma 3.2.2.

3.3.2 *Structured spinners*' equivalent definition

Note that from the initial definition of the *Structured spinners* one can conclude that each structured matrix $\mathbf{G}_{struct} \in \mathbb{R}^{n \times n}$ from the family of structured spinners is a product of three main structured blocks, i.e.:

$$\mathbf{G}_{struct} = \mathbf{B}_3\mathbf{B}_2\mathbf{B}_1, \tag{3.10}$$

where matrices $\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3$ satisfy two conditions that are given below.

Condition 3.3.1. *Matrices:* $\mathbf{B}_1$ *and* $\mathbf{B}_2\mathbf{B}_1$ *are* $(\delta(n), p(n))$*-balanced isometries.*

Condition 3.3.2. *Pair of matrices* $(\mathbf{B}_2, \mathbf{B}_3)$ *is* $(K, \Lambda_F, \Lambda_2)$*-random.*

Below the definition of $(K, \Lambda_F, \Lambda_2)$-randomness is given.

Definition 3.3.1 ($(K, \Lambda_F, \Lambda_2)$-randomness)**.** *A pair of matrices* $(\mathbf{Y}, \mathbf{Z}) \in \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times n}$ *is* $(K, \Lambda_F, \Lambda_2)$-random *if there exists* $\mathbf{r} \in \mathbb{R}^k$*, and a set of linear isometries* $\phi = \{\phi_1, \ldots, \phi_n\}$*, where* $\phi_i : \mathbb{R}^n \to \mathbb{R}^k$*, such that:*

- $\mathbf{r}$ *is either a* ± 1*-vector with i.i.d. entries or Gaussian with identity covariance matrix,*
- *for every* $\mathbf{x} \in \mathbb{R}^n$ *the* j^{th} *element* $(\mathbf{Zx})_j$ *of* $\mathbf{Zx}$ *is of the form:* $\mathbf{r}^T \cdot \phi_j(\mathbf{x})$*,*
- *there exists a set of i.i.d. sub-Gaussian random variables* $\{\rho_1, \ldots, \rho_n\}$ *with sub-Gaussian norm at most* K*, mean* 0*, the same second moments and a* (Λ_F, Λ_2)*-smooth set of matrices* $\{\mathbf{W}^i\}_{i=1,\ldots,n}$ *such that for every* $\mathbf{x} = (x_1, \ldots, x_n)^T$*, the following holds:* $\phi_i(\mathbf{Yx}) = \mathbf{W}^i(\rho_1 x_1, \ldots, \rho_n x_n)^T$*.*

3.3.3 Proof of Lemma 3.2.2

Finally, we can prove Lemma 3.2.2 which is recalled here, within the modified model.

Lemma 3.2.2 *The following matrices:* $\mathbf{G}_{circ}\mathbf{D}_2\mathbf{HD}_1$*,* $\sqrt{n}\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ *and* $\sqrt{n}\mathbf{HD}_{g_1,\ldots,g_n}\mathbf{HD}_2\mathbf{HD}_1$*, where* $\mathbf{G}_{circ}$ *is Gaussian circulant, are valid structured spinners for* $\delta(n) = \log(n)$*,* $p(n) = 2ne^{-\frac{\log^2(n)}{8}}$*,* $K = 1$*,* $\Lambda_F = O(\sqrt{n})$ *and* $\Lambda_2 = O(1)$*. The same is true if one replaces* $\mathbf{G}_{circ}$ *by a Gaussian Hankel or Toeplitz matrix.*

Proof. Let us first assume the $\mathbf{G}_{circ}\mathbf{D}_2\mathbf{HD}_1$-setting (analysis for Toeplitz Gaussian or Hankel Gaussian is completely analogous). In that setting, it is easy to see that one can take $\mathbf{r}$ to be a Gaussian vector (this vector corresponds to the

first row of $\mathbf{G}_{circ}$). Furthermore linear mappings ϕ_i are defined as:

$$\phi_i((x_0, x_1, \ldots, x_{n-1})^T) = (x_{n-i}, x_{n-i+1}, \ldots, x_{i-1})^T, \tag{3.11}$$

where operations on indices are modulo n. The value of $\delta(n)$ and $p(n)$ come from the fact that matrix $\mathbf{HD}_1$ is used as a $(\delta(n), p(n))$-balanced matrix and from Remark 3.2.1. In that setting, sequence $(\rho_1, \ldots, \rho_n)$ is discrete and corresponds to the diagonal of $\mathbf{D}_2$. Thus, $K = 1$.

To calculate Λ_F and Λ_2, note first that matrix $\mathbf{W}^1$ is defined as $\mathbf{I}$ and subsequent $\mathbf{W}^i$s are given as circulant shifts of the previous ones (i.e. each row is a circulant shift of the previous row). That observation comes directly from the circulant structure of $\mathbf{G}_{circ}$. Thus there is: $\Lambda_F = O(\sqrt{n})$ and $\Lambda_2 = O(1)$. The former is true since each $\mathbf{A}^{i,j} := (\mathbf{W}^j)^T\mathbf{W}^i$ has $O(n)$ nonzero entries and these are all 1s. The latter is true since each nontrivial $\mathbf{A}^{i,j}$ in that setting is an isometry (this is straightforward from the definition of $\{\mathbf{W}^i\}_{i=1,\ldots,n}$).

Finally, all other conditions regarding $\mathbf{W}^i$-matrices are clearly satisfied (each column of each $\mathbf{W}^i$ has unit L_2 norm and corresponding columns from different $\mathbf{W}^i$ and $\mathbf{W}^j$ are clearly orthogonal).

Now let us consider the setting, where the structured matrix is of the form: $\sqrt{n}\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$. In that case, $\mathbf{r}$ corresponds to a discrete vector (namely, the diagonal of $\mathbf{D}_3$). Linear mappings ϕ_i are defined as:

$$\phi_i((x_1, \ldots, x_n)^T) = (\sqrt{n}h_{i,1}x_1, \ldots, \sqrt{n}h_{i,n}x_n)^T, \tag{3.12}$$

where $(h_{i,1}, \ldots, h_{i,n})^T$ is the i^{th} row of $\mathbf{H}$. One can also notice that the set $\{\mathbf{W}^i\}_{i=1,\ldots,n}$ is defined as: $w^i_{a,b} = \sqrt{n}h_{i,a}h_{a,b}$. Let us first compute the Frobenius norm of the matrix $\mathbf{A}^{i,j}$, defined based on the aforementioned sequence $\{\mathbf{W}^i\}_{i=1,\ldots,n}$:

$$\begin{aligned}\|\mathbf{A}^{i,j}\|_F^2 &= \sum_{l,t\in\{1,\ldots,n\}} (\mathbf{A}^{i,j}_{l,t})^2 \\ &= \sum_{l,t\in\{1,\ldots,n\}} (\sum_{k=1}^{n} w^j_{k,l}w^i_{k,t})^2 = n^2 \sum_{l,t\in\{1,\ldots,n\}} (\sum_{k=1}^{n} h_{j,k}h_{k,l}h_{i,k}h_{k,t})^2\end{aligned} \tag{3.13}$$

To compute the expression above, note first that for $r_1 \neq r_2$ the following holds:

$$\theta = \sum_{k,l} h_{r_1,k}h_{r_1,l}h_{r_2,k}h_{r_2,l} = \sum_k h_{r_1,k}h_{r_2,k} \sum_l h_{r_1,l}h_{r_2,l} = 0, \tag{3.14}$$

where the last equality comes from fact that different rows of $\mathbf{H}$ are orthogonal. From the fact that $\theta = 0$, there is:

$$\|\mathbf{A}^{i,j}\|_F^2 = n^2 \sum_{r=1,\ldots,n} \sum_{k,l} h^2_{i,r}h^2_{j,r}h^2_{r,k}h^2_{r,l} = n \cdot n^2(\frac{1}{\sqrt{n}})^8 \cdot n^2 = n. \tag{3.15}$$

Thus there is: $\Lambda_F \leq \sqrt{n}$.

Now $\|\mathbf{A}^{i,j}\|_2$ is computed. Notice that from the definition of $\mathbf{A}^{i,j}$ there is

$$\mathbf{A}^{i,j} = \mathbf{E}^{i,j}\mathbf{F}^{i,j}, \tag{3.16}$$

where the l^{th} row of $\mathbf{E}^{i,j}$ is of the form $(h_{j,1}h_{1,l}, \ldots, h_{j,n}h_{n,l})$ and the t^{th} column

of $\mathbf{F}^{i,j}$ is of the form $(h_{i,1}h_{1,t},\ldots,h_{i,n}h_{n,t})^T$. Thus one can easily verify that $\mathbf{E}^{i,j}$ and $\mathbf{H}^{i,j}$ are isometries (since $\mathbf{H}$ is) thus $\mathbf{A}^{i,j}$ is also an isometry and therefore $\Lambda_2 = 1$. As in the previous setting, remaining conditions regarding matrices $\mathbf{W}^i$ are trivially satisfied (from the basic properties of Hadamard matrices). That completes the proof. □

Now, taking proof's goal of Section 3.3.1, the *Structured spinners* are shown to be an accurate counterpart to unstructured matrices in the randomized setting[7].

3.3.4 Accuracy of the *Structured Spinners* in the randomized setting

The following definition is needed.

Definition 3.3.2 (b-convexity of a set)**.** *A set $\mathcal{S}$ is b-convex if it is a union of at most b pairwise disjoint convex sets.*

Fix a function $f_i : \mathbb{R}^{d_i \cdot m} \rightarrow \mathcal{V}$, for some domain $\mathcal{V}$. Our main result states that for any $\mathcal{S} \subseteq \mathcal{V}$ such that $f_i^{-1}(\mathcal{S})$ is measurable and b-convex for b not too large, the probability that $f_i(\mathbf{q}_{f_i})$ belongs to $\mathcal{S}$ is close to the probability that $f_i'(\mathbf{q}_{f_i'})$ belongs to $\mathcal{S}$.

Theorem 3.3.1 (structured random setting)**.** *Let $\mathcal{A}$ be a randomized algorithm using unstructured Gaussian matrices $\mathbf{G}$ and let s, d and f_is be as at the beginning of the section. Replace the unstructured matrix $\mathbf{G}$ by one of the structured spinners defined in Section 3.2 with blocks of m rows each. Then for n large enough, $\epsilon = o_{md}(1)$ and fixed f_i with probability p_{succ} at least:*

$$1 - 2p(n)d - 2\binom{md}{2}e^{-\Omega(\min(\frac{\epsilon^2 n^2}{K^4\Lambda_F^2\delta^4(n)}, \frac{\epsilon n}{K^2\Lambda_2\delta^2(n)}))} \tag{3.17}$$

with respect to the random choices of $\mathbf{M}_1$ and $\mathbf{M}_2$ the following holds for any $\mathcal{S}$ such that $f_i^{-1}(\mathcal{S})$ is measurable and b-convex:

$$|\mathbb{P}[f_i(q_{f_i}) \in \mathcal{S}] - \mathbb{P}[f_i'(q_{f_i'}) \in \mathcal{S}]| \leq b\eta, \tag{3.18}$$

where the probabilities in the last formula are with respect to the random choice of $\mathbf{M}_3$, $\eta = \frac{\delta^3(n)}{n^{\frac{2}{5}}}$, and $\delta(n), p(n), K, \Lambda_F, \Lambda_2$ are as in the definition of structured spinners from Section 3.2.

Overview of Theorem 3.3.1's proof Let us briefly give an overview of the proof before presenting it in detail. Challenges regarding proving accuracy results for structured matrices come from the fact that, for any given $\mathbf{x} \in \mathbb{R}^n$, different dimensions of $\mathbf{y} = \mathbf{G}_{struct}\mathbf{x}$ are no longer independent (as it is the case for the unstructured setting). For matrices from the family of structured spinners one can, however, show that with high probability different elements of $\mathbf{y}$ correspond to projections of a given vector $\mathbf{r}$ (see Section 3.2) into directions that are close to orthogonal. The "close-to-orthogonality" characteristic is obtained with the use of the *Hanson-Wright inequality* that focuses on concentration results regarding quadratic forms involving vectors of sub-Gaussian random variables. If $\mathbf{r}$ is Gaussian, then from the well-known fact that projections of the Gaussian vector into orthogonal directions are independent, one

[7] Theoretical guarantees for the adaptive setting can be found in Appendix C.

can conclude that dimensions of $\mathbf{y}$ are "close to independent". If $\mathbf{r}$ is a discrete vector then there is a need to show that for n large enough, it "resembles" the Gaussian vector. This is where the aforementioned techniques regarding multivariate Berry-Esseen-type central limit theorem results need to be applied.

Proof. Notation from Section 3.2 and previous sections is used. It is assumed that the model with structured matrices stacked vertically, each of m rows, is applied. Without loss of generality, one can assume that there is just one block since different blocks are chosen independently. Let $\mathbf{G}_{struct}$ be a matrix from the family of structured spinners. Let us assume that $\mathbf{G}_{struct}$ is used by a function f operating in the d-dimensional space and let us denote by $\mathbf{x}^1, \ldots, \mathbf{x}^d$ some fixed orthonormal basis of that space.

$\mathbf{B}_1$ application. The first goal is to compute:

$$\mathbf{y}^1 = \mathbf{G}_{struct}\mathbf{x}^1, \ldots, \mathbf{y}^d = \mathbf{G}_{struct}\mathbf{x}^d. \tag{3.19}$$

Denote by $\tilde{\mathbf{x}}^i$ the linearly transformed version of $\mathbf{x}$ after applying block $\mathbf{B}_1$, i.e. $\tilde{\mathbf{x}}^i = \mathbf{B}_1\mathbf{x}^i$. Since $\mathbf{B}_1$ is $(\delta(n), p(n))$-balanced), with probability at least: $p_{balanced} \geq 1 - dp(n)$ each element of each $\tilde{\mathbf{x}}^i$ has absolute value at most $\frac{\delta(n)}{\sqrt{n}}$. One can shortly say that each $\tilde{\mathbf{x}}^i$ is $\delta(n)$-balanced. This event is called $\mathcal{E}_{balanced}$.

Note that by the definition of structured spinners, each $\mathbf{y}^i$ is of the form:

$$\mathbf{y}^i = (\mathbf{r}^T \cdot \phi_1(\mathbf{B}_2\tilde{\mathbf{x}}^i), \ldots, \mathbf{r}^T \cdot \phi_m(\mathbf{B}_2\tilde{\mathbf{x}}^i))^T. \tag{3.20}$$

$\mathbf{B}_2$ application. For clarity and to reduce notation, $\mathbf{r}$ is assumed to be n-dimensional. To obtain results for vectors $\mathbf{r}$ of different dimensionality D, it suffices to replace in our analysis and theoretical statements n by D. Let us denote

$$\mathcal{A} = \{\phi_1(\mathbf{B}_2\tilde{\mathbf{x}}^1), \ldots, \phi_m(\mathbf{B}_2\tilde{\mathbf{x}}^1), \ldots, \phi_1(\mathbf{B}_2\tilde{\mathbf{x}}^d), \ldots, \phi_m(\mathbf{B}_2\tilde{\mathbf{x}}^d))\}. \tag{3.21}$$

Our goal is to show that with high probability (in respect to random choices of $\mathbf{B}_1$ and $\mathbf{B}_2$) for all $\mathbf{v}^i, \mathbf{v}^j \in \mathcal{A}$, $i \neq j$ the following is true:

$$|(\mathbf{v}^i)^T \cdot \mathbf{v}^j| \leq t \tag{3.22}$$

for some given $0 < t \ll 1$.

Fix some $t > 0$, the lower bound on the corresponding probability is searched. Let us fix two vectors $\mathbf{v}^1, \mathbf{v}^2 \in \mathcal{A}$ and denote them as: $\mathbf{v}^1 = \phi_i(\mathbf{B}_2\mathbf{x})$, $\mathbf{v}^2 = \phi_j(\mathbf{B}_2\mathbf{y})$ for some $\mathbf{x} = (x_1, \ldots, x_n)^T$ and $\mathbf{y} = (y_1, \ldots, y_n)^T$. With denotation from Section, there is 3.2):

$$\phi_i(\mathbf{B}_2\mathbf{x}) = (w^i_{11}\rho_1 x_1 + \ldots + w^i_{1,n}\rho_n x_n, \ldots, w^i_{n,1}\rho_1 x_1 + \ldots + w^i_{n,n}\rho_n x_n)^T \tag{3.23}$$

and

$$\phi_j(\mathbf{B}_2\mathbf{y}) = (w^j_{11}\rho_1 y_1 + \ldots + w^j_{1,n}\rho_n y_n, \ldots, w^j_{n,1}\rho_1 y_1 + \ldots + w^j_{n,n}\rho_n y_n)^T. \tag{3.24}$$

Then, the following holds:

$$(\mathbf{v}^1)^T \cdot \mathbf{v}^2 = \sum_{l\in\{1,\ldots,n\}, u\in\{1,\ldots,n\}} \rho_l\rho_u(\sum_{k=1}^{n} x_l y_u w^i_{k,u} w^j_{k,l}). \tag{3.25}$$

Under assumptions from Theorem 3.3.1, the expected value of the expression above is now shown to be 0. There is:

$$\mathbb{E}[(\mathbf{v}^1)^T \cdot \mathbf{v}^2] = \mathbb{E}[\sum_{l \in \{1,\ldots,n\}} \rho_l^2 x_l y_l (\sum_{k=1}^{n} w_{k,l}^i w_{k,l}^j)], \tag{3.26}$$

since $\rho_1, \ldots, \rho_n$ are independent and have expectations equal to 0. Now notice that if $i \neq j$ then from the assumption that corresponding columns of matrices $\mathbf{W}^i$ and $\mathbf{W}^j$ are orthogonal, the above expectation is 0. Now assume that $i = j$. But then $\mathbf{x}$ and $\mathbf{y}$ have to be different and thus they are orthogonal (since they are taken from the orthonormal system transformed by an isometry). In that setting it holds:

$$\mathbb{E}[(\mathbf{v}^1)^T \cdot \mathbf{v}^2] = \mathbb{E}[\sum_{l \in \{1,\ldots,n\}} \rho_l^2 x_l y_l (\sum_{k=1}^{n} (w_{k,l}^i)^2)] = \tau w \sum_{l=1}^{n} x_l y_l = 0, \tag{3.27}$$

where τ stands for the second moment of each ρ_i, w is the squared L_2-norm of each column of $\mathbf{W}^i$ (τ and w are well defined due to the properties of structured spinners). The last inequality comes from the fact that $\mathbf{x}$ and $\mathbf{y}$ are orthogonal. Now if matrices $\mathbf{A}^{i,j}$ are defined as in the definition of the model of structured spinners then one can see that

$$(\mathbf{v}^1)^T \cdot \mathbf{v}^2 = \sum_{l,u \in \{1,\ldots,n\}} \rho_l \rho_u T_{l,u}^{i,j}, \tag{3.28}$$

where $T_{l,u}^{i,j} = x_l y_u \mathbf{A}_{l,u}^{i,j}$.

Now the following inequality is used:

Theorem 3.3.2 (Hanson-Wright Inequality)**.** *Let $\mathbf{X} = (X_1, \ldots, X_n)^T \in \mathbb{R}^n$ be a random vector with independent components X_i which satisfy: $\mathbb{E}[X_i] = 0$ and have sub-Gaussian norm at most K for some given $K > 0$. Let $\mathbf{A}$ be an $n \times n$ matrix. Then for every $t \geq 0$ the following is true:*

$$\mathbb{P}[\mathbf{X}^T \mathbf{A} \mathbf{X} - \mathbb{E}[\mathbf{X}^T \mathbf{A} \mathbf{X}] > t] \leq 2e^{-c \min(\frac{t^2}{K^4 \|\mathbf{A}\|_F^2}, \frac{t}{K^2 \|\mathbf{A}\|_2})}, \tag{3.29}$$

where c is some universal positive constant.

Note that, assuming $\delta(n)$-balancedness, there is: $\|\mathbf{T}^{i,j}\|_F \leq \frac{\delta^2(n)}{n} \|\mathbf{A}^{i,j}\|_F$ and $\|\mathbf{T}^{i,j}\|_2 \leq \frac{\delta^2(n)}{n} \|\mathbf{A}^{i,j}\|_2$.

Now $\mathbf{X} = (\rho_1, \ldots, \rho_n)^T$ and $\mathbf{A} = \mathbf{T}^{i,j}$ are taken in the theorem above. Applying the Hanson-Wright inequality in that setting, taking the union bound over all pairs of different vectors $\mathbf{v}^i, \mathbf{v}^j \in \mathcal{A}$ (this number is exactly: $\binom{md}{2}$) and the event $\mathcal{E}_{balanced}$, finally taking the union bound over all s functions f_i, there is with probability at least:

$$p_{good} = 1 - p(n)ds - 2\binom{md}{2} s e^{-\Omega(\min(\frac{t^2 n^2}{K^4 \Lambda_F^2 \delta^4(n)}, \frac{tn}{K^2 \Lambda_2 \delta^2(n)}))} \tag{3.30}$$

for every f any two different vectors $\mathbf{v}^i, \mathbf{v}^j \in \mathcal{A}$ satisfy: $|(\mathbf{v}^i)^T \cdot \mathbf{v}^j| \leq t$.

Note that from the fact that $\mathbf{B}_2\mathbf{B}_1$ is $(\delta(n), p(n))$-balanced and from Equation 3.30, it holds with probability at least:

$$p_{right} = 1 - 2p(n)ds - 2\binom{md}{2}se^{-\Omega(\min(\frac{t^2n^2}{K^4\Lambda_F^2\delta^4(n)}, \frac{tn}{K^2\Lambda_2\delta^2(n)}))}. \tag{3.31}$$

for every f any two different vectors $\mathbf{v}^i, \mathbf{v}^j \in \mathcal{A}$ satisfy: $|(\mathbf{v}^i)^T \cdot \mathbf{v}^j| \leq t$ and furthermore each $\mathbf{v}^i$ is $\delta(n)$-balanced.

$\mathbf{B}_3$ application. Assume now that this event happens. Consider the vector

$$\mathbf{q}' = ((\mathbf{y}^1)^T, \ldots, (\mathbf{y}^d)^T)^T \in \mathbb{R}^{md}. \tag{3.32}$$

Note that $\mathbf{q}'$ can be equivalently represented as:

$$\mathbf{q}' = (\mathbf{r}^T \cdot \mathbf{v}^1, \ldots, \mathbf{r}^T \cdot \mathbf{v}^{md}), \tag{3.33}$$

where: $\mathcal{A} = \{\mathbf{v}^1, \ldots, \mathbf{v}^{md}\}$. From the fact that $\phi_i\mathbf{B}_2$ and $\mathbf{B}_1$ are isometries one can conclude that: $\|\mathbf{v}^i\|_2 = 1$ for $i = 1, \ldots, md$.

Now the following Berry-Esseen type result is needed for random vectors:

Theorem 3.3.3 (Bentkus [2003])**.** *Let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be independent vectors taken from $\mathbb{R}^k$ with common mean $\mathbb{E}[\mathbf{X}_i] = 0$. Let $\mathbf{S} = \mathbf{X}_1 + \ldots + \mathbf{X}_n$. Assume that the covariance operator $\mathbf{C}^2 = \mathrm{cov}(\mathbf{S})$ is invertible. Denote $\beta_i = \mathbb{E}[\|\mathbf{C}^{-1}\mathbf{X}_i\|_2^3]$ and $\beta = \beta_1 + \ldots + \beta_n$. Let $\mathcal{C}$ be the set of all convex subsets of $\mathbb{R}^k$. Denote $\Delta(\mathcal{C}) = \sup_{A\in\mathcal{C}} |\mathbb{P}[S \in A] - \mathbb{P}[Z \in A]|$, where Z is the multivariate Gaussian distribution with mean 0 and covariance operator $\mathbf{C}^2$. Then:*

$$\Delta(\mathcal{C}) \leq ck^{\frac{1}{4}}\beta \tag{3.34}$$

for some universal constant c.

Denote $\mathbf{X}_i = (r_iv_i^1, \ldots, r_iv_i^k)^T$ for $k = md$, $\mathbf{r} = (r_1, \ldots, r_n)^T$ and $\mathbf{v}^j = (v_1^j, \ldots, v_n^j)$. Note that $\mathbf{q}' = \mathbf{X}_1 + \ldots + \mathbf{X}_n$. Clearly there is: $\mathbb{E}[\mathbf{X}_i] = 0$ (the expectation is taken with respect to the random choice of $\mathbf{r}$). Furthermore, given the choices of $\mathbf{v}^1, \ldots, \mathbf{v}^k$, random vectors $\mathbf{X}_1, \ldots, \mathbf{X}_n$ are independent.

Let us calculate now the covariance matrix of $\mathbf{q}'$, $\Sigma_{\mathbf{q}'} := \mathbb{E}[(\mathbf{q}' - \mathbb{E}[\mathbf{q}'])(\mathbf{q}' - \mathbb{E}[\mathbf{q}'])^T] = \mathbf{E}[\mathbf{q}'\mathbf{q}'^T]$ since, because of definition of $\mathbf{r}$, $\mathbf{E}[\mathbf{q}'] = 0$. There is:

$$\mathbf{q}'_i = r_1v_1^i + \ldots + r_nv_n^i, \tag{3.35}$$

where $\mathbf{q}' = (\mathbf{q}'_1, \ldots, \mathbf{q}'_k)$.

Thus for i_1, i_2 there is:

$$(\Sigma_{\mathbf{q}'})_{i_1,i_2} = \mathbb{E}[\mathbf{q}'_{i_1}\mathbf{q}'_{i_2}] = \sum_{j=1}^{n} v_j^{i_1}v_j^{i_2}\mathbb{E}[r_j^2] + 2\sum_{1\leq j_1<j_2\leq n} v_{j_1}^{i_1}v_{j_2}^{i_2}\mathbb{E}[r_{j_1}r_{j_2}] = (\mathbf{v}^{i_1})^T \cdot \mathbf{v}^{i_2}, \tag{3.36}$$

where the last equation comes from the fact r_j are either Gaussian from $\mathcal{N}(0,1)$ or discrete with entries from $\{-1, +1\}$ and furthermore different r_js are independent.

Therefore if $i_1 = i_2 = i$, since each $\mathbf{v}^i$ has unit L_2-norm, it holds that

$$\mathbb{E}[\mathbf{q}'_i\mathbf{q}'_i] = 1, \tag{3.37}$$

and for $i_1 \neq i_2$ there is:

$$|\mathbb{E}[\mathbf{q}'_{i_1}\mathbf{q}'_{i_2}]| \leq t. \tag{3.38}$$

As a conclusion the covariance matrix $\Sigma_{\mathbf{q}'}$ of the distribution $\mathbf{q}'$ is a matrix with entries 1 on the diagonal and other entries of absolute value at most t.

For $t = o_k(1)$ small enough, from:

- the fact that $\mathbf{C}^{-1}$ is a constant,
- the $\delta(n)$-balancedness of vectors $\mathbf{v}^1, \ldots, \mathbf{v}^k$,
- and from the well-known fact that for any vector $\mathbf{z} \in \mathbb{R}^k$, $\|\mathbf{z}\|_2 \leq \sqrt{k}\|\mathbf{z}\|_\infty$,

one can conclude that:

$$\beta_i = \mathbb{E}[\|\mathbf{C}^{-1}\mathbf{X}_i\|_2^3] = O(\mathbb{E}[\|\mathbf{X}_i\|_2^3]) = O\left(\sqrt{\left(\frac{k}{n}\right)^3}\delta^3(n)\right), \tag{3.39}$$

Now, using Theorem 3.3.3, there is

$$\sup_{A \in \mathcal{C}} |\mathbb{P}[\mathbf{q}' \in A] - \mathbb{P}[Z \in A]| = O\left(k^{\frac{1}{4}} n \cdot \frac{k^{\frac{3}{2}}}{n^{\frac{3}{2}}}\delta^3(n)\right) = O\left(\frac{\delta^3(n)}{\sqrt{n}} k^{\frac{7}{4}}\right), \tag{3.40}$$

where:

- Z is taken from the multivariate Gaussian distribution with covariance matrix $\mathbf{I} + \mathbf{E}$,
- diagonal coefficients of $\mathbf{E}$ are equal to 0 and the absolute value of all its off-diagonal entries is at most ϵ,
- $\mathcal{C}$ is the set of all convex sets.

Now if the above inequality is applied to the pairwise disjoint convex sets $A_1, \ldots, A_j$, where $A_1 \cup \ldots \cup A_j = f_i^{-1}(\mathcal{S})$ and $l \leq b$ (such sets exist from the b-convexity of $f_i^{-1}(\mathcal{S})$), take $\eta = \frac{\delta^3(n)}{\sqrt{n}} k^{\frac{7}{4}}$, $\epsilon = t = o_{md}(1)$ and take n large enough, the statement of the theorem follows, recall that:

- $\mathbb{P}[\mathbf{q}' \in A] = \mathbb{P}[\mathbf{q}' \in f_i^{-1}(\mathcal{S})] = \mathbb{P}[f_i(\mathbf{q}') \in \mathcal{S}]$,
- $k = md$,
- we can take $\mathbb{P}[Z \in A] = \mathbb{P}[f'_i(\mathbf{q}_{f'_i}) \in \mathcal{S}]$.

□

Remark 3.3.1. *Theorem 3.3.1 does not require any strong regularity conditions regarding f_is (such as differentiability or even continuity). In practice, b is often a small constant. For instance, for the angular kernel approximation where f_is are non-continuous and for $\mathcal{S}$-singletons, one can take $b = 1$ (see Appendix C).*

Now let us think of f_i and f'_i as random variables, where randomness is generated by vectors $\mathbf{q}_{f_i}$ and $\mathbf{q}_{f'_i}$ respectively. Then, from Theorem 3.3.1, there is:

Theorem 3.3.4. *Denote by F_X the cdf of the random variable X and by ϕ_X its characteristic function. If f_i is convex or concave in respect to q_{f_i}, then for every t*

the following holds: $|F_{f_i}(t) - F_{f_i'}(t)| = O\left(\frac{\delta^3(n)}{n^{\frac{2}{5}}}\right)$. *Furthermore, if* f_i *is bounded then:* $|\phi_{f_i}(t) - \phi_{f_i'}(t)| = O\left(\frac{\delta^3(n)}{n^{\frac{2}{5}}}\right)$.

Proof. Let us assume that f_i is a convex function of $\mathbf{q}_{f_i}$ (if f_i is concave then the proof is completely analogous). For any $t \in \mathbb{R}$ let $\mathcal{S}_t = \{\mathbf{q}_{f_i} : f_i(\mathbf{q}_{f_i}) \leq t\}$ for f_i and $\mathcal{S}_t = \{\mathbf{q}_{f_i'} : f_i'(\mathbf{q}_{f_i'}) \leq t\}$ for f_i'. From the convexity assumption it holds that $\mathcal{S}_t$ is a convex set. Thus Theorem 3.3.1 can be directly applied and the result regarding cdf functions follows. To obtain the result regarding the characteristic functions, notice first that there is:

$$\phi_X(t) := \int_{-1}^{1} \mathbb{P}[\cos(tX) > s]ds + i\int_{-1}^{1} \mathbb{P}[\sin(tX) > s]ds \tag{3.41}$$

The event $\{\cos(tX) > s\}$ for $t \neq 0$ is equivalent to: $X \in \cup_{I \in \mathcal{I}} I$ for some family of intervals $\mathcal{I}$ thanks to the periodicity of cos function. Similar observation is true for the event $\{\sin(tX) > s\}$.

In our scenario, from the fact that f_i is bounded, the corresponding families $\mathcal{I}$ are finite. Furthermore, the probability of belonging to a particular interval can be expressed by the values of the cdf function in the endpoints of that interval. From this observation and the result on cdfs that has been just obtained, the result for the characteristic functions follows immediately. □

Theorem 3.3.1 implies strong accuracy guarantees for the specific structured spinners. As a corollary there is:

Theorem 3.3.5. *Under assumptions from Theorem 3.3.1 the probability* p_{succ} *from Theorem 3.3.1 reduces to:* $1 - 4ne^{-\frac{\log^2(n)}{8}}d - 2\binom{md}{2}e^{-\Omega(\frac{\epsilon^2 n}{\log^4(n)})}$ *for the structured matrices* $\sqrt{n}\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$, $\sqrt{n}\mathbf{HD}_{g_1,\ldots,g_n}\mathbf{HD}_2\mathbf{HD}_1$ *as well as for the structured matrices of the form* $\mathbf{G}_{struct}\mathbf{D}_2\mathbf{HD}_1$, *where* $\mathbf{G}_{struct}$ *is Gaussian circulant, Gaussian Toeplitz or Gaussian Hankel matrix.*

Proof. This comes directly from Theorem 3.3.1 and Lemma 3.2.2. □

As a corollary of Theorem 3.3.5, the following result is obtained showing the effectiveness of the Cross-polytope LSH with structured matrices $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ that was only heuristically confirmed before Andoni et al. [2015b].

Theorem 3.3.6. *Let* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ *be two unit* L_2*-norm vectors. Let* $\mathbf{v}_{\mathbf{x},\mathbf{y}}$ *be the vector indexed by all* $(2m)^2$ *ordered pairs of canonical directions* $(\pm\mathbf{e}_i, \pm\mathbf{e}_j)$, *where the value of the entry indexed by* $(\mathbf{u}, \mathbf{w})$ *is the probability that:* $h(\mathbf{x}) = \mathbf{u}$ *and* $h(\mathbf{y}) = \mathbf{w}$, *and* $h(\mathbf{v})$ *stands for the hash of* $\mathbf{v}$. *Then with probability at least:*

$$p_{success} = 1 - 8ne^{-\frac{\log^2(n)}{8}} - 2\binom{2m}{2}e^{-\Omega\left(\frac{\epsilon^2 n}{\log^4(n)}\right)} \tag{3.42}$$

the version of the stochastic vector $\mathbf{v}^1_{\mathbf{x},\mathbf{y}}$ *for the unstructured Gaussian matrix* $\mathbf{G}$ *and its structured counterpart* $\mathbf{v}^2_{\mathbf{x},\mathbf{y}}$ *for the matrix* $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ *satisfy:*

$$\|\mathbf{v}^1_{\mathbf{x},\mathbf{y}} - \mathbf{v}^2_{\mathbf{x},\mathbf{y}}\|_\infty \leq \log^3(n)n^{-\frac{2}{5}} + c\epsilon, \tag{3.43}$$

for n large enough, where $c > 0$ is a universal constant. The probability above is taken with respect to random choices of $\mathbf{D}_1$ and $\mathbf{D}_2$.

For angles between $\mathbf{v}^1_{\mathbf{x},\mathbf{y}}$ and $\mathbf{v}^2_{\mathbf{x},\mathbf{y}}$ in the range $[0, \frac{\pi}{3}]$ the result above leads to the same asymptotics of the probabilities of collisions as these in Theorem 1 of Andoni et al. [2015b] given for the unstructured Cross-polytope LSH (see Theorem 2.2.1 in Section 2.2.2, p. 21).

The proof for the discrete structured setting applies Berry-Esseen-type results for random vectors showing that for n large enough ± 1 random vectors $\mathbf{r}$ act similarly to Gaussian vectors.

Proof. For clarity the structured matrix is assumed to consist of just one block of m rows and we will compare its performance with the unstructured variant of m rows (the more general case when the structured matrix is obtained by stacking vertically many blocks is analogous since the blocks are chosen independently).

Consider the two-dimensional linear space $\mathcal{H}$ spanned by $\mathbf{x}$ and $\mathbf{y}$. Fix some orthonormal basis $\mathcal{B} = \{\mathbf{u}^1, \mathbf{u}^2\}$ of $\mathcal{H}$. Take vectors $\mathbf{q}$ and $\mathbf{q}'$. Note that they are $2m$-dimensional, where m is the number of rows of the block used in the structured setting. From Theorem 3.3.5 with probability at least $p_{success}$, where $p_{success}$ is as in the statement of the theorem, the following holds for any convex $2m$-dimensional set A:

$$|\mathbb{P}[\mathbf{q}(\epsilon) \in A] - \mathbb{P}[\mathbf{q}' \in A]| \leq \eta, \tag{3.44}$$

where $\eta = \frac{\log^3(n)}{n^{\frac{2}{5}}}$. Take two corresponding entries of vectors $\mathbf{v}^1_{\mathbf{x},\mathbf{y}}$ and $\mathbf{v}^2_{\mathbf{x},\mathbf{y}}$ indexed by a pair $(\mathbf{e}_i, \mathbf{e}_j)$ for some fixed $i, j \in \{1, \ldots, m\}$ (for the case when the pair is not of the form $(\mathbf{e}, \mathbf{e}_j)$, but of a general form: $(\pm\mathbf{e}_i, \pm\mathbf{e}_j)$ the analysis is exactly the same). Call them p^1 and p^2 respectively. The goal is to compute $|p^1 - p^2|$. Notice that p^1 is the probability that $h(\mathbf{x}) = \mathbf{e}_i$ and $h(\mathbf{y}) = \mathbf{e}_j$ for the unstructured setting and p^2 is that probability for the structured variant.

Let us consider now the event $E^1 = \{h(\mathbf{x}) = \mathbf{e}_i \wedge h(\mathbf{y}) = \mathbf{e}_j\}$, where the setting is unstructured. Denote the corresponding event for the structured setting as E^2. Denote $\mathbf{q} = (q_1, \ldots, q_{2m})$. Assume that $\mathbf{x} = \alpha_1\mathbf{u}^1 + \alpha_2\mathbf{u}^2$ for some scalars $\alpha_1, \alpha_2 > 0$. Denote the unstructured Gaussian matrix by $\mathbf{G}$. There is:

$$\mathbf{G}\mathbf{x} = \alpha_1\mathbf{G}\mathbf{u}^1 + \alpha_2\mathbf{G}\mathbf{u}^2 \tag{3.45}$$

Note that there is: $\mathbf{G}\mathbf{u}^1 = (q_1, \ldots, q_m)^T$ and $\mathbf{G}\mathbf{u}^2 = (q_{m+1}, \ldots, q_{2m})^T$. Denote by $A(\mathbf{e}_i)$ the set of all the points in $\mathbb{R}^m$ such that their angular distance to $\mathbf{e}_i$ is at most the angular distance to all other $m-1$ canonical vectors. This set can be interpreted as the set of all points that are nearer (in the sense of the angular distance) from $\mathbf{e}_i$ than for all $\mathbf{e}_j$ for $j \in \{1, \ldots, m\}$, $j \neq i$. Note that this is definitely a convex set. Now denote:

$$Q(\mathbf{e}_i) = \{(q_1, \ldots, q_{2m})^T \in \mathbb{R}^{2m} : \alpha_1(q_1, \ldots, q_m)^T + \alpha_2(q_{m+1}, \ldots, q_{2m})^T \in A(\mathbf{e}_i)\}. \tag{3.46}$$

Note that since $A(\mathbf{e}_i)$ is convex, one can conclude that $Q(\mathbf{e}_i)$ is also convex. Note that

$$\{h(\mathbf{x}) = \mathbf{e}_i\} = \{\mathbf{q} \in Q(\mathbf{e}_i)\}. \tag{3.47}$$

By repeating the analysis for the event $\{h(\mathbf{y}) = \mathbf{e}_j\}$,

$$\{h(\mathbf{x}) = \mathbf{e}_i \wedge h(\mathbf{y}) = \mathbf{e}_j\} = \{\mathbf{q} \in Y(\mathbf{e}_i, \mathbf{e}_j)\} \tag{3.48}$$

for convex set $Y(\mathbf{e}_i, \mathbf{e}_j) = Q(\mathbf{e}_i) \cap Q(\mathbf{e}_j)$. Now observe that

$$|p^1 - p^2| = |\mathbb{P}[\mathbf{q} \in Y(\mathbf{e}_i, \mathbf{e}_j)] - \mathbb{P}[\mathbf{q}' \in Y(\mathbf{e}_i, \mathbf{e}_j)]| \tag{3.49}$$

Thus there is by triangle inequality:

$$\begin{aligned} |p^1 - p^2| \leq |\mathbb{P}[\mathbf{q} \in Y(\mathbf{e}_i, \mathbf{e}_j)] - \mathbb{P}[\mathbf{q}(\epsilon) \in Y(\mathbf{e}_i, \mathbf{e}_j)]| \\ + |\mathbb{P}[\mathbf{q}(\epsilon) \in Y(\mathbf{e}_i, \mathbf{e}_j)] - \mathbb{P}[\mathbf{q}' \in Y(\mathbf{e}_i, \mathbf{e}_j)]| \end{aligned} \tag{3.50}$$

Therefore there is by application of Theorem 3.3.5:

$$|p^1 - p^2| \leq |\mathbb{P}[\mathbf{q} \in Y(\mathbf{e}_i, \mathbf{e}_j)] - \mathbb{P}[\mathbf{q}(\epsilon) \in Y(\mathbf{e}_i, \mathbf{e}_j)]| + \eta. \tag{3.51}$$

Thus it is just needed to upper-bound:

$$\xi := |\mathbb{P}[\mathbf{q} \in Y(\mathbf{e}_i, \mathbf{e}_j)] - \mathbb{P}[\mathbf{q}(\epsilon) \in Y(\mathbf{e}_i, \mathbf{e}_j)]|. \tag{3.52}$$

Denote the covariance matrix of the distribution $\mathbf{q}(\epsilon)$ as $\mathbf{I} + \mathbf{E}$. Note that $\mathbf{E}$ is equal to 0 on the diagonal and the absolute value of all other off-diagonal entries is at most ϵ.

Denote $k = 2m$. There is $\xi = |A - B|$, where

$$A = \frac{1}{(2\pi)^{\frac{k}{2}}\sqrt{\det(I+E)}} \int_{Y(\mathbf{e}_i, \mathbf{e}_j)} e^{-\frac{\mathbf{x}^T(\mathbf{I}+\mathbf{E})^{-1}\mathbf{x}}{2}} d\mathbf{x} \tag{3.53}$$

and

$$B = \frac{1}{(2\pi)^{\frac{k}{2}}} \int_{Y(\mathbf{e}_i, \mathbf{e}_j)} e^{-\frac{\mathbf{x}^T\mathbf{x}}{2}} d\mathbf{x}. \tag{3.54}$$

Expanding: $(\mathbf{I}+\mathbf{E})^{-1} = \mathbf{I} - \mathbf{E} + \mathbf{E}^2 - \ldots$, noticing that $|\det(I+E) - 1| = O(\epsilon^{2m})$ by definition of the determinant as a sum over the set of permutations, and using the above formula, it is easily obtained that:

$$\xi = O(\epsilon). \tag{3.55}$$

That completes the proof. □

3.4 Experiments with Locality-Sensitive Hashing (LSH)

In this Section, the *Structured spinners* are applied to Locality-Sensitive Hashing (LSH)[8].

3.4.1 Experimental setup

Experiments were conducted using Python. In particular, NumPy is linked against a highly optimized BLAS library (Intel MKL). Fast Fourier Transform is performed using numpy.fft and Fast Hadamard Transform is using ffht from Andoni et al. [2015b].

[8] Experiments for kernel approximations, Newton sketches and neural networks can be found in Appendix C.

To have a fair comparison, it has been set up: OMP_NUM_THREADS = 1 so that every experiment is done on a single thread. Every parameter of the structured spinner matrix is computed in advance, such that obtained speedups take only matrix-vector products into account. All figures should be read in color.

3.4.2 Collision probabilities with Cross-polytope LSH

In this experiment, state-of-the-art Cross-polytope LSH is considered.

Experimental protocol

In Figure 3.2, collision probabilities are compared for the low dimensional case ($n = 256$, $m = 64$), where for each interval, the collision probability has been computed for 20000 points. Results are shown for one hash function (averaged over 100 runs) and are reported for a random 256×64 Gaussian matrix $\mathbf{G}$ and five other types of matrices from a family of structured spinners (descending order of the number of parameters):

- $\mathbf{G}_{circ}\mathbf{K}_2\mathbf{K}_1$,
- $\mathbf{G}_{Toeplitz}\mathbf{D}_2\mathbf{HD}_1$,
- $\mathbf{G}_{skew-circ}\mathbf{D}_2\mathbf{HD}_1$,
- $\mathbf{HD}_{g_1,\ldots,g_n}\mathbf{HD}_2\mathbf{HD}_1$,
- and $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$,

where $\mathbf{K}_i$, $\mathbf{G}_{Toeplitz}$, and $\mathbf{G}_{skew-circ}$ are respectively a Kronecker matrix with discrete entries, Gaussian Toeplitz and Gaussian skew-circulant matrices.

Results

All matrices from the family of structured spinners show high collision probabilities for small distances and low ones for large distances. As theoretically predicted, structured spinners do not lead to accuracy losses. All considered matrices give almost identical results.

3.5 Conclusion

This Chapter introduced a general structured paradigm for large scale machine learning computations with random matrices, providing computational speedups and storage compression. The application considered in this Chapter is in particularly the approximate nearest neighbors search with LSH method. This framework is also accompanied by theoretical guarantees on the effectiveness of the structured approach.

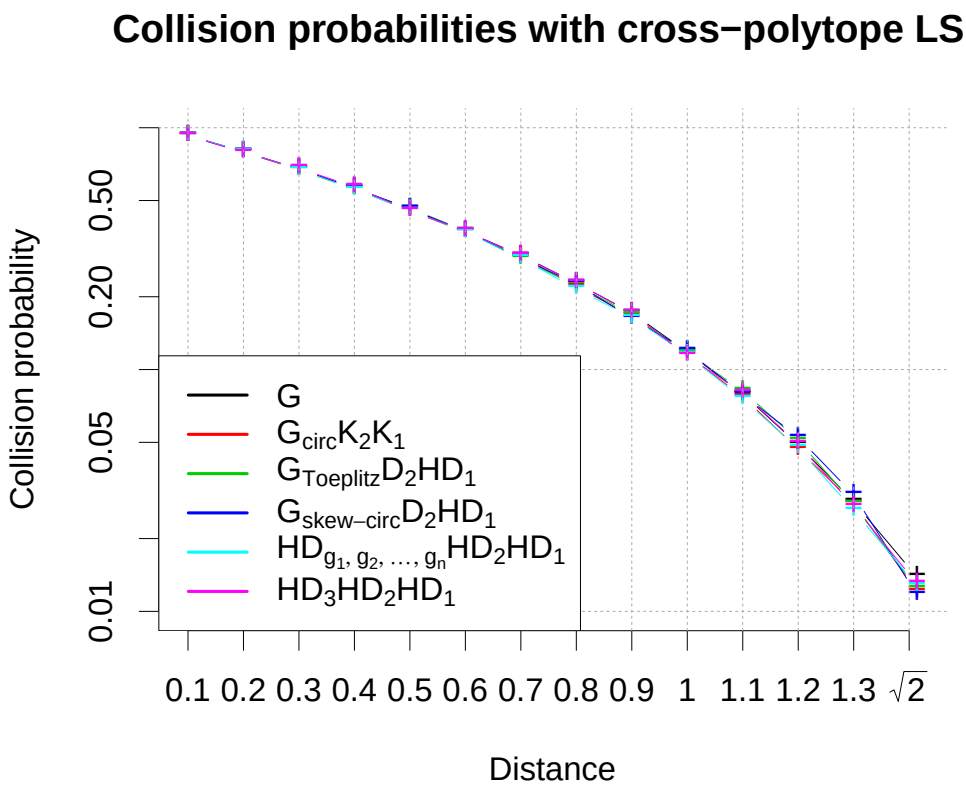
> This work brings the first theoretical guarantees for the fastest known Cross-polytope LSH [Andoni et al., 2015b] based on the $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ structured matrix.

However this framework and the theoretical guarantees do not only apply to general dimensionality reduction algorithms and LSH-based algorithms but also to various other applications:

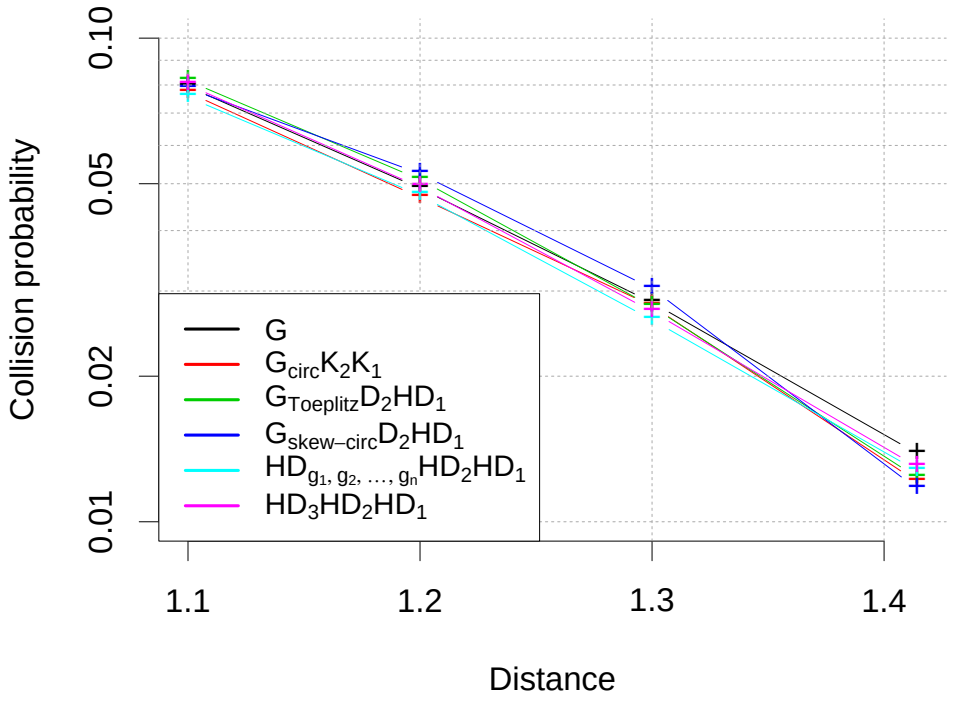
- quantization with random projection trees,

- kernel approximations via random feature maps,
- convex optimization via Newton sketches,
- deep learning, etc.

In Appendix C, the last three points are addressed. A remaining open question is: Can one obtain computation speedups for matrices from the *Structured spinners* model for which the Fast Fourier Transform trick does not work ?



(a)



(b)

FIGURE 3.2: Cross-polytope LSH - collision probabilities for distance between 0 and $\sqrt{2}$ with $n = 256$, $m = 64$ (a). (b) A zoom on higher distances enables to distinguish the curves which are almost superposed.

Chapter 4

Learning compact binary codes from massive data streams with Hypercubic hashing for Nearest Neighbors search

Contents

This Chapter concerns a collaboration with Antoine Souloumiac[1] *and Krzysztof Choromanski*[2]*. A first version of this work has been published at IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2018 under the title "Streaming Binary Sketching based on Subspace Tracking and Diagonal Uniformization".*

4.1 Introduction

Nearest neighbors (NN) search is a key task involved in many machine learning applications such as classification or clustering. For large-scale datasets e.g. in computer vision or metagenomics, indexing efficiently high-dimensional data becomes necessary

[1] CEA
[2] Google Brain Robotics

for reducing space needs and speeding up similarity search. This can be classically achieved by hashing techniques which map data onto lower-dimensional representations.

We recall from Chapter 2 that two hashing paradigms exist: data-independent and data-dependent hashing methods. On the one hand, Locality-Sensitive Hashing (LSH) [Andoni and Indyk, 2008] and its variants [Terasawa and Tanaka, 2007, Andoni et al., 2015b, Yu et al., 2014] belong to the data-independent paradigm. They rely on some random projection onto a c-lower dimensional space followed by a scalar quantization returning the nearest vertex from the set $\{-1, 1\}^c$ for getting the binary codes (e.g. the sign function is applied point-wise). On the other hand, data-dependent methods [Wang et al., 2018] learn this projection from data instead and have been found to be more accurate for computing similarity-preserving binary codes. Among them, the unsupervised data-dependent Hypercubic hashing methods, embodied by ITerative Quantization (ITQ) [Gong et al., 2013], use Principal Component Analysis (PCA) to reduce data dimensionality to c: the data is projected onto the first c principal components chosen as the ones with the highest explained variance as they carry more information on variability. If we then directly mapped each resulting direction to one bit, each of them would get represented by the same volume of binary code (1 bit), although the c^{th} direction should carry less information than the first one. Thus, one can intuitively understand why PCA projection application solely leads to poor performance of the obtained binary codes in the NN search task. This is why data get often mixed though an isometry after PCA-projection so as to balance variance over the kept directions. See Figure 4.1, a recall from Section 2.3.2, for the general scheme of Hypercubic hashing methods.

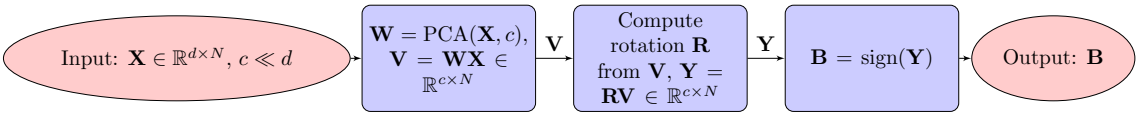


FIGURE 4.1: General (offline) scheme for Hypercubic hashing methods with notation from Section 4.2. The first step consists in computing the $c \times d$-projection matrix with PCA. The second rotates the PCA-projected data. Finally, the sign function is applied pointwise to obtain binary coefficients.

Works from Jégou et al. [2010] and Chen et al. [2017] use a random rotation[3] while the rotation can also be learned [Gong et al., 2013, Kong and Li, 2012] to that purpose. This has led to the development of variant online techniques such as Online Sketching Hashing (OSH) [Leng et al., 2015a] or FasteR Online Hashing (FROSH) [Chen et al., 2017] which are deployable in the streaming setting when large high-dimensional data should be processed with limited memory. These are currently the state-of-the-art of online unsupervised hashing methods for learning on the fly similarity-preserving binary embeddings.

Nevertheless, even if one can now use these unsupervised online methods for processing high-dimensional streams of data, there is still no theoretical justification that equalizing the variance or in other words choosing directions with isotropic

[3] In the sequel, the terms orthogonal matrix and rotation are used equivalently.

variance leads to optimal results.

Contributions In this Chapter, the contributions are two-fold:

1. First, theoretical guarantees to several state of the art quantization-based hashing techniques are brought, by formally proving the need for a rotation when the methods rely on PCA-like preprocessing.

2. Second, a novel streaming algorithm with convergence guarantees is introduced where the data are seen only once and the principal subspace plus the balancing rotation are updated as new data are seen. To obtain the principal subspace and the rotation, this requires additionally only the storage of two $c \times c$ matrices, instead of the whole initial and projected datasets as for ITQ or IsoHash. Our algorithm outperforms the known state-of-the-art online unsupervised method Online Sketching Hashing (OSH) Leng et al. [2015a] while being far less computationally demanding.

We also introduce experiments accompanying the theoretical results and compare the existing online unsupervised hypercubic quantization-based hashing methods.

Plan of the Chapter For the online setting, the problem of finding compact binary codes is stated in Section 4.2. This Section also describes the offline inspirations of the new proposed algorithm from the family of Hypercubic hashing. Before digging into the details of this novel online binary hashing method in Section 4.4, Section 4.3 brings theoretical elements justifying the design of our method. Experiments show the usefulness of the technique in Section 4.5.1 and Section 4.6 concludes this Chapter.

4.2 Preliminaries

Recall that for any matrix $\mathbf{M}$, $\mathbf{\Sigma_M} = \mathbf{M}\mathbf{M}^T$[4]. For any vector $\mathbf{z}$, $\mathbf{z}^{(i)}$ denotes its i^{th} entry. Let be a stream of N zero-centered data points $\{\mathbf{x}_t \in \mathbb{R}^d\}_{1 \leq t \leq N}$. The considered hashing methods aim at obtaining the binary codes, for $t \in [N]$:

$$\mathbf{b}_t = \text{sign}(\tilde{\mathbf{W}}\mathbf{x}_t) \in \{-1, 1\}^c \tag{4.1}$$

where c denotes the code length, $c \ll d$, and $\tilde{\mathbf{W}} \in \mathbb{R}^{c \times d}$ is the dimensionality reduction operator. In other words, for each bit $k \in [c]$, the hashing function is defined as

$$h_k(\mathbf{x}_t) = \text{sign}(\tilde{\mathbf{w}}_k^T \mathbf{x}_t) \tag{4.2}$$

where $\tilde{\mathbf{w}}_k$ are column vectors of hyperplane coefficients. So $\tilde{\mathbf{w}}_k^T$ is a row of $\tilde{\mathbf{W}} \in \mathbb{R}^{c \times d}$ for each k. In the framework of Hypercubic quantization hashing functions,

$$\tilde{\mathbf{W}} = \mathbf{R}\mathbf{W} \tag{4.3}$$

where $\mathbf{W}$ is the linear dimensionality reduction embedding applied to data and $\mathbf{R}$ is a suitable $c \times c$ orthogonal matrix. Typically, one can take $\mathbf{W}$ as the matrix whose row vectors $\mathbf{w}_k^T$ are the c first principal components of the covariance matrix $\mathbf{\Sigma_X}$ where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{d \times N}$. Note that this problem statement includes offline and online methods: $\mathbf{W}$ can be either the PCA or a tracked principal subspace as a new data point is seen. What is more specific to the methods is the way of learning

[4]This is a slight abuse of notation as the normalization factor is discarded. See Eq. 2.28 p.27.

the appropriate orthogonal matrix for rotating the data which have been previously projected onto this principal subspace.

After application of the (possibly online estimated) PCA algorithm, the stream becomes $\{\mathbf{v}_t \in \mathbb{R}^c\}_{1\leq t\leq N}$ s.t.

$$\mathbf{v}_t = \mathbf{W}\mathbf{x}_t. \tag{4.4}$$

Then, for all $t \in [N]$,

$$\mathbf{y}_t = \mathbf{R}\mathbf{v}_t \tag{4.5}$$

and

$$\mathbf{b}_t = \text{sign}(\mathbf{y}_t). \tag{4.6}$$

In the sequel, depending on the context, we will be considering either one data point $\mathbf{x}_t$, its initial projection $\mathbf{v}_t = \mathbf{W}\mathbf{x}_t$, its rotated projection $\mathbf{y}_t$ and binary sketch $\mathbf{b}_t$, or the whole associated sets $\mathbf{X}$, $\mathbf{V} = \mathbf{W}\mathbf{X}$, $\mathbf{Y} = \mathbf{R}\mathbf{V}$ and $\mathbf{B} = \text{sign}(\mathbf{Y}) = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \{-1, 1\}^{c\times N}$.

ITerative Quantization (ITQ) For ITQ [Gong et al., 2013], as previously described in Section 2.3.2, $\mathbf{R}$ is the solution of an orthogonal Procustes problem which consists in minimizing the quantization error $Q(\mathbf{B}, \mathbf{R})$ of mapping the resulting data to the vertices of the 2^c hypercube:

$$Q(\mathbf{B}, \mathbf{R}) = \|\mathbf{B} - \tilde{\mathbf{W}}\mathbf{X}\|_F^2 = \|\mathbf{B} - \mathbf{R}\mathbf{W}\mathbf{X}\|_F^2 = \|\mathbf{B} - \mathbf{R}\mathbf{V}\|_F^2. \tag{4.7}$$

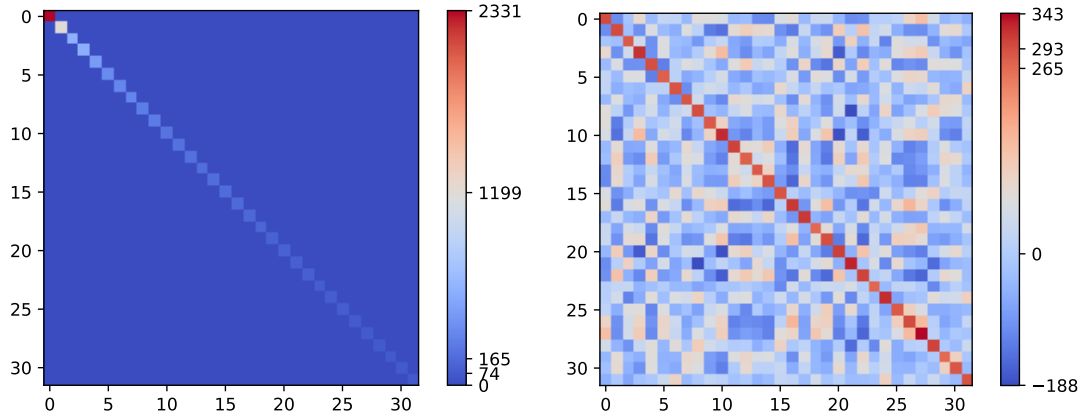
This is currently the best method among Hypercubic quantization techniques. Its drawback is it is a full offline process and there is no convergence guarantees for obtaining $\mathbf{R}$.

Our goal is to propose an online version of the algorithm. Since the quantization error minimization from Equation 4.7 in the online setting is not an obvious task, we looked for a by-product effect of ITQ, hopefully easier to reproduce. Figure 4.2 shows the covariance matrix of CIFAR dataset after PCA and ITQ projection before the sign application. It can be observed that ITQ tends to uniformize the diagonal coefficients of this matrix. Intuitively, we can see on Figure 4.3 that it makes sense. Consequently, an interesting idea would be to determine a rotation which directly equalizes the diagonal coefficient of the projected data covariance. This is basically the principle of IsoHash algorithm [Kong and Li, 2012] explained before in Section 2.3.2. Our proposed online algorithm shares also the same objective function but our rotation solution is easier to compute and is integrated in a full online algorithm (see Section 4.4).

Isotropic Hashing (IsoHash) Let us again consider $\sigma_1^2, ..., \sigma_c^2$ the diagonal coefficients of $\mathbf{\Sigma_V}$, hence $\sigma_1^2 \geq ... \geq \sigma_c^2$. IsoHash [Kong and Li, 2012] but also our algorithm UnifDiag (see Section 4.4) looks for a matrix $\mathbf{R}$ balancing the variance over the c directions, i.e. equalizing the diagonal coefficients of $\mathbf{\Sigma_Y}$ to the same value

$$\tau = \text{Tr}(\mathbf{\Sigma_V})/c. \tag{4.8}$$

We recall from Section 2.3.2 that IsoHash, determines the rotation with a gradient descent converging to the intersection between the set of orthogonal matrices and the set of transfer matrices making $\mathbf{\Sigma_Y}$ diagonal.



(a) PCA (b) ITQ

FIGURE 4.2: Covariance matrix of projected data from CIFAR dataset with $c = 32$.

4.3 Theoretical justification of optimality of a rotation R for Hypercubic quantization hashing

Assuming some distribution on data, we deliver in this Section two results on the theoretical justification of applying **R** after PCA projection in terms of efficiency of the binary codes:

1. We prove the optimality of choosing **R** as a rotation uniformizing the diagonal of the covariance matrix.

2. We provide some lower bound on the probability of getting different binary codes for two data points initially close to each other.

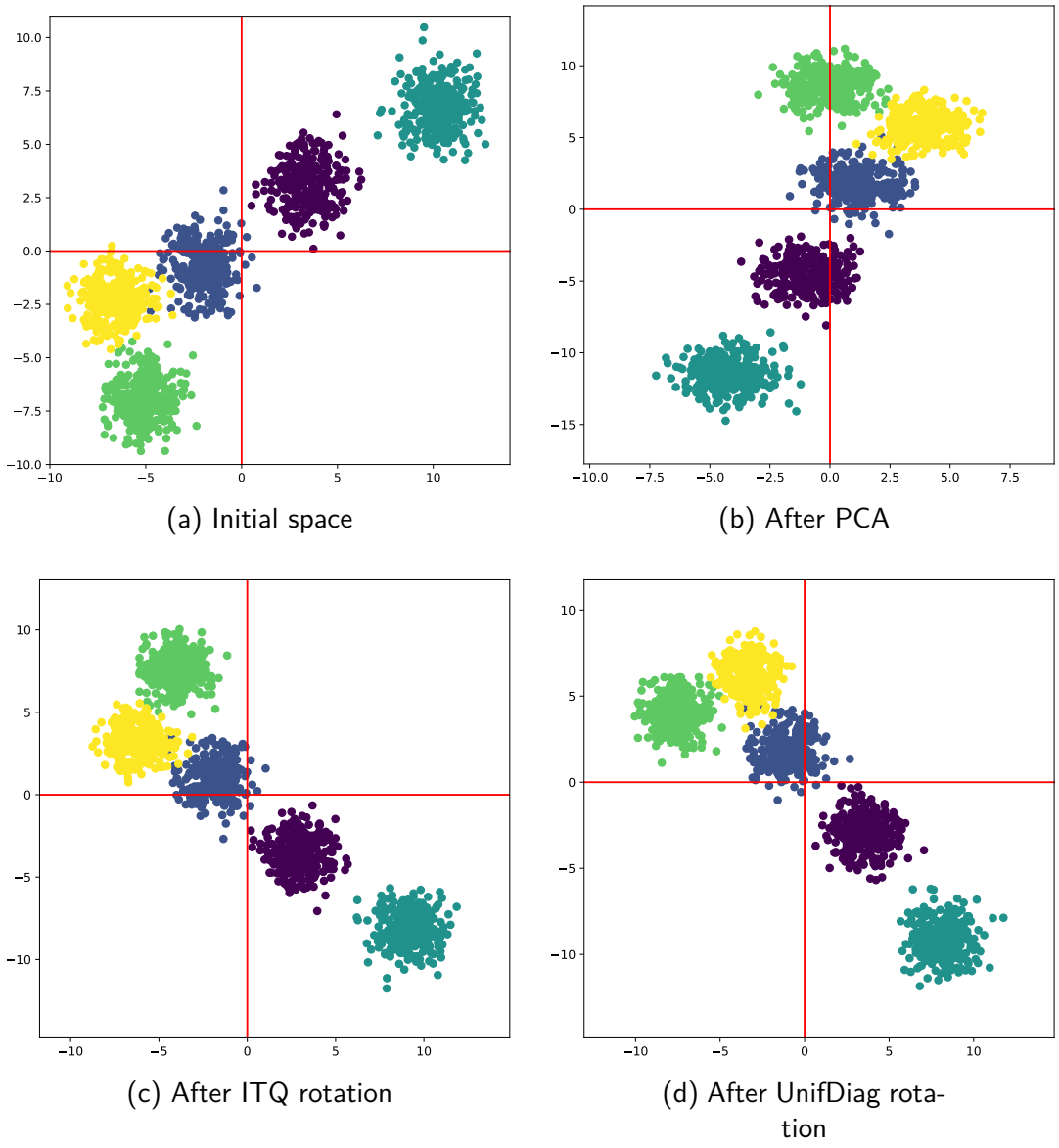
In Appendix D, we also propose an extension to the case with no assumption on data distribution but some on matrix **R**: when matrix **R** is random. It corresponds to the case of Online Sketching Hashing (OSH) algorithm [Leng et al., 2015a]. Given an upper bound on the distance between two data points in the initial space, we give a lower bound on the number of bits in common in their binary codes.

We assume here $\mathbf{R} \in \mathbb{R}^{c\times c}$ is a rotation[5] matrix ($\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I}_c$). Moreover, we make another assumption in the form of Hypothesis 4.3.1:

Hypothesis 4.3.1 (H1). *We assume:* $\forall t \in [N]$, $(\mathbf{v}_t^{(1)}, \mathbf{v}_t^{(2)}, \ldots, \mathbf{v}_t^{(c)})^T \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathbf{V}}^{th})$ *s.t. diagonal coefficients of* $\mathbf{\Sigma}_{\mathbf{V}}^{th}$ *are* $({\sigma_1^{th}}^2, \ldots, {\sigma_c^{th}}^2)$. *In particular,* $\forall t \in [N], \forall i \in [c]$, $\mathbf{v}_t^{(i)} \sim \mathcal{N}(0, {\sigma_i^{th}}^2)$. $\mathbf{\Sigma}_{\mathbf{V}}^{th}$ *is not necessary diagonal.*

Before introducing Theorem 4.3.1, note that two neighboring data points can have very dissimilar binary codes if their projections on PCA have many coefficients near zero, not on the same side of the hyperplanes delimiting the orthants. Indeed, in this case, the sign function will attribute opposite bits (see again Figure 4.3 for an illustration).

[5] Again, this is a slight abuse of notation since we do not need a determinant equal to 1.



(a) Initial space

(b) After PCA

(c) After ITQ rotation

(d) After UnifDiag rotation

FIGURE 4.3: Effect of the rotation. (a) Five random blobs/clusters have been drawn in 2D from some Gaussian distributions (we call them respectively the yellow, the light green, the blue, the violet and the emerald clusters). (b) They have been projected onto the $c = 2$ principal components. We see that the light green, the blue and the violet clusters are crossing the horizontal axis after PCA-projection: their points are spread accross two (or more for the blue cluster) orthants. Hence, some points of the *same* cluster will have *different* binary codes since the sign function of a vector is determined by from which sides of the hyperplanes delimiting the orthants the vector is. An idea would be to rotate the data so that to minimize the number of points estranged to the main ones in their cluster. (c) The effect of the ITQ rotation. (d) The effect of the UnifDiag rotation. On this kind of data, the best idea is indeed to set the clusters in the center of orthants, i.e. to equalize the variance on both axes.

Therefore, after the dimensionality reduction of the data points in the original space, a good hashing method tends to keep the coefficients of the projected data away from zero, i.e. away from the hyperplanes delimiting the orthants. The challenge is then to determine how to move these data points away from the hyperplanes delimiting the orthants.

Theorem 4.3.1. *Assume $\{\mathbf{x}_t \in \mathbb{R}^d\}_{1\leq t\leq N}$ is a stream of N zero-centered data points following Hypothesis 4.3.1. Then, choosing $\mathbf{R}$ so that it uniformizes the diagonal of the covariance matrix $\mathbf{\Sigma}_{\mathbf{Y}}^{th}$ is equivalent to minimizing some upper bound on the probability that the data points are close to an hyperplane delimiting an orthant.*

Proof. Let $\epsilon > 0$. For $i \in [c]$, $t \in [N]$, let p_i^ϵ be the probability (independent of t) that $\mathbf{y}_t^{(i)} = (\mathbf{R}\mathbf{v}_t)^{(i)}$ is closer than ϵ from the orthant:

$$p_i^\epsilon = \mathbb{P}[\ |\mathbf{y}_t^{(i)}| < \epsilon\ ] = \int_{-\epsilon}^{\epsilon} \frac{1}{\sqrt{2\pi(\mathbf{R}\mathbf{\Sigma}_{\mathbf{V}}^{th}\mathbf{R}^T)_{ii}}} e^{\frac{-s^2}{2(\mathbf{R}\mathbf{\Sigma}_{\mathbf{V}}^{th}\mathbf{R}^T)_{ii}}} ds \tag{4.9}$$

$$= \frac{2\epsilon}{\sqrt{2\pi(\mathbf{R}\mathbf{\Sigma}_{\mathbf{V}}^{th}\mathbf{R}^T)_{ii}}} + \underbrace{o(\epsilon^2)}_{\leq 0} \leq \frac{2\epsilon}{\sqrt{2\pi(\mathbf{R}\mathbf{\Sigma}_{\mathbf{V}}^{th}\mathbf{R}^T)_{ii}}}. \tag{4.10}$$

Hence,

$$\min_{\mathbf{R}} \mathbb{P}[\bigcup_{i\in[c]} \left(|\mathbf{y}_t^{(i)}| < \epsilon\right)] \leq \min_{\mathbf{R}} \sum_{i\in[c]} p_i^\epsilon \tag{4.11}$$

$$\leq \min_{\mathbf{R}} \left(\frac{2\epsilon}{\sqrt{2\pi}} \sum_{i\in[c]} \frac{1}{\sqrt{(\mathbf{R}\mathbf{\Sigma}_{\mathbf{V}}^{th}\mathbf{R}^T)_{ii}}} \right). \tag{4.12}$$

Now, let us define:

$$\mathbf{R}^* \in \operatorname*{argmin}_{\mathbf{R}} \left(\frac{2\epsilon}{\sqrt{2\pi}} \sum_{i\in[c]} \frac{1}{\sqrt{(\mathbf{R}\mathbf{\Sigma}_{\mathbf{V}}^{th}\mathbf{R}^T)_{ii}}} \right). \tag{4.13}$$

We denote for all $i \in [c]$,

$$\gamma_i^{\mathbf{R}} = \sqrt{(\mathbf{R}\mathbf{\Sigma}_{\mathbf{V}}^{th}\mathbf{R}^T)_{ii}} \tag{4.14}$$

and

$$\gamma^{\mathbf{R}} = (\gamma_1^{\mathbf{R}}, \gamma_2^{\mathbf{R}}, \ldots, \gamma_c^{\mathbf{R}})^T. \tag{4.15}$$

Then, Cauchy-Schwartz inequality gives: $\langle \mathbf{1}, \gamma^{\mathbf{R}} \rangle^2 \leq \langle \mathbf{1}, \mathbf{1} \rangle \langle \gamma^{\mathbf{R}}, \gamma^{\mathbf{R}} \rangle$ i.e. $\left(\sum_{i\in[c]} \gamma_i^{\mathbf{R}} \right)^2 \leq c \cdot \sum_{i\in[c]} (\gamma_i^{\mathbf{R}})^2$ which rewrites:

$$\left(\sum_{i\in[c]} \gamma_i^{\mathbf{R}} \right)^{-1} \geq c^{-\frac{1}{2}} \cdot \left(\sum_{i\in[c]} (\gamma_i^{\mathbf{R}})^2 \right)^{-\frac{1}{2}} \tag{4.16}$$

Besides, $c^2 = \left\langle (\gamma^{\mathbf{R}})^{-\frac{1}{2}}, (\gamma^{\mathbf{R}})^{\frac{1}{2}} \right\rangle^2 \leq \sum_{i\in[c]} (\gamma_i^{\mathbf{R}})^{-1} \cdot \sum_{i\in[c]} \gamma_i^{\mathbf{R}}$ rewrites:

$$\sum_{i\in[c]} (\gamma_i^{\mathbf{R}})^{-1} \geq c^2 \cdot \left(\sum_{i\in[c]} \gamma_i^{\mathbf{R}} \right)^{-1} \tag{4.17}$$

$\mathbf{R}$ is a rotation, hence $\mathrm{Tr}(\mathbf{R}\mathbf{\Sigma}_{\mathbf{V}}^{th}\mathbf{R}^T) = \mathrm{Tr}(\mathbf{\Sigma}_{\mathbf{V}}^{th}) = \sum_{i\in[c]} (\sigma_i^{th})^2$. So, $\sum_{i\in[c]} (\gamma_i^{\mathbf{R}})^2 = \sum_{i\in[c]} (\mathbf{R}\mathbf{\Sigma}_{\mathbf{V}}^{th}\mathbf{R}^T)_{ii} = \mathrm{Tr}(\mathbf{\Sigma}_{\mathbf{V}}^{th})$ is a constant of $\mathbf{R}$. Then, Equation 4.16 and 4.17 give:

$$\sum_{i\in[c]} (\gamma_i^{\mathbf{R}})^{-1} \geq c^{\frac{3}{2}} \cdot \left(\sum_{i\in[c]} (\gamma_i^{\mathbf{R}})^2 \right)^{-\frac{1}{2}} = c^{\frac{3}{2}} C^{-\frac{1}{2}}. \tag{4.18}$$

Minimal value of $\sum_{i\in[c]} (\gamma_i^{\mathbf{R}})^{-1}$ is reached if and only if equality holds in the Cauchy-Schwartz inequalities i.e. if and only if, for all $i \in [c]$, $\gamma_i^{\mathbf{R}}$ are equal. Hence, for all $i \in [c]$, $\gamma_i^{\mathbf{R}^*} = \gamma_1^{\mathbf{R}^*}$. Conversely, if $\gamma_i^{\mathbf{R}^*} = \gamma_1^{\mathbf{R}^*}$ for all $i \in [c]$, then

$$\mathbf{R}^* \in \underset{\mathbf{R}}{\mathrm{argmin}} \left(\frac{2\epsilon}{\sqrt{2\pi}} \sum_{i\in[c]} \frac{1}{\sqrt{(\mathbf{R}\mathbf{\Sigma}_{\mathbf{V}}^{th}\mathbf{R}^T)_{ii}}} \right). \tag{4.19}$$

This completes the proof. □

If moreover $\mathbf{\Sigma}_{\mathbf{V}}^{th}$ is diagonal, inequality from Equation 4.11 becomes an equality. Thus, choosing $\mathbf{R}$ so that it uniformizes the diagonal of the covariance matrix $\mathbf{\Sigma}_{\mathbf{Y}}^{th}$ is exactly equivalent to minimizing the probability that the data points are close to an hyperplane delimiting an orthant plus $o(\epsilon^2)$, where ϵ is the distance to the orthant.

Please note that in practice the algorithm uniformizes the diagonal coefficients of the *empirical* covariance matrix $\mathbf{\Sigma}_{\mathbf{V}}$. This still makes sense because $\frac{1}{N}\mathbf{\Sigma}_{\mathbf{V}}$ is a consistent estimator of the theoretical covariance matrix $\mathbf{\Sigma}_{\mathbf{V}}^{th}$.

Now we can bound the probability of getting dissimilar codes for some data points close to each other, as stated in Theorem 4.3.2 below:

Theorem 4.3.2. *Let $\mathbf{x}_{t_1} \in \mathbb{R}^d$ and $\mathbf{x}_{t_2} \in \mathbb{R}^d$ be two data points following Hypothesis 4.3.1, $\epsilon > 0$ so that $\|\mathbf{x}_{t_1} - \mathbf{x}_{t_2}\|_2 \leq \epsilon$ and $\mathbf{b}_{t_1} \in \{-1, 1\}^c$, $\mathbf{b}_{t_2} \in \{-1, 1\}^c$ with $\mathbf{b}_{t_i} = \mathrm{sign}(\mathbf{R}\mathbf{W}\mathbf{x}_{t_i})$ for $i \in \{1, 2\}$. Then, the probability of getting dissimilar binary codes is upper bounded as follows:*

$$\mathbb{P}[\ dist_H(\mathbf{b}_{t_1}, \mathbf{b}_{t_2}) > 0] \leq 2\epsilon \sqrt{\frac{2}{\pi}} c^{\frac{3}{2}} \left(\mathrm{Tr}(\mathbf{\Sigma}_{\mathbf{V}}^{th}) \right)^{-\frac{1}{2}}. \tag{4.20}$$

Proof. As PCA performs a projection, one has:

$$\|\mathbf{v}_{t_1} - \mathbf{v}_{t_2}\|_2 \leq \|\mathbf{x}_{t_1} - \mathbf{x}_{t_2}\|_2 \leq \epsilon. \tag{4.21}$$

Then, $\|\mathbf{y}_{t_1} - \mathbf{y}_{t_2}\|_2 \leq \epsilon$ since $\mathbf{R}$ preserves the norm as a rotation. Thus, in particular, for all $i \in [c]$, $|\mathbf{y}_{t_1}^{(i)} - \mathbf{y}_{t_2}^{(i)}| \leq \epsilon$. Then,

$$\mathbb{P}[\ dist_H(\mathbf{b}_{t_1}, \mathbf{b}_{t_2}) > 0] = \mathbb{P}[\bigcup_{i\in[c]} \left(\mathbf{y}_{t_1}^{(i)}\mathbf{y}_{t_2}^{(i)} < 0\right)] \tag{4.22}$$

$$\leq \mathbb{P}[\bigcup_{i\in[c]} \left(|\mathbf{y}_{t_1}^{(i)}| < \epsilon \ \cap \ |\mathbf{y}_{t_2}^{(i)}| < \epsilon\right)] \tag{4.23}$$

because $\left(|\mathbf{y}_{t_1}^{(i)}| \geq \epsilon \cup |\mathbf{y}_{t_2}^{(i)}| \geq \epsilon\right) \implies \left(\mathbf{y}_{t_1}^{(i)}\mathbf{y}_{t_2}^{(i)} \geq 0\right)$. Moreover,

$$\mathbb{P}[\bigcup_{i\in[c]} \left(|\mathbf{y}_{t_1}^{(i)}| < \epsilon \ \cap \ |\mathbf{y}_{t_2}^{(i)}| < \epsilon\right)] \leq \mathbb{P}[\bigcup_{i\in[c]} |\mathbf{y}_{t_1}^{(i)}| < \epsilon] + \mathbb{P}[\bigcup_{i\in[c]} |\mathbf{y}_{t_2}^{(i)}| < \epsilon]. \tag{4.24}$$

Since $\mathbf{y}_{t_1}$ and $\mathbf{y}_{t_2}$ have the same distribution and using Theorem 4.3.1,

$$\mathbb{P}[\bigcup_{i\in[c]} |\mathbf{y}_{t_1}^{(i)}| < \epsilon] + \mathbb{P}[\bigcup_{i\in[c]} |\mathbf{y}_{t_2}^{(i)}| < \epsilon] = 2\mathbb{P}[\bigcup_{i\in[c]} |\mathbf{y}_{t_1}^{(i)}| < \epsilon] \tag{4.25}$$

$$\leq 2\sum_{i\in[c]} \mathbb{P}[\ |\mathbf{y}_{t_1}^{(i)}| < \epsilon] = 2\epsilon\sqrt{\frac{2}{\pi}}c^{\frac{3}{2}}\left(\mathrm{Tr}(\mathbf{\Sigma}_{\mathbf{V}}^{th})\right)^{-\frac{1}{2}} \tag{4.26}$$

The result follows. □

4.4 The proposed online algorithm: UnifDiag Hashing

4.4.1 The principle of UnifDiag

In the online setting, what one would like to maintain dynamically is only the cheap to store $c \times c$ symmetric matrix $\mathbf{\Sigma}_{\mathbf{V},t} = \mathbf{V}_t\mathbf{V}_t^T$, the covariance matrix of projected data seen until data t $\mathbf{V}_t = \mathbf{W}_t\mathbf{X}_t$. This is straightforward while updating $\mathbf{W}_t$ with OPAST algorithm [Abed-Meraim et al., 2000].

Then, again, similarly to IsoHash [Kong and Li, 2012], the rotation $\mathbf{R}_t$ is learned for each t to balance the variance over the c directions given by the c principal components of $\mathbf{\Sigma}_{\mathbf{V},t}$. After application of $\mathbf{R}_t$, $\mathbf{\Sigma}_{\mathbf{Y},t}$ is expected to have each of its diagonal coefficients equal to τ: $\forall i \in [c]$, $(\mathbf{\Sigma}_{\mathbf{Y},t})_{ii} = \tau = \mathrm{Tr}(\mathbf{\Sigma}_{\mathbf{V},t})/c$. In the sequel, for clarity we drop the subscript t. In our model, $\mathbf{R}$ is defined as a product of $c-1$ Givens rotations $\mathbf{G}(i,j,\theta)$. Definition 4.4.1 details what is a Givens rotation.

Definition 4.4.1. *A Givens rotation $\mathbf{G}(i,j,\theta)$ is a matrix of the form:*

$$\mathbf{G}(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \tag{4.27}$$

where for $i > j$, $c := \cos(\theta)$ and $s := \sin(\theta)$ are at the intersections of the i-th and j-th rows and columns. The nonzero elements are consequently: $\forall k \neq i, j$, $g_{k,k} = 1$, $g_{i,i} = g_{j,j} = c$, $g_{j,i} = -s$ and $g_{i,j} = s$ for $i > j$. All remaining coefficients are set to 0. If $i < j$, $\mathbf{G}(i,j,\theta) = \mathbf{G}(j,i,-\theta)$ taken from the definition for $i > j$.

The principle is the following: the Givens rotations are iteratively applied left and right to $\mathbf{\Sigma_V}$ during the iterative Jacobi eigenvalue algorithm for matrix diagonalization [Golub and van der Vorst, 2000] with $c-1$ steps. For the step $r \in [c-1]$, given i_r, j_r, θ_r,

$$(\mathbf{\Sigma_Y})_r \leftarrow \mathbf{G}(i_r, j_r, \theta_r)\ (\mathbf{\Sigma_Y})_{r-1}\ \mathbf{G}(i_r, j_r, \theta_r)^T \tag{4.28}$$

$$\mathbf{R}_r \leftarrow \mathbf{R}_{r-1}\ \mathbf{G}(i_r, j_r, \theta_r)^T, \tag{4.29}$$

where $(\mathbf{\Sigma_Y})_0 = \mathbf{\Sigma_V}$, $\mathbf{R}_0 = \mathbf{I}_c$.

Now how to determine i_r, j_r and θ_r? At each step r, i_r and j_r are chosen to be the indices of some diagonal coefficients of $(\mathbf{\Sigma_Y})_{r-1}$ below, respectively above τ. θ_r is computed accordingly following the result of Theorem 4.4.1 so that at the end of step r, r diagonal coefficients of $(\mathbf{\Sigma_Y})_r$ are equal to τ. So at each step r, only two diagonal coefficients are modified: at least one of the both becomes exactly equal to τ. Indeed, one can see that left (resp. right) multiplication by $\mathbf{G}(i_r, j_r, \theta_r)$ impacts only i_r^{th} and j_r^{th} rows (resp. columns). Thus, the update in Equation 4.28 at step r only changes i_r^{th} and j_r^{th} rows and columns of $(\mathbf{\Sigma_Y})_{r-1}$ and the two updated diagonal coefficients (i_r, i_r) and (j_r, j_r) depend solely on $((\mathbf{\Sigma_Y})_{r-1})_{i_r i_r}$, $((\mathbf{\Sigma_Y})_{r-1})_{j_r j_r}$, $((\mathbf{\Sigma_Y})_{r-1})_{j_r i_r}$ and θ_r which reduces the optimization of θ_r at each step r to a 2-dimensional problem, a classical trick when using the Givens rotations [Golub and van der Vorst, 2000]. By defining

$$a := ((\mathbf{\Sigma_Y})_{r-1})_{j_r j_r}, \tag{4.30}$$

$$d := ((\mathbf{\Sigma_Y})_{r-1})_{i_r i_r}, \tag{4.31}$$

$$b := ((\mathbf{\Sigma_Y})_{r-1})_{j_r i_r} = ((\mathbf{\Sigma_Y})_{r-1})_{i_r j_r}, \tag{4.32}$$

the 2-dimensional problem is:

$$\begin{pmatrix} a' & b' \\ b' & d' \end{pmatrix} := \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} a & b \\ b & d \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix}. \tag{4.33}$$

Theorem 4.4.1 gives finally how to compute θ_r at each step given i_r and j_r. The subscript r is dropped for readability.

Theorem 4.4.1. *If* $\min(a,d) \leq \tau \leq \max(a,d)$ *(sufficient condition)*[6] *then there exists one* $\theta \in [-\pi/2, \pi/2]$ *s.t.* $a' = \tau$, $d' = a + d - \tau$ *and* $b' = -s_2\sqrt{\left(\frac{a-d}{2}\right)^2 + b^2}$ *with* $\cos(\theta) = \sqrt{\frac{1+c_1c_2-s_1s_2}{2}}$ *and* $\sin(\theta) = -\frac{c_1s_2+c_2s_1}{2\cos\theta}$, $c_1 = \left(\frac{a-d}{2}\right)/\sqrt{\left(\frac{a-d}{2}\right)^2 + b^2}$, $s_1 = b/\sqrt{\left(\frac{a-d}{2}\right)^2 + b^2}$, $c_2 = (\tau - \frac{a+d}{2})/\sqrt{\left(\frac{a-d}{2}\right)^2 + b^2}$ *and* $s_2 = \sqrt{1-c_2^2} \in [0,1]$.

[6] Theorem 4.4.1 uses only a sufficient condition. A weaker necessary and sufficient one to guarantee $|c_2| \leq 1$ and $s_2 \in [0,1]$ is $\frac{a+d}{2} - \sqrt{\left(\frac{a-d}{2}\right)^2 + b^2} \leq \tau \leq \frac{a+d}{2} + \sqrt{\left(\frac{a-d}{2}\right)^2 + b^2}$.

Proof. Equation 4.33 implies:

$$a' = \frac{a+d}{2} + \begin{pmatrix} \frac{a-d}{2} & b \end{pmatrix} . \begin{pmatrix} \cos(2\theta) \\ -\sin(2\theta) \end{pmatrix} \tag{4.34}$$

$$d' = \frac{a+d}{2} - \begin{pmatrix} \frac{a-d}{2} & b \end{pmatrix} . \begin{pmatrix} \cos(2\theta) \\ -\sin(2\theta) \end{pmatrix} \tag{4.35}$$

$$b' = \begin{pmatrix} \frac{a-d}{2} & b \end{pmatrix} . \begin{pmatrix} \sin(2\theta) \\ \cos(2\theta) \end{pmatrix} \tag{4.36}$$

As, the Givens angle θ should be parameterized s.t. the diagonal coefficients are set to a same value τ,

$$\begin{pmatrix} \frac{a-d}{2} & b \end{pmatrix} . \begin{pmatrix} \cos(2\theta) \\ -\sin(2\theta) \end{pmatrix} = \tau - \frac{a+d}{2} \tag{4.37}$$

Recall that:

$$c_1 := \cos(\theta_1) = \left(\frac{a-d}{2}\right) / \sqrt{\left(\frac{a-d}{2}\right)^2 + b^2} \tag{4.38}$$

$$s_1 := \sin(\theta_1) = b / \sqrt{\left(\frac{a-d}{2}\right)^2 + b^2} \tag{4.39}$$

$$c_2 := \cos(\theta_2) = \left(\tau - \frac{a+d}{2}\right) / \sqrt{\left(\frac{a-d}{2}\right)^2 + b^2} \tag{4.40}$$

$$s_2 := \sin(\theta_2) = \sqrt{1 - c_2^2}. \tag{4.41}$$

The condition $\min(a,d) < \tau < \max(a,d)$ guarantees c_2 to be well defined i.e. $|c_2| \leq 1$ and $s_2 \in \mathbb{R}^+$. Then, Equation 4.37 becomes:

$$\begin{pmatrix} c_1 & s_1 \end{pmatrix} \begin{pmatrix} \cos(2\theta) \\ -\sin(2\theta) \end{pmatrix} = c_2. \tag{4.42}$$

This is clear that a solution of Equation 4.42 is:

$$\begin{pmatrix} \cos(2\theta) \\ -\sin(2\theta) \end{pmatrix} = \begin{pmatrix} c_1 c_2 - s_1 s_2 \\ c_1 s_2 + c_2 s_1 \end{pmatrix}. \tag{4.43}$$

In that case, one can take:

$$\cos(\theta) = \sqrt{\frac{1+\cos(2\theta)}{2}} = \sqrt{\frac{1 + c_1 c_2 - s_1 s_2}{2}} \tag{4.44}$$

$$\sin(\theta) = \frac{\sin(2\theta)}{2\cos(\theta)} = -\frac{c_1 s_2 + c_2 s_1}{2\cos\theta} \tag{4.45}$$

and the corresponding Givens rotation gives:

$$a' = \tau \tag{4.46}$$
$$d' = a + d - \tau \tag{4.47}$$
$$b' = \begin{pmatrix} \frac{a-d}{2} & b \end{pmatrix} \begin{pmatrix} \sin(2\theta) \\ \cos(2\theta) \end{pmatrix} \tag{4.48}$$
$$= \sqrt{\left(\frac{a-d}{2}\right)^2 + b^2} \begin{pmatrix} c_1 & s_1 \end{pmatrix} \begin{pmatrix} -c_1 s_2 - c_2 s_1 \\ c_1 c_2 - s_1 s_2 \end{pmatrix} \tag{4.49}$$
$$= -s_2 \sqrt{\left(\frac{a-d}{2}\right)^2 + b^2} \tag{4.50}$$

with s_2 completely defined by Equation 4.41 and 4.40. □

Note that there is no need to compute explicitly θ, θ_1 or θ_2. The method is summarized in Algorithm 5 where pop($list$) and add($list, e$) are subroutines to delete and return the first element of $list$, respectively to add e at the end of $list$. The mean of $\mathbf{\Sigma_Y}$ diagonal coefficients being equal to τ, these indices sets are not empty:

$$iInf := \{\ l \in \{1,\ ...,\ c\} \mid \mathbf{\Sigma_Y}_{ll} < \tau\} \tag{4.51}$$

and

$$iSup := \{\ l \in \{1,\ ...,\ c\} \mid \mathbf{\Sigma_Y}_{ll} > \tau\}. \tag{4.52}$$

Taking one index j from $iInf$ and the other one i from $iSup$ guarantees the condition of Theorem 4.4.1, which allows to set $\mathbf{\Sigma_Y}_{jj}$ to the value τ. The index j can then be removed from $iInf$ and as $\mathbf{\Sigma_Y}_{ii}$ is set to $a + d - \tau$, the index i reassigned to $iInf$ if $\tau > \frac{a+d}{2}$, or in $iSup$ if $\tau < \frac{a+d}{2}$. The number of diagonal coefficients of $\mathbf{\Sigma_Y}$ different from τ has been decreased by one. Finally, the necessary number of iterations to completely empty $iInf$ and $iSup$, i.e. uniformizing $\mathbf{\Sigma_Y}$ diagonal, is bounded by $c-1$.

Finally, Figure 4.4 shows the covariance matrix of data from CIFAR dataset resulting from the rotation with IsoHash and UnifDiag for $c = 32$. The particular structure of covariance for UnifDiag is due to the sparsity of the rotation.

4.4.2 Time and space complexities comparison with existing online works

Our algorithm requires the storage of two $c \times c$ matrices, besides obviously $\mathbf{W}_t$ and $\mathbf{R}_t$: one with OPAST to obtain $\mathbf{W}_t$ and $\mathbf{\Sigma_V}_{,t}$ for $\mathbf{R}_t$. One update with OPAST for $\mathbf{W}_t$ and $\mathbf{\Sigma_V}_{,t}$ costs $4dc + O(c^2)$. Then, to compute $\mathbf{R}_t$, at most $c-1$ Givens rotations are needed, each implying four column or row multiplications i.e. $4c$ flops. So the final time complexity of our algorithm is $4dc + O(c^2)$.

We compare here the spatial and time costs of our method with, to the best of our knowledge, the only online unsupervised method, **Online Sketching Hashing** (OSH) [Leng et al., 2015a] which is the most similar to ours, i.e. unsupervised, hyperplanes-based, from the Hypercubic hashing family and reading one data point at a time. Recall from Section 2.3.2, despite what is announced, OSH is fundamentally mini-batch: the stream is divided into chunks of data for which a matrix $\mathbf{S} \in \mathbb{R}^{d \times l}$ as a sketch of the whole dataset $\mathbf{X} \in \mathbb{R}^{d \times N}$ is maintained. Then the principal components are computed from the updated sketch $\mathbf{S}$. The projection of the data followed by the random rotation can be applied only after this step. Therefore there are actually two passes over the data by reading twice data of each chunk. Without counting the

Algorithm 5 Diagonal Uniformization algorithm (UnifDiag)

Inputs : $\mathbf{\Sigma_V}$ ($c \times c$, symmetric), tolerance: tol
$R \leftarrow \mathbf{I}_c$ // $c \times c$ Identity matrix
$\tau \leftarrow \mathrm{Tr}(\mathbf{\Sigma}_V)/c$
$\mathbf{\Sigma_Y} \leftarrow \mathbf{\Sigma_V}$
$it = 0$
$iInf = \{\ l \in \{1,\ ...,\ c\} \mid (\mathbf{\Sigma_Y})_{ll} < \tau - tol\}$
$iSup = \{\ l \in \{1,\ ...,\ c\} \mid (\mathbf{\Sigma_Y})_{ll} > \tau + tol\}$
while $it < c - 1$ & not isEmpty($iInf$) & not isEmpty($iSup$) **do**
  // Givens rotation parameters computation:
  $j \leftarrow pop(iInf)$
  $i \leftarrow \mathrm{pop}(iSup)$
  $a \leftarrow \mathbf{\Sigma_Y}[j, j]$
  $b \leftarrow \mathbf{\Sigma_Y}[i, j]$
  $d \leftarrow \mathbf{\Sigma_Y}[i, i]$
  c, s from formula in Theorem 4.4.1, p. 82.
  $it \leftarrow it + 1$
  // $\mathbf{\Sigma_Y}$ *update*:
  $row_j \leftarrow \mathbf{\Sigma_Y}[j, :]$
  $row_i \leftarrow \mathbf{\Sigma_Y}[i, :]$
  $\mathbf{\Sigma_Y}[j, :] = c \times row_j - s \times row_i$;
  $\mathbf{\Sigma_Y}[i, :] = s \times row_j + c \times row_i$
  $\mathbf{\Sigma_Y}[:, j] = \mathbf{\Sigma_Y}[j, :]$
  $\mathbf{\Sigma_Y}[:, i] = \mathbf{\Sigma_Y}[i, :]$
  $\mathbf{\Sigma_Y}[j, j] = a'$
  $\mathbf{\Sigma_Y}[i, i] = d'$
  $\mathbf{\Sigma_Y}[j, i] = b'$ from formula in Theorem 4.4.1, p. 82.
  // Rotation update:
  $col_j \leftarrow \mathbf{R}[:, j]$
  $col_i \leftarrow \mathbf{R}[:, i]$
  $\mathbf{R}[:, j] = c \times col_j - s \times col_i$
  $\mathbf{R}[:, i] = s \times col_j + c \times col_i$
  // Indices list update:
  if $\frac{a+d}{2} < \tau - tol$ **then**
    add($iInf, i$)
  if $\frac{a+d}{2} > \tau + tol$ **then**
    add($iSup, i$)
return $\mathbf{R}$

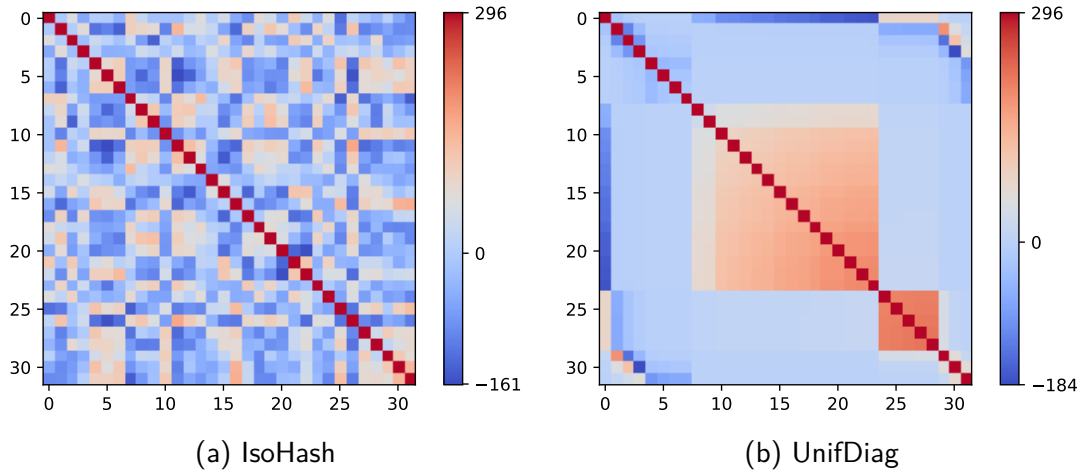


FIGURE 4.4: Covariance matrix of rotated PCA-projected data from CIFAR dataset with $c = 32$. The particular structure of covariance for UnifDiag is due to the sparsity of the rotation.

projection matrix and the rotation, OSH needs spatially to maintain the sketch $\mathbf{S}$ which costs $O(d \times l)$ with $c \ll l \ll d$. The SVD decomposition then needs $O(dl + l^2)$ space. In comparison, we only need $O(c^2)$. For each round, OSH takes $O(dl^2 + l^3)$ time to learn the principal components, i.e. $O(dl + l^2)$ for each new data seen.

We also compare with **IsoHash** [Kong and Li, 2012]. Although it counts as an offline method because no technique is proposed to approximatively estimate the principal subspace, IsoHash rotation can be applied after for instance OPAST. IsoHash rotation computation involves an integration of a differential equation using Adams-Bashforth-Moulton PECE solver which costs $O(c^3)$ time. Even if c is small in comparison to d and the complexities do not either depend on N, our model has the advantages to have a lower time cost and to be much more simple than IsoHash. Thus, our method shows advantages in terms of spatial and time complexities over OSH and IsoHash. Moreover, binary hash codes can be directly computed as a new data point is seen, while OSH, as a mini-batch method, has a delay.

Finally, Figure 4.5 illustrates the fully online pipeline for the hashing technique with OPAST and UnifDiag.

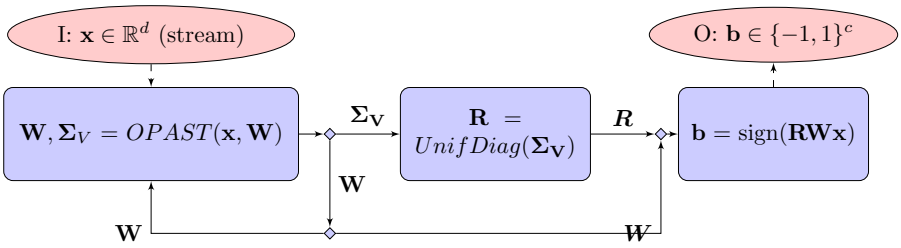


FIGURE 4.5: The fully online pipeline for the hashing technique with OPAST and UnifDiag. For clarity, the subscript t is dropped.

4.5 Experiments

Experiments have been carried out on a single processor machine (Intel Core i7-5600U CPU @ 2.60GHz, 4 hyper-threads) with 16GB RAM and implemented in Python[7].

4.5.1 Comparison of Hypercubic Quantization Hashing methods for the Nearest Neighbors search task

We propose first an experimental comparison for different code lengths in the batch and the online settings of our algorithm with the existing competing methods.

The quality of the hashing is assessed here on the Nearest Neighbors (NN) search task: the retrieved results of the NN search task performed on the c-bits codes of hashed data points are compared with the true nearest neighbors induced by the Euclidean distance applied on the initial d-dimensional real-valued descriptors. Mean Average Precision (MAP), commonly used in Information Retrieval tasks, measures then the accuracy of the result by taking into account the number of well-retrieved nearest neighbors and their rank. Based on the MAP criteria, two types of experiments are presented:

1. UnifDiag algorithm is compared with batch-based methods to show that the streaming constraint does not lose too much accuracy in NN search results.

2. In the online context, the algorithm is set against to existing online methods in order to exhibit its efficiency. In both cases, tests were conducted on two datasets: CIFAR-10[8] and GIST1M[9].

CIFAR-10 (CIFAR) contains 60000 32×32 color images equally divided into 10 classes. 960-D GIST descriptors were extracted from those data. GIST1M (GIST) contains 1 million 960-D GIST descriptors, from which 60000 instances were randomly chosen from the first half of the learning set. To perform the NN search task, 1000 queries are randomly sampled and the 59000 remaining data points are used as training set. Then, the sets of neighbors and non-neighbors of the queries are determined by a nominal threshold which is arbitrarily chosen to be the average distance to the 600^{th} nearest neighbor in the training set (1% of each dataset). After binary hashing of the query and training sets, MAP at 2000 is computed over the first 2000 retrieved nearest neighbors from the training set according to the sorted values of the Hamming distance between the binary codes. Indeed, we are obviously interested in the fact that the nearest neighbors are returned first, so MAP@2000 is enough. Results are averaged over 5 random training/test partitions.

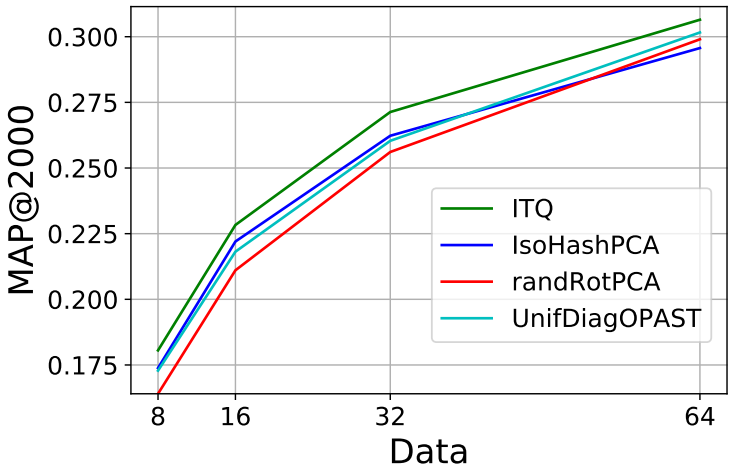
Comparison with batch-based methods

In the offline setting, the hashing function is learned over the whole training set and the final MAP is printed for different values of c. Figure 4.6 shows the MAP results for UnifDiag against three unsupervised batch-based methods: ITQ ($K = 50$), IsoHash (the original version preceded by a PCA projection) and PCA followed by a random rotation. The evaluation is made after having seen the whole training set. The online estimation of the principal subspace via OPAST instead of the classical PCA does not lead to a loss of accuracy, since UnifDiag reaches similar performances to batch-based methods for every tested code lengths.

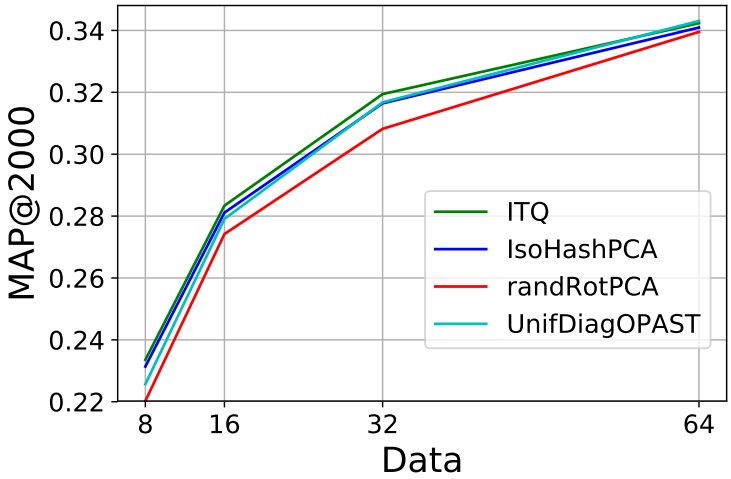
[7] Code available on Github:
https://github.com/annemorvan/UnifDiagStreamBinSketching

[8] http://www.cs.toronto.edu/~kriz/cifar.html

[9] http://corpus-texmex.irisa.fr/



(a) CIFAR



(b) GIST

FIGURE 4.6: MAP@2000 in the batch setting for various code lengths $c \in \{8, 16, 32, 64\}$.

Comparison with online methods

In the online setting, we print the MAP after every 5 data points until the 3000 ones because a plateau is then reached (but the NN are still computed over the whole training set). We compared here four unsupervised online baseline methods that follow the basic hashing scheme

$$\Phi(\mathbf{x}_t) = \text{sgn}(\tilde{\mathbf{W}}_t \mathbf{x}_t), \tag{4.53}$$

where the projection matrix $\tilde{\mathbf{W}}_t \in \mathbb{R}^{c \times d}$ is determined according to the chosen method:

1. **OSH** [Leng et al., 2015a]: the number of chunks/rounds is set to 100 and $l = 200$.
2. **RandRotOPAST**: W_t is the PCA matrix obtained with OPAST and $\mathbf{R}_t$ a constant random rotation.
3. **IsoHashOPAST**: $\mathbf{R}_t$ is obtained with IsoHash.
4. **UnifDiagOPAST**.

Figure 4.7 and 4.8 (best viewed in color) show the MAP for both datasets for different code lengths. Not surprisingly, UnifDiagOPAST and the online version of IsoHash with OPAST exhibit similar behavior for both datasets. Moreover, for small values of code length ($c < 64$), UnifDiagOPAST outperforms OSH and randRotOPAST while all have similar results for $c = 64$.

4.5.2 Effect of the rotation on the binary codes

In this Section, in the offline context, the experiments shed light on the theory from Section 4.3 by showing why a rotation gives better small binary codes than simply PCA projection. Rotations considered are random or learned from ITQ, IsoHash and UnifDiag.

First, for CIFAR and GIST datasets, we compute the cumulative distribution function of $\mathbb{P}[|\mathbf{y}_t^{(i)}| < \epsilon]$, i.e. the probability for all $t \in [N]$ of $\mathbf{y}_t \in \mathbb{R}^c$ to have entries near zero before and after the rotation application. Figure 4.9 plots $\mathbb{P}[|\mathbf{y}_t^{(i)}| < \epsilon]$ for $c = 32$ (averaged on 5 runs). Similar results are obtained for other code lengths. For all rotation-based methods, this probability is always lower similarly than the one associated to only PCA projection.

Secondly, we provide a visualization of the rotation efficiency in the clustering task. This experiment is made on simulated data since a ground truth partition is required. We consider C equally distributed clusters of N data points such that the nearest neighbors of a data point are the points from the same cluster. We choose the centroids from these clusters randomly. The expected result is a small variance of the binary codes within the same cluster. Figure 4.10 displays the small binary embeddings obtained with and without rotation for $C = 6$, $n = 6000$ points with $d = 960$ and $c = 32$. Each column is a binary code: a yellow case represents a bit equal to 1 and a red pixel stands for -1. Each cluster, delimited by a blue vertical line, contains 1000 points plotted in order. An interpretation of the results is that the rotation tends to move data away from the hyperplanes delimiting the orthant since more binary codes after application of the rotation have bits in common. This is illustrated by the obtained blocks of the same color, as opposed to the "blurry" visualization implied by the PCA alone.

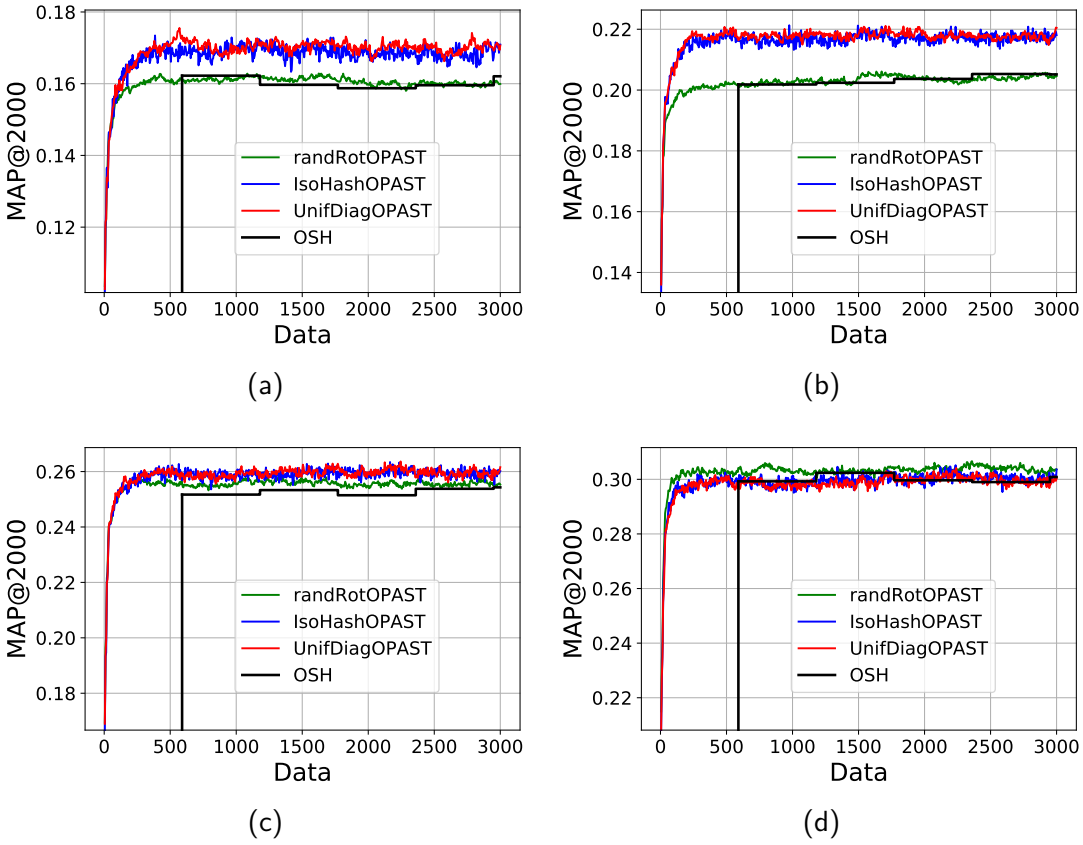


FIGURE 4.7: MAP@2000 in the online setting for different code lengths and CIFAR: (a) $c = 8$, (b) $c = 16$, (c) $c = 32$, (d) $c = 64$.

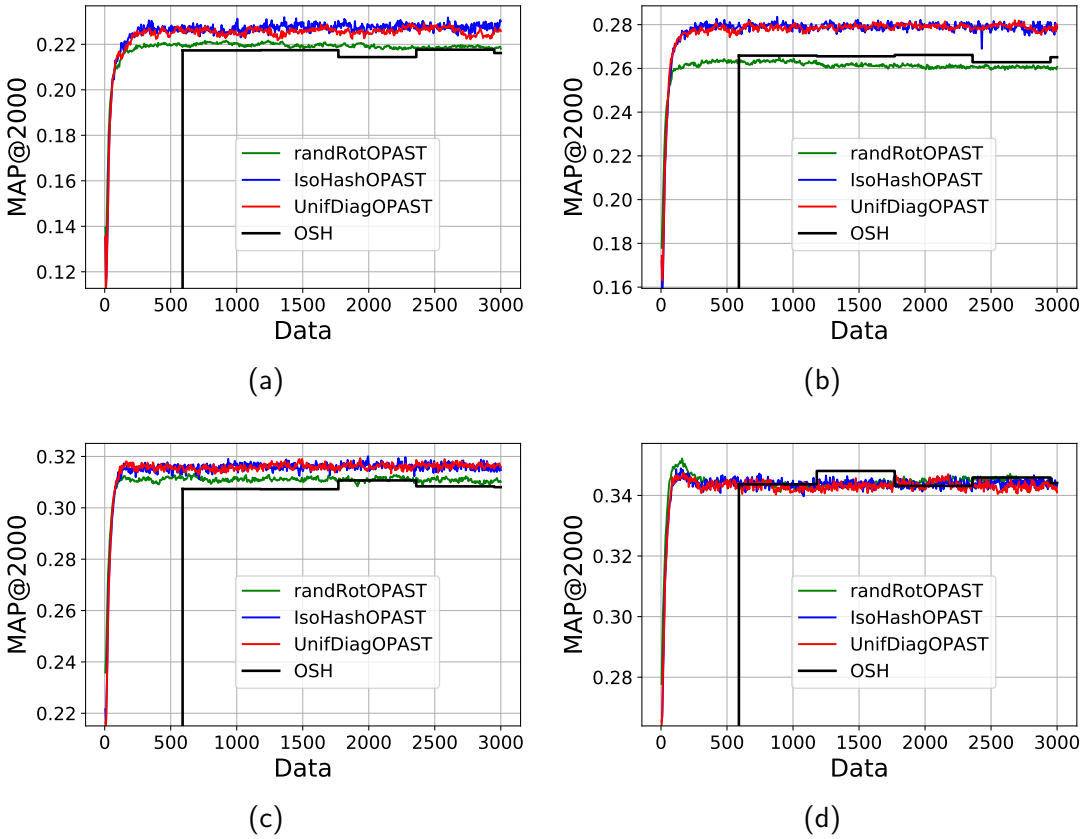


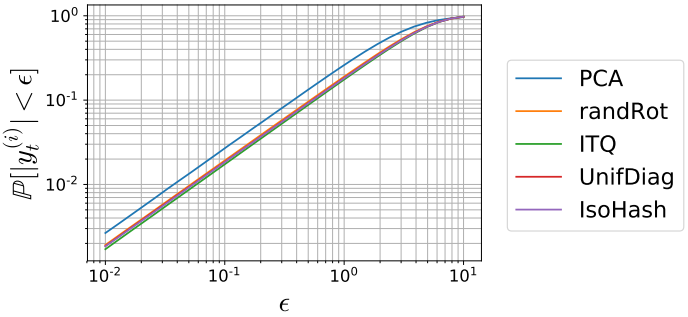
FIGURE 4.8: MAP@2000 in the online setting for different code lengths and GIST: (a) $c = 8$, (b) $c = 16$, (c) $c = 32$, (d) $c = 64$.

More quantitatively, Table 4.1 compiles the variance of the compact binary representations averaged on the 6 clusters for 10 partitions. Not surprisingly, PCA gives the worst results: the high variance in the binary codes explains the previous "blurry" visualization. Conversely, all rotation-based methods tend to reduce the variance in the binary codes.

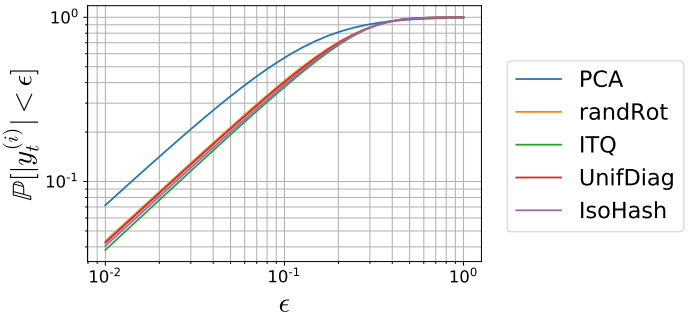
4.6 Conclusion

State-of-the-art unsupervised Hypercubic quantization hashing methods preprocess data with Principal Component Analysis and then rotate the projected data to balance variance over the different directions. It has been shown experimentally that the compact binary representations that they compute with a rotation give better accuracy in the nearest neighbors search task than simply the PCA-projection. Nevertheless, to the best of our knowledge, this is the first time that theoretical guarantees are provided. In particular, rotations uniformizing the diagonal of the data covariance matrix are interesting since they enable to deploy the hashing algorithm in the streaming setting.

Thus, we introduced a novel method for learning distance-preserving binary embeddings of high-dimensional data streams with convergence guarantees. Unlike classical state-of-the-art methods, our algorithm does not need to store the whole dataset and enables to obtain without a delay a binary code as a new data point is seen. Our approach shows promising results as evidenced by the experiments. It can achieve



(a) CIFAR



(b) GIST

FIGURE 4.9: Cumulative distribution function for CIFAR and GIST and different hashing methods: $\forall i \in [c], \mathbb{P}[|\mathbf{y}_t^{(i)}| < \epsilon]$ for $c = 32$.

TABLE 4.1: Mean variance for the binary codes (averaged on 10 runs) obtained for 6 convex clusters with random centroids in $d = 960$.

	8	16	32	64
PCA	7.1×10^{-2}	6.5×10^{-2}	4.0×10^{-2}	2.2×10^{-2}
randRot	3.9×10^{-4}	2.6×10^{-4}	1.9×10^{-4}	7.4×10^{-5}
ITQ	0.0	2.0×10^{-4}	1.1×10^{-4}	1.3×10^{-4}
UnifDiag	4.1×10^{-4}	2.2×10^{-4}	2.5×10^{-4}	1.1×10^{-4}
IsoHash	1.4×10^{-4}	3.3×10^{-4}	2.1×10^{-4}	1.3×10^{-4}

better accuracy than state-of-the-art online unsupervised methods while saving considerable computation time and spatial requirements.

Besides, it has been shown that the Givens rotations, that are a classical tool for QR factorization, singular and eigendecomposition or joint diagonalization, can also be used to uniformize the diagonal of a symmetric matrix via an original Givens angle tuning technique.

As there exists possibly an infinity of rotations uniformizing the covariance matrix diagonal, further investigation would be to evaluate among them which ones perform better. Another interesting perspective is to evaluate the performance of the compact binary codes in other machine learning applications: instead of using the original data, one could use directly these binary embeddings to perform unsupervised or supervised learning while preserving the accuracy.

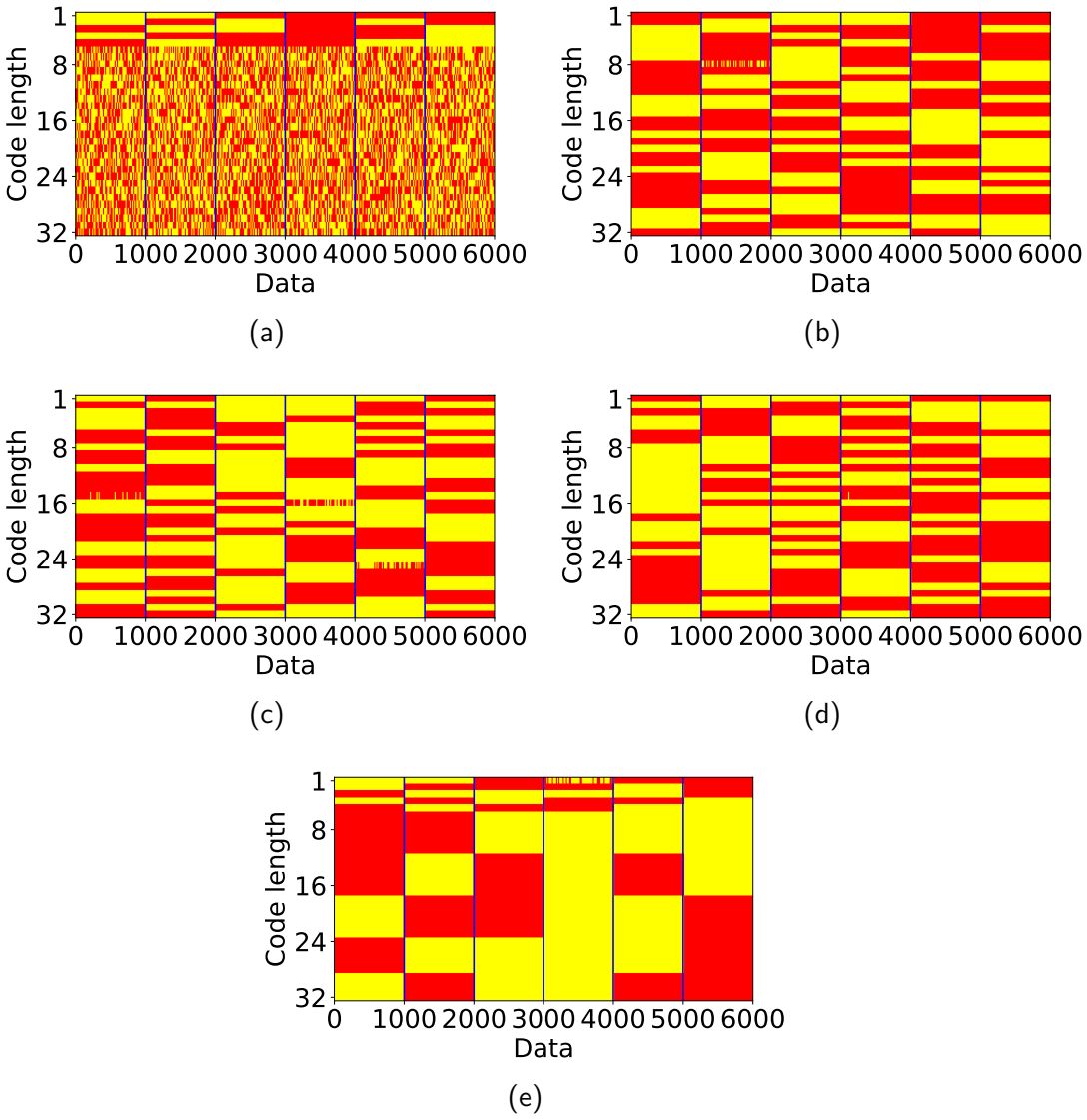


FIGURE 4.10: Effect of the rotation on the small binary codes for simulated data: 6 clusters with $d = 960$ and $c = 32$: (a) PCA, (b) RandRot, (c) ITQ, (d) IsoHash, (e) UnifDiag.

Chapter 5

Graph sketching-based massive data clustering

Contents

This Chapter is a compilation of two published papers in separate collaborations with Krzysztof Choromanski[1] on the one hand and Rafaël Pinot[2] and Florian Yger[3] on the other hand. The corresponding papers have been published in the two following international conferences:

- *SIAM International Conference on Data Mining (SDM) 2018 under the title "Graph sketching-based Space-efficient Data Clustering",*

[1]Google Brain Robotics
[2]CEA, Université Paris-Dauphine
[3]Université Paris-Dauphine

- *International conference on Uncertainty in Artificial Intelligence (UAI) 2018 under the title "Graph-based Clustering under Differential Privacy".*

5.1 Introduction

As mentioned earlier in this thesis, clustering is one of the principal data mining tasks consisting in grouping related objects in an unsupervised manner. It is expected that objects belonging to the same cluster are more similar to each other than to objects belonging to different clusters. Sections 1.1.2 and 2.3.4 testify there exists a variety of algorithms performing this task. Recall that methods like k-means [Lloyd, 1982], k-medians [Jain and Dubes, 1988] or k-medoids [Kaufman and Rousseeuw, 1987] are useful unless the number and the shape of the clusters are unknown which is unfortunately often the case in real-world applications. They are typically unable to find clusters with a non-convex shape. Although DBSCAN [Ester et al., 1996] does not have these disadvantages, its resulting clustering still depends on the chosen parameter values.

One of the successful approaches relies on a graph representation of the data. Given a set of N data points $\{x_1, \ldots, x_N\}$, a graph can be built based on the dissimilarity of the data where the points of the dataset are the vertices and the weighted edges express dissimilarities or "distances" between these objects. Besides, the dataset can be already a graph $\mathcal{G}$ modeling a network in many fields, such as bioinformatics - where gene-activation dependencies are described through a network - or social, computer, information, transportation network analysis. The clustering task consequently aims at detecting clusters as groups of nodes close in terms of some specific similarity. For instance, these clusters can be seen as groups of nodes that are densely connected with each other and sparsely connected to vertices of other groups. In this context, Spectral Clustering [Nascimento and de Carvalho, 2011] is a popular tool to recover clusters with some particular structures for which classical k-means algorithm fails. When dealing with large scale datasets, a main bottleneck of the technique is to perform the partial eigendecomposition of the associated graph Laplacian matrix, though. Another inherent difficulty is to handle the huge number of nodes and edges of the induced dissimilarity graph: storing all edges can cost up to $O(N^2)$ where N is the number of nodes. Over the last decade, it has been established that the dynamic streaming model [Muthukrishnan, 2005] associated with linear sketching techniques [Ahn et al., 2012a] - also suitable for distributed processing -, is a good way for tackling this last issue.

Contributions In this Chapter, our contributions are threefold:

1. A new clustering algorithm DBMSTClu is presented providing a solution to the following issues:

   (a) detecting arbitrary-shaped data clusters,

   (b) with no parameter,

   (c) in a space-efficient manner by working on a limited number of linear measurements, a *sketched* version of the dissimilarity graph $\mathcal{G}$.

   DBMSTClu returns indeed a partition of the N points to cluster by relying only on a Minimum Spanning Tree (MST) of the dissimilarity graph $\mathcal{G}$. This MST takes $O(N)$ memory space. Moreover, it can be space-efficiently approximatively retrieved in the dynamic semi-streaming model by handling $\mathcal{G}$ as a

stream of edge weight updates; $\mathcal{G}$ is sketched in only one pass over the stream of edge weight updates into a compact structure taking $O(N \operatorname{polylog}(N))$ space. DBMSTClu then automatically identifies the right number of non-convex clusters by cutting off suitable edges of the resulting approximate MST in $O(N)$ time.

2. DBMSTClu belongs to the family of MST-based clustering algorithms. The first theoretical justifications of MST-based clustering algorithms are given in order to motivate the design of our algorithm.

3. Finally, DBMSTClu is endowed with specific theoretical guarantees on the recovery of a good clustering partition.

Plan of the Chapter The remaining of this Chapter is organized as follows.

Before introducing in Section 5.3 the new proposed MST-based algorithm for clustering - DBMSTClu (DB for Density-Based) - theoretical motivation is given to justify the relying on a Minimum Spanning Tree in Section 5.2.

Theoretical guarantees of DBMSTClu are discussed in Section 5.4 and the implementation enabling its scalability is detailed in Section 5.5.

Then, Section 5.6 recalls the fundamentals of the sketching technique that can be used to obtain an approximate MST of the dissimilarity graph required as input of DBMSTClu[4].

Section 5.7 presents the experimental results comparing the proposed clustering algorithm to some other existing methods.

Finally, Section 5.8 concludes the Chapter and discusses future directions.

5.2 Theoretical framework motivating MST-based clustering methods

As previously stated, DBMSTClu is a member of the MST-based clustering algorithms family. The latter, however efficient, lacked of proper motivation. To this end, this Section closes the gap by providing a theoretical framework for MST-based clustering methods.

Before justifying the use of MST-based clustering algorithms by defininig properly the concept of a cluster in Section 5.2.1, used notations are defined in Section 5.2.2.

Note that the term MST is kept in the whole Chapter for simplicity, but the conceived algorithm also works on a Minimum Spanning Forest (MSF) if the initial graph is disconnected.

5.2.1 Further notations

We keep notation from Section 2.4 but for convenience, we recall it here with some new ones. Let $\mathcal{G} = (V, E, w)$ be a simple undirected weighted graph with a vertex set V, an edge set E, and a weight function $w := E \to \mathbb{R}$ but we assume the weight function $w := E \to (0, 1]$ for the clustering algorithm. One will respectively call the edge set and the node set of a graph $\mathcal{G}$ using the applications $E(\mathcal{G})$ and $V(\mathcal{G})$. Given a node set $S \subset V$, one denotes by $\mathcal{G}_{|S}$ the subgraph induced by S.

[4]Full details of the sketching technique are given in Chapter 2, Section 2.4.2 from p.45.

We call $G = (V, E)$ the topology of the graph. Cursive letters are used to represent weighted graphs and straight letters refer to topological arguments.

Since graphs are simple, the path $\mathcal{P}_{u-v}$ between two vertices u and v is characterized either as the ordered sequence of vertices $\{u, \dots, v\}$ or the corresponding binding edges depending on the context. Besides, edges e_{ij} denote an edge between nodes i and j.

In practice, either such an underlying network already exists and is the graph data $\mathcal{G}$ or $\mathcal{G}$ can be built as the dissimilarity graph between the points from a dataset: the points of the dataset are the vertices and the weighted edges express dissimilarities or "distances" between these objects. For instance, one can take the Euclidean distance (see Definition A.0.4 p.128 for a recall).

5.2.2 Theoretical justification by notion of Cluster

The notion of Cluster later explicited relies on the following definition of the *minimum path distance* between two nodes in the graph.

Definition 5.2.1 (Minimum path distance)**.** *Let be $\mathcal{G} = (V, E, w)$ and $u, v \in V$. The* minimum path distance *between u and v is*

$$dist_{\mathcal{G}}(u, v) = \min_{\mathcal{P}_{u-v}} \sum_{e \in \mathcal{P}_{u-v}} w(e) \tag{5.1}$$

with $\mathcal{P}_{u-v}$ a path (edge version) from u to v in $\mathcal{G}$.

Definition 5.2.2 (Cluster)**.** *Let be $\mathcal{G} = (V, E, w)$ a graph, $w := E \to (0, 1]$, $(V, dist_{\mathcal{G}})$ a metric space based on the minimum path distance $dist_{\mathcal{G}}$ defined on $\mathcal{G}$ and $D \subset V$ a node set. $C \subset D$ is a cluster if and only if $|C| > 2$ and $\forall C_1, C_2$ s.t. $C = C_1 \cup C_2$ and $C_1 \cap C_2 = \emptyset$, one has:*

$$\underset{z \in D \setminus C_1}{\operatorname{argmin}} \{ \min_{v \in C_1} dist_{\mathcal{G}}(z, v) \} \subset C_2 \tag{5.2}$$

Assuming that a cluster is built of at least 3 points makes sense since singletons or groups of 2 nodes can be legitimately considered as noise. For simplicity of the proofs, the following theorems hold in the case where noise is neglected. However, they are still valid in the setting where noise is considered as singletons (with each singleton representing a generalized notion of cluster).

Theorem 5.2.1. *Let be $\mathcal{G} = (V, E, w)$ a graph and $\mathcal{T}$ a minimum spanning tree of $\mathcal{G}$. Let also be C a cluster in the sense of Definition 5.2.2 and two vertices $v_1, v_2 \in C$. Then, $V_{\mathcal{P}_{v_1-v_2}} \subset C$ with $\mathcal{P}_{v_1-v_2}$ a path from v_1 to v_2 in $\mathcal{G}$, and $V_{\mathcal{P}_{v_1-v_2}}$ the set of vertices contained in $\mathcal{P}_{v_1-v_2}$.*

Proof. Let be $v_1, v_2 \in C$. If v_1 and v_2 are neighbors, the result is trivial. Otherwise, as $\mathcal{T}$ is a tree, there exists a unique path within $\mathcal{T}$ between v_1 and v_2 denoted by $\mathcal{P}_{v_1-v_2} = \{v_1, \dots, v_2\}$. Let now prove by *reductio ad absurdum* that $V_{\mathcal{P}_{v_1-v_2}} \subset C$. Suppose there is $h \in V_{\mathcal{P}_{v_1-v_2}}$ s.t. $h \notin C$. We will see that it leads to a contradiction. We set C_1 to be the largest connected component (regarding the number of vertices) of $\mathcal{T}$ s.t. $v_1 \in C_1$, and every nodes from C_1 are in C. Because of h's definition, $v_2 \notin C_1$. Let be $C_2 = C \setminus C_1$. $C_2 \neq \emptyset$ since $v_2 \in C_2$. Let be $z^* \in \underset{z \in V \setminus C_1}{\operatorname{argmin}} \{ \min_{v \in C_1} dist_{\mathcal{G}}(z, v) \}$ and $e^* = (z^*, v^*)$ an edge that reaches this minimum. Let us show that $z^* \notin C$. If $z^* \in C$, then two possibilities hold:

1. There is an edge $e_{z^*} \in \mathcal{T}$, s.t. $e_{z^*} = (z^*, z')$ with $z' \in C_1$. This is impossible, otherwise by definition of a connected component, $z^* \in C_1$. Contradiction.

2. For all $e_{z^*} = (z^*, z')$ s.t $z' \in C_1$, one has $e_{z^*} \notin \mathcal{T}$. In particular, $e^* \notin \mathcal{T}$. Since h is the neighbor of C_1 in $\mathcal{G}$, there is also $e_h \in \mathcal{T}$, s.t. $e_h = (h, h')$ with $h' \in C_1$. Once again two possibilities hold:

   (a) $w(e_{z^*}) = \min_{z \in V \setminus C_1} \{ \min_{v \in C_1} dist_{\mathcal{G}}(z, v) \} < w(e_h)$. Then, if we replace e_h by e_{z^*} in $\mathcal{T}$, its total weight decreases. So $\mathcal{T}$ is not a minimum spanning tree. Contradiction.

   (b) $w(e_{z^*}) = w(e_h)$, therefore $h \in \operatorname{argmin}_{z \in V \setminus C_1} \{ \min_{v \in C_1} dist_{\mathcal{G}}(z, v) \}$. Since $h \notin C$, one gets that $\operatorname{argmin}_{z \in V \setminus C_1} \{ \min_{v \in C_1} dist_{\mathcal{G}}(z, v) \} \not\subset C_2$. Thus, C is not a cluster. Contradiction.

It has been proven that $z^* \notin C$. In particular, $z^* \notin C_2$. Then, $\operatorname{argmin}_{z \in V \setminus C_1} \{ \min_{v \in C_1} dist_{\mathcal{G}}(z, v) \} \not\subset C_2$. Thus, C is not a cluster. Contradiction. Finally $h \in C$ and $V_{\mathcal{P}_{v_1 - v_2}} \subset C$. □

Theorem 5.2.1 states that, given a graph $\mathcal{G}$, an MST $\mathcal{T}$, and any two nodes of C, every node in the path between them is in C. This means that a cluster can be characterized by a subtree of $\mathcal{T}$. It justifies the use of all MST-based methods for data clustering or node clustering in a graph.

> All the clustering algorithms based on successively cutting edges in an MST to obtain a subtree forest are meaningful in the sense of Theorem 5.2.1.

In particular, this Theorem holds for the use of DBMSTClu presented in Section 5.3.

5.3 DBMSTClu, MST-based clustering

Now that the use of MST-based clustering methods has been legitimized, this Section introduces the proposed DBMSTClu algorithm.

5.3.1 DBMSTClu principle

Let us consider $\mathcal{T}$ an MST of $\mathcal{G}$, as the unique input of the clustering algorithm DBMSTClu. The clustering partition results then from successive cuts on $\mathcal{T}$ so that a new cut in $\mathcal{T}$ splits a connected component into two new ones. Each final connected component, a subtree of $\mathcal{T}$, represents a cluster. Initially, $\mathcal{T}$ is one cluster containing all nodes. Then, at each iteration, an edge is cut if some criterion, called *Validity Index of a Clustering Partition* (DBCVI) is improved. This edge is greedily chosen to locally maximize the DBCVI at each step. When no improvement on DBCVI can be further made, the algorithm stops. The basic scheme of this algorithm is described in Figure 5.1.

The DBCVI is defined as the weighted average of all *cluster validity indices* which are based on two positive quantities, the *Dispersion* and the *Separation* of a cluster:

5.3.2 Separation, dispersion and validity indices concepts

Definition 5.3.1 (Cluster Dispersion)**.** *The* Dispersion *of a cluster C_i (*DISP*) is defined as the maximum edge weight of C_i. If the cluster is a singleton (i.e. contains*

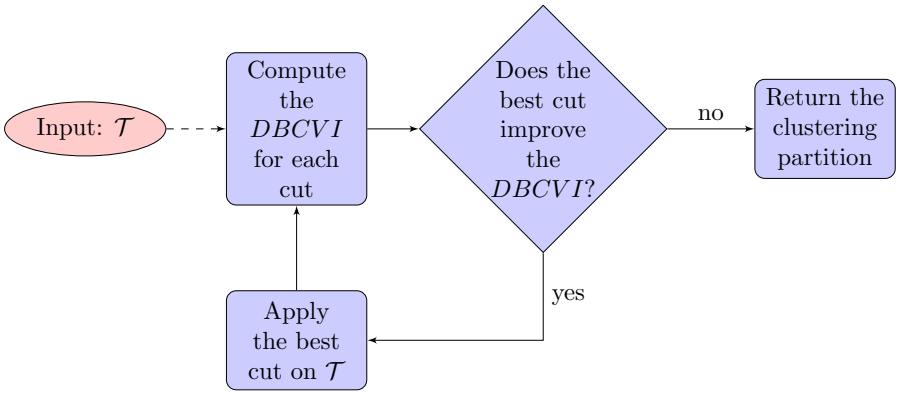


FIGURE 5.1: Basic scheme illustrating DBMSTClu algorithm.

only one node), the associated Dispersion is set to 0*. More formally:*

$$\forall i \in [K],\ \mathrm{DISP}(C_i) := \begin{cases} \max\limits_{j,\ e_j \in C_i} w_j & \text{if } |E(C_i)| \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

Definition 5.3.2 (Cluster Separation)**.** *The* Separation *of a cluster* C_i *(*SEP*) is defined as the minimum distance between the nodes of* C_i *and the ones of all other clusters* $C_j, j \neq i, 1 \leq i, j \leq K, K \neq 1$ *where* K *is the total number of clusters. In practice, it corresponds to the minimum weight among all already cut edges from* $\mathcal{T}$ *comprising a node from* C_i*. If* $K = 1$*, the Separation is set to* 1*. More formally, with* $incCuts(C_i)$ *denoting cut edges incident to* C_i*,*

$$\forall i \in [K],\ \mathrm{SEP}(C_i) := \begin{cases} \min\limits_{j,\ e_j \in incCuts(C_i)} w_j & \text{if } K \neq 1 \\ 1 & \text{otherwise.} \end{cases} \tag{5.4}$$

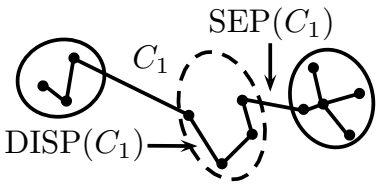


FIGURE 5.2: SEP and DISP definitions with $N = 12$, $K = 3$ for dashed cluster C_1 in the middle.

Figure 5.2 sums up the introduced definitions. The higher the Separation, the farther is the cluster separated from the other clusters, while low values suggest that the cluster is close to the nearest one.

Definition 5.3.3 (Validity Index of a Cluster)**.** *The* Validity Index of a Cluster C_i *is defined as:*

$$V_C(C_i) := \frac{\text{SEP}(C_i) - \text{DISP}(C_i)}{\max(\text{SEP}(C_i), \text{DISP}(C_i))} \in [-1; 1] \tag{5.5}$$

The Validity Index of a Cluster (illustration in Figure 5.3) is defined s.t. $-1 \leq V_C(C_i) \leq 1$ where 1 stands for the best validity index and -1 for the worst one. No division by zero (i.e. $\max(\text{DISP}(C_i), \text{SEP}(C_i)) = 0$) happens because Separation is always strictly positive. When Dispersion is higher than Separation, $-1 < V_C(C_i) < 0$. Conversely, when Separation is higher than Dispersion, $0 < V_C(C_i) < 1$. So our clustering algorithm will naturally encourage clusters with a higher Separation over those with a higher Dispersion.

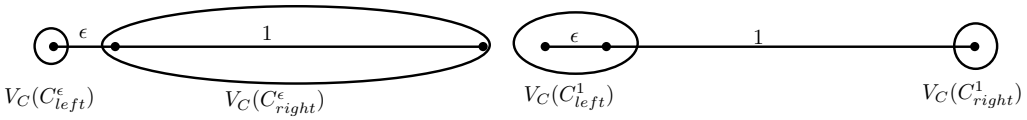


FIGURE 5.3: Validity Index of a Cluster's example with $N = 3$. For a small ϵ, cutting edge with weight ϵ or 1 gives respectively the left and right partitions.
(left) $V_C(C^{\epsilon}_{left}) = 1$; $V_C(C^{\epsilon}_{right}) = \epsilon - 1 < 0$.
(right) $V_C(C^{1}_{left}) = 1 - \epsilon > 0$; $V_C(C^{1}_{right}) = 1$.
The right partition, for which validity indices of each cluster are positive, is preferred.

Definition 5.3.4 (Density-Based Validity Index of a Clustering Partition)**.** *The* Density-Based Validity Index of a Clustering partition $\Pi = \{C_i\}, 1 \leq i \leq K$ *noted* DBCVI(Π) *is defined as the weighted average of the Validity Indices of all clusters in the partition where N is the number of vertices.*

$$\text{DBCVI}(\Pi) := \sum_{i=1}^{K} \frac{|C_i|}{N} V_C(C_i) \in [-1, 1] \tag{5.6}$$

The Density-Based Validity Index of Clustering lies also between -1 and 1 where 1 stands for an optimal density-based clustering partition while -1 stands for the worst one.

Our defined quantities are significantly distinct from the *separation* and *sparseness* defined in work from Moulavi et al. [2014]. Indeed, firstly, their quantities are not well defined for the special cases when clusters have less than four nodes or a partition containing a lonely cluster. Secondly, the way they differentiate internal and external nodes or edges does not properly recover easy clusters like convex blobs. Moreover, our DBCVI differs from the Silhouette Coefficient [Rousseeuw, 1987]. The latter does not perform well with non-convex-shaped clusters and although this is based on close concepts like *tightness* and also *separation*, the global coefficient is based on the average values of Silhouette coefficients of each point, while our computation of DBCVI begins at the cluster level.

5.3.3 DBMSTClu algorithm

DBMSTClu is summarized in Algorithm 6. In the latter, evaluateCut(.) computes the DBCVI when the cut in parameter is applied to $\mathcal{T}$.

The algorithm starts from a partition with one cluster containing the whole dataset whereas the associated initial DBCVI is set to the worst possible value: -1. As long as there exists a cut which makes the DBCVI greater from (or equal to) the one of the current partition, a cut is greedily chosen by maximizing the obtained DBCVI among all the possible cuts.

When no direct improvement is possible, the algorithm stops. It is guaranteed that the cut edge locally maximizes the DBCVI at each iteration since by construction, the algorithm will try each possible cut. In practice, the algorithm stops after a reasonable number of cuts, getting trapped in a local maximum corresponding to a meaningful cluster partition. This prevents from obtaining a partition where all the points are in singleton clusters. Indeed, such a result ($K = N$) is not desirable, although it is optimal in the sense of the DBCVI, since in this case, $\forall i \in [K]$, $\text{DISP}(C_i) = 0$ and $V_C(C_i) = 1$. Moreover, the non-parametric characteristic helps achieving stable partitions. In the next Section, several insights are given to support theoretically these claims.

Algorithm 6 DBMSTClu($\mathcal{T}$)

Input: $\mathcal{T}$, the MST
$dbcvi \leftarrow -1.0$
$clusters \leftarrow \emptyset$
$cut\_list \leftarrow \{E(\mathcal{T})\}$
while $dbcvi < 1.0$ **do**
  $cut\_tp \leftarrow \emptyset$
  $dbcvi\_tp \leftarrow dbcvi$
  for each cut in cut_list **do**
    $newDbcvi \leftarrow \text{evaluateCut}(\mathcal{T}, cut)$
    if $newDbcvi \geq dbcvi\_tp$ **then**
      $cut\_tp \leftarrow cut$
      $dbcvi\_tp \leftarrow newDbcvi$
  if $cut\_tp \neq \emptyset$ **then**
    $clusters \leftarrow \text{cut}(clusters, cut\_tp)$
    $dbcvi \leftarrow dbcvi\_tp$
    $cut\_list \leftarrow cut\_list \backslash \{cut\_tp\}$
  else
    break
return $clusters$, $dbcvi$

5.4 Theoretical guarantees for DBMSTClu

Section 5.4.1 gives theoretical guarantees on the exact recovery of an underlying clustering partition. In Section 5.4.2, further theoretical results are presented to give more intuition on the way the algorithm works.

All presented results are completely independent on the way an exact or approximate MST is computed. In particular, they are independent of the sketching

phase which can be used to retrieve efficently an approximate MST.

5.4.1 DBMSTClu exact clustering recovery proof

In this Section, please note that the notion of cluster follows Definition 5.2.2 to achieve theoretical guarantees. Let us first begin by introducing some definitions.

Definition 5.4.1 (Cut)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K clusters, $\mathcal{T}$ an MST of $\mathcal{G}$. Let denote $(C_i^*)_{i\in[K]}$ the set of the clusters. Then, $Cut_{\mathcal{G}}(\mathcal{T}) := \{e_{kl} \in \mathcal{T} \mid k \in C_i^*,\ l \in C_j^*,\ i, j \in [K]^2,\ i \neq j\}$. In the sequel, for simplicity, we denote $e^{(ij)} \in Cut_{\mathcal{G}}(\mathcal{T})$ the edge between cluster C_i^* and C_j^*.*

$Cut_{\mathcal{G}}(\mathcal{T})$ is basically the set of effective cuts to perform on $\mathcal{T}$ in order to ensure the exact recovery of the clustering partition. More generally, trees on which $Cut_{\mathcal{G}}(.)$ enables to find the right partition are said to be a *partitioning topology* (see Definition 5.4.2).

Definition 5.4.2 (Partitionning topology)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K clusters $C_1^*, \ldots, C_K^*$. A spanning tree $\mathcal{T}$ of $\mathcal{G}$ is said to have a* partitioning topology *if $\forall i, j \in [K]$, $i \neq j$, $|\{e = (u, v) \in Cut_{\mathcal{G}}(\mathcal{T}) \mid u \in C_i^*, v \in C_j^*\}| = 1$.*

Definitions 5.4.1 and 5.4.2 introduce a topological condition on the tree as input of the algorithm. Nevertheless, conditions on weights are necessary too. Hence, we define *homogeneous separability* which expresses the fact that within a cluster the edge weights are spread in a controlled manner.

Definition 5.4.3 (Homogeneous separability condition)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$, $s \in E$ and $\mathcal{T}$ a tree of $\mathcal{G}$. $\mathcal{T}$ is said to be* homogeneously separable *by s, if*

$$\alpha_{\mathcal{T}} \max_{e\in E(\mathcal{T})} w(e) < w(s) \text{ with } \alpha_{\mathcal{T}} = \frac{\max\limits_{e\in E(\mathcal{T})} w(e)}{\min\limits_{e\in E(\mathcal{T})} w(e)} \geq 1. \tag{5.7}$$

One will write for simplicity that $H_{\mathcal{T}}(s)$ is verified.

The accuracy of DBMSTClu is proven under the *weak homogeneity condition* given by Definition 5.4.4.

Definition 5.4.4 (Weak homogeneity condition of a Cluster)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K clusters $C_1^*, \ldots, C_K^*$. A given cluster C_i^*, $i \in [K]$, is* weakly homogeneous *if: for all $\mathcal{T}$ an MST of $\mathcal{G}$, and $\forall j \in [K]$, $j \neq i$, s.t. $e^{(ij)} \in Cut_{\mathcal{G}}(\mathcal{T})$, $H_{\mathcal{T}_{|C_i^*}}(e^{(ij)})$ is verified. For simplicity, one denote $\underline{\alpha}_i := \max\limits_{\mathcal{T} \text{ MST of } \mathcal{G}} \alpha_{\mathcal{T}_{|C_i^*}}$*

Now the proof on accuracy is finally stated by Corollary 5.4.1 which relies on the Theorems 5.4.1, 5.4.2 and 5.4.3.

Theorem 5.4.1. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K homogeneous clusters $C_1^*, \ldots, C_K^*$ and $\mathcal{T}$ an MST of $\mathcal{G}$. Let now assume that at step $k < K-1$, DBMSTClu built $k+1$ subtrees $\mathcal{C}_1, \ldots, \mathcal{C}_{k+1}$ by cutting $e_1,\ e_2,\ \ldots,\ e_k \in E$.*

Then, $Cut_k := Cut_{\mathcal{G}}(\mathcal{T}) \setminus \{e_1,\ e_2,\ \ldots,\ e_k\} \neq \emptyset \implies \text{DBCVI}_{k+1} \geq DBCVI_k$, i.e. if there are still edges in Cut_k, the algorithm will continue to perform some cut.

Proof of Theorem 5.4.1 relies on the following lemma:

Lemma 5.4.1. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K clusters $C_1^*, \ldots, C_K^*$ and $\mathcal{T}$ an MST of $\mathcal{G}$. If for all $i \in [K]$, C_i^* is weakly homogeneous, then $\operatorname{argmax}\limits_{e\in\mathcal{T}} w(e) \subset Cut_{\mathcal{G}}(\mathcal{T})$ i.e. the heaviest edges in $\mathcal{T}$ are in $Cut_{\mathcal{G}}(\mathcal{T})$.*

Proof. Let us consider C_i^* a cluster of $\mathcal{G}$. As C_i^* is weakly homogeneous, $\forall j \in [K]$ s.t. $e^{(ij)} \in Cut_{\mathcal{G}}(\mathcal{T})$, $\max_{e \in \mathcal{T}_{|C_i^*}} w(e) < w(e^{(ij)})$. Hence, $\operatorname{argmax}_{e \in E(\mathcal{T})} w(e) \subset Cut_{\mathcal{G}}(\mathcal{T})$. □

Now proof of Theorem 5.4.1 can be given.

Proof. Let note DBCVI at step k, $DBCVI_k = \sum_{i=1}^{k+1} \frac{|\mathcal{C}_i|}{N} V_C(\mathcal{C}_i)$. Let assume that $Cut_k \neq \emptyset$. Therefore, there is $e^* \in Cut_k$ and $i \in \{1, \ldots, k+1\}$ s.t. $e^* \in E(\mathcal{C}_i)$. Since $e^* \in Cut_{\mathcal{G}}(\mathcal{T})$, using Lemma 5.4.1, one can always take $e^* \in \operatorname{argmax}_{e \in E(\mathcal{C}_i)} w(e)$. Then, if we denote $\mathcal{C}_i^1$, $\mathcal{C}_i^2$ the two subtrees of $\mathcal{C}_i$ induced by the cut of e^* (see Figure 5.4 for an illustration) and $DBCVI_{k+1}(e^*)$ the associated DBCVI value,

$$\Delta = DBCVI_{k+1}(e^*) - DBCVI_k \tag{5.8}$$

$$= \frac{|\mathcal{C}_i^1|}{N} \underbrace{\left(\frac{\text{SEP}(\mathcal{C}_i^1) - \text{DISP}(\mathcal{C}_i^1)}{\max(\text{SEP}(\mathcal{C}_i^1), \text{DISP}(\mathcal{C}_i^1))} \right)}_{V_C(\mathcal{C}_i^1)} + \frac{|\mathcal{C}_i^2|}{N} \underbrace{\left(\frac{\text{SEP}(\mathcal{C}_i^2) - \text{DISP}(\mathcal{C}_i^2)}{\max(\text{SEP}(\mathcal{C}_i^2), \text{DISP}(\mathcal{C}_i^2))} \right)}_{V_C(\mathcal{C}_i^2)}$$

$$- \frac{|\mathcal{C}_i|}{N} \underbrace{\left(\frac{\text{SEP}(\mathcal{C}_i) - \text{DISP}(\mathcal{C}_i)}{\max(\text{SEP}(\mathcal{C}_i), \text{DISP}(\mathcal{C}_i))} \right)}_{V_C(\mathcal{C}_i)}. \tag{5.9}$$

There are two possible cases:

1. $V_C(\mathcal{C}_i) \leq 0$, then $\text{SEP}(\mathcal{C}_i) \leq \text{DISP}(\mathcal{C}_i) = w(e^*)$. As for $l \in \{1,2\}$, $\text{SEP}(\mathcal{C}_i^l) \geq \text{SEP}(\mathcal{C}_i)$ and $\text{DISP}(\mathcal{C}_i^l) \leq \text{DISP}(\mathcal{C}_i)$ because $e^* \in \operatorname{argmax}_{e \in E(\mathcal{C}_i)} w(e)$, then, for $l \in \{1,2\}$,

$$\frac{\text{SEP}(\mathcal{C}_i^l) - \text{DISP}(\mathcal{C}_i^l)}{\max(\text{SEP}(\mathcal{C}_i^l), \text{DISP}(\mathcal{C}_i^l))} \geq \frac{\text{SEP}(\mathcal{C}_l) - \text{DISP}(\mathcal{C}_i)}{\max(\text{SEP}(\mathcal{C}_i), \text{DISP}(\mathcal{C}_i))} = \frac{\text{SEP}(\mathcal{C}_i)}{w(e)} - 1 \tag{5.10}$$

   and $\Delta \geq 0$.

2. $V_C(\mathcal{C}_i) \geq 0$, then $\text{SEP}(\mathcal{C}_i) \geq \text{DISP}(\mathcal{C}_i) = w(e^*)$ i.e. $\max(\text{SEP}(\mathcal{C}_i), \text{DISP}(\mathcal{C}_i)) = \text{SEP}(\mathcal{C}_i)$, for $l \in \{1,2\}$, $\text{DISP}(\mathcal{C}_i^l) \leq \text{DISP}(\mathcal{C}_i)$ i.e. $\text{DISP}(\mathcal{C}_i^l) \leq w(e^*)$, $\text{SEP}(\mathcal{C}_i^l) = w(e^*)$ hence $\text{SEP}(\mathcal{C}_i^l) \geq \text{DISP}(\mathcal{C}_i^l)$. Thus, $V_C(\mathcal{C}_i) = 1 - \frac{\text{DISP}(\mathcal{C}_i)}{\text{SEP}(\mathcal{C}_i)}$ and for $l \in \{1,2\}$, $V_C(\mathcal{C}_i^l) = 1 - \frac{\text{DISP}(\mathcal{C}_i^l)}{\text{SEP}(\mathcal{C}_i^l)}$. Then, for $l \in \{1,2\}$, $V_C(\mathcal{C}_i^l) \geq V_C(\mathcal{C}_i)$ and $\Delta \geq 0$.

For both cases, $\Delta = DBCVI_{k+1}(e^*) - DBCVI_k \geq 0$. Hence, at least the cut of e^* improves the current DBCVI, so the algorithm will perform a cut at this stage.

□

Theorem 5.4.2. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K homogeneous clusters $C_1^*, \ldots, C_K^*$ and $\mathcal{T}$ an MST of $\mathcal{G}$.*

Assume now that at step $k < K-1$, DBMSTClu built $k+1$ subtrees $\mathcal{C}_1, \ldots, \mathcal{C}_{k+1}$ by cutting $e_1, e_2, \ldots, e_k \in E$. We still denote $Cut_k := Cut_{\mathcal{G}}(\mathcal{T}) \backslash \{e_1, e_2, \ldots, e_k\}$.

If $Cut_k \neq \emptyset$ then $\operatorname{argmax}_{e \in \mathcal{T} \backslash \{e_1, e_2, \ldots, e_k\}} DBCVI_{k+1}(e) \subset Cut_k$ i.e. the cut edge at step $k+1$ is in Cut_k.

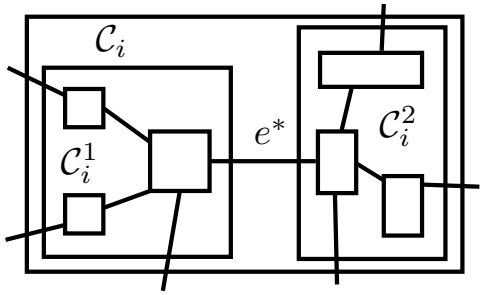


FIGURE 5.4: Illustration for Theorem 5.4.1's proof.

Proof. It is sufficient to show that, at step k, if there exists an edge e^* from Cut_k whose cut builds two clusters, then e^* maximizes DBCVI among all possible cuts in the union of itself and both resulting clusters. Indeed, showing this for two clusters, one can easily generalize to the whole graph as a combination of couples of clusters (see Figure 5.6 for an illustration): if for each couple, the best local solution is in Cut_k, then the best general solution is necessary in Cut_k.

Let us consider at step k of the algorithm two clusters C_1^* and C_2^* such that e^* the edge separating them in $\mathcal{T}$ is in Cut_k (see Figure 5.5 for an illustration). For readability we denote $\mathcal{T}_{|C_1^*} = \mathcal{C}_1^*$ and $\mathcal{T}_{|C_2^*} = \mathcal{C}_2^*$ Let us proof that for all $\tilde{e} \in \mathcal{T}_{|C_1^* \cup C_2^*}$, one has: $DBCVI_{k+1}(e^*) > DBCVI_{k+1}(\tilde{e})$. W.l.o.g. let assume $\tilde{e} \in \mathcal{C}_1^*$ and let denote $\mathcal{C}_{1,1}^*$ and $\mathcal{C}_{1,2}^*$ the resulting subtrees from the cut of $\tilde{e}$. We still denote $DBCVI_{k+1}(e)$ the value of the DBCVI at step $k+1$ for the cut of e.

$$\Delta := DBCVI_{k+1}(e^*) - DBCVI_{k+1}(\tilde{e}) \tag{5.11}$$

$$= \underbrace{\frac{|\mathcal{C}_1^*|}{N}\left(\frac{\text{SEP}(\mathcal{C}_1^*) - \text{DISP}(\mathcal{C}_1^*)}{\max(\text{SEP}(\mathcal{C}_1^*), \text{DISP}(\mathcal{C}_1^*))}\right) + \frac{|\mathcal{C}_2^*|}{N}\left(\frac{\text{SEP}(\mathcal{C}_2^*) - \text{DISP}(\mathcal{C}_2^*)}{\max(\text{SEP}(\mathcal{C}_2^*), \text{DISP}(\mathcal{C}_2^*))}\right)}_{A}$$

$$- \underbrace{\left(\frac{|\mathcal{C}_{1,1}^*|}{N}\left(\frac{\text{SEP}(\mathcal{C}_{1,1}^*) - \text{DISP}(\mathcal{C}_{1,1}^*)}{\max(\text{SEP}(\mathcal{C}_{1,1}^*), \text{DISP}(\mathcal{C}_{1,1}^*))}\right) + \frac{|\mathcal{C}_{1,2}^*|}{N}\left(\frac{\text{SEP}(\mathcal{C}_{1,2}^*) - \text{DISP}(\mathcal{C}_{1,2}^*)}{\max(\text{SEP}(\mathcal{C}_{1,2}^*), \text{DISP}(\mathcal{C}_{1,2}^*))}\right)\right)}_{B} \tag{5.12}$$

By weak homogeneity of C_1^* and C_2^*,

$$A = \frac{|\mathcal{C}_1^*|}{N}\left(1 - \frac{\text{DISP}(\mathcal{C}_1^*)}{\text{SEP}(\mathcal{C}_1^*)}\right) + \frac{|\mathcal{C}_2^*|}{N}\left(1 - \frac{\text{DISP}(\mathcal{C}_2^*)}{\text{SEP}(\mathcal{C}_2^*)}\right) > 0, \tag{5.13}$$

$$B = \underbrace{\frac{|\mathcal{C}_{1,1}^*|}{N}\left(\frac{\text{SEP}(\mathcal{C}_{1,1}^*) - \text{DISP}(\mathcal{C}_{1,1}^*)}{\max(\text{SEP}(\mathcal{C}_{1,1}^*), \text{DISP}(\mathcal{C}_{1,1}^*))}\right)}_{B_1} + \underbrace{\frac{|\mathcal{C}_{1,2}^*|}{N}\left(\frac{\text{SEP}(\mathcal{C}_{1,2}^*) - \text{DISP}(\mathcal{C}_{1,2}^*)}{\max(\text{SEP}(\mathcal{C}_{1,2}^*), \text{DISP}(\mathcal{C}_{1,2}^*))}\right)}_{B_2}. \tag{5.14}$$

By Lemma 5.4.1's proof, $e^* \in \underset{e \in E(\mathcal{T}_{|C_1^* \cup C_2^*})}{\operatorname{argmax}} w(e)$ so $\mathrm{DISP}(\mathcal{C}_{1,2}^*) = w(e^*)$.

Since $e^* \in Cut_{\mathcal{G}}(\mathcal{T})$, one has $w(e^*) \geq \max(\mathrm{SEP}(\mathcal{C}_1^*), \mathrm{SEP}(\mathcal{C}_2^*))$. Moreover, as $\mathcal{C}_2^*$ is a subtree of $\mathcal{C}_{1,2}^*$, then $\mathrm{SEP}(\mathcal{C}_{1,2}^*) \leq \mathrm{SEP}(\mathcal{C}_2^*)$. Thus, $w(e^*) \geq \mathrm{SEP}(\mathcal{C}_{1,2}^*)$. Finally, $B_2 = \frac{|\mathcal{C}_{1,2}^*|}{N}\left(\frac{\mathrm{SEP}(\mathcal{C}_{1,2}^*)}{\mathrm{DISP}(\mathcal{C}_{1,2}^*)} - 1\right) \leq 0$.

Besides, $w(\tilde{e}) \leq \mathrm{SEP}(\mathcal{C}_1^*) \implies \mathrm{SEP}(\mathcal{C}_{1,1}^*) = w(\tilde{e}) \leq \max_{e \in E(\mathcal{C}_1^*)} w(e)$ and $\mathrm{DISP}(\mathcal{C}_{1,1}^*) = \max_{e \in E(\mathcal{C}_{1,1}^*)} w(e) \geq \min_{e \in E(\mathcal{C}_1^*)} w(e)$. Then, two possibilities hold:

1. $B_1 < 0 \implies B < 0 < A$.
2. $B_1 \geq 0$, thus one has $B_1 = \frac{|\mathcal{C}_{1,1}^*|}{N}\left(1 - \frac{\mathrm{DISP}(\mathcal{C}_{1,1}^*)}{\mathrm{SEP}(\mathcal{C}_{1,1}^*)}\right) \leq \frac{|\mathcal{C}_{1,1}^*|}{N}\left(1 - \frac{\min\limits_{e \in \mathcal{C}_1^*} w(e)}{\max\limits_{e \in \mathcal{C}_1^*} w(e)}\right)$.

   Under weak homogeneity condition, there is: $\frac{\mathrm{DISP}(C_1^*)}{\mathrm{SEP}(C_1^*)} < \frac{\min\limits_{e \in \mathcal{C}_1^*} w(e)}{\max\limits_{e \in \mathcal{C}_1^*} w(e)}$. Thus,

$$B_1 < \frac{|C_{1,1}^*|}{N}\left(1 - \frac{\mathrm{DISP}(\mathcal{C}_1^*)}{\mathrm{SEP}(\mathcal{C}_1^*)}\right) \tag{5.15}$$

$$< \frac{|\mathcal{C}_1^*|}{N}\left(1 - \frac{\mathrm{DISP}(\mathcal{C}_1^*)}{\mathrm{SEP}(\mathcal{C}_1^*)}\right) \text{ because } \mathcal{C}_{1,1}^* \text{ is a subtree of } \mathcal{C}_1^* \tag{5.16}$$

$$< A \tag{5.17}$$

So, $B_1 + B_2 = B < A = DBCVI_{k+1}(e^*)$.

Since $B < A$, $\Delta > 0$ and e^* maximizes DBCVI among all possible cuts in the union of itself and both resulting clusters. Q.E.D. □

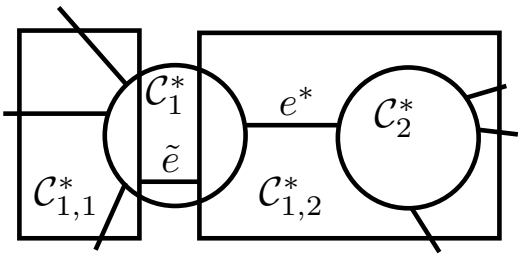


FIGURE 5.5: Illustration for Theorem 5.4.2's proof.

Theorem 5.4.3. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K weakly homogeneous clusters $C_1^*, \ldots, C_K^*$ and $\mathcal{T}$ an MST of $\mathcal{G}$. Let now assume that at step $K-1$, DBMSTClu built K subtrees $\mathcal{C}_1, \ldots, \mathcal{C}_K$ by cutting $e_1, e_2, \ldots, e_{K-1} \in E$. We still denote $Cut_{K-1} := Cut_{\mathcal{G}}(\mathcal{T}) \backslash \{e_1, e_2, \ldots, e_{K-1}\}$.*

Then, for all $e \in \mathcal{T} \backslash \{e_1, e_2, \ldots, e_{K-1}\}$, $DBCVI_K(e) < DBCVI_{K-1}$ i.e. the algorithm stops: no edge gets cut during step K.

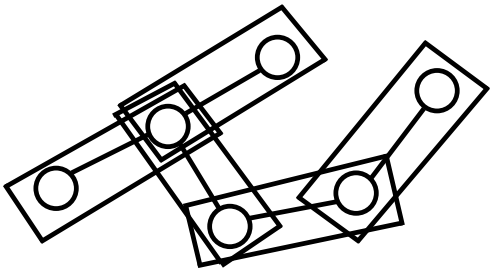


FIGURE 5.6: Illustration for Theorem 5.4.2's proof. Each circle corresponds to a cluster. The six clusters are handled within five couples of clusters.

Proof. According to Theorem 5.4.1 and Theorem 5.4.2, for all $k < K$, if $Cut_k \neq \emptyset$, the algorithm performs some cut from $Cut_{\mathcal{G}}(\mathcal{T})$. We still denote for all $j \in [K]$ $\mathcal{C}_j^* = \mathcal{T}_{|C_j^*}$. Since $|Cut_{\mathcal{G}}(\mathcal{T})| = K-1$, the $K-1$ first steps produce $K-1$ cuts from $Cut_{\mathcal{G}}(\mathcal{T})$. Therefore, $\text{DBCVI}_{K-1} = \sum_{j\in[K-1]} \frac{|\mathcal{C}_j^*|}{N} V_C(\mathcal{C}_j^*)$.

Let be e the (expected) edge cut at step K, splitting the tree $\mathcal{C}_i^*$ into $\mathcal{C}_{i,1}^*$ and $\mathcal{C}_{i,2}^*$.

$$\Delta = \text{DBCVI}_{K-1} - \text{DBCVI}_K \tag{5.18}$$

$$= \frac{|\mathcal{C}_i^*|}{N} V_C(\mathcal{C}_i^*) - \frac{|\mathcal{C}_{i,1}^*|}{N} V_C(\mathcal{C}_{i,1}^*) - \frac{|\mathcal{C}_{i,2}^*|}{N} V_C(\mathcal{C}_{i,2}^*) \tag{5.19}$$

$$= \frac{|\mathcal{C}_i^*|}{N} \frac{\text{SEP}(\mathcal{C}_i^*) - \text{DISP}(\mathcal{C}_i^*)}{\max(\text{SEP}(\mathcal{C}_i^*), \text{DISP}(\mathcal{C}_i^*))} - \frac{|\mathcal{C}_{i,1}^*|}{N} \frac{\text{SEP}(\mathcal{C}_{i,1}^*) - \text{DISP}(\mathcal{C}_{i,1}^*)}{\max(\text{SEP}(\mathcal{C}_{i,1}^*), \text{DISP}(\mathcal{C}_{i,1}^*))}$$
$$- \frac{|\mathcal{C}_{i,2}^*|}{N} \frac{\text{SEP}(\mathcal{C}_{i,2}^*) - \text{DISP}(\mathcal{C}_{i,2}^*)}{\max(\text{SEP}(\mathcal{C}_{i,2}^*), \text{DISP}(\mathcal{C}_{i,2}^*))} \tag{5.20}$$

Since C_i^* is a weakly homogeneous cluster, therefore $\text{SEP}(\mathcal{C}_i^*) \geq \text{DISP}(\mathcal{C}_i^*)$. Then, minimal value of Δ, Δ_{min} is reached when $\text{SEP}(\mathcal{C}_{i,1}^*) \geq \text{DISP}(\mathcal{C}_{i,1}^*)$, $\text{SEP}(\mathcal{C}_{i,2}^*) \geq \text{DISP}(\mathcal{C}_{i,2}^*)$, $\text{SEP}(\mathcal{C}_{i,1}^*) = \text{SEP}(\mathcal{C}_{i,2}^*) = \min_{e'\in E(\mathcal{C}_i^*)} w(e')$ and $\text{DISP}(\mathcal{C}_{i,1}^*) =$

$\text{DISP}(\mathcal{C}^*_{i,2}) = \max_{e' \in E(\mathcal{C}^*_i)} w(e')$. Then,

$$N \times \Delta_{min} = |\mathcal{C}^*_i| \left(1 - \frac{\text{DISP}(\mathcal{C}^*_i)}{\text{SEP}(\mathcal{C}^*_i)}\right) - |\mathcal{C}^*_{i,1}| \left(1 - \frac{\text{DISP}(\mathcal{C}^*_{i,1})}{\text{SEP}(\mathcal{C}^*_{i,1})}\right) - |\mathcal{C}^*_{i,2}| \left(1 - \frac{\text{DISP}(\mathcal{C}^*_{i,2})}{\text{SEP}(\mathcal{C}^*_{i,2})}\right) \tag{5.21}$$

$$= |\mathcal{C}^*_i| \left(1 - \frac{\text{DISP}(\mathcal{C}^*_i)}{\text{SEP}(\mathcal{C}^*_i)}\right) - |\mathcal{C}^*_{i,1}| \left(1 - \frac{\max_{e' \in E(\mathcal{C}^*_i)} w(e')}{\min_{e' \in E(\mathcal{C}^*_i)} w(e')}\right) - |\mathcal{C}^*_{i,2}| \left(1 - \frac{\max_{e' \in E(\mathcal{C}^*_i)} w(e')}{\min_{e' \in E(\mathcal{C}^*_i)} w(e')}\right) \tag{5.22}$$

$$= |\mathcal{C}^*_i| \left(-\frac{\text{DISP}(\mathcal{C}^*_i)}{\text{SEP}(\mathcal{C}^*_i)} + \frac{\max_{e' \in E(\mathcal{C}^*_i)} w(e')}{\min_{e' \in E(\mathcal{C}^*_i)} w(e')}\right) \tag{5.23}$$

By weak homogeneity condition on C^*_i, $\frac{\text{DISP}(\mathcal{C}^*_i)}{\text{SEP}(\mathcal{C}^*_i)} < \frac{\min_{e' \in E(\mathcal{C}^*_i)} w(e')}{\max_{e' \in E(\mathcal{C}^*_i)} w(e')} \leq \frac{\max_{e' \in E(\mathcal{C}^*_i)} w(e')}{\min_{e' \in E(\mathcal{C}^*_i)} w(e')}$.

Therefore, $\Delta_{min} > 0$ and $\Delta > 0$. □

Corollary 5.4.1. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K weakly homogeneous clusters $C^*_1, \ldots, C^*_K$ and $\mathcal{T}$ an MST of $\mathcal{G}$. $DBMSTClu(\mathcal{T})$ stops after $K-1$ iterations and the K subtrees produced match exactly the clusters i.e. under homogeneity condition, the algorithm finds automatically the underlying clustering partition.*

Proof. Theorems 5.4.1 and 5.4.3 ensure that under homogeneity condition on all clusters, the algorithm performs the $K-1$ distinct cuts within $Cut_{\mathcal{G}}(\mathcal{T})$ and stops afterwards. By definition of $Cut_{\mathcal{G}}(\mathcal{T})$, it means the DBMSTClu correctly builds the K clusters. □

5.4.2 Analysis of DBMSTClu algorithm

Previous Section demonstrates the exact recovery of the underlying clustering partition with DBMSTClu assuming some hypotheses. This Section gives now some further analysis of the algorithm to show more insights on the way how the algorithm works. The main results are the following:

1. Besides the fact it does not require the number of expected clusters in advance, DBMSTClu differs significantly from the naive approach of SEMST by preferring a first cut which is not necessarily corresponding to the heaviest edge (Propositions 5.4.1 and 5.4.2).

2. As long as the current partition contains at least one cluster with a negative validity index, DBMSTClu will find a cut improving the global index (Proposition 5.4.3).

3. Conditions are given to determine in advance if and which cut will be performed in a cluster with a positive validity index (Propositions 5.4.4 and 5.4.5).

Propositions 5.4.1 and 5.4.2 rely on the two basic lemmas regarding the first cut in an MST:

Lemma 5.4.2 (Highest weighted edge case)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ and $\mathcal{T}$ an MST of $\mathcal{G}$. If at step $k = 1$, cutting edge $e' = \underset{e \in E(\mathcal{T})}{\text{argmax}}\ w(e)$ leads to $DBCVI_1(e') \geq 0$.*

Proof. For the first cut, at step $k = 1$, both separations of obtained clusters C_1 and C_2 are equal to the weight of the considered edge for cut. Here, this is the one with the highest weight. Thus, for $i = 1, 2$, $\text{DISP}(C_i) \leq \text{SEP}(C_i) \implies V_C(C_i) \geq 0$. Finally, the DBCVI of the partition, as a convex sum of two nonnegative quantities, is clearly nonnegative. □

Lemma 5.4.3 (Lowest weighted edge case)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ and $\mathcal{T}$ an MST of $\mathcal{G}$. If at step $k = 1$, cutting edge $e' = \underset{e \in E(\mathcal{T})}{\text{argmin}}\ w(e)$ leads to $DBCVI_1(e') \leq 0$.*

Proof. Same reasoning in the opposite case s.t. $\text{SEP}(C_i) - \text{DISP}(C_i) \leq 0$ for $i \in \{1, 2\}$. □

Proposition 5.4.1 (When the first cut is not the heaviest)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ and $\mathcal{T}$ an MST of $\mathcal{G}$ s.t $N = |V|$. Let us consider this specific case: $\exists e_1, e_2 \in E(\mathcal{T})$ s.t. $w(e_1) = w_1$, $w(e_2) = w_2$ and $\forall e \in E(\mathcal{T}) \backslash \{e_1, e_2\}$, $w(e) = w$ s.t. $w_1 > w_2 > w > 0$. At first step $k = 1$, DBMSTClu does not cut any edge with weight w and cuts e_2 instead of e_1 iff:*

$$w_2 > \frac{2n_2w_1 - n_1 + \sqrt{n_1^2 + 4w_1(n_2^2w_1 + N^2 - Nn_1 - n_2^2)}}{2(N - n_1 + n_2)} \tag{5.24}$$

where n_1 (resp. n_2) is the number of nodes in the first cluster resulting from the cut of e_1 (resp. e_2). Otherwise, e_1 gets cut.

Proof. Let $DBCVI_1(e_1)$ (resp. $DBCVI_1(e_2)$) be the DBCVI after the cut of e_1 (resp. e_2) at step $k = 1$. As w (resp. w_1) is the minimum (resp. maximal) weight, the algorithm does not cut e since the resulting DBCVI would be negative (cf. Lemma 5.4.3) while $DBCVI_1(e_1)$ is guaranteed to be positive (cf. Lemma 5.4.2). So, the choice will be between e_1 and e_2 but e_2 gets cut iff $DBCVI_1(e_2) > DBCVI_1(e_1)$. $DBCVI_1(e_1)$ and $DBCVI_1(e_2)$ expressions are simplified w.l.o.g. by scaling the weights by w s.t. $w \leftarrow 1$, $w_1 \leftarrow w_1/w$, $w_2 \leftarrow w_2/w$, hence $w_1 > w_2 > 1$. Then,

$$\begin{aligned}
&DBCVI_1(e_2) > DBCVI_1(e_1) > 0 \\
&\iff \frac{n_2}{N}\left(\frac{w_2}{w_1} - 1\right) + \left(1 - \frac{n_2}{N}\right)\left(1 - \frac{1}{w_2}\right) - \frac{n_1}{N}\left(1 - \frac{1}{w_1}\right) + \left(1 - \frac{n_1}{N}\right)\left(1 - \frac{w_2}{w_1}\right) > 0 \\
&\iff w_2^2 \underbrace{(N + n_2 - n_1)}_{a} + w_2 \underbrace{(n_1 - 2n_2w_1)}_{b} + \underbrace{(n_2 - N)w_1}_{c<0} > 0.
\end{aligned} \tag{5.25}$$

Clearly, $\Delta = b^2 - 4ac$ is positive and c/a is negative. But $w_2 > 0$, then $w_2 > \frac{-b + \sqrt{b^2 - 4ac}}{2a}$ which gives the final result after some simplifications. □

Proposition 5.4.1 emphasizes that the algorithm is cleverer than simply cutting the heaviest edge first. Indeed, although $w_2 < w_1$, cutting e_2 could be preferred over

e_1. Moreover, no edge with weight w can get cut at the first iteration as they have the minimal weight in the tree. Indeed it really happens since an approximate MST with discrete rounded weights is used when sketching is applied.

Proposition 5.4.2 (First cut on the heaviest edge in the middle)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ and $\mathcal{T}$ an MST of $\mathcal{G}$ s.t $N = |V|$. Let us consider this specific case: $\exists e_1, e_2 \in E(\mathcal{T})$ s.t. $w(e_1) = w_1$, $w(e_2) = w_2$ and $\forall e \in E(\mathcal{T})\backslash\{e_1, e_2\}$, $w(e) = w$ s.t. $w_1 > w_2 > w > 0$. Denote n_1 (resp. n_2) the number of nodes in the first cluster resulting from the cut of e_1 (resp. e_2). In the particular case where edge e_1 with maximal weight w_1 stands between two subtrees with the same number of points, i.e. $n_1 = N/2$, e_1 is always preferred over e_2 as the first optimal cut i.e. $DBCVI_1(e_1) > DBCVI_1(e_2)$.*

Proof. A *reductio ad absurdum* is made by showing that at step $k = 1$, cutting edge e_2 i.e. $DBCVI_1(e_2) > DBCVI_1(e_1)$ leads to the contradiction $w_1/w < 1$. With the scaling process from Proposition 5.4.1'proof:

$$DBCVI_1(e_1) = \frac{1}{2}(1 - \frac{1}{w_1}) + \frac{1}{2}(1 - \frac{w_2}{w_1}) = 1 - \frac{1}{2w_1} - \frac{w_2}{2w_1} \tag{5.26}$$

$$\begin{aligned} DBCVI_1(e_2) &= \frac{n_2}{N}(\frac{w_2}{w_1} - 1) + (1 - \frac{n_2}{N})(1 - \frac{1}{w_2}) \\ &= 1 - \frac{1}{w_2} + \frac{n_2}{N}(\underbrace{\frac{w_2}{w_1} + \frac{1}{w_2} - 2}_{=A}) \end{aligned} \tag{5.27}$$

There is $w_2 > w = 1$, so $\frac{1}{w_2} < 1$. Besides $w_2 < w_1$ so $\frac{w_2}{w_1} < 1$ thus, $A < 0$. Let now consider w.l.o.g. that edge e_2 is on the "right side" (right cluster/subtree) of e_1 (similar proof if e_2 is on the left side of e_1). Hence, it is clear that for maximizing $DBCVI_1(e_2)$ as a function of n_2, we need $n_2 = n_1 + 1$. Then,

$$DBCVI_1(e_2) > DBCVI_1(e_1)$$

$$\iff -\frac{1}{w_2} + (\frac{1}{2} + \frac{1}{N})(\frac{w_2}{w_1} - 2 + \frac{1}{w_2}) > -\frac{1}{w_1} - \frac{w_2}{w_1} \tag{5.28}$$

$$\begin{aligned} \iff &(\frac{1}{2w_1} + \frac{1}{Nw_1} + \frac{1}{2w_1})w_2 - 1 - \frac{2}{N} + \frac{1}{2w_1} \\ &+ (-1 + \frac{1}{2} + \frac{1}{N})\frac{1}{w_2} > 0 \end{aligned} \tag{5.29}$$

$$\iff \underbrace{(1 + \frac{1}{N})}_{a>0} w_2^2 + w_2 \underbrace{(\frac{1}{2} - w_1(1 + \frac{2}{N}))}_{b<0} + w_1 \underbrace{(\frac{1}{N} - \frac{1}{2})}_{c<0} > 0 \tag{5.30}$$

As $c/a < 0$ and $w_2 > 0$, $w_2 > \frac{N}{2(N+1)}$ [$w_1(1 + \frac{2}{N}) - \frac{1}{2} + \sqrt{\Delta}$] with $\Delta = (w_1(1 + \frac{2}{N}) - \frac{1}{2})^2 + 4(1 + \frac{1}{N})(\frac{1}{2} - \frac{1}{N})w_1$. This inequality is incompatible with

$w_1 > w_2$ since:

$$w_1 > w_2 \iff w_1 > \frac{N}{2(N+1)} \left[w_1(1+\frac{2}{N}) - \frac{1}{2} + \sqrt{\Delta} \right] \tag{5.31}$$

$$\iff w_1 + \frac{1}{2} > \sqrt{\Delta} \tag{5.32}$$

$$\iff \frac{4}{N} w_1^2 (1+\frac{1}{N}) + \frac{4}{N} w_1(-1-\frac{1}{N}) < 0 \tag{5.33}$$

$$\iff w_1 < 1 \ : ILLICIT \tag{5.34}$$

Indeed, after the scaling process, $w_1 < 1 = w$ is not possible since by hypothesis, $w_1 > w$. Finally, it is not allowed to cut e_2, the only remaining possible edge to cut is e_1. □

Remark 5.4.1. *Let us consider the MST in Figure 5.7 with $N = 8$, $w_1 = 1$, $w_2 = w_3 = 1 - \epsilon$, and the other weights set to ϵ. Clearly, it is not always preferred to cut e_1 in the middle since for $\epsilon = 0.1$, $DBCVI_1(e_2) \approx 0.27 > DBCVI_1(e_1) = \epsilon = 0.1$. So, it is a counter-example to a possible generalization of Proposition 5.4.2 where there would be more than three possible distinct weights in $\mathcal{T}$.*

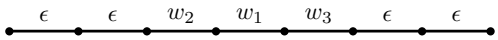


FIGURE 5.7: Counter-example for Remark 5.4.1.

These last Propositions hold for every iteration in the algorithm.

Proposition 5.4.3 (Fate of negative V_C cluster)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ and $\mathcal{T}$ an MST of $\mathcal{G}$ s.t $N = |V|$. Let be $C_1, \ldots, C_k$ the current found clusters in the clustering partition at iteration $t = k - 1$. If $\{i \in [k] \mid V_C(C_i) < 0\} \neq \emptyset$, then $\exists j \in \{i \in [k] \mid V_C(C_i) < 0\}$ s.t. DBMSTClu will cut an edge at stage t in cluster C_j.*

Proof. Let $i \in [k]$ s.t. $V_C(C_i) < 0$ i.e. $\text{SEP}(C_i) < \text{DISP}(C_i)$. We denote w^l_{sep} the minimal weight outing cluster C_i and w_{max} the maximal weight in subtree S_i of C_i i.e. $\text{SEP}(C_i) := w^l_{sep}$ and $\text{DISP}(C_i) := w_{max}$. Hence, $w^l_{sep} < w_{max}$. By cutting the cluster C_i on the edge with weight w_{max}, we define C^l_i and C^r_i resp. the left and right resulting clusters.

Let us look at $V_C(C^l_i)$. If $\text{SEP}(C^l_i) \geq \text{DISP}(C^l_i)$ then $V_C(C^l_i) \geq 0 \geq V_C(C_i)$. Else $V_C(C^l_i) = \frac{\text{SEP}(C^l_i)}{\text{DISP}(C^l_i)} - 1$. The definition of the separation as a minimum and our cut imply that $\text{SEP}(C^l_i) \geq \min(\text{SEP}(C_i), w_{max}) \geq \text{SEP}(C_i)$. Also the definition of the Dispersion as a maximum implies that $\text{DISP}(C^l_i) \leq \text{DISP}(C_i)$. Hence we get that $\frac{\text{SEP}(C^l_i)}{\text{DISP}(C^l_i)} - 1 \geq \frac{\text{SEP}(C_i)}{\text{DISP}(C_i)} - 1$ i.e. $V_C(C^l_i) \geq V_C(C_i)$ in this case too. The same reasoning holds for C^r_i showing that $V_C(C^r_i) \geq V_C(C_i)$. Finally, by denoting $DBCVI_t$ the DBCVI at step t after the cut and $DBCVI_{t-1}$ the

DBCVI before the cut:

$$DBCVI_t = \sum_{j \neq i} \frac{n_j}{N} V_C(C_j) + \frac{n_i^l}{N} V_C(C_i^l) + \frac{n_i^r}{N} V_C(C_i^r) \tag{5.35}$$

$$\geq \sum_{j \neq i} \frac{n_j}{N} V_C(C_j) + \frac{n_i^l}{N} V_C(C_i) + \frac{n_i^r}{N} V_C(C_i) = DBCVI_{t-1}. \tag{5.36}$$

Hence cutting the edge with maximal weight in C_i improves the resulting DBCVI. □

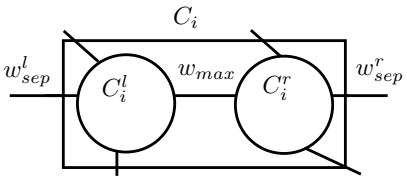


FIGURE 5.8: Generic example of Proposition 5.4.3 and 5.4.5's proofs.

Thus, at the end of the clustering algorithm, each cluster will have a nonnegative V_C so at each step of the algorithm, this bound holds for the final DBCVI:

$$DBCVI \geq \sum_{i=1}^{k} \frac{|C_i|}{N} \max(V_C(C_i), 0). \tag{5.37}$$

Proposition 5.4.4 (Fate of positive V_C cluster I)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ and $\mathcal{T}$ an MST of $\mathcal{G}$. Let be $C_1, \ldots, C_k$ the current found clusters in the clustering partition at iteration $t = k - 1$. $\forall i \in [k]$ s.t. $V_C(C_i) > 0$ and $\text{SEP}(C_i) = s > 0$, then DBMSTClu does not cut an edge e of cluster C_i with weight $w < s$ if both resulting clusters have at least one edge with weight greater than w.*

Proof. Let us be $i \in [k]$ s.t. $V_C(C_i) > 0$ and $\text{SEP}(C_i) = s > 0$. Let us consider clusters C_i^l and C_i^r resulting from the cut of edge e in C_i. Assume that in the associated subtree of C_i^l (resp. C_i^r), there is an edge e_l (resp. e_r) with a weight w_l (resp. w_r) higher than w s.t. without loss of generality, $w_l > w_r$. Since $V_C(C_i) > 0$, $s > w_l > w_r > w$. But cutting edge e implies that $\text{DISP}(C_i^l) > \text{SEP}(C_i^l) = w$ (resp. $\text{DISP}(C_i^r) > \text{SEP}(C_i^r) = w$), and thus $V_C(C_i^l) < 0$ (resp. $V_C(C_i^r) < 0$). Cutting edge e would therefore mean to replace a cluster C_i s.t. $V_C(C_i) > 0$ by two clusters C_i^l and C_i^r s.t. $V_C(C_i^l) < 0$ and $V_C(C_i^r) < 0$ which obviously decreases the current DBCVI. Thus, e does not get cut at this step of the algorithm. □

Proposition 5.4.5 (Fate of positive V_C cluster II)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ and $\mathcal{T}$ an MST of $\mathcal{G}$. Let be $C_1, \ldots, C_k$ the current found clusters in the clustering partition at iteration $t = k - 1$. Consider a partition with K clusters s.t. some cluster C_i, $i \in [K]$ with $V_C(C_i) > 0$ is in the setting of Figure 5.8 i.e. cutting the heaviest edge e with weight w_{max} results in two clusters: the left (resp. right) cluster C_i^l (resp. C_i^r) with n_1 points (resp. n_2) s.t. $\text{DISP}(C_i^l) = d_1$, $\text{SEP}(C_i^l) = w_{sep}^l$,*

$\mathrm{DISP}(C_i^r) = d_2$ *and* $\mathrm{SEP}(C_i^r) = w_{sep}^r$. *Assuming w.l.o.g.* $w_{sep}^l > w_{sep}^r$, *cutting edge e improves the DBCVI iff:*

$$\frac{\left(\frac{n_1 d_1 + n_2 d_2}{n_1 + n_2}\right)}{w_{max}} \leq \frac{w_{max}}{w_{sep}^r}. \tag{5.38}$$

Proof. As $V_C(C_i) > 0$, there is $\mathrm{SEP}(C_i) = w_{sep}^r > w_{max}$. Then, the DBCVI before (K clusters) and after cut of w_{max} ($K+1$ clusters) are:

$$DBCVI_K = \sum_{j \neq i}^{K} V_C(C_j) + \frac{n_1 + n_2}{N}\left(1 - \frac{w_{max}}{w_{sep}^r}\right) \tag{5.39}$$

$$DBCVI_{K+1} = \sum_{j \neq i}^{K} V_C(C_j) + \frac{n_1}{N}\left(1 - \frac{d_1}{w_{max}}\right) + \frac{n_2}{N}\left(1 - \frac{d_2}{w_{max}}\right) \tag{5.40}$$

DBMSTClu cuts w_{max} iff $DBCVI_{K+1} \geq DBCVI_K$. So the result after simplification. □

5.5 Implementation for linear time and space complexities

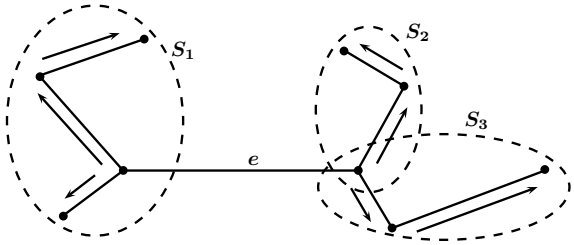


FIGURE 5.9: Illustration of the recursive relationship for left and right Dispersions resulting from the cut of edge e: $\mathrm{DISP}_{left}(e) = \max(w(S_1))$, $\mathrm{DISP}_{right}(e) = \max(w(S_2), w(S_3))$ where $w(.)$ returns the edge weights. Separation works analogically.

For a dataset with N points, $\mathcal{T}$ contains $N-1$ edges and $K-1$ cuts give K connected components. Note that independently of the technique used to obtain an MST (the sketching method presented in Section 5.6 is just a possible one), the space complexity of the algorithm is $O(N)$ which is better than the $O(N^2)$ of Spectral Clustering.

What is about time complexity? For each iteration, all possible edges are greedily tested as a cut. So for K cuts, the time complexity of a naive implementation of DBMSTClu described in Algorithm 6 is $O(NK)$. Although this is clearly less than the $O(N^3)$ one implied by Spectral Clustering, this is not satisfying for large-scale application.

We now explain how to make the implementation efficient in order to achieve the linear time and space complexities in N[5]. The principle is based on two tricks:

1. As observed in Section 5.4: for a performed cut in a current cluster C_i, the $V_C(C_j)$ for any $j \neq i$ remain unchanged. Hence, if the $V_C(C_j)$ are stored for each untouched cluster after a given cut, only the edges of C_i^l and C_i^r respectively the left and right clusters induced by e's cut in cluster C_i need to be evaluated again to determine the DBCVI in case of cut. Thus the number of operations to find the optimal cut decreases drastically over time as the number of edges in clusters becomes smaller through the cuts.

2. However finding the first cut already costs $O(N)$ time (by testing all the $N-1$ possible cuts) hence paying this price for each cut evaluation at each iteration would lead to $O((N-1)+(N-2)+\ldots+1) = O(N^2)$ operations. Fortunately, this can be avoided as SEP and DISP exhibit some recurrence relationship in $\mathcal{T}$: when knowing these values for a given cut, we can deduce the value for a neighboring cut (cf. Figure 5.9). To determine the first cut, $\mathcal{T}$ should be hence completely crossed following the iterative version of the Depth-First Search. The difficulty though is that the recursive relationship between the quantities to update is directional: left and right w.r.t. the edge to cut. So we develop here *double Depth-First search* (see principle in Algorithm 7): from any given edge of $\mathcal{T}$, edges left and right are all visited consecutively with a Depth-First search, and SEP and DISP are updated recursively thanks to a carefully defined order in the priority queue of edges to handle.

Algorithm 7 Generic Double Depth-First Search

Input: $\mathcal{T}$, the MST; e, the edge of $\mathcal{T}$ where the search starts; n_src, source node of e
Q = deque() *//Empty priority double-ended queue*
for incident edges to n_src **do**
  pushBack(Q, ($incident\_e, n\_src, FALSE$))
pushBack(Q, ($e, n\_src, TRUE$))
for incident edges to n_trgt **do**
  pushBack(Q, ($incident\_e, n\_trgt, FALSE$))
pushBack(Q, ($e, n\_trgt, TRUE$))
while Q is not empty **do**
  $e, node, marked$ = popFront(Q)
  $opposite\_node$ = getOtherNodeEdge($e, node$)
  if not $marked$ **then**
    for incident edges to $node$ **do**
      pushBack(Q, ($incident\_e, node, F$))
    pushBack(Q, ($e, node, T$))
    pushFront(Q, ($e, opposite\_node, T$))
  else
    doTheJob(e) *//Perform here the task*
return

[5]Code available in Python using igraph library on Github: https://github.com/annemorvan/DBMSTClu/.

5.6 Retrieval of an approximate MST via sketching

At this point, one can argue that the usefulness of DBMSTClu depends on an efficient way of computing a minimum spanning tree. To that purpose:

1. From the stream of edge weights, a sketch of the dissimilarity graph is built,
2. Then an approximate MST is computed with no more information than the previously obtained sketch.

Sections 5.6.1 and 5.6.2 respectively briefly explain the two steps.

5.6.1 Streaming graph sketching

For the graph sketching, data are processed in the dynamic streaming model [Muthukrishnan, 2005]:

1. The graph is handled as a stream s of edge weight updates:

   $$s := \langle s_1, ... \; s_j, ... \rangle$$

   where s_j is the j-th update in the stream corresponding to the tuple

   $$s_j := (i, \; w_{old,i}, \; \Delta w_i)$$

   with i denoting the index of the edge to update, $w_{old,i}$ its previous weight and Δw_i the update to perform. Thus, after reading s_j in the stream, the i-th edge is assigned the new weight $w_i = w_{old,i} + \Delta w_i \geq 0$.
2. The graph sketching method makes only one pass over this stream.
3. Edges can be both inserted or deleted (*turnstile* model) in the graph sketch. Weights can be increased or decreased but have always to be nonnegative. They can change regularly, as in social networks where individuals can be friends for some time then not anymore.

The algorithm from Ahn et al. [2012a] is used here to produce in an online fashion a limited number of linear measurements summarizing the edge weights of $\mathcal{G}$, as the new data s_j are read from the stream s of edge weight updates. Its general principle is briefly described here. See Section 2.4.2 for more details. The basic idea is to compress the information on the edge weights through few linear measurements. These are updated *dynamically* as new data s_j from stream s is seen only once.

For a given small ϵ_1, $\mathcal{G}$ is seen as a set of unweighted subgraphs G_k containing all the edges with weight lower than $(1+\epsilon_1)^k$, hence $G_k \subset G_{k+1}$. The G_k are embodied as N virtual vectors $\mathbf{v}^{(i)} \in \{-1, 0, 1\}^M$ for $i \in [N]$ expressing for each node the belonging to an existing edge: for $j \in [M]$, $\mathbf{v}_j^{(i)}$ equals to 0 if node i is not in e_j, 1 (resp. -1) if e_j exists and i is its left (resp. right) node. All $\mathbf{v}^{(i)}$ are described at L different "levels", i.e. L virtual copies of the true vectors are made with some entries randomly set to zero s.t. the $\mathbf{v}^{(i),l}$ get sparser as the corresponding level $l \in [L]$ increases. The $\mathbf{v}^{(i),l}$ for each level are explicitly coded in memory by three counters:

- $\phi := \sum_{j=1}^{M} \mathbf{v}_j^{(i),l}$
- $\iota := \sum_{j=1}^{M} j \; \mathbf{v}_j^{(i),l}$

- $\tau := \sum_{j=1}^{M} \mathbf{v}_j^{(i),l} \; z^j \mod p$, with p a suitably large prime and $z \in \mathbb{Z}/p\mathbb{Z}$.

The resulting compact data structure further named a *sketch* enables to draw almost uniformly at random a nonzero weighted edge of the G_k at any time among the levels vectors $\mathbf{v}^{(i),l}$ which are 1-sparse (with exactly one nonzero coefficient). This is made possible thanks to ℓ_0-sampling [Cormode and Firmani, 2014]:

Definition 5.6.1 (ℓ_0-sampling)**.** *An* (ϵ, δ) ℓ_0-sampler *for a nonzero vector* $\mathbf{x} \in \mathbb{R}^n$ *fails with a probability at most* δ *or returns some* $i \in [n]$ *with probability* $(1 \pm \epsilon)\frac{1}{|\operatorname{supp} \mathbf{x}|}$ *where* $\operatorname{supp} \mathbf{x} = \{i \in [n] \mid \mathbf{x}_i \neq 0\}$.

The sketch requires $O(N \log^3(N))$ space. It follows that the sketching is technically *semi-streamed*[6] but in practice only one pass over the data is needed and the space cost is significantly lower than the theoretical $O(N^2)$ bound. The time cost for each update of the sketch is polylog(N). This sketch is the first one to support both insertion and deletion of edges. In this context, the number of nodes is known while the edges, whose weights can be increased or decreased (but have always to stay positive) are summarized into those sketches.

5.6.2 Recovery of an approximate MST from the graph sketch

Ahn et al. [2012a] also proposed an algorithm to compute in a single pass the approximate weight $\tilde{W}$ of an MST $\mathcal{T}$ - the sum of all its edge weights - by appropriate samplings from the sketch in $O(N \operatorname{polylog}(N))$ time. They show that

$$W \leq \tilde{W} \leq (1 + \epsilon_1)\, W \tag{5.41}$$

where W stands for the true weight and

$$\tilde{W} = N - (1 + \epsilon_1)^{r+1}\, cc(G_r) + \sum_{k=0}^{r} \lambda_k\, cc(G_k) \tag{5.42}$$

with $\lambda_k = (1+\epsilon_1)^{k+1} - (1+\epsilon_1)^i$, $r = \lceil \log_{1+\epsilon_1}(w_{max}) \rceil$ such that w_{max} is the maximal weight of $\mathcal{G}$ and cc denotes the number of connected components of the graph in parameter.

Here an extended method is applied for obtaining rather an approximate MST - and not simply its weight - by registering edges as they are sampled. Referring to the proof of Lemma 3.4 in work from Ahn et al. [2012a] and its corrected version Lemma 2.4.3 in Section 2.4.2, the approach is simply justified by applying Kruskal's algorithm where edges with the lower weights are first sampled.

Similarly, note that the sketching technique enables also to recover a Minimum Spanning Forest (MSF) if the initial graph is disconnected.

5.7 Experiments

Experiments were conducted to show the two following properties:

1. Using an approximate MST by sketching instead of a real one is safe regarding the precision of the result.

2. DBMSTClu is scalable for large values of N.

[6] In a *streaming* algorithm, the space cost should be *linear* in the number of instances in the database.

Tight lower and upper bounds are given in Section 2.4.2 for the weight of the approximate MST retrieved by sketching. First experiments in Section 5.7.1 show that the clustering results do not suffer from the use of an approximate MST instead of a real one. Experiments from Section 5.7.2 prove then the scalability of DBMSTClu for large values of N.

Experiments were conducted using Python and scikit-learn library [Pedregosa et al., 2011] on a single-thread process on an intel processor based node. All Figures should be read in color.

5.7.1 Safety of the sketching

The results of DBMSTClu are first compared with DBSCAN [Ester et al., 1996] because it can compete with DBMSTClu as:

1. Non-convex-shaped clusters are recognized.
2. It does not require explicitly the number of expected clusters.
3. It is both time and space-efficient (resp. $O(N \log N)$ and $O(N)$).

Then, the results of another MST-based algorithm are shown for comparison. The latter called Standard Euclidean MST (SEMST) [Zahn, 1971] cuts the $K-1$ heaviest edges of a standard Euclidean MST given a targeted number of clusters K. For DBMSTClu, the dissimilarity graph is built from the computation of the Euclidean distance between the data points and passed into the sketch phase to produce an approximate version of the exact MST.

Synthetic datasets

Experiments were performed on three classic synthetic datasets from the Euclidean space: three blobs, noisy circles and noisy moons. Each dataset contains 1000 data points in 20 dimensions: the first two dimensions are randomly drawn from predefined 2D-clusters, as shown in Figure 5.10, 5.11 and 5.12, while the other 18 dimensions are random Gaussian noise.

Real dataset

DBMSTClu performances are also measured on the mushroom dataset[7]. It contains 8124 records of 22 categorical attributes corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota family. 118 binary attributes are created from the 22 categorical ones, then the complete graph (about 33 millions of edges) is built by computing the normalized Hamming distance (i.e. the number of distinct bits divided by the dimension, see Definition A.0.5 p.128 for a recall) between the points.

Results

Figure 5.10, 5.11, 5.12 and 5.13 show the results for all previously defined datasets and aforementioned methods when providing to DBMSTClu an approximate MST obtained from the sketching phase. The synthetic datasets were projected onto 2D spaces for visualization purposes. They were produced with a noise level such that SEMST fails and DBSCAN does not perform well without parameters optimization. In particular, for DBSCAN all the cross points correspond to noise.

[7] https://archive.ics.uci.edu/ml/datasets/mushroom

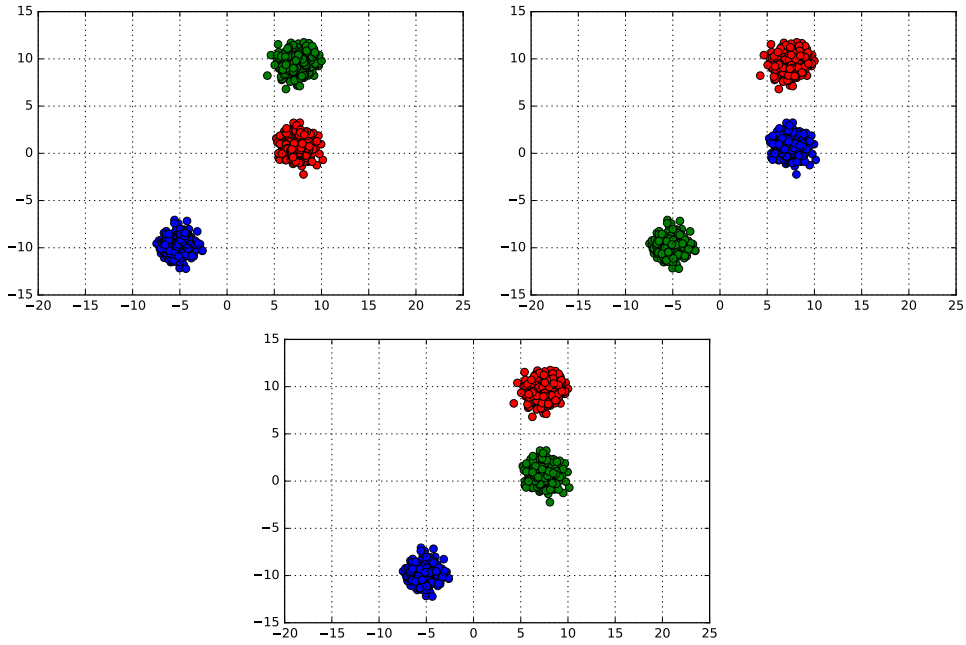


FIGURE 5.10: Three blobs: SEMST, DBSCAN ($\epsilon = 1.4$, $minPts = 5$), DBMSTClu with an approximate MST.

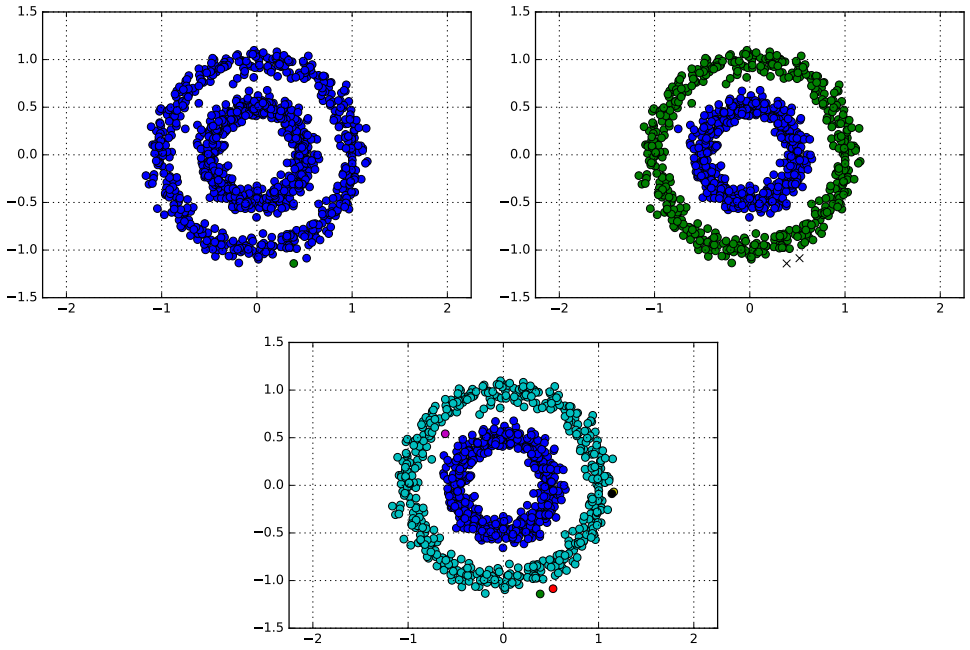


FIGURE 5.11: Noisy circles: SEMST, DBSCAN ($\epsilon = 0.15$, $minPts = 5$), DBMSTClu with an approximate MST.

- With the three blobs, each method performs well: they all manage to retrieve the three clusters.
- With the concentric circles, SEMST does not cut on the consistent edges, hence leads to an isolated singleton cluster. DBSCAN classifies the same point plus a near one as noise while recovering the two circles well. Finally, DBMSTClu finds the two main clusters and also creates five singleton clusters which can be legitimately considered as noise as well.

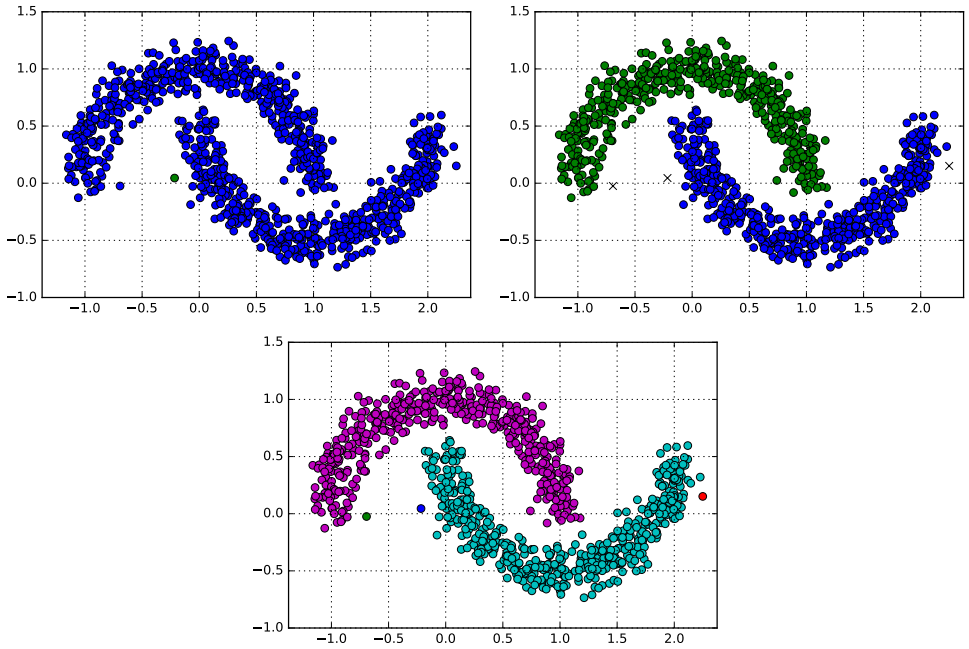


FIGURE 5.12: Noisy moons: SEMST, DBSCAN ($\epsilon = 0.16$, $minPts = 5$), DBMSTClu with an approximate MST.

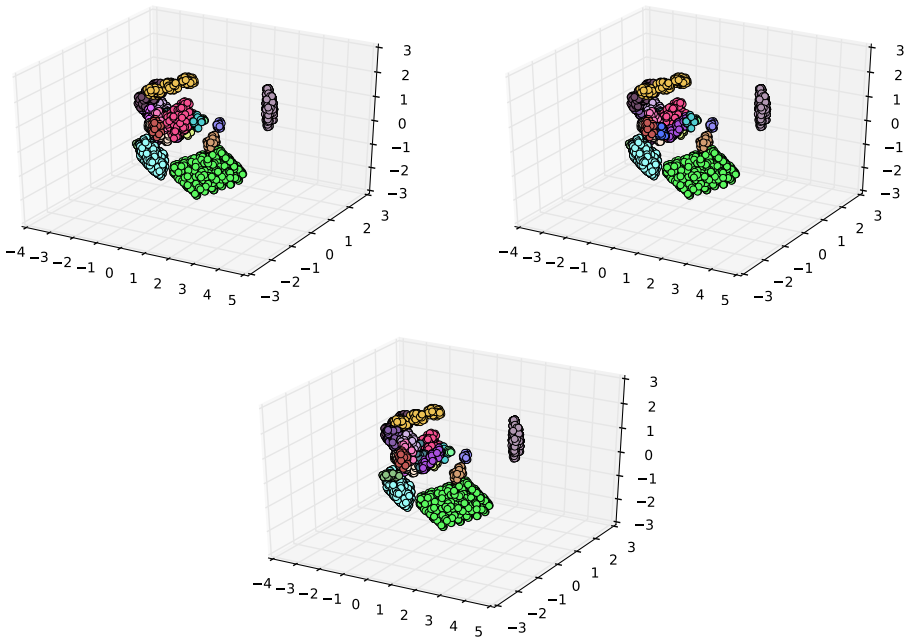


FIGURE 5.13: Mushroom dataset: SEMST, DBSCAN ($\epsilon = 1.5$, $minPts = 2$), DBMSTClu with an approximate MST (projection on the first three principal components).

- With noisy moons, while DBSCAN considers three outliers, DBMSTClu detects the same as singleton clusters.
- For the mushroom dataset, if suitable parameters are given to SEMST and DBSCAN, the right 23 clusters get found while DBMSTClu retrieves them without tuning any parameter.

As theoretically proved above, experiments emphasize the fact that our algorithm is more subtle than simply cutting the heaviest edges as the failure of SEMST shows. Moreover our algorithm exhibits an ability to detect outliers, which could be labeled as noise in a post-processing phase. Another decisive advantage of our algorithm is the absence of any required parameters, contrarily to DBSCAN.

Quantitative results for the synthetic datasets are shown in Table 5.1: the achieved Silhouette coefficient [Rousseeuw, 1987], the Adjusted Rand Index (ARI) [Rand, 1971] and the DBCVI. For all the indices, the higher, the better.

- The Silhouette coefficient (between -1 and 1) is used to measure a clustering partition without any external information. For DBSCAN it is computed by considering noise points as singletons. We see that this measure is not very suitable for non-convex clusters like noisy circles or moons.
- The ARI (between 0 and 1) measures the similarity between the experimental clustering partition and the known ground truth. DBSCAN and DBMSTClu give similar almost optimal results.
- Finally, the obtained DBCVIs are consistent, since the best ones are reached for DBMSTClu.

For the mushroom dataset, the corresponding DBCVI and Silhouette coefficient are respectively 0.75 and 0.47.

	Silhouette coefficient			Adjusted Rand Index			DBCVI		
SEMST	**0.84**	**0.16**	−0.12	**1**	0	0	**0.84**	0.001	0.06
DBSCAN	**0.84**	0.02	**0.26**	**1**	**0.99**	**0.99**	**0.84**	−0.26	**0.15**
DBMSTClu	**0.84**	−0.26	**0.26**	**1**	**0.99**	**0.99**	**0.84**	**0.18**	**0.15**

TABLE 5.1: Silhouette coefficients, Adjusted Rand Index and DBCVI for the blobs, noisy circles and noisy moons datasets with SEMST, DBSCAN and DBMSTClu.

5.7.2 Scalability of the clustering

For the mushroom dataset which is not a toy dataset, DBMSTClu's execution time (averaged on 5 runs) is 3.36s while DBSCAN requires 9.00s. This gives a first overview of its ability to deal with a high number of clusters. Further experiments on execution time were conducted on large-scale random weighted graphs generated from the Stochastic Block Model with varying equal-sized number of clusters K and N. The scalability of DBMSTClu is shown in Figure 5.14 and Table 5.2 by exhibiting the linear time complexity in N. Graphs with 1 million of nodes and 100 clusters were

$K \backslash N$	1000	10000	50000	100000	250000	500000	750000	1000000
5	0.34	2.96	14.37	28.91	73.04	148.85	218.11	292.25
20	0.95	8.73	43.71	88.51	223.18	449.37	669.29	889.88
100	4.36	40.25	201.76	398.41	995.42	2011.79	3015.61	4016.13
"100/5"	12.82	13.60	14.04	13.78	13.63	13.52	13.83	13.74

TABLE 5.2: Numerical values for DBMSTClu's execution time (in s) varying N and K (averaged on 5 runs). The last row shows the execution time ratio between $K = 100$ and $K = 5$.

easily clustered. In Table 5.2, the row with the execution time ratio between $K = 100$ and $K = 5$ illustrates the first trick from Section 5.5 as the observed time ratio is around 2/3 of the theoretical one $100/5 = 20$.

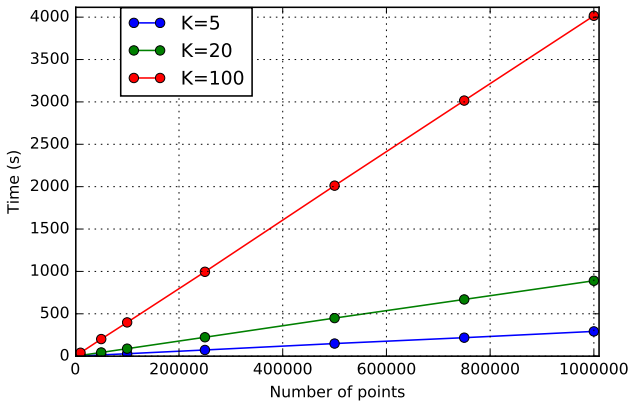


FIGURE 5.14: DBMSTClu's execution time with values of $N \in \{1\text{K}, 10\text{K}, 50\text{K}, 100\text{K}, 250\text{K}, 500\text{K}, 750\text{K}, 1\text{M}\}$.

5.8 Conclusion

In this Chapter 5, a novel *space-efficient* Density-Based Clustering algorithm, named DBMSTClu, is introduced. It only relies on a Minimum Spanning Tree (MST) of the dissimilarity graph $\mathcal{G}$. The spatial and time costs are $O(N)$ with N the number of data points. This enables to deal easily with graphs of million of nodes.

Moreover, DBMSTClu is *non-parametric*, unlike most existing clustering methods: it automatically determines the right number of non-convex clusters.

Although the approach is fundamentally independent from the sketching phase, its robustness has been assessed by using as input an approximate MST of the sketched $\mathcal{G}$ rather than an exact one. The graph sketch is computed dynamically on the fly as the new edge weight updates are read in only one pass over the data. This brings a space-efficient solution for finding an MST of $\mathcal{G}$ when the $O(N^2)$ edges cannot fit in memory. Hence, our algorithm adapts to the semi-streaming setting with $O(N \operatorname{polylog} N)$ space.

Our approach shows promising results, as evidenced by the experimental part regarding time and space scalability and clustering performance even with sketching.

Further work would consist in using this algorithm in privacy issues, as the lost information when sketching might ensure data privacy (see Appendix E for the obtained results). Moreover, as it is already the case for the graph sketching, we could look for the adaptation of both the MST recovery and DBMSTClu to the fully online setting. It would consist in modifying both dynamically a current MST and the clustering partition as a new edge weight update from the stream is seen.

Chapter 6

Conclusion

Contents

This Chapter presents a discussion of the work and some research perspectives.

6.1 Summary of the contributions

The main purpose of this thesis is to investigate how to perform efficiently unsupervised machine learning applications preserving the distance and the data structure such as the nearest neighbors search and clustering for high dimensional streams. The proposed models rely on approximation algorithms based on compact representations of the data which have the attractive advantages to reduce the required memory space, to increase the data processing rate of flow while maintaining an acceptable accuracy of the result in comparison with the exact equivalent algorithms.

To this end, three main improvements can be acknowledged, belonging each to the three main categories for distance and structure preserving methods: *data-independent*, *data-dependent* and *graph-based* approaches which cover almost all usages for the user.

- The first one concerns a theoretical framework proving the accuracy of the *data-independent* fastest known version of the Cross-polytope LSH algorithm [Terasawa and Tanaka, 2007] for the nearest neighbor search. The classical Cross-polytope LSH involves some matrix-vector products where the matrix is a random Gaussian one. Instead of this classical Gaussian matrix, Andoni et al. [2015a] proposed the structured one $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ where $\mathbf{H}$ stands for the normalized Hadamard transform and for all $i \in \{1, 2, 3\}$, $\mathbf{D}_i$ are random independent diagonal matrices with coefficients taken from $\{+1, -1\}$. This leads to considerable memory savings by storing less parameters since the Hadamard transform does not need to be stored in order to be applied to an input vector. While the accuracy of the $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ matrix was only experimentally checked, our work gives a comprehensive framework justifying the role of each block $\mathbf{HD}_i$. The theory holds for a variety of other structured matrices such

as the circulant, Toeplitz, Hankel etc. matrices. The speedups, in comparison with the unstructured counterpart, are enabled by the use of the Fast Fourier Transform to compute the matrix-vector products. (Chapter 3)

Moreover, the applications are not limited to the approximate nearest neighbors search. One can use them in kernel approximation via random feature maps, convex optimization with Newton sketches, vector quantization, neural networks or machine learning applications comprising matrix-vector products where the involved matrix is learned or random and can be replaced by a structured one. (Appendix C).

- Secondly, when *data-independent* methods do not provide enough accuracy, a novel streaming *data-dependent* method is proposed to *learn* compact binary codes from high-dimensional data points in only one pass over the stream. In practice, the memory footprint is really small since only a projection matrix and small square matrices with dimension equal to the targeted code length are stored. The latter are fully updated in a streaming fashion as the new data points are seen. Theoretical guarantees support the method and the quality of the obtained sketches are assessed on the approximate nearest neighbors search task. (Chapter 4)

- As a last contribution, a space-efficient parameter-free clustering algorithm recognizing non-convex cluster shapes is presented. The approach of the algorithm is *graph-based*. To enable linear time and space complexities of the method in the number of data points, it works on a Minimum Spanning Tree (MST) of the data dissimilarity graph and performs relevant cuts on this tree to make the clusters appear from the resulting connected components. In practice, the algorithm takes as input rather an approximate MST which has been recovered from the beforehand sketched data dissimilarity graph. The efficient sketching phase is streamed by processing the data dissimilarity graph by a sequence of edges. (Chapter 5)

  Finally, the clustering algorithm is applied in the differential privacy framework. (Appendix E)

6.2 Discussion

In the streaming setting, the three development axes to pursue in this thesis for existing machine learning algorithms are:

- limiting the required memory space,
- reducing the computation time cost,
- controlling the approximation error.

The matrix-vector product compression and speedup (the *Structured spinners*, contribution 1 in Chapter 3) and the learning of small binary codes from high-dimensional vectors (UnifDiag, contribution 2 in Chapter 4) completely respect this given framework. Moreover, the tree-based clustering algorithm (DBMSTClu, contribution 3 in Chapter 5) follows carefully the three previously stated points. The space cost is indeed limited to the storage of a minimum spanning tree which is linear in the number of vertices. In addition, some implementation efforts have been made to ensure the linear computation time in the number of vertices. At last but not least, accuracy of the algorithm is guaranteed:

1. by *theory* on optimality of MST-based clustering algorithms on the one hand and specific good clustering partition recovery proofs for DBMSTClu on the other hand,

2. by *experimental* results with approximate MST instead of a true one when the latter results from the graph sketching technique from Ahn et al. [2012a] or from the differentially private release algorithm from Pinot [2018] (PTClust, see Appendix E).

Nevertheless, one can argue that even if the computation of the dissimilarity graph sketch is streamed, the retrieval of an approximate MST and the clustering algorithm are offline. Indeed, let us consider a fixed graph sketch after stopping the edge stream. From it, an MST and the underlying clustering partition with DBMSTClu are recovered. If the edge stream is read again, the graph sketch is easily revised as a new edge update is seen while the MST and the clustering partition need to be re-computed from the beginning. This is a clear limitation of the method. However, the reasonable time to compute an MST and the clustering partition - respectively $O(N \log^3(N))$ and $O(N)$ where N is the number of vertices - makes the clustering algorithm still useful despite its limitation.

6.3 Perspectives

6.3.1 How to overcome the current limitations

DBMSTClu and the streaming setting

One future research perspective ensues directly from the current limitation of the clustering algorithm DBMSTClu. The idea is to adapt DBMSTClu to the fully streaming setting. The difficulties are two-fold:

1. Computing in an online fashion an MST with high space constraints, depending on the fact that deletions are allowed or not, is not straightforward and is indeed a research topic itself.

   To give some insights on the difficulty, let us first consider the case where *only additions* are accepted. The MST is built incrementally so it is initially an empty set of edges. Step by step, edges from the stream are added to the current formed MST which is rather a Minimum Spanning Forest (MSF) as long as the tree is disconnected. Then, an edge from the current MSF may be removed to respect the constraint on the absence of cycle and the minimization of the edge weights sum.

   Concretely, as long as the new edge seen from the stream contains at least one vertex which has not been yet encountered before, the edge can be directly added in the maintained MSF without further calculations. Otherwise, let us add this edge in the forest anyway: this may create a cycle if it is not connecting two separated connected components. In that case, it is necessary to compute the connected component incident to this edge and if there is a cycle, to remove the edge with the heaviest weight.

2. From a given current MST and the made change according to the current observed edge in a stream, how to update efficiently the clustering partition? Indeed, DBMSTClu is divisive. Depending on the added edge, some clusters may require to be merged. However, clusters merging has not been implemented in the algorithm.

On the need of rotations in Hypercubic hashing

UnifDiag, our proposed online method for learning compact binary codes from high-dimensional vectors, is a streaming hashing method from the family of Hypercubic hashing. Recall, in Hypercubic hashing, the compact binary codes for high dimensional data are obtained by projecting the data points with a linear embedding followed by a suitable rotation. Then, the sign function is applied pointwise to get binary coefficients.

From this family, ITQ [Gong et al., 2013] is the algorithm giving the best results for the nearest neighbors retrieval task. With an expected code length c, the method projects the data points onto the c first principal components and then learns iteratively a rotation which correlates the rotated data points with their binary version (see Chapter 2, Section 2.3.2 for more details).

For producing a streaming version of this algorithm, the rotation is optimized in UnifDiag method so that to uniformize the diagonal coefficients of the PCA-projected data covariance matrix. The thing is, such a rotation is not unique. For instance, UnifDiag defines the rotation as a product of a sequence of Givens rotations while IsoHash algorithm [Kong and Li, 2012] computes the rotation by solving a particular objective function with gradient descent. Hence, the following questions remain open:

- How to find the optimal rotation among the ones uniformizing the diagonal of the PCA-projected covariance matrix?
- Since ITQ in the offline setting still performs better than UnifDiag, how to solve the objective function from ITQ (Equation 4.7, p. 76) in an online fashion? Actually, we do not know yet how to do this. This is why UnifDiag has been designed so that to optimize this kind of surrogate function aiming at equalizing the diagonal coefficients of the PCA-projected data covariance.

Besides, the proposed embedding is linear by relying on an approximate PCA projection and a rotation. This can be limiting if data are not assumed to be linearly separable. A field to investigate could be non-linear binary embedding and neural network methods.

6.3.2 Applications

Another aspect of the thesis that could be more developed is real-world applications. From the implementation point of view, for the moment, the stream has been only simulated by a *for* loop. The done work could be more valued if the algorithms are implemented in streaming frameworks like Kafka streams, Spark streaming or Redis.

Concretely, applications in metagenomics could be investigated for UnifDiag and DBMSTClu, e.g. to reduce the dimensionality of some metagenomics descriptors and to perform clustering when the number of clusters is not known in advance.

Appendix A

Further mathematical tools: distances

This Appendix reports some further mathematical tools which are used in the thesis.

Nearest neighbors search and clustering quality results depend heavily on the chosen distance. Some recall is made on classical distances.

Definition A.0.1 (distance)**.** *A* distance *is a function* $dist : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ *that obeys the following four axioms. For any* $x, y \in \mathcal{X}$*,*

1. $dist(x, y) \geq 0$ *(non-negativity), and* $dist(x, y) = 0$ *if and only if* $x = y$ *(identity of indiscernibles),*
2. $dist(x, y) = dist(y, x)$ *(symmetry)*
3. $\forall z \in \mathcal{X}$, $dist(x, z) \leq dist(x, y) + dist(y, z)$ *(triangle inequality)*

Such a distance function is known as a metric and with a set defines a metric space. In this thesis in particular, distance functions will be defined for:

- $\mathcal{X} = V$ where V is the set of vertices of a simple undirected weighted graph $\mathcal{G}(V, E, w)$ where E is the edge set and $w := E \to \mathbb{R}$ a weight function.
- $\mathcal{X} = \mathbb{R}^d$ where the data points are real d-dimensional vectors.

For graphs i.e. $\mathcal{X} = V$ a set of vertices in a graph, we will use the *minimum path distance*.

Definition A.0.2 (Minimum path distance)**.** *Let be* $\mathcal{G} = (V, E, w)$ *and* u, $v \in V$*. The* minimum path distance *between* u *and* v *is*

$$dist_{\mathcal{G}}(u, v) = \min_{\mathcal{P}_{u-v}} \sum_{e \in \mathcal{P}_{u-v}} w(e)$$

with $\mathcal{P}_{u-v}$ *a path from* u *to* v *in* $\mathcal{G}$*.*

For the classical case which we will cover more carefully $\mathcal{X} = \mathbb{R}^d$ for some $d > 0$, the L_1 and L_2-distances are in particular considered.

Definition A.0.3 (L_1-distance)**.** *For* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$*, the* L_1*-distance between* $\mathbf{x}$ *and* $\mathbf{y}$ *is defined as:*

$$dist_1(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_1 = \sum_{i=1}^{d} |\mathbf{x}_i - \mathbf{y}_i|. \tag{A.1}$$

The L_1-distance is also called *Manhattan distance*.

Definition A.0.4 (L_2 distance)**.** *For* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$*, the* L_2*-distance between* $\mathbf{x}$ *and* $\mathbf{y}$ *is defined as:*

$$dist_2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\sum_{i=1}^{d} (\mathbf{x}_i - \mathbf{y}_i)^2}. \tag{A.2}$$

The L_2-distance is well-known as the *Euclidean distance.*
Two special cases of L_1 and L_2-distances respectively should be mentioned.

1. When the data and the queries are in the hypercube $\{0,1\}^d$, L_1-distance is equivalent to the Hamming distance.

   Definition A.0.5 (Hamming distance)**.** *For* $\mathbf{x}, \mathbf{y} \in \{0,1\}^d$*, the* Hamming distance *between* $\mathbf{x}$ *and* $\mathbf{y}$ *is defined as:*

   $$dist_H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} (\mathbf{x}_i \oplus \mathbf{y}_i) \tag{A.3}$$

   where $\oplus$ *denotes the XOR operator.*

   The Hamming distance has the particular advantage to be cheap to compute since this is the basic operation of counting the number of different bits between two binary numbers.

2. When the data and the queries are on the unit sphere $S^{d-1} \subset \mathbb{R}^d$, the L_2-distance is equivalent to the cosine or the angular distances.

   Definition A.0.6 (Cosine similarity)**.** *For* $\mathbf{x}, \mathbf{y} \in S^{d-1} \subset \mathbb{R}^d$*, the* cosine similarity *between* $\mathbf{x}$ *and* $\mathbf{y}$ *is defined as:*

   $$sim_{cos}(\mathbf{x}, \mathbf{y}) := \cos(\theta) = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2} = \frac{\sum_{i=1}^{d} \mathbf{x}_i \mathbf{y}_i}{\sqrt{\sum_{i=1}^{d} \mathbf{x}_i^2} \sqrt{\sum_{i=1}^{d} \mathbf{y}_i^2}}. \tag{A.4}$$

   Definition A.0.7 (Cosine distance)**.** *For* $\mathbf{x}, \mathbf{y} \in S^{d-1} \subset \mathbb{R}^d$*, the* cosine distance *between* $\mathbf{x}$ *and* $\mathbf{y}$ *is defined as:*

   $$dist_{cos}(\mathbf{x}, \mathbf{y}) = 1 - sim_{cos}(\mathbf{x}, \mathbf{y}) \tag{A.5}$$

   Definition A.0.8 (Angular distance)**.** *For* $\mathbf{x}, \mathbf{y} \in S^{d-1} \subset \mathbb{R}^d$*, the* angular distance *between* $\mathbf{x}$ *and* $\mathbf{y}$ *is defined as:*

   $$dist_{ang}(\mathbf{x}, \mathbf{y}) = \frac{\cos^{-1}(sim_{cos}(\mathbf{x}, \mathbf{y}))}{\pi} \tag{A.6}$$

Appendix B

Further structured matrices as complements of Chapter 2

This Appendix reports some structured matrices which have been used in the thesis.

The product between a matrix and a vector arises in many problems of machine learning: dimensionality reduction techniques, random projection or LSH are just a few examples. With a dense $d \times d$-matrix $\mathbf{A}$ and any d-dimensional vector $\mathbf{x}$, the standard multiplication cost is $O(d^2)$. For large values of d, this is prohibitive. We show in this Section a class of structured matrices which, although they are dense, depend on at most $O(d)$ parameters. Hence, the memory cost of these matrices goes from $O(d^2)$ to $O(d)$. Moreover, we show how to compute the matrix-vector product $\mathbf{y} = \mathbf{A}\mathbf{x}$ in only $O(d \log d)$ time where $\mathbf{x}$, $\mathbf{y} \in \mathbb{R}^d$ and $\mathbf{A}$ is a $d \times d$-matrix. The members of this matrix family are:

- the Hadamard matrix (already defined above),
- the circulant matrix,
- the skew-circulant matrix,
- the Toeplitz matrix,
- the Hankel matrix,
- the Kronecker matrix.

They have been successfully used for instance in work from Vybíral [2011], Hinrichs and Vybíral [2011], Zhang et al. [2015], Andoni et al. [2015a]. Let us begin with a definition of these matrices and then their efficient matrix-vector products. The Kronecker matrix will be presented separately.

Applied to a d-dimensional input vector, the Hadamard matrix is called the Fast (Walsh)-Hadamard transform (FHT). An efficient implementation of the FHT relies on a divide-and-conquer strategy that recursively decomposes a Hadamard transform of size d into two ones of size $d/2$ [Fino and Algazi, 1976]. This directly follows the recursive definition of the Hadamard matrix. Then, the cost of the Fast Walsh-Hadamard transform is $O(d \log d)$ time while there is nothing to store.

In the sequel, the remaining selected matrices of the considered family of structured matrices are defined before explaining for all how to compute efficiently the matrix-vector product.

Definition B.0.1 (Circulant matrix)**.** *A* circulant matrix *is of the form:*

$$\mathbf{C}_d := \mathbf{C}(a_1, \ldots, a_d) = \begin{pmatrix} a_1 & a_d & a_{d-1} & \ldots & a_2 \\ a_2 & a_1 & a_d & \ldots & a_3 \\ a_3 & a_2 & a_1 & \ldots & a_4 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_d & a_{d-1} & a_{d-2} & \ldots & a_1 \end{pmatrix} \tag{B.1}$$

A circulant matrix is completely determined by the first column, the other ones are obtained by a shift of the previous one. Hence, it relies on d parameters.

Definition B.0.2 (Skew-circulant matrix)**.** *A* skew-circulant matrix *is of the form:*

$$\mathbf{S}_d := \mathbf{S}(a_1, \ldots, a_d) = \begin{pmatrix} a_1 & -a_d & -a_{d-1} & \ldots & -a_2 \\ a_2 & a_1 & -a_d & \ldots & -a_3 \\ a_3 & a_2 & a_1 & \ldots & -a_4 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_d & a_{d-1} & a_{d-2} & \ldots & a_1 \end{pmatrix} \tag{B.2}$$

A skew-circulant matrix is completely determined by the first column and the other ones are obtained by a shift of the previous one. Moreover, the upper triangle is of opposite sign. Hence, it relies on d parameters.

Definition B.0.3 (Toeplitz matrix)**.** *A* Toeplitz matrix *is of the form:*

$$\mathbf{T}_d := \mathbf{T}(a_{-d+1}, \ldots, a_0, \ldots, a_{d-1}) = \begin{pmatrix} a_0 & a_1 & a_2 & \ldots & a_{d-1} \\ a_{-1} & a_0 & a_1 & \ldots & a_{d-2} \\ a_{-2} & a_{-1} & a_0 & \ldots & a_{d-3} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{-d+1} & a_{-d+2} & a_{-d+3} & \ldots & a_0 \end{pmatrix} \tag{B.3}$$

A Toeplitz matrix is completely determined by the first column and the first row. Hence it depends on $2d-1$ parameters. The entries are constant on the diagonals parallel to the main one.

Definition B.0.4 (Hankel matrix)**.** *A* Hankel matrix *is of the form:*

$$\mathbf{L}_d := \mathbf{L}(a_{-d+1}, \ldots, a_0, \ldots, a_{d-1}) = \begin{pmatrix} a_{-d+1} & a_{-d+2} & a_{-d+3} & \ldots & a_0 \\ a_{-d+2} & a_{-d+3} & a_{-d+4} & \ldots & a_1 \\ a_{-d+3} & a_{-d+4} & a_{-d+5} & \ldots & a_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_0 & a_1 & a_2 & \ldots & a_{d-1} \end{pmatrix} \tag{B.4}$$

An Hankel matrix is completely determined by the first column and the last row. Hence it depends on $2d-1$ parameters. The entries are constant on the anti-diagonals, the ones perpendicular to the main diagonal.

For the latter matrices, an efficient implementation of the matrix-vector product is based on the fact that they can be decomposed by Fourier matrices.

Definition B.0.5 (Fourier matrix)**.** *A* Fourier matrix *of order d is defined as the following:*

$$\mathbf{F}_d := \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_d & \omega_d^2 & \dots & \omega_d^{d-1} \\ 1 & \omega_d^2 & \omega_d^4 & \dots & \omega_d^{2(d-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & \omega_d^{d-1} & \omega_d^{2(d-1)} & \dots & \omega_d^{(d-1)(d-1)} \end{pmatrix} \tag{B.5}$$

where $\omega_d = e^{-\frac{2\pi i}{d}}$ is an d-th root of unity.

The multiplication of the Fourier matrix with any vector is the *Discrete Fourier transform* (DFT) whose efficient implementation is the Fast Fourier transform (FFT) [Cooley and Tukey, 1965]. $\mathbf{F}_d$ is a unitary matrix, i.e $\mathbf{F}_d\mathbf{F}_d^*$ and so does $\mathbf{F}_d^*$ where $\mathbf{M}^*$ denotes the adjoint operator for any matrix $\mathbf{M}$. The efficient product of $\mathbf{F}_d^*$ with any vector is the Inverse Fast Fourier transform (IFFT). Both FFT and IFFT can be performed in $O(d \log d)$ time [Cooley and Tukey, 1965].

Circulant product The fast algorithm for matrix-vector product with a circulant matrix is based on the following result [Davis, 1970]. For any vector $\mathbf{a} = (a_1, a_2, \dots, a_d)^T$, a circulant matrix $\mathbf{C}(a_1, \dots, a_d)$ can be diagonalized by the Fourier matrix i.e.

$$\mathbf{C}(a_1, \dots, a_d) = \mathbf{F}_d^* \cdot \mathrm{diag}(\mathbf{F}_d\mathbf{a}) \cdot \mathbf{F}_d \tag{B.6}$$

Therefore, for any vector $\mathbf{x}$, $\mathbf{C}_d\mathbf{x}$ can be computed following Algorithm 8. The time and space costs of the algorithm depend only on the FFT and IFFT's ones. So the final time complexity is $O(d \log d)$.

Algorithm 8 Matrix-vector product with a circulant matrix

1: **Input:** $\mathbf{x} \in \mathbb{R}^d$, the input vector; $\mathbf{a} = (a_1, \dots, a_d)^T$ the first column of the circulant matrix
2: $\mathbf{f} = \mathrm{FFT}(\mathbf{x})$
3: $\mathbf{g} = \mathrm{FFT}(\mathbf{a})$
4: $\mathbf{h} = \mathbf{f} \,.*\, \mathbf{g}$ // *element-wise vector-vector product*
5: $\mathbf{z} = \mathrm{IFFT}(\mathbf{h})$
6: **return** $\mathbf{z}$

Skew-circulant product Similarly, the product $\mathbf{S}_d\mathbf{x}$ can be computed with Algorithm 9 [Pan, 2001]. Please note that if the input vector $\mathbf{x}$ is complex, directly return $\mathbf{z}'$ from Algorithm 9 and not the real part.

Toeplitz product The trick for computing the product of a Toeplitz matrix $\mathbf{T}_d$ and any d-dimensional vector $\mathbf{x}$ is based on the observation that $\mathbf{T}_d$ can be embedded into a $2d \times 2d$ circulant matrix $\mathbf{C}_{2d}$ [Kailath and Sayed, 1999] such that:

$$\mathbf{C}_{2d} = \begin{pmatrix} \mathbf{T}_d & \mathbf{E}_d \\ \mathbf{E}_d & \mathbf{T}_d \end{pmatrix} \tag{B.7}$$

Algorithm 9 Matrix-vector product with a skew-circulant matrix

1: **Input:** $\mathbf{x} \in \mathbb{R}^d$, the input vector; $\mathbf{a} = (a_1, \ldots, a_d)^T$ the first column of the skew-circulant matrix; $\omega = (1, e^{\frac{\pi i}{d}}, e^{\frac{2\pi i}{d}}, \ldots, e^{\frac{(d-1)\pi i}{d}})^T$
2: $\mathbf{x}' = \mathbf{x} \,.*\, \omega$ // *element-wise vector-vector product*
3: $\mathbf{a}' = \mathbf{a} \,.*\, \omega$
4: $\mathbf{f} = \mathrm{FFT}(\mathbf{x}')$
5: $\mathbf{g} = \mathrm{FFT}(\mathbf{a}')$
6: $\mathbf{h} = \mathbf{f} \,.*\, \mathbf{g}$
7: $\mathbf{z} = \mathrm{IFFT}(\mathbf{h})$
8: $\mathbf{z}' = \mathbf{z} \,.*\, \overline{\omega}$
9: **return** $\Re(\mathbf{z}')$

where

$$\mathbf{E}_d = \mathbf{E}(a_{-d+1}, \ldots, a_0, \ldots, a_{d-1}) = \begin{pmatrix} 0 & a_{-d+1} & a_{-d+2} & \ldots & a_{-1} \\ a_{d-1} & 0 & a_{-d+1} & \ldots & a_{-2} \\ a_{d-2} & a_{d-1} & 0 & \ldots & a_{-3} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_1 & a_2 & a_3 & \ldots & 0 \end{pmatrix} \tag{B.8}$$

Therefore, it holds:

$$\mathbf{C}_{2d} \cdot \begin{pmatrix} \mathbf{x} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{T}_d & \mathbf{E}_d \\ \mathbf{E}_d & \mathbf{T}_d \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{T}_d\mathbf{x} \\ \mathbf{E}_d\mathbf{x} \end{pmatrix} \tag{B.9}$$

It suffices to multiply $\mathbf{C}_{2d}$ with $\mathbf{x}$ completed with d zeros following Algorithm 8 and to truncate the resulting vector to the first d coefficients to get the final result. Hence, this algorithm can be executed in $O(d \log d)$ time.

Hankel product By denoting

$$\mathbf{J}_d := \begin{pmatrix} 0 & 0 & \ldots & 0 & 1 \\ 0 & 0 & \ldots & 1 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 1 & \ldots & 0 & 0 \\ 1 & 0 & \ldots & 0 & 0 \end{pmatrix} \tag{B.10}$$

the backward identity matrix, there is $\mathbf{J}_d = \mathbf{J}_d^T = \mathbf{J}_d^{-1}$. Then, it is easy to see that $\mathbf{J}_d\mathbf{L}_d$ is a Toeplitz matrix for any Hankel matrix $\mathbf{L}_d$ (similarly $\mathbf{J}_d\mathbf{T}_d$ is a Hankel matrix for any Toeplitz matrix $\mathbf{T}_d$). Thus, to compute the product $\mathbf{L}_d\mathbf{x}$ in $O(d \log d)$ time [Kailath and Sayed, 1999]:

1. Compute the product $\mathbf{z} = (\mathbf{J}_d\mathbf{L}_d)\mathbf{x}$ as a product between the Toeplitz matrix $\mathbf{J}_d\mathbf{L}_d$ and $\mathbf{x}$ following Equation B.9.
2. Returns $\mathbf{J}_d\mathbf{z}$ since $\mathbf{J}_d = \mathbf{J}_d^{-1}$.

Kronecker matrix-vector product

Definition B.0.6 (Kronecker product)**.** *Let us consider two matrices* $\mathbf{A} \in \mathbb{R}^{k \times l}$ *and* $\mathbf{B} \in \mathbb{R}^{m \times n}$*. The* Kronecker product *of* $\mathbf{A}$ *and* $\mathbf{B}$ *is* $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{km \times ln}$ *such that:*

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} \mathbf{A}_{1,1}\mathbf{B} & \dots & \mathbf{A}_{1,l}\mathbf{B} \\ \mathbf{A}_{2,1}\mathbf{B} & \dots & \mathbf{A}_{2,l}\mathbf{B} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{k,1}\mathbf{B} & \dots & \mathbf{A}_{k,l}\mathbf{B} \end{pmatrix} \tag{B.11}$$

Now our Definition B.0.7 of a Kronecker matrix is introduced.

Definition B.0.7 (Kronecker matrix)**.** *A* Kronecker matrix $\mathbf{K} \in \mathbb{R}^{k \times d}$ *is defined as a product of a sequence of element matrices* $\mathbf{A}_1, \dots, \mathbf{A}_M$ *such that:*

$$\mathbf{K} = \mathbf{A}_1 \otimes \dots \otimes \mathbf{A}_M = \otimes_{i=1}^{M} \mathbf{A}_i \tag{B.12}$$

and $\forall i \in [M]$, $\mathbf{A}_i \in \mathbb{R}^{k_i \times d_i}$, $\Pi_{i=1}^{M} k_i = k$ *and* $\Pi_{i=1}^{M} d_i = d$.

Hence, the space cost of the Kronecker matrix is $O(\sum_{i=1}^{M} k_i d_i)$ which is significantly less than $O(kd)$. The efficient multiplication of a Kronecker matrix with a vector is based on Proposition B.0.1. for which two operations should be introduce [van Loan, 2000]:

1. For any d-dimensional vector $\mathbf{x}$ and two integers k, l such that $kl = d$, $\text{mat}(\mathbf{x}, k, l)$ builds a $k \times l$-matrix from $\mathbf{x}$ column-wise i.e. one has, for $\mathbf{x} = (x_1, \dots, x_d)^T$,

$$\text{mat}\left(\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_d \end{pmatrix}, k, l\right) = \begin{pmatrix} x_1 & x_{k+1} & \dots & x_{(l-1)k+1} \\ x_2 & x_{k+2} & \dots & x_{(l-1)k+2} \\ \vdots & \vdots & \vdots & \vdots \\ x_{k-1} & x_{2k-1} & \dots & x_{kl-1} \\ x_k & x_{2k} & \dots & x_d \end{pmatrix} \tag{B.13}$$

2. For any matrix $\mathbf{M} \in \mathbb{R}^{k \times l}$,

$$\text{vec}(\mathbf{M}) = (\mathbf{M}_{1,1}, \mathbf{M}_{2,1}, \dots, \mathbf{M}_{k,1}, \mathbf{M}_{1,2}, \dots, \mathbf{M}_{k-1,l}, \mathbf{M}_{k,l})^T \in \mathbb{R}^{kl}. \tag{B.14}$$

   i.e. $\text{vec}(\mathbf{M})$ returns a vector resulting from the stacking of the columns of $\mathbf{M}$.

Please notice that in particular, for $\mathbf{x} \in \mathbb{R}^d$ and $k, l \in \mathbb{N}$ such that $d = kl$, $\text{vec}(\text{mat}(\mathbf{x}, k, l) = \mathbf{x}$.

In the sequel, similarly to the work from Zhang et al. [2015], we assume for simplicity - and because we will use later this kind of matrix in practice - that a Kronecker matrix is square i.e. $k = d$ and that all elementary matrices are $d_e \times d_e$ for small $d_e > 0$, i.e. d is a multiple of d_e. In practice, one can take $d_e = 2$.

Proposition B.0.1 (Efficient Kronecker product)**.** *Let us consider elementary matrices* $\mathbf{A}_1, \dots, \mathbf{A}_M \in \mathbb{R}^{d_e \times d_e}$ *for small* $d_e > 0$, $\mathbf{K} = \otimes_{i=1}^{M} \mathbf{A}_i \in \mathbb{R}^{d \times d}$, $d = d_e^M$, *and* $\mathbf{x} \in \mathbb{R}^d$*. It holds:*

$$\mathbf{K}\mathbf{x} = (\otimes_{i=1}^{M} \mathbf{A}_i)\mathbf{x} = \text{vec}((\otimes_{i=2}^{M} \mathbf{A}_i)\, \text{mat}(\mathbf{x}, d/d_e, d_e)\mathbf{A}_1^T) \tag{B.15}$$

Let us consider first the product $\text{mat}(\mathbf{x}, d/d_e, d_e)\mathbf{A}_1^T \in \mathbb{R}^{d/d_e \times d_e}$ in Equation B.15. There are d coefficients to compute and for each coefficient, d_e multiplications and

$d_e - 1$ additions are required. Hence, totally dd_e multiplications and $dd_e - d$ additions are performed, i.e. $d(2d_e - 1)$ operations. Please notice that the Kronecker matrix $(\otimes_{i=2}^{M} \mathbf{A}_i)$ is a $d/d_e \times d/d_e$ matrix since $\mathbf{A}_1$ is missing and $d = d_e^M$. Then, computing $(\otimes_{i=2}^{M} \mathbf{A}_i)\,\mathrm{mat}(\mathbf{x}, d/d_e, d_e)\mathbf{A}_1^T$ can be seen as d_e individual matrix-vector products with d/d_e-dimensional vectors as columns of the matrix $\mathrm{mat}(\mathbf{x}, d/d_e, d_e)\mathbf{A}_1^T$. Therefore, the following recursive scheme appears:

$$f(d, d_e) = \underbrace{d(2d_e - 1)}_{\mathrm{mat}(\mathbf{x}, d/d_e, d_e)\mathbf{A}_1^T} + d_e f(d/d_e, d_e) \tag{B.16}$$

by developping to the next level of recursion:

$$= d(2d_e - 1) + d_e[d/d_e(2d_e - 1) + d_e f(d/d_e^2, de)] \tag{B.17}$$

after simplification:

$$= 2d(2d_e - 1) + d_e^2 f(d/d_e^2, d_e) \tag{B.18}$$

$$= \ldots$$

$$= (M - 1)d(2d_e - 1) + d_e^{M-1} f(d_e, d_e) \tag{B.19}$$

after the last level of recursion:

$$= Md(2d_e - 1) + df(1, d_e) \tag{B.20}$$

after replacing by the value of M:

$$= d(2d_e - 1)\log_{d_e}(d) + df(1, d_e) \tag{B.21}$$

Consequently, the matrix-vector product with a Kronecker matrix which will be used in the sequel costs $d(2d_e - 1)\log_{d_e}(d)$ operations. Typically, $d_e = 2$ will be chosen, so the time complexity can be reduced to $O(d \log d)$.

From square matrices to rectangular matrices Previously, it has been explicated how to perform efficiently in $O(d \log d)$ time matrix-vector products for a wide range of $d \times d$ structured matrices: Hadamard, circulant, skew-circulant, Toeplitz, Hankel and Kronecker matrices. They can be an efficient counterpart of random Gaussian projection matrices.

However, for dimensionality reduction, the required projection matrix should be $c \times d$ where $c \ll d$. In that case, we perform the $d \times d$ corresponding structured matrix-vector product and then take only the c first coefficients. Finally, the time cost is $O(d \log d)$.

Remark B.0.1. *In the Appendix C, an application is shown for kernel approximation with the case of $k \times d$-structured matrices where $k >> d$. Here some $d \times d$-structured matrices are stacked vertically and the final time complexity is $O(d \log d \times k/d) = O(k \log d)$, assuming that k is a multiple of d. For both cases, the rule to keep in mind is that the time complexity for computing the matrix-vector multiplication is $O(\max($input vector dimension, output vector dimension$) \times \log($input vector dimension$))$. See also the scheme from Section 3.2.2.*

Appendix C

Some other *Structured spinners'* applications

Contents

C.1 Introduction

The applications range of the *Structured spinners* family previously described in Chapter 3 do not restrict to dimensionality reduction algorithms and LSH-based schemes. For the *randomized* setting, they are also:

- kernel approximation techniques based on random feature maps produced from linear projections with Gaussian matrices followed by nonlinear mappings [Rahimi and Recht, 2007, Le et al., 2013, Choromanski and Sindhwani, 2016, Huang et al., 2014, Choromanska et al., 2016],
- algorithms solving convex optimization problems with random sketches of Hessian matrices [Pilanci and Wainwright, 2015, 2014],
- quantization techniques using random projection trees, where splitting in each node is determined by a projection of data onto some Gaussian direction [Dasgupta and Freund, 2008],
- and many more.

Regarding the *adaptive* setting, the classical example of machine learning nonlinear models where linear projections are learned is a multi-layered neural network [LeCun et al., 2015, Goodfellow et al., 2016], where the operations of linear projection via matrices with learned parameters followed by the pointwise nonlinear feature transformation are the building blocks of the network's architecture. These two operations are typically stacked multiple times to form a deep network.

Further contributions in this Appendix For the *randomized* setting, we show also in this Appendix applications in kernel approximation via random features maps and convex optimization algorithms via Newton sketches while for the *adaptive* setting, neural networks are given as an example. For the latter to the best of our knowledge, this work is the first to theoretically explain the effectiveness of structured neural network architectures (see Section C.4).

Plan of this Appendix Section C.2 gives the corresponding complementary related work for kernel approximation via random features map, convex optimization with Newton sketches and structured neural networks. Additional theoretical results are demonstrated regarding the randomized setting in Section C.3, and in Section C.4 for the accuracy guarantees in the adaptive setting. Finally, further experiments have been conducted.

C.2 Complementary related work

C.2.1 Kernel approximation via random features map

More recently, the so-called $\mathbf{\Psi}$-regular structured matrices (the Toeplitz and circulant matrices belong to this wider family of matrices) were used to approximate angular distances [Choromanska et al., 2016] and signed Circulant Random Matrices were used to approximate Gaussian kernels [Feng et al., 2015]. Another work from Choromanski and Sindhwani [2016] applies structured matrices coming from the so-called *P-model*, which further generalizes the $\mathbf{\Psi}$-regular family, to speed up random feature map computations of some special kernels (angular, arc-cosine and Gaussian). These techniques did not work for discrete structured constructions, such as the $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ matrices, or their direct non-discrete modifications, since they require matrices with low (polylog) chromatic number of the corresponding coherence graphs.

C.2.2 Newton sketches for convex optimization

Furthermore, a recently proposed technique based on the so-called *Newton Sketch* provides yet another example of application for structured matrices. The method from Pilanci and Wainwright [2015, 2014] is used for speeding up algorithms solving convex optimization problems by approximating Hessian matrices using so-called *sketch matrices*. Initially, the sub-Gaussian sketches based on i.i.d. sub-Gaussian random variables were used. The disadvantage of the sub-Gaussian sketches lies in the fact that computing the sketch of the given matrix of size $n \times d$ requires $O(mnd)$ time, where $m \times n$ in the size of the sketch matrix. Thus the method is too slow in practice and could be accelerated with the use of structured matrices. Some structured approaches were already considered, e.g. sketches based on randomized orthonormal systems were proposed in work from Pilanci and Wainwright [2015].

C.2.3 Neural networks

All previously considered methods focus on the randomized setting, where the structured matrix is fully random. In the context of adaptive setting, the parameters are being learned instead. This Appendix focuses on the example of multi-layer neural networks. It should be emphasized though that this approach is much more general and extends beyond this setting.

Structured neural networks were considered before, for instance in work from Yang et al. [2015], where the so-called *Deep Fried Neural Convnets* were proposed. Those architectures are based on the adaptive version of the Fastfood transform used for approximating various kernels [Le et al., 2013], which is a special case of the structured spinner matrices. Deep Fried Convnets apply adaptive structured matrices for the fully connected layers of the convolutional networks. The structured matrix is of the form: $\mathbf{SHG\Pi HB}$, where $\mathbf{S}$, $\mathbf{G}$, and $\mathbf{B}$ are adaptive diagonal matrices, $\mathbf{\Pi}$ is a random permutation matrix, and $\mathbf{H}$ is the Walsh-Hadamard matrix. The method reduces the storage and the the computational costs of the matrix multiplication step from, often prohibitive, $O(nd)$ down to $O(n)$ storage and $O(n \log d)$ computational cost, where d and n denote the size of consecutive layers of the network. At the same time, this approach does not sacrifice the network's predictive performance. Another work from Moczulski et al. [2016] that offers an improvement over Deep Fried Convnets, looks at a structured matrix family that is very similar to $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ (however is significantly less general than the family of *Structured spinners*). Their theoretical results rely on the analysis in work from Huhtanen and Perämäki [2015].

The Adaptive Fastfood approach elegantly complements previous works dedicated to address the problem of huge overparametrization of deep models with structured matrices. Indeed, the method of Denil et al. [2013] represents the parameter matrix as a product of two low rank factors and, similarly to Adaptive Fastfood, applies both at train and test time. Besides, Sainath et al. [2013] introduces low-rank matrix factorization to reduce the size of the fully connected layers at train time. Li [2013] uses low-rank factorizations with SVD after training the full model. These methods, as well as approaches that consider kernel methods in deep learning [Denil et al., 2013, Mairal et al., 2014, Dai et al., 2014, Huang et al., 2014], are conveniently discussed in work from Yang et al. [2015].

Structured neural networks are also considered in work from Sindhwani et al. [2015], where low-displacement rank matrices are applied for linear projections. The advantage of this approach over Deep Fried Convnets is due to the high parametrization of low-displacement rank matrices family that allows the adjustment of the number of learned parameters based on the accuracy and speedup requirements.

The class of *Structured spinners* proposed in this work is more general than Deep Fried Convnets or low displacement rank matrices, but it also provides much easier structured constructions, such as $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ matrices, where $\mathbf{D}_i$s are adaptive diagonal matrices. Furthermore, to the best of our knowledge, this work is the first to prove theoretically that structured neural networks learn good quality models, by analyzing the capacity of the family of *Structured spinners*.

C.3 Complements of the theoretical results for the randomized setting

C.3.1 Recall of notation

Note that the setting in Section 3.3 covers a wide range of machine learning algorithms: in particular, kernel approximation via random feature maps. Recall the notation from Section 3.3.1, p.59:

Let $\mathcal{A}_{\mathcal{G}}$ be a machine learning algorithm applied to a fixed dataset $\mathcal{X} \subseteq \mathbb{R}^n$ and parametrized by a set $\mathcal{G}$ of matrices $\mathbf{G} \in \mathbb{R}^{m \times n}$, where each $\mathbf{G}$ is either learned or Gaussian with independent entries taken from $\mathcal{N}(0,1)$. Assume furthermore, that $\mathcal{A}_{\mathcal{G}}$ consists of functions $f_1, \ldots, f_s$, where each f_i applies a certain matrix $\mathbf{G}_i$ from $\mathcal{G}$ to vectors from some linear space $\mathcal{L}_i$ of dimensionality at most d. Note that for a fixed dataset $\mathcal{X}$ function f_i is a function of a random vector

$$\mathbf{q}_{f_i} = ((\mathbf{G}_i \mathbf{x}^1)^T, \ldots, (\mathbf{G}_i \mathbf{x}^{d_i})^T)^T \in \mathbb{R}^{d_i \cdot m}, \tag{C.1}$$

where $dim(\mathcal{L}_i) = d_i \leq d$ and $\mathbf{x}^1, \ldots, \mathbf{x}^{d_i}$ stands for some fixed basis of $\mathcal{L}_i$.

Denote by f_i' the structured counterpart of f_i, where $\mathbf{G}_i$ is replaced by the structured spinner (for which vector $\mathbf{r}$ is either learned or random). The goal is to show that $f_i's$ "resemble" f_is distribution-wise.

Under this notation, one has:

Remark C.3.1. *In the kernel approximation setting with random feature maps, one can match each pair of vectors $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ to a different $f = f_{\mathbf{x},\mathbf{y}}$. Each f computes the approximate value of the kernel for vectors $\mathbf{x}$ and $\mathbf{y}$. Thus in that scenario $s = \binom{|\mathcal{X}|}{2}$ and $d = 2$ (since one can take: $\mathcal{L}_{f(\mathbf{x},\mathbf{y})} = \mathrm{span}(\mathbf{x}, \mathbf{y})$).*

Remark C.3.2. *In the vector quantization algorithms using random projection trees one can take $s = 1$ (the algorithm $\mathcal{A}$ itself is a function f outputting the partitioning of space into cells) and $d = d_{intrinsic}$, where $d_{intrinsic}$ is an intrinsic dimensionality of a given dataset $\mathcal{X}$ (random projection trees are often used if $d_{intrinsic} \ll n$).*

C.3.2 b-convexity for angular kernel approximation

Recall that general accuracy results of the *Structured spinners* in the randomized setting are given by Theorem 3.3.1, Section 3.3.4 p.53. We rewrite it here for clarity.

Theorem 3.3.1 (structured random setting). *Let $\mathcal{A}$ be a randomized algorithm using unstructured Gaussian matrices $\mathbf{G}$ and let s, d and f_is be as at the beginning of the section. Replace the unstructured matrix $\mathbf{G}$ by one of the structured spinners defined in Section 3.2 with blocks of m rows each. Then for n large enough, $\epsilon = o_{md}(1)$ and fixed f_i with probability p_{succ} at least:*

$$1 - 2p(n)d - 2\binom{md}{2} e^{-\Omega(\min(\frac{\epsilon^2 n^2}{K^4 \Lambda_F^2 \delta^4(n)}, \frac{\epsilon n}{K^2 \Lambda_2 \delta^2(n)}))} \tag{C.2}$$

with respect to the random choices of $\mathbf{M}_1$ and $\mathbf{M}_2$ the following holds for any $\mathcal{S}$ such that $f_i^{-1}(\mathcal{S})$ is measurable and b-convex:

$$|\mathbb{P}[f_i(q_{f_i}) \in \mathcal{S}] - \mathbb{P}[f_i'(q_{f_i'}) \in \mathcal{S}]| \leq b\eta, \tag{C.3}$$

where the probabilities in the last formula are with respect to the random choice of $\mathbf{M}_3$, $\eta = \frac{\delta^3(n)}{n^{\frac{2}{5}}}$, *and* $\delta(n), p(n), K, \Lambda_F, \Lambda_2$ *are as in the definition of the structured spinners from Section 3.2 p.56.*

Let us now consider the particular setting where linear projections are used to approximate angular kernels between pairs of vectors via random feature maps. In this case, the linear projection is followed by the pointwise nonlinear mapping, where the applied nonlinear mapping is the sign function. The angular kernel is retrieved from the Hamming distance between $\{-1, +1\}$-hashes obtained in such a way. Note that in this case, to each pair $\mathbf{x}, \mathbf{y}$ of vectors from a database can be assigned a function $f_{\mathbf{x},\mathbf{y}}$ that outputs the binary vector whose length is the size of the hash and whose indices are turned on for the hashes of $\mathbf{x}$ and $\mathbf{y}$ which disagree. Such a binary vector uniquely determines the Hadamard distance between the hashes. Notice that for a fixed-length hash, $f_{\mathbf{x},\mathbf{y}}$ produces only finitely many outputs. If $\mathcal{S}$ is a set-singleton consisting of one of the possible outputs, then one can notice (straightforwardly from the way the hash is created) that $f^{-1}_{\mathbf{x},\mathbf{y}}(\mathcal{S})$ is an intersection of the convex sets (as a function of $\mathbf{q}_{f_{\mathbf{x},\mathbf{y}}}$). Thus it is convex and thus for sets $\mathcal{S}$ which are singletons one can take $b = 1$ in Theorem 3.3.1.

C.4 Accuracy of the *Structured spinners* in the adaptive setting

The following theorem explains that the *Structured spinners* can be used to replace unstructured fully connected neural network layers performing dimensionality reduction (such as hidden layers in certain autoencoders) provided that input data has low intrinsic dimensionality. These theoretical findings were confirmed in experiments that will be presented in the next Section. Notation from Theorem 3.3.1 will be used.

Theorem C.4.1. *Consider a matrix* $\mathbf{M} \in \mathbb{R}^{m \times n}$ *encoding the weights of connections between a layer* l_0 *of size* n *and a layer* l_1 *of size* m *in some learned unstructured neural network model. Assume that the input to layer* l_0 *is taken from the* d*-dimensional space* $\mathcal{L}$ *(although potentially embedded in a much higher dimensional space). Then with probability at least*

$$1 - 2p(n)d - 2\binom{md}{2} e^{-\Omega(\min(\frac{t^2 n^2}{K^4 \Lambda_F^2 \delta^4(n)}, \frac{tn}{K^2 \Lambda_2 \delta^2(n)}))} \tag{C.4}$$

for $t = \frac{1}{md}$ *and with respect to random choices of* $\mathbf{M}_1$ *and* $\mathbf{M}_2$, *there exists a vector* r *defining* $\mathbf{M}_3$ *(see definition of the structured spinner in Section 3.2 p.56) such that the structured spinner* $\mathbf{M}_{struct} = \mathbf{M}_3\mathbf{M}_2\mathbf{M}_1$ *equals to* M *on* $\mathcal{L}$.

In the proof of Theorem C.4.1, it is shown that by learning vector $\mathbf{r} \in \mathbb{R}^k$ defined following Condition 3.2.3 p.57, one can approximate well any matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ learned by the neural network, providing that the size k of $\mathbf{r}$ is large enough in comparison with the number of projections and the intrinsic dimensionality d of the dataset $\mathcal{X}$.

Take the parametrized structured spinner matrix $\mathbf{M}_{struct} \in \mathbb{R}^{m \times n}$ with a learnable vector $\mathbf{r}$. Let $\mathbf{M} \in \mathbb{R}^{m \times n}$ be a matrix learned in the unstructured setting. Let $\mathcal{B} = \{\mathbf{x}^1, \ldots, \mathbf{x}^d\}$ be some orthonormal basis of the linear space, where data $\mathcal{X}$ is taken from. One has the following proof for Theorem C.4.1:

Proof. Note that from the definition of the parametrized structured spinner model, with probability at least $p_1 = 1 - p(n)$ with respect to the choices of $\mathbf{M}_1$ and $\mathbf{M}_2$ each $\mathbf{M}_{struct}\mathbf{x}^i$ is of the form:

$$\mathbf{M}_{struct}\mathbf{x}^i = (\mathbf{r}^T \cdot \mathbf{z}_1(\mathbf{q}^i), \ldots, \mathbf{r}^T \cdot \mathbf{z}_m(\mathbf{q}^i))^T, \tag{C.5}$$

where each $\mathbf{z}_j(\mathbf{q}^i)$ is of the form:

$$\mathbf{z}_j(\mathbf{q}^i) = (w^j_{1,1}\rho_1 q^i_1 + w^j_{1,n}\rho_n q^i_n, \ldots, w^j_{k,1}\rho_1 q^i_1 + w^j_{k,n}\rho_n q^i_n)^T \tag{C.6}$$

and $\mathcal{B}' = \{\mathbf{q}^1, \ldots, \mathbf{q}^d\}$ is an orthonormal basis such that for $i \in [n]$:

$$\|\mathbf{q}^i\|_\infty \leq \frac{\delta(n)}{\sqrt{n}}. \tag{C.7}$$

Note that the system of equations for $i = 1, \ldots, d$ with fixed $\mathbf{M}\mathbf{x}^i$:

$$\mathbf{M}_{struct}\mathbf{x}^i = \mathbf{M}\mathbf{x}^i \tag{C.8}$$

has the solution in $\mathbf{r}$ if the vectors from the set $\mathcal{A} = \{\mathbf{z}_j(\mathbf{q}^i) : j \in [m],\ i \in [d]\}$ are independent.

Construct a matrix $\mathbf{G} \in \mathbb{R}^{md \times k}$, where rows are vectors from $\mathcal{A}$. The goal is to show that $rank(\mathbf{G}) = md$, otherwise $rank(\mathbf{G}) < md$ which implies that rows of $\mathbf{G}$ are not independent. It suffices to show that $\det(\mathbf{G}\mathbf{G}^T) \neq 0$. Indeed for $\mathbf{G}\mathbf{G}^T \in \mathbb{R}^{md \times md}$ if $\det(\mathbf{G}\mathbf{G}^T) \neq 0$, then $\mathbf{G}\mathbf{G}^T$ is invertible and $rank(\mathbf{G}\mathbf{G}^T) = md$. But by composition property, $rank(\mathbf{G}\mathbf{G}^T) \leq rank(\mathbf{G})$. Hence, $rank(\mathbf{G}) = md$. Denote $\mathbf{B} = \mathbf{G}\mathbf{G}^T$. Note that $B_{i,j} = (\mathbf{v}^i)^T\mathbf{v}^j$, where $\mathcal{A} = \{\mathbf{v}^1, \ldots, \mathbf{v}^{md}\}$. Take two vectors $\mathbf{v}^a, \mathbf{v}^b \in \mathcal{A}$. Note that from the definition of $\mathcal{A}$ there is:

$$(\mathbf{v}^a)^T\mathbf{v}^b = \sum_{l \in \{1,\ldots,n\}, u \in \{1,\ldots,n\}} \rho_l \rho_u x_l y_u \Big(\sum_{s=1}^{k} w^i_{s,l} w^j_{s,u}\Big) \tag{C.9}$$

for some i, j and some vectors $\mathbf{x} = (x_1, \ldots, x_n)^T$, $\mathbf{y} = (y_1, \ldots, y_n)^T$. Furthermore,

- $i = j$ and $\mathbf{x} = \mathbf{y}$ if $a = b$,
- $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$ by hypothesis,
- $\mathbf{x}^T\mathbf{y} = 0$ or $\mathbf{x} = \mathbf{y}$ and $i \neq j$ for $a \neq b$.

Indeed for the last point, if $a \neq b$:

- either $\mathbf{x} = \mathbf{y}$: in that case, $i \neq j$ otherwise $a = b$,
- or $\mathbf{x} \neq \mathbf{y}$ and $\mathbf{x}^T\mathbf{y} = 0$.

There is also:

$$\mathbb{E}[(\mathbf{v}^a)^T\mathbf{v}^b] = \mathbb{E}\Big[\sum_{l \in \{1,\ldots,n\}} \rho_l^2 x_l y_l \Big(\sum_{s=1}^{k} w^i_{s,l} w^j_{s,u}\Big)\Big]. \tag{C.10}$$

From the previous observations and the properties of matrices $\mathbf{W}^1, \ldots, \mathbf{W}^n$, the entries of the diagonal of $\mathbf{B}$ are equal to 1. Furthermore, all other entries

are 0 on expectation. Using Hanson-Wright inequality, for any $t > 0$ it holds: $|B_{i,j}| \leq t$ for all $i \neq j$ with probability at least:

$$p_{succ} = 1 - 2p(n)d - 2\binom{md}{2} e^{-c\min(\frac{t^2 n^2}{K^4 \Lambda_F^2 \delta^4(n)}, \frac{tn}{K^2 \Lambda_2 \delta^2(n)})}. \tag{C.11}$$

If this is the case, let $\tilde{\mathbf{B}} \in \mathbb{R}^{(md)\times(md)}$ be a matrix with diagonal entries $\tilde{\mathbf{B}}_{i,i} = 0$ and off-diagonal entries $\tilde{\mathbf{B}}_{\mathbf{i,j}} = -\mathbf{B}_{i,j}$. Furthermore, let $\mathbf{B}^* \in \mathbb{R}^{(md)\times(md)}$ be a matrix with diagonal entries $\mathbf{B}^*_{i,i} = 0$ and off-diagonal entries $\mathbf{B}^*_{i,j} = t$.

Following a similar argument from Brent et al. [2014], note that $\mathbf{B}^* = t(\mathbf{J} - \mathbf{I})$ where $\mathbf{J}$ is the matrix of all ones (thus of rank 1) and $\mathbf{I}$ is the identity matrix. Then the eigenvalues of $\mathbf{B}^*$ are $t(md-1)$ with multiplicity 1 and $t(0-1)$ with multiplicity $(md-1)$. Thereby, $\det(\mathbf{I} - \mathbf{B}^*) = (1 - t(md-1))(1+t)^{md-1}$ can be explicitly computed.

If $\rho(\mathbf{B}^*) \leq 1$, Theorem C.4.2 of Brent et al. [2014] can be applied by replacing $\mathbf{F}$ with $\mathbf{B}^*$ and $\mathbf{E}$ with $\tilde{\mathbf{B}}$. For the convenience of the reader, their theorem is stated here:

Theorem C.4.2 (Brent et al. [2014]). *Let $\mathbf{F} \in \mathbb{R}^{n\times n}$ with nonnegative entries and $\rho(F) \leq 1$. Let $\mathbf{E} \in \mathbb{R}^{n\times n}$ with entries $|\, e_{i,j} \,| \leq f_{i,j}$, then $\det(\mathbf{I} - \mathbf{E}) \geq \det(\mathbf{I} - \mathbf{F})$.*

That is: if $\rho(\mathbf{B}^*) \leq 1$, then

$$\det(\mathbf{I} - \mathbf{B}^*) = (1 - t(md-1))(1+t)^{md-1} \leq \det(\mathbf{I} - \tilde{\mathbf{B}}) = \det(\mathbf{B}). \tag{C.12}$$

The final step is to observe that:
$\rho(\mathbf{B}^*) \leq 1 \iff \max\{|\, t(md-1)\,|, |\, -t\,|\} = t(md-1) \leq 1 \iff t \leq \frac{1}{md-1}$.
Hence, using this result, it can be seen that $\det(\mathbf{B}) \geq (1 - t(md-1))(1+t)^{md-1} \geq 0$, in particular $\det(\mathbf{B}) > 0$ for $t = \frac{1}{md}$. That completes the proof. □

C.5 Experiments

C.5.1 Kernel approximation

In this experiment, the Gaussian and angular kernels are approximated using Random Fourier features. The Gaussian random matrix (with i.i.d. Gaussian entries) can be used to sample random Fourier features with a specified σ. This Gaussian random matrix is replaced with specific matrices from a family of *Structured spinners* for Gaussian and angular kernels. The obtained feature maps are compared. To test the quality of the structured kernels' approximations, the Gram-matrix reconstruction error is computed as in the work from Choromanski and Sindhwani [2016] : $\frac{\|\mathbf{K} - \tilde{\mathbf{K}}\|_F}{\|\mathbf{K}\|_F}$, where $\mathbf{K}, \tilde{\mathbf{K}}$ are respectively the exact and approximate Gram-matrices, as a function of the number of random features. When the number of random features k is greater than data dimensionality n, the block-mechanism described in Section 3.2.2 is applied.

For the Gaussian kernel, $\mathbf{K}_{ij} = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}}$ and for the angular kernel, $\mathbf{K}_{ij} = 1 - \frac{\theta}{\pi}$ with $\theta = cos^{-1}(\frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\|_2 \|\mathbf{x}_j\|_2})$. For the approximation, $\tilde{\mathbf{K}}_{i,j} = \frac{1}{\sqrt{d'}} s(\mathbf{A}\mathbf{x}_i)^T \frac{1}{\sqrt{d'}} s(\mathbf{A}\mathbf{x}_j)$ where $s(x) = e^{\frac{-ix}{\sigma}}$ and $\tilde{\mathbf{K}}_{i,j} = 1 - \frac{dist_H(s(\mathbf{A}\mathbf{x}_i), s(\mathbf{A}\mathbf{x}_j))}{d'}$ where $s(x) = \text{sign}(x)$ respectively. In both cases, function s is applied pointwise. $dist_H$ stands for the Hamming distance and $\mathbf{x}_i$, $\mathbf{x}_j$ are points from the dataset.

Two datasets are used: G50C (550 points, $n = 50$) and USPST (test set, 2007 points, $n = 256$).

- G50C dataset contains 550 points of dimensionality 50 drawn from multivariate Gaussians.
- USPST is a dataset of 2007 handwritten digits (from 0 to 9) with dimension 256 collected by the US Postal Service.

For the Gaussian kernel, the bandwidth σ is set to 17.4734 for G50C and to 9.4338 for USPST. The choice of σ comes from Choromanski and Sindhwani [2016] in order to have comparable results. The results are averaged over 10 runs and the following matrices have been tested: Gaussian random matrix $\mathbf{G}$, $\mathbf{G}_{circ}\mathbf{K}_2\mathbf{K}_1$, $\mathbf{G}_{Toeplitz}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$, $\mathbf{G}_{skew-circ}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$, $\mathbf{H}\mathbf{D}_{g_1,\ldots,g_n}\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$ and $\mathbf{H}\mathbf{D}_3\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$.

Figure C.1 (resp. C.2) shows results for the G50C (resp. USPST) dataset. In case of G50C dataset, for both kernels, all matrices from the family of *Structured spinners* perform similarly to a random Gaussian matrix. $\mathbf{H}\mathbf{D}_3\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$ performs better than all other matrices for a wide range of sizes of random feature maps. In case of USPST dataset, for both kernels, all matrices from the family of *Structured spinners* again perform similarly to a random Gaussian matrix (except $\mathbf{G}_{circ}\mathbf{K}_2\mathbf{K}_1$ which gives relatively poor results) and $\mathbf{H}\mathbf{D}_3\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$ is giving the best results. Finally, the efficiency of the *Structured spinners* does not depend on the dataset.

Table C.1 shows substantial speedups obtained by the structured spinner matrices. The speedups are computed as time($\mathbf{G}$)/time($\mathbf{T}$), where time($\mathbf{G}$) and time($\mathbf{T}$) are the runtimes for respectively a random Gaussian matrix and a structured spinner matrix.

MATRIX DIM.	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
$\mathbf{G}_{Toeplitz}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$	x1.4	x3.4	x6.4	x12.9	x28.0	x42.3	x89.6
$\mathbf{G}_{skew-circ}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$	x1.5	x3.6	x6.8	x14.9	x31.2	x49.7	x96.5
$\mathbf{H}\mathbf{D}_{g_1,\ldots,g_n}\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$	x2.3	x6.0	x13.8	x31.5	x75.7	x137.0	x308.8
$\mathbf{H}\mathbf{D}_3\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$	x2.2	x6.0	x14.1	x33.3	x74.3	x140.4	x316.8

TABLE C.1: Speedups for the Gaussian kernel approximation via the *Structured spinners*. It has been tested for square matrices with dimension 2^k for k up to 15. Indeed, the used machine with 16Go RAM is not able to store square matrices with dimension 2^k for $k > 15$. For instance, for dimension 2^{15}, the kernel computation costs $1.382s$ in the unstructured case and $4363\mu s$ with $\mathbf{H}\mathbf{D}_3\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$, which constitutes the best obtained speedup.

C.5.2 Convex optimization via *Newton sketches*

This experiment focuses on the Newton sketch approach [Pilanci and Wainwright, 2015], a generic optimization framework. It guarantees super-linear convergence with exponentially high probability for self-concordant functions [Boyd and Vandenberghe, 2004], and a reduced computational complexity compared to the original second-order Newton method. The method relies on the use of a sketched version of the Hessian matrix, in place of the original one. In the subsequent experiment it is demonstrated that matrices from the family of *Structured spinners* can be used for this purpose, thus can speed up several convex optimization problems solvers.

In particular, the unconstrained large scale logistic regression problem is considered, i.e. given a set of n observations $\{(\mathbf{a}_i, y_i)\}_{i=1,\dots,n}$, with $\mathbf{a}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, the goal is to find $\mathbf{x} \in \mathbb{R}^d$ minimizing the cost function

$$f(\mathbf{x}) = \sum_{i=1}^{n} \log(1 + \exp(-y_i \mathbf{a}_i^T \mathbf{x})). \tag{C.13}$$

The Newton approach to solve this optimization problem entails solving at each iteration the least squares equation $\nabla^2 f(\mathbf{x}^t)\boldsymbol{\Delta}^\mathbf{t} = -\nabla f(\mathbf{x}^t)$, where

$$\nabla^2 f(\mathbf{x}^t) = \mathbf{A}^T \operatorname{diag}\left(\frac{1}{1+\exp(-\mathbf{a}_i^T\mathbf{x})}(1 - \frac{1}{1+\exp(-\mathbf{a}_i^T\mathbf{x})})\right)\mathbf{A} \in \mathbb{R}^{d\times d} \tag{C.14}$$

is the Hessian matrix of $f(\mathbf{x}^t)$, $\mathbf{A} = [\mathbf{a}_1^T \mathbf{a}_2^T \cdots \mathbf{a}_n^T] \in \mathbb{R}^{n\times d}$, $\boldsymbol{\Delta}^t = \mathbf{x}^{t+1} - \mathbf{x}^t$ is the increment at iteration t and $\nabla f(\mathbf{x}^t) \in \mathbb{R}^d$ is the gradient of the cost function. In work from Pilanci and Wainwright [2015] it is proposed to consider the sketched version of the least square equation, based on a Hessian square root of $\nabla^2 f(\mathbf{x}^t)$, denoted $\nabla^2 f(\mathbf{x}^t)^{1/2} = \operatorname{diag}\left(\frac{1}{1+\exp(-\mathbf{a}_i^T\mathbf{x})}(1 - \frac{1}{1+\exp(-\mathbf{a}_i^T\mathbf{x})})\right)^{1/2}\mathbf{A} \in \mathbb{R}^{n\times d}$. The least squares problem at each iteration t is of the form:

$$\left((\mathbf{S}^t\nabla^2 f(\mathbf{x}^t)^{1/2})^T\mathbf{S}^t\nabla^2 f(\mathbf{x}^t)^{1/2}\right)\boldsymbol{\Delta}^t = -\nabla f(\mathbf{x}^t), \tag{C.15}$$

where $\mathbf{S}^t \in \mathbb{R}^{m\times n}$ is a sequence of isotropic sketch matrices. Let us finally recall that the gradient of the cost function is

$$\nabla f(\mathbf{x}^t) = \sum_{i=1}^{n}\left(\frac{1}{1+\exp(-y_i\mathbf{a}_i^T\mathbf{x})} - 1\right)y_i\mathbf{a}_i. \tag{C.16}$$

In this experiment, the goal is to find $\mathbf{x} \in \mathbb{R}^d$, which minimizes the logistic regression cost, given a dataset $\{(\mathbf{a}_i, y_i)\}_{i=1,\dots,n}$, with $\mathbf{a}_i \in \mathbb{R}^d$ sampled according to a Gaussian centered multivariate distribution with covariance $\boldsymbol{\Sigma}_{\mathbf{i,j}} = 0.99^{|i-j|}$ and $y_i \in \{-1, 1\}$, generated at random. Various sketching matrices $\mathbf{S}^t \in \mathbb{R}^{m\times n}$ are considered.

In Figure C.3a the convergence of the Newton sketch algorithm is reported, as measured by the optimality gap defined in work from Pilanci and Wainwright [2015], versus the iteration number. As expected, the structured sketched versions of the algorithm do not converge as quickly as the exact Newton-sketch approach, however various matrices from the family of *Structured spinners* exhibit equivalent convergence properties as shown in the figure.

When the dimensionality of the problem increases, the cost of computing the Hessian in the exact Newton-sketch approach becomes very large [Pilanci and Wainwright, 2015], scaling as $O(nd^2)$. The complexity of the structured Newton-sketch approach with the matrices from the family of *Structured spinners* is instead only $O(dn\log(n) + md^2)$. Figure C.3b also illustrates the wall-clock times of computing single Hessian matrices and confirms that the increase in the number of iterations of the Newton sketch compared to the exact Newton method is compensated by the efficiency of the sketched computations, in particular the Hadamard-based sketches yield improvements at the lowest dimensions.

C.5.3 Neural networks

Finally, experiments have been performed with neural networks using two different network architectures. The first one is a fully-connected network with two fully connected layers (later called MLP), where the size of the hidden layer is referred as h, and the second one is a convolutional network with the following architecture:

- Convolution layer with filter size 5×5, 4 feature maps + ReLU + Max Pooling (region 2×2 and step 2×2)
- Convolution layer with filter size 5×5, 6 feature maps + ReLU + Max Pooling (region 2×2 and step 2×2)
- Fully-connected layer (h outputs) + ReLU
- Fully-connected layer (10 outputs)
- LogSoftMax.

Experiments were performed on the MNIST data set. In both experiments, each weights matrix of fully connected layers is re-parametrized with a structured $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ matrix from a family of *Structured spinners*. This setting is compared with the case where the unstructured parameter matrix is used. Note that in case when $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ is used only a linear number of parameters is learned (the Hadamard matrix is deterministic and even does not need to be explicitly stored, instead Walsh-Hadamard transform is used). Thus the network has significantly less parameters than in the unstructured case, e.g. for the MLP network there are $O(h)$ instead of $O(\text{input size} \times h)$ parameters.

In Figure C.4 and Table C.2 respectively the test error and the running time of the unstructured and structured approaches are compared. Figure C.4 shows that for large enough h, neural networks with *Structured spinners* achieve similar performance to those with unstructured projections, while at the same time using *Structured spinners* leads to significant computational savings as shown in Table C.2. As mentioned before, the $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$-neural network is a simpler construction than the Deep Friend Convnet, however one can replace it with any structured spinner to obtain compressed neural network architecture of a good capacity.

h	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}
unstructured	42.9	51.9	72.7	99.9	163.9	350.5	716.7	1271.5	2317.4
$\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$	109.2	121.3	109.7	114.2	117.4	123.9	130.6	214.3	389.8

TABLE C.2: Running time (in $[\mu s]$) for the MLP - unstructured matrices vs *Structured spinners*.

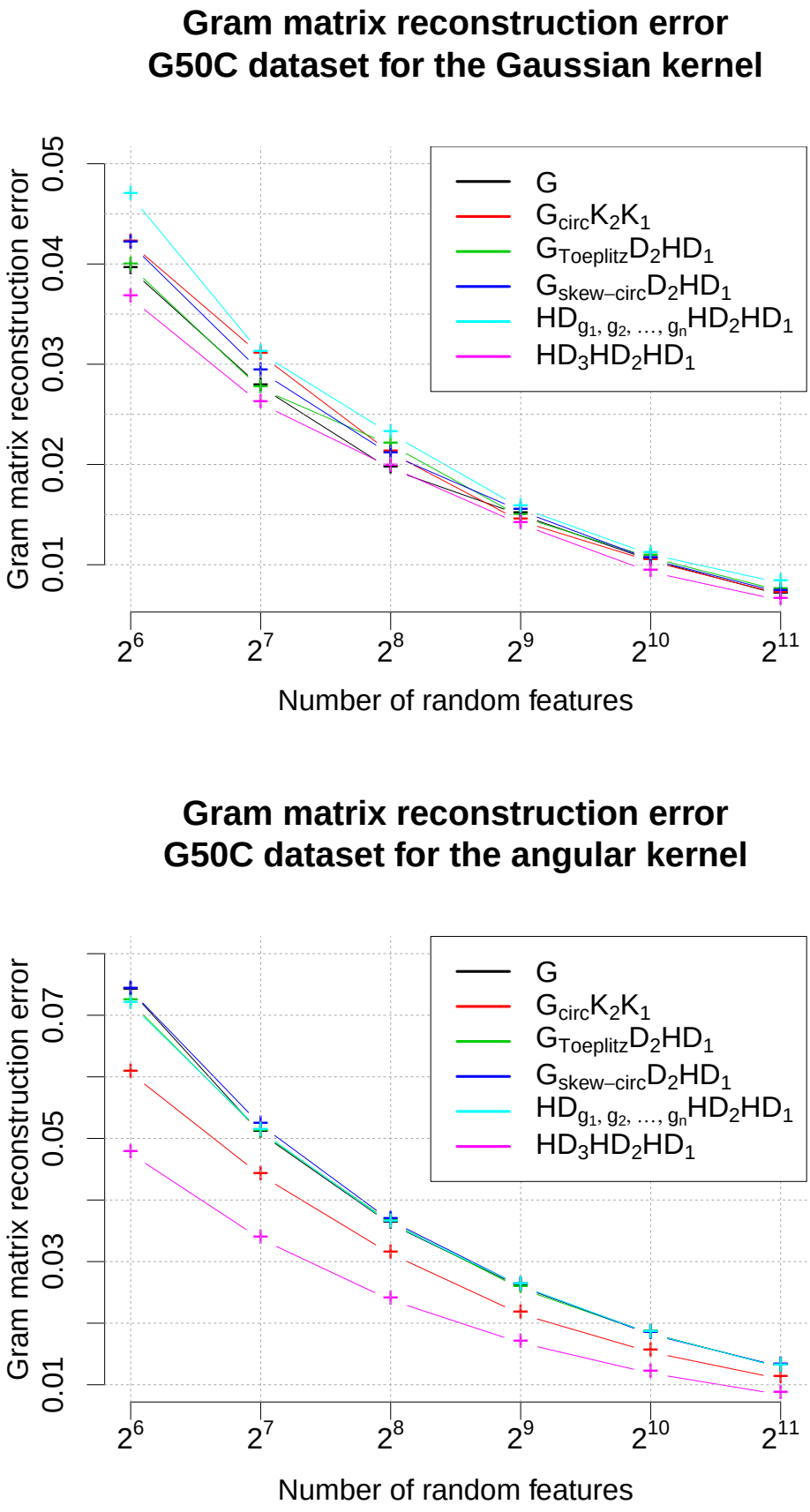
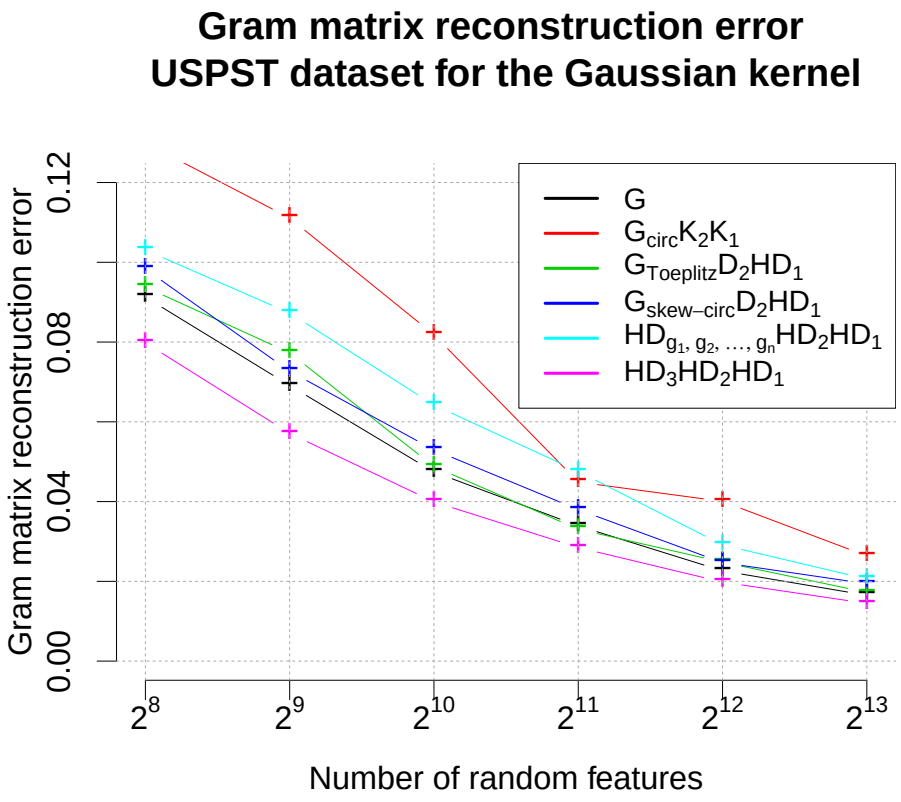


FIGURE C.1: Accuracy of random feature map kernel approximation for the G50C dataset.



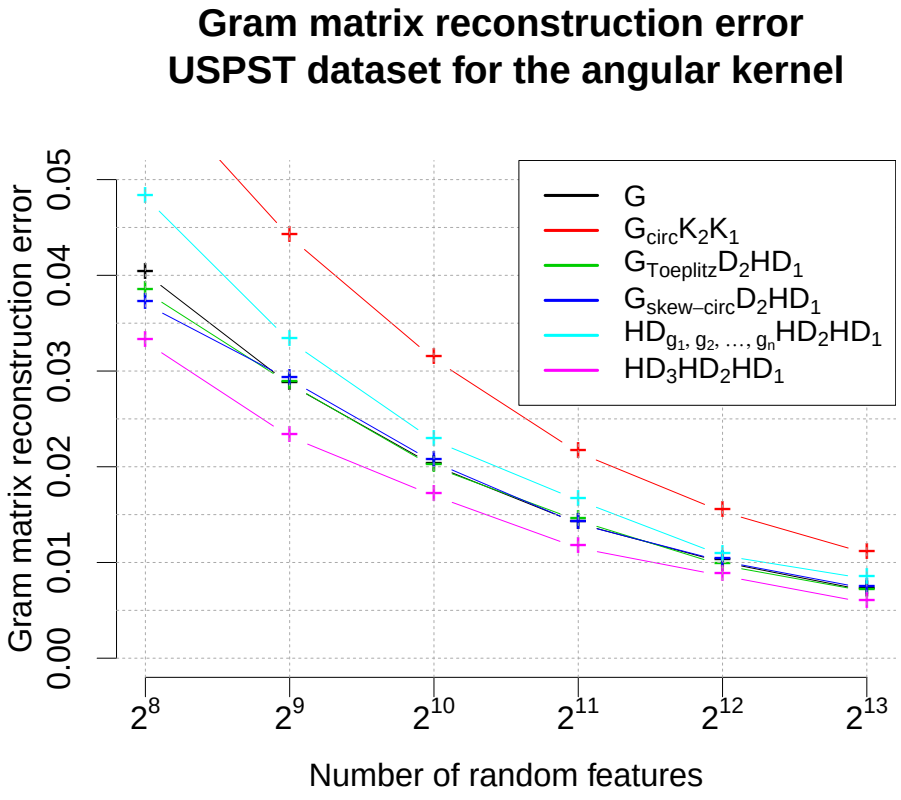
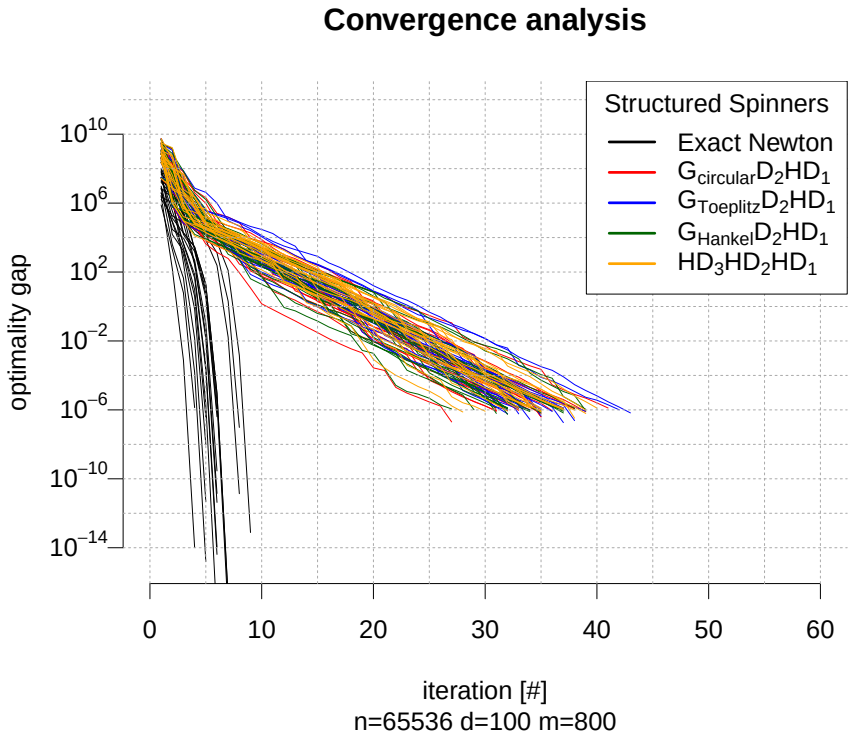
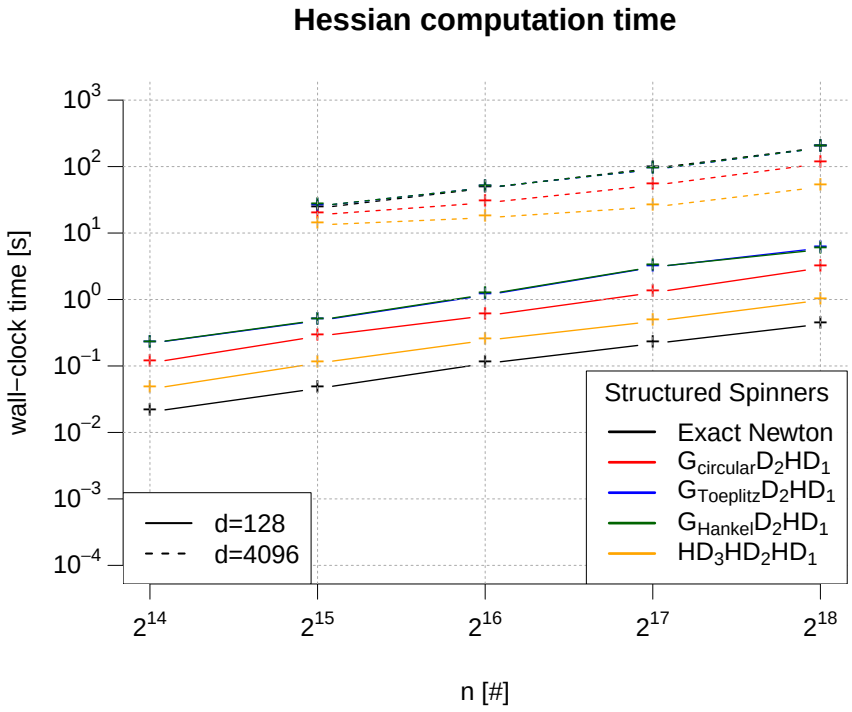


FIGURE C.2: Accuracy of random feature map kernel approximation for the USPST dataset.



(a)



(b)

FIGURE C.3: Numerical illustration of the convergence (a) and computational complexity (b) of the Newton sketch algorithm with various *Structured spinners*. (a) Various sketching structures are compared in terms of the convergence against iteration number. (b) Wall-clock times of *Structured spinners* are compared in various dimensionality settings.

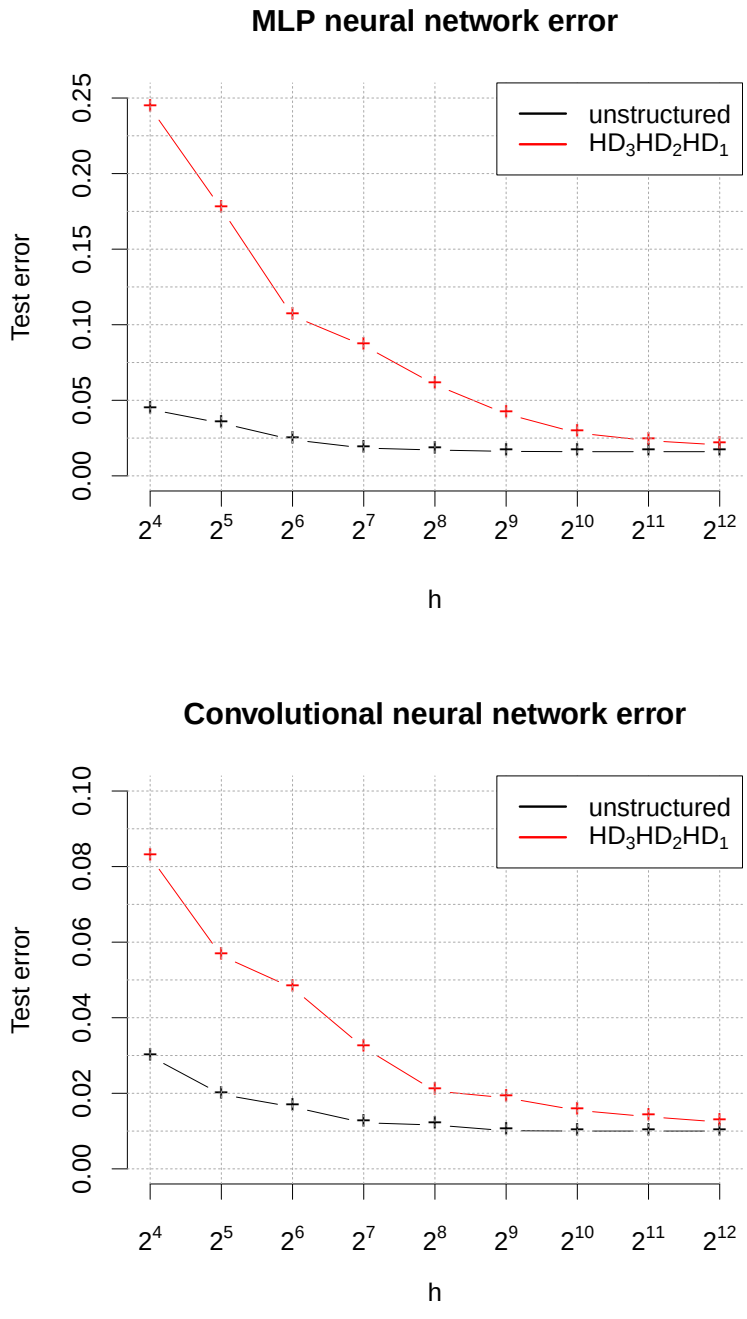


FIGURE C.4: Test error for MLP (left) and convolutional network (right).

Appendix D

Complements of Chapter 4: When matrix R is random

In this Appendix, theoretical optimality justification of a rotation $\mathbf{R}$ in Hypercubic quantization hashing is given for the competing existing online method Online Sketching Hashing (OSH) [Leng et al., 2015a]. Notation from Chapter 4 is kept.

We now show that if we assume $\mathbf{R}$ is a random matrix that resembles a random rotation (i.e. we do not optimize $\mathbf{R}$ and furthermore $\mathbf{R}$ is not a valid rotation matrix) we can still get strong guarantees regarding the quality of the hashing mechanism (even though not as strong as in the case when $\mathbf{R}$ is optimized). We will assume that $\mathbf{R}$ is a Gaussian matrix with entries taken independently at random from $\mathcal{N}(0,1)$. We further make some assumptions regarding the input data and the quality of the PCA projection mechanism.

Hypothesis D.0.1 (H2)**.** *We assume that the variance of the norm of vectors $\mathbf{x}_t$ is upper-bounded, namely we assume that $l(1-\delta) \leq \|\mathbf{x}_t\|_2 \leq l(1+\delta)$ for some fixed $l, \delta > 0$. We also assume that the PCA projection encoded by matrix $\mathbf{W}$ satisfies for every t: $\|\mathbf{W}\mathbf{x}_t - \mathbf{x}_t\|_2 \leq \epsilon\|\mathbf{x}_t\|_2$ for some $\epsilon > 0$. The latter condition means that the fraction of the L_2-norm of the vector lost by performing a PCA projection is upper-bounded (better quality PCA projections are characterized by smaller values of ϵ).*

We now state the main theoretical result of this Section:

Theorem D.0.1. *Let $\mathbf{x}_{t_1}, \mathbf{x}_{t_2} \in \mathbb{R}^d$ be two data points following Hypothesis D.0.1 for some constants $l, \delta, \epsilon > 0$. Then for every $\rho > 0$ the following holds: if $\|\mathbf{x}_{t_1} - \mathbf{x}_{t_2}\|_2 \leq \rho$ and $\mathbf{b}_{t_1} \in \{-1,1\}^c$ and $\mathbf{b}_{t_2} \in \{-1,1\}^c$ with $\mathbf{b}_{t_1} = \mathrm{sign}(\mathbf{R}\mathbf{W}\mathbf{x}_{t_1})$ and $\mathbf{b}_{t_2} = \mathrm{sign}(\mathbf{R}\mathbf{W}\mathbf{x}_{t_2})$ then for any $\eta > 0$ with probability at least $1 - e^{-\frac{\eta^2 c}{2}}$ the number of bits in common between $\mathbf{b}_{t_1}$ and $\mathbf{b}_{t_2}$ is lower bounded by q satisfying*

$$q = c(1 - \eta - \frac{1}{\pi}\arccos(A)) \tag{D.1}$$

with

$$A = \frac{(1-\epsilon)^2(1-\delta)^2}{(1+\delta)^2} - \frac{\rho^2}{2l^2(1+\delta)^2} - \frac{\rho\epsilon}{l(1+\delta)} - 2\epsilon^2 \tag{D.2}$$

Proof. Note that by Hypothesis D.0.1 and triangle inequality, we know that $\mathbf{v}_{t_1} = \mathbf{W}\mathbf{x}_{t_1}$ and $\mathbf{v}_{t_2} = \mathbf{W}\mathbf{x}_{t_2}$ satisfy

$$\begin{aligned}\|\mathbf{v}_{t_1} - \mathbf{v}_{t_2}\|_2 &= \|\mathbf{v}_{t_1} - \mathbf{x}_{t_1} - (\mathbf{v}_{t_2} - \mathbf{x}_{t_2}) + \mathbf{x}_{t_1} - \mathbf{x}_{t_2}\|_2 \\ &\leq \rho + 2\epsilon(1+\delta)l. \end{aligned} \tag{D.3}$$

Now consider a particular entry $i \in \{1, \ldots, c\}$ of two binary codes $\mathbf{b}_{t_1}$ and $\mathbf{b}_{t_2}$, namely $\mathbf{b}_{t_1}^{(i)}$ and $\mathbf{b}_{t_2}^{(i)}$. Note that $\mathbf{b}_{t_1}^{(i)} = \text{sign}((\mathbf{r}^i)^\top \mathbf{v}_{t_1})$ and $\mathbf{b}_{t_2}^{(i)} = \text{sign}((\mathbf{r}^i)^\top \mathbf{v}_{t_2})$, where $\mathbf{r}^i$ stands for the transpose of the i^{th} row of $\mathbf{R}$. We have:

$$(\mathbf{r}^i)^\top \mathbf{v}_{t_1} = (\mathbf{r}^i_{\text{proj}} + \mathbf{r}^i_{\text{ort}})^\top \mathbf{v}_{t_1} = (\mathbf{r}^i_{\text{proj}})^\top \mathbf{v}_{t_1}, \tag{D.4}$$

where $\mathbf{r}^i_{\text{proj}}$ stands for the orthogonal projection of the vector $\mathbf{r}^i$ into a 2-dimensional linear space spanned by $\{\mathbf{v}_{t_1}, \mathbf{v}_{t_2}\}$ and $\mathbf{r}^i_{\text{ort}}$ denoted the part of the vector $\mathbf{r}^i$ that is orthogonal to that subspace.

Similarly, we obtain:

$$(\mathbf{r}^i)^\top \mathbf{v}_{t_2} = (\mathbf{r}^i_{\text{proj}} + \mathbf{r}^i_{\text{ort}})^\top \mathbf{v}_{t_2} = (\mathbf{r}^i_{\text{proj}})^\top \mathbf{v}_{t_2}. \tag{D.5}$$

Therefore we have: $\mathbf{b}_{t_1}^{(i)} = \text{sign}((\mathbf{r}^i_{\text{proj}})^\top \mathbf{v}_{t_1})$ and $\mathbf{b}_{t_2}^{(i)} = \text{sign}((\mathbf{r}^i_{\text{proj}})^\top \mathbf{v}_{t_2})$.

Denote by $\mathcal{L}$ the subset of the two-dimensional subspace spanned by $\{\mathbf{v}_{t_1}, \mathbf{v}_{t_2}\}$ that consists of these vectors $\mathbf{v} \in \mathbb{R}^c$ that satisfy: $(\mathbf{v}^\top \mathbf{v}_{t_1})(v^\top \mathbf{v}_{t_2}) > 0$. Now note that the projection $\mathbf{r}^i_{\text{proj}}$ of the Gaussian vector $\mathbf{r}^i$ into two-dimensional deterministic linear subspace spanned by $\{\mathbf{v}_{t_1}, \mathbf{v}_{t_2}\}$ is still Gaussian. Furthermore, Gaussian vectors satisfy the isotropic property. The probability that $\mathbf{b}_{t_1}^{(i)}$ and $\mathbf{b}_{t_2}^{(i)}$ are the same is exactly the probability that $\mathbf{r}^i_{\text{proj}} \in \mathcal{L}$. From the fact that $\mathbf{r}^i_{\text{proj}}$ is Gaussian and the isotropic property of Gaussian vectors [Charikar, 2002] we conclude that:

$$\mathbb{P}[\mathbf{b}_{t_1}^{(i)} = \mathbf{b}_{t_2}^{(i)}] = \mathbb{P}[\mathbf{r}^i_{\text{proj}} \in \mathcal{L}] = 1 - \frac{\theta_{\mathbf{v}_{t_1}, \mathbf{v}_{t_2}}}{\pi}, \tag{D.6}$$

where $\theta_{\mathbf{v}_{t_1}, \mathbf{v}_{t_2}}$ stands for an angle between $\mathbf{v}_{t_1}$ and $\mathbf{v}_{t_2}$. Besides, we have:

$$\cos(\theta_{\mathbf{v}_{t_1}, \mathbf{v}_{t_2}}) = \frac{\|\mathbf{v}_{t_1}\|_2^2 + \|\mathbf{v}_{t_2}\|_2^2 - \|\mathbf{v}_{t_1} - \mathbf{v}_{t_2}\|_2^2}{2\|\mathbf{v}_{t_1}\|_2 \|\mathbf{v}_{t_2}\|_2} \tag{D.7}$$

Note that by property of the PCA, one has: $||\mathbf{v}_{t_1}||_2 \leq ||\mathbf{x}_{t_1}||_2$ and $||\mathbf{v}_{t_2}||_2 \leq ||\mathbf{x}_{t_2}||_2$. Moreover, triangle inequality gives: $\|\mathbf{v}_{t_1}\|_2 \geq \|\mathbf{x}_{t_1}\|_2 - \|\mathbf{v}_{t_1} - \mathbf{x}_{t_1}\|_2$. Thus, $\|\mathbf{v}_{t_1}\|_2 \geq (1-\epsilon)(1-\delta)l$ by using Hypothesis D.0.1. Similarly, $\|\mathbf{v}_{t_2}\|_2 \geq (1-\epsilon)(1-\delta)l$. From the above and by using Equation D.3, we get:

$$\begin{aligned}\cos(\theta_{\mathbf{v}_{t_1}, \mathbf{v}_{t_2}}) &\geq \frac{2(1-\epsilon)^2(1-\delta)^2 l^2 - (\rho + 2\epsilon(1+\delta)l)^2}{2l^2(1+\delta)^2} \\ &\geq \frac{(1-\epsilon)^2(1-\delta)^2}{(1+\delta)^2} - \frac{\rho^2}{2l^2(1+\delta)^2} - \frac{\rho\epsilon}{l(1+\delta)} - 2\epsilon^2 \end{aligned} \tag{D.8}$$

Hence,

$$\theta_{\mathbf{v}_{t_1}, \mathbf{v}_{t_2}} \leq \arccos\left(\frac{(1-\epsilon)^2(1-\delta)^2}{(1+\delta)^2} - \frac{\rho^2}{2l^2(1+\delta)^2} - \frac{\rho\epsilon}{l(1+\delta)} - 2\epsilon^2\right). \tag{D.9}$$

Now denote by X_i a random variable that is $+1$ if $\mathbf{b}_{t_1}^{(i)}$ and $\mathbf{b}_{t_2}^{(i)}$ are the same and is 0 otherwise. Note that the probability that X_i is nonzero is exactly

$$p = 1 - \frac{\theta_{\mathbf{v}_{t_1},\mathbf{v}_{t_2}}}{\pi}. \tag{D.10}$$

Note also that different X_i are independent since different rows $\mathbf{r}^i$ of the matrix $\mathbf{R}$ are independent. If we denote by X a random variable defined as:

$$X := X_1 + \ldots + X_c, \tag{D.11}$$

then note that $\mathbb{E}[X] = pc$. This random variable X counts the number of entries of $\mathbf{b}_{t_1}$ and $\mathbf{b}_{t_2}$ that are the same. Now, using standard concentration results such as Azuma's inequality [Azuma, 1967], we can conclude that for all $a > 0$:

$$\mathbb{P}[X - pc < -a] \leq e^{-\frac{a^2}{2c}}. \tag{D.12}$$

In particular, for $a = \eta c$,

$$\mathbb{P}[X - pc < -\eta c] \leq e^{-\frac{\eta^2 c}{2}}. \tag{D.13}$$

Then with probability at least $1 - e^{-\frac{\eta^2 c}{2}}$, one has: $X \geq c(p - \eta)$. Putting the formula on p from Equation D.10 and the obtained upper bound on $\theta_{\mathbf{v}_{t_1},\mathbf{v}_{t_2}}$ from Equation D.9, we complete the proof. □

Appendix E

Graph-based clustering under differential privacy

Contents

This Appendix concerns a collaboration with Rafaël Pinot[1] *and Florian Yger*[2]*. This work has been published at the International conference on Uncertainty in Artificial Intelligence (UAI) 2018 under the title "Graph-based Clustering under Differential Privacy". It is an applicative Appendix to Chapter 5.*

In this Appendix, the first differentially-private clustering method for arbitrary-shaped node clusters in a graph is presented. This algorithm takes as input only an approximate Minimum Spanning Tree (MST) $\mathcal{T}$ released under weight differential privacy constraints from the graph. Then, the underlying non-convex clustering partition is successfully recovered from cutting optimal cuts on $\mathcal{T}$. As opposed to existing methods, this algorithm is theoretically well-motivated. Experiments support the theoretical findings.

This is a direct application of clustering algorithm DBMSTClu presented in Chapter 5 for differentially-private clustering.

[1]CEA, Université Paris-Dauphine, PSL Research University
[2]Université Paris-Dauphine, PSL Research University

E.1 Introduction

Similarly to Chapter 2, Section 2.4 and Chapter 5, we consider in this Appendix simple undirected weighted graphs where the weighted edges express "distances" between the objects represented by the vertices. For understanding the underlying structure in this kind of graphs [Schaeffer, 2007], graph clustering is then one of the key tools: the resulting clusters can be seen as groups of nodes close in terms of some specific similarity.

> Nevertheless, it is critical that the data representation used in machine learning applications protects the private characteristics contained into it.

Let us consider an application where one wants to identify groups of similar web pages in the sense of traffic volume i.e. web pages with similar audience. In that case, the nodes stand for the websites. The link between two vertices represents the fact that some people consult them both. The edge weights are the number of common users and thus, carry sensitive information about individuals. During any graph data analysis, no private user surfing behavior should be breached i.e. browsing from one page to another should remain private. As a standard for data privacy preservation, differential privacy [Dwork et al., 2006b] has been designed:

> An algorithm is differentially private if, given two close databases, it produces statistically indistinguishable outputs.

Since then, its definition has been extended to weighted graphs. Though, machine learning applications ensuring data privacy remain rare, in particular for clustering which encounters severe theoretical and practical limitations. Indeed, some clustering methods lack of theoretical support and most of them restrict the data distribution to convex-shaped clusters [Nissim et al., 2007, Blum et al., 2008, McSherry, 2009, Dwork, 2011] or unstructured data [Ho and Ruan, 2013, Chen et al., 2015].

> Hence, the aim of this Appendix is to offer a theoretically motivated private graph clustering. Moreover, to the best of our knowledge, this is the first weight differentially-private clustering algorithm able to detect clusters with an arbitrary shape for weighted graph data.

Our method belongs to the family of Minimum Spanning Tree (MST)-based approaches. Recall that an MST represents a useful summary of the graph, and appears to be a natural object to describe it at a lower cost. For clustering purposes, Section 2.4.1 showed that it has the appealing property to help retrieving non-convex shapes [Zahn, 1971, Asano et al., 1988, Grygorash et al., 2006]. Moreover, they appear to be well-suited for incorporating privacy constraints as it will be formally proved in this work.

Contributions The contributions in this Appendix are two-fold:

1. A differentially-private version of DBMSTClu is introduced.
2. Several results on its privacy/utility tradeoff are given.

E.2 Preliminaries

Notations from Section 5.2.1 are kept. Moreover, $\mathcal{W}_E$ denotes the set of all possible weight functions mapping the edge set E to weights in $\mathbb{R}$.

E.2.1 Differential privacy in graphs

As opposed to *node*-differential privacy [Kasiviswanathan et al., 2013] and *edge*-differential privacy [Hay et al., 2009], both based on the graph topology, the privacy framework considered here is *weight*-differential privacy where the graph topology $G = (V, E)$ is assumed to be public and the private information to protect is the weight function $w := E \to \mathbb{R}$. Under this model introduced by Sealfon [2016], two graphs are said to be neighbors if they have the same topology, and *close* weight functions. This framework allows one to release an almost MST with weight-approximation error of $O(|V| \log |E|)$ for fixed privacy parameters. Differential privacy is ensured in that case by using the Laplace mechanism on every edges weight to release a spanning tree based on a perturbed version of the weight function. The privacy of the spanning tree construction is thus provided by post-processing (cf. Theorem E.2.5).

However, under a similar privacy setting, Pinot [2018] recently manages to produce the topology of a tree under differential privacy without relying on the post-processing of a more general mechanism such as the *Laplace mechanism*. The algorithm from Pinot [2018], called PAMST, privately releases the topology of an almost MST thanks to an iterative use of the *Exponential mechanism* instead. For fixed privacy parameters, the weight approximation error is $O\left(\frac{|V|^2}{|E|} \log |V|\right)$, which outperforms the former method from Sealfon [2016] on arbitrary weighted graphs under weak assumptions on the graph sparseness. Thus, we keep here privacy setting from Pinot [2018].

Definition E.2.1 (neighboring weight functions [Pinot, 2018])**.** *For any edge set E, two weight functions $w, w' \in \mathcal{W}_E$ are* neighboring*, denoted $w \sim w'$, if $||w - w'||_\infty := \max_{e \in E} |w(e) - w'(e)| \leq \mu$.*

μ represents the sensitivity of the weight function and should be chosen according to the application and the range of this function. The neighborhood between such graphs is clarified in the following definition.

Definition E.2.2 (neighboring graphs)**.** *Let $\mathcal{G} = (V, E, w)$ and $\mathcal{G}' = (V', E', w')$, two weighted graphs, $\mathcal{G}$ and $\mathcal{G}'$ are said to be* neighbors *if $V = V'$, $E = E'$ and $w \sim w'$.*

The so-called weight-differential privacy for graph algorithms is now formally defined.

Definition E.2.3 (weight-differential-private graph algorithm [Sealfon, 2016])**.** *For any graph topology $G = (V, E)$, let $\mathcal{A}$ be a randomized algorithm that takes as input a weight function $w \in \mathcal{W}_E$. $\mathcal{A}$ is called* (ϵ, δ)-differentially private on $G = (V, E)$ *if for all pairs of neighboring weight functions $w, w' \in \mathcal{W}_E$, and for all set of possible outputs S, one has*

$$\mathbb{P}[\mathcal{A}(w) \in S] \leq e^{\epsilon} \mathbb{P}[\mathcal{A}(w') \in S] + \delta. \tag{E.1}$$

If $\mathcal{A}$ is (ϵ, δ)-differentially private on every graph topology in a class $\mathcal{C}$, it is said to be (ϵ, δ)-differentially private on $\mathcal{C}$.

One of the first, and most used differentially private mechanisms is the *Laplace mechanism*. It is based on the process of releasing a numerical query perturbed by a noise drawn from a centered Laplace distribution scaled to the sensitivity of the query. We present here its graph-based reformulation.

Definition E.2.4 (function sensitivity, reformulation of [Dwork et al., 2006b])**.** *Given some graph topology $G = (V, E)$, for any $f_G : \mathcal{W}_E \to \mathbb{R}^k$, the* sensitivity *of the function is defined as $\Delta f_G := \max_{w \sim w' \in \mathcal{W}_E} ||f_G(w) - f_G(w')||_1$.*

Definition E.2.5 (graph-based Laplace mechanism, reformulation of [Dwork et al., 2006b]). *Given some graph topology $G = (V, E)$, any function $f_G : \mathcal{W}_E \to \mathbb{R}^k$, any $\epsilon > 0$, and $w \in \mathcal{W}_E$, the* graph-based Laplace mechanism *is $\mathcal{M}_L(w, f_G, \epsilon) = f_G(w) + (Y_1, \ldots, Y_k)$ where Y_i are i.i.d. random variables drawn from $Lap(\Delta f_G/\epsilon)$, and $Lap(b)$ denotes the Laplace distribution with scale b* (i.e *probability density* $\frac{1}{2b} \exp\left(-\frac{|x|}{b}\right)$).

Theorem E.2.1 (Dwork et al. [2006b]). *The Laplace mechanism is ϵ-differentially private.*

We define hereafter the *graph-based Exponential mechanism*. In the sequel we refer to it simply as *Exponential mechanism*. The Exponential mechanism represents a way of privately answering arbitrary range queries. Given some range of possible responses to the query $\mathcal{R}$, it is defined according to a utility function $u_G := \mathcal{W}_E \times \mathcal{R} \to \mathbb{R}$, which aims at providing some total preorder on the range $\mathcal{R}$ according to the total order in $\mathbb{R}$. The sensitivity of this function is denoted $\Delta u_G := \max_{r \in \mathcal{R}} \max_{w \sim w' \in \mathcal{W}_E} |u_G(w, r) - u_G(w', r)|$.

Definition E.2.6 (graph-based Exponential mechanism). *Given some graph topology $G = (V, E)$, some output range $\mathcal{R} \subset E$, some privacy parameter $\epsilon > 0$, some utility function $u_G := \mathcal{W}_E \times \mathcal{R} \to \mathbb{R}$, and some $w \in \mathcal{W}_E$ the* graph-based Exponential mechanism $\mathcal{M}_{Exp}(G, w, u_G, \mathcal{R}, \epsilon)$ *selects and outputs an element $r \in \mathcal{R}$ with probability proportional to* $\exp\left(\frac{\epsilon u_G(w,r)}{2\Delta u_G}\right)$.

The Exponential mechanism defines a distribution on a potentially complex and large range $\mathcal{R}$. As the following theorem states, sampling from such a distribution preserves ϵ-differential privacy.

Theorem E.2.2 (reformulation of [McSherry and Talwar, 2007]). *For any non-empty range $\mathcal{R}$, given some graph topology $G = (V, E)$, the graph-based Exponential mechanism preserves ϵ-differential privacy,* i.e *if $w \sim w' \in \mathcal{W}_E$,*

$$\mathbb{P}\left[\mathcal{M}_{Exp}(G, w, u_G, \mathcal{R}, \epsilon) = r\right] \leq e^{\epsilon} \mathbb{P}\left[\mathcal{M}_{Exp}(G, w', u_G, \mathcal{R}, \epsilon) = r\right]. \tag{E.2}$$

Further, Theorem E.2.3 highlights the trade-off between privacy and accuracy for the Exponential mechanism when $0 < |\mathcal{R}| < +\infty$. Theorem E.2.4 presents the ability of differential privacy to comply with composition while Theorem E.2.5 introduces its post-processing property.

Theorem E.2.3 (reformulation of [Dwork and Roth, 2013]). *Given some graph topology $G = (V, E)$, some $w \in \mathcal{W}_E$, some output range $\mathcal{R}$, some privacy parameter $\epsilon > 0$, some utility function $u_G := \mathcal{W}_E \times \mathcal{R} \to \mathbb{R}$, and denoting $OPT_{u_G}(w) := \max_{r \in \mathcal{R}} u_G(w, r)$, one has $\forall\ t \in \mathbb{R}$,*

$$u_G\left(G, w, \mathcal{M}_{Exp}(w, u_G, \mathcal{R}, \epsilon)\right) \leq OPT_{u_G}(w) - \frac{2\Delta u_G}{\epsilon}\left(t + \ln|\mathcal{R}|\right) \tag{E.3}$$

with probability at most $\exp(-t)$.

Theorem E.2.4 (Dwork et al. [2006a]). *For any $\epsilon > 0$, $\delta \geq 0$ the adaptive composition of k (ϵ, δ)-differentially private mechanisms is $(k\epsilon, k\delta)$-differentially private.*

Theorem E.2.5 (Post-Processing [Dwork and Roth, 2013]). *Let $\mathcal{A} : \mathcal{W}_E \to B$ be a randomized algorithm that is (ϵ, δ)-differentially private, and $h : B \to B'$ a deterministic mapping. Then $h \circ \mathcal{A}$ is (ϵ, δ)-differentially private.*

E.2.2 Differentially-private clustering

Differentially-private clustering for unstructured datasets has been first discussed in work from Nissim et al. [2007]. The latter introduced the first method for differentially-private clustering based on the k-means algorithm. Since then, most of the work in the field focused on adaptation of this method [Blum et al., 2008, McSherry, 2009, Dwork, 2011]. The main drawback of this work is that it is not able to deal with arbitrary-shaped clusters. This issue has been recently investigated by Ho and Ruan [2013] and Chen et al. [2015]. They proposed two new methods to find arbitrary-shaped clusters in unstructured datasets respectively based on density clustering and wavelet decomposition. Even though both of them allow one to produce non-convex clusters, they only deal with unstructured datasets and thus are not applicable to node clustering in a graph.

Our work focuses on node clustering in a graph under weight-differential privacy. Graph clustering has already been investigated in a topology-based privacy framework [Mülle et al., 2015, Nguyen et al., 2016], however, this work does not consider weight-differential privacy.

Our work is, to the best of our knowledge, the first attempt to define node clustering in a graph under weight-differential privacy.

E.3 Differentially-private tree-based clustering

We aim at producing a private clustering method while providing bounds on the accuracy loss. Our method is an adaptation of an existing clustering algorithm DBMSTClu.

This Section presents our new node clustering algorithm PTClust for weight-differential privacy. It relies on a mixed adaptation of PAMST algorithm [Pinot, 2018] for recovering a differentially-private MST of a graph and DBMSTClu (see Chapter 5).

E.3.1 PAMST algorithm

Given a simple undirected weighted graph $\mathcal{G} = (V, E, w)$, PAMST outputs an almost minimal weight spanning tree topology under differential privacy constraints. It relies on a Prim-like MST algorithm, and an iterative use of the graph-based Exponential mechanism. PAMST takes as an input a weighted graph, and a utility function. It outputs the topology of a spanning tree whose weight is almost minimal. Algorithm 10 presents this new method, with illustrative Figure E.1, using the following utility function:

$$\begin{array}{rccc} u_G & : & \mathcal{W}_E \times \mathcal{R} & \to & \mathbb{R} \\ & & (w, r) & \mapsto & -|w(r) - \min\limits_{r' \in \mathcal{R}} w(r')|. \end{array} \tag{E.4}$$

PAMST starts by choosing an arbitrary node to construct iteratively the tree topology. At every iteration, it uses the Exponential mechanism to find the next edge to be added to the current tree topology while keeping the weights private. This algorithm is the state of the art to find a spanning tree topology under differential privacy. For readability, let us introduce some additional notations. Let S be a set of nodes from G, and $\mathcal{R}_S$ the set of edges that are incident to one and only one node in S (also denoted xor-incident). For any edge r in such a set, the incident node to r that is not in S is denoted $r_{\to}$. Finally, the restriction of the weight function to an edge set $\mathcal{R}$ is denoted $w_{|\mathcal{R}}$.

Algorithm 10 PAMST(G, u_G, w, ϵ)

Input: $\mathcal{G} = (V, E, w)$ a weighted graph (separately the topology G and the weight function w), ϵ a degree of privacy and u_G utility function.
Pick $v \in V$ at random
$S_V \leftarrow \{v\}$
$S_E \leftarrow \emptyset$
while $S_V \neq V$ **do**
  $r = \mathcal{M}_{Exp}(\mathcal{G}, w, u_G, \mathcal{R}_{S_V}, \frac{\epsilon}{|V|-1})$
  $S_V \leftarrow S_V \cup \{r_{\rightarrow}\}$
  $S_E \leftarrow S_E \cup \{r\}$
return S_E

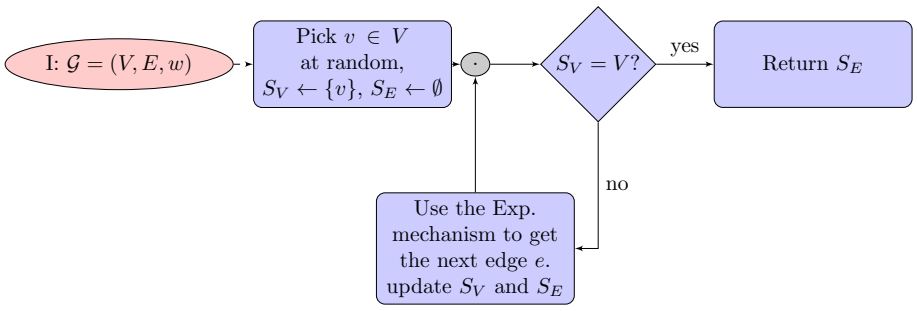


FIGURE E.1: PAMST algorithm from Pinot [2018]. I: Input.

Theorem E.3.1 states that using PAMST to get an almost MST topology preserves weight-differential privacy.

Theorem E.3.1. *Let $G = (V, E)$ be the topology of a simple undirected graph, then $\forall \epsilon > 0$, PAMST$(G, u_G, \bullet, \epsilon)$ is ϵ-differentially-private on G.*

E.3.2 Differentially-private clustering

The overall goal of this Section is to show that one can obtain a differentially-private clustering algorithm by combining PAMST and DBMSTClu algorithms. However, PAMST does not output a weighted tree which is inappropriate for clustering purposes. To overcome this, one could rely on a sanitizing mechanism such as the Laplace mechanism. Moreover, since DBMSTClu only takes weights from $(0, 1]$, two normalizing parameters τ and p are introduced, respectively to ensure lower and upper bounds to the weights that fit within DBMSTClu needs. This sanitizing mechanism is called the *Weight-Release mechanism*. Coupled with PAMST, it will allows us to produce a weighted spanning tree with differential privacy, that will be exploited in our private graph clustering.

Definition E.3.1 (Weight-Release mechanism)**.** *Let $\mathcal{G} = (G, w)$ be a weighted graph, $\epsilon > 0$ a privacy parameter, s a scaling parameter, $\tau \geq 0$, and $p \geq 1$ two normalization parameters. The* Weight-Release mechanism *is defined as*

$$\mathcal{M}_{w.r}(G, w, s, \tau, p) := \left(G, w' = \frac{w + (Y_1, ..., Y_{|E|}) + \tau}{p}\right) \tag{E.5}$$

where Y_i are i.i.d. random variables drawn from Lap$(0, s)$. With $w + (Y_1, ..., Y_{|E|})$ meaning that if one gives an arbitrary order to the edges $E = (e_i)_{i \in [|E|]}$, one has $\forall i \in [|E|]$, $w'(e_i) = w(e_i) + Y_i$.

The following Theorem presents the privacy guarantees of the Weight-Release mechanism.

Theorem E.3.2. *Let $G = (V, E)$ be the topology of a simple undirected graph, $\tau \geq 0$, $p \geq 1$, then $\forall \epsilon > 0$, $\mathcal{M}_{w.r}\left(G, \bullet, \frac{\mu}{\epsilon}, \tau, p\right)$ is ϵ- differentially private on G.*

> *Proof.* Given $\tau \geq 0$, $p \geq 1$, and $\epsilon > 0$, the Weight release mechanism scaled to $\frac{\mu}{\epsilon}$ can be break down into a Laplace mechanism and a post-processing consisting in adding τ to every edge and dividing them by p. Using Theorems E.2.1 and E.2.5, one gets the expected result. □

So far we have presented DBMSTClu and PAMST algorithms, and the Weight-Release mechanism. Let us now introduce how to compose those blocks to obtain a Private node clustering in a graph, called PTClust (see Algorithm 11 and Figure E.2). The algorithm takes as an input a weighted graph (dissociated topology and weight function), a utility function, a privacy degree and two normalization parameters. It outputs a clustering partition. To do so, a spanning tree topology is produced using PAMST. Afterward a randomized and normalized version of the associated weight function is released using the Weight-release mechanism. Finally the obtained weighted tree is given as an input to DBMSTClu that performs a clustering partition. The following Theorem ensures that our method preserves ϵ-differential privacy.

Theorem E.3.3. *Let $G = (V, E)$ be the topology of a simple undirected graph, $\tau \geq 0$, and $p \geq 1$, then $\forall \epsilon > 0$, PTClust$(G, \bullet, u_G, \epsilon, \tau, p)$ is ϵ-differentially private on G.*

Algorithm 11 PTClust$(G, w, u_G, \epsilon, \tau, p)$

1: **Input:** $\mathcal{G} = (V, E, w)$ a weighted graph (separately the topology G and the weight function w), ϵ a degree of privacy and u_G utility function.
2: $T = PAMST(G, w, u_G, \epsilon/2)$
3: $\mathcal{T}' = \mathcal{M}_{w.r}(T, w_{|E(T)}, \frac{2\mu}{\epsilon}, \tau, p)$
4: **return** DBMSTClu$(\mathcal{T}')$

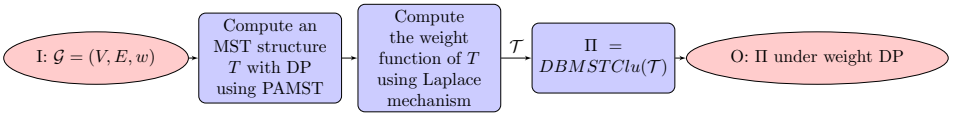


FIGURE E.2: Our new method: PTClust returning a clustering partition Π under weight Differential Privacy (DP). I: Input, O: Output.

Proof. Using Theorem E.3.1 one has that T is produced with $\epsilon/2$-differential privacy, and using Theorem E.3.2 one has that w' is obtained with $\epsilon/2$-differential privacy as well. Therefore using Theorem E.2.4, $\mathcal{T}'$ is released with ϵ-differential privacy. Using the post-processing property (Theorem E.2.5) one gets the expected result. □

E.3.3 Differential privacy trade-off of clustering

The results stated in this Section present the security/accuracy trade-off of our new method in the differentially-private framework. PTClust relies on two differentially-private mechanisms, namely PAMST and the Weight-Release mechanism. Evaluating the accuracy of PTClust amounts to check whether using PAMST and the Weight-Release mechanism for ensuring privacy does not deteriorate the final clustering partition. The accuracy is preserved if PAMST outputs the same topology as the MST-based clustering and if the Weight-Release mechanism preserves enough the weight function. According to Definition 5.4.2 p.103, if a tree has a partitioning topology, then it fits the tree-based clustering. Theorem E.3.4 states that with high probability PAMST outputs a tree with a partitioning topology.

While the accuracy of DBMSTClu (in the deterministic setting) was proved under the weak homogeneity condition (cf. Definition 5.4.4 p.103 on clusters), Theorem E.3.4 gives accuracy of its differentially-private version under the strong homogeneity condition. Strong homogeneity condition appears to be naturally more constraining on the edge weights than the weak one.

Definition E.3.2 (Strong homogeneity condition of a Cluster)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K clusters $C_1^*, \ldots, C_K^*$. A given cluster C_i^*, $i \in [K]$, C_i^* is strongly homogeneous if: for all $\mathcal{T}$ a spanning tree (ST) of $\mathcal{G}$, and $\forall j \in [K]$, $j \neq i$, s.t. $e^{(ij)} \in Cut_{\mathcal{G}}(\mathcal{T})$, $H_{\mathcal{T}_{|C_i^*}}(e^{(ij)})$ is verified. For simplicity, one denote* $\bar{\alpha}_i := \max_{\mathcal{T} \text{ ST of } \mathcal{G}} \alpha_{\mathcal{T}_{|C_i^*}}$.

We show that the weak homogeneity condition is implied by the strong homogeneity condition.

Proposition E.3.1. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K clusters $C_1^*, \ldots, C_K^*$. If a given cluster C_i^*, $i \in [K]$ is strongly homogeneous, then, it is weakly homogeneous.*

Proof. If $\mathcal{T}$ a spanning tree of $\mathcal{G}$, and $\forall j \in [K]$, $j \neq i$, s.t. $e^{(ij)} \in Cut_{\mathcal{G}}(\mathcal{T})$, $H_{\mathcal{T}_{|C_i^*}}(e^{(ij)})$ is verified, then in particular, it is true for any MST. □

Theorem E.3.4. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K strongly homogeneous clusters $C_1^*, \ldots, C_K^*$ and $T = PAMST(\mathcal{G}, u_{\mathcal{G}}, w, \epsilon)$, $\epsilon > 0$. T has a partitioning topology with probability at least*

$$1 - \sum_{i=1}^{K} (|C_i^*| - 1) \exp\left(-\frac{A}{2\Delta u_{\mathcal{G}}(|V| - 1)}\right) \tag{E.6}$$

with $A = \epsilon \left(\bar{\alpha}_i \max_{e \in E\left(\mathcal{G}_{|C_i^*}\right)}(w(e)) - \min_{e \in E\left(\mathcal{G}_{|C_i^*}\right)}(w(e)) \right) + \ln |E|$.

Proof. Let $\mathcal{T} = \text{PAMST}(\mathcal{G}, u_{\mathcal{G}}, w, \epsilon)$, $\{\mathcal{R}_1, ..., \mathcal{R}_{|V|-1}\}$ denotes the ranges used in the successive calls of the Exponential mechanism in $\text{PAMST}(\mathcal{G}, u_{\mathcal{G}}, w, \epsilon)$, $r_k = \mathcal{M}_{Exp}(\mathcal{G}, w, u_{\mathcal{G}}, \mathcal{R}_k, \underbrace{\frac{\epsilon}{|V| - 1}}_{\epsilon'})$, and $\text{Steps}(C_i^*)$ the set of steps k of the algorithm were $\mathcal{R}_k$ contains at least one edge from $\mathcal{G}_{|C_i^*}$. Finally for readability we denote $u_k = u_{\mathcal{G}}(w, r_k)$. Then,

$$\begin{aligned} &\mathbb{P}[\mathcal{T} \text{ has a partitioning topology}] \\ =&\mathbb{P}[\underbrace{\forall i, j \in [K],\ i \neq j,\ |\{(u, v) \in E(\mathcal{T}),\ u \in C_i^*,\ v \in C_j^*\}| = 1}_{A}] = 1 - \mathbb{P}[\neg A] \end{aligned} \tag{E.7}$$

If we denote $B =$ "$\forall i \in [K],\ \forall k > 1 \in \text{Steps}(C_i^*)$, if $r_{k-1} \in E(\mathcal{G}_{|C_i^*})$ then $r_k \in E(\mathcal{G}_{|C_i^*})$". One easily has: $B \implies A$, therefore $\mathbb{P}[\neg A] \leq \mathbb{P}(\neg B)$. Moreover, by using the privacy/accuracy trade-off of the Exponential mechanism, one has:

$$\forall t \in \mathbb{R}, \forall i \in [K], \forall k \in \text{Steps}(C_i^*)\ \mathbb{P}\left[\underbrace{u_k \leq -\frac{2\Delta u_{\mathcal{G}}}{\epsilon'}(t + \ln |\mathcal{R}_k|)}_{A_k(t)}\right] \leq \exp(-t). \tag{E.8}$$

Moreover one can major $\mathbb{P}[\neg B]$ as follows:

$$\mathbb{P}\left[\exists i \in [K], \exists k \in \text{Steps}(C_i^*) \text{ s.t } r_{k-1} \in E(\mathcal{G}_{|C_i^*}) \text{ and } r_k \notin E(\mathcal{G}_{|C_i^*})\right] \tag{E.9}$$

By using the union bound, one gets:

$$\leq \sum_{i \in [K]} \mathbb{P}\left[\exists k \in \text{Steps}(C_i^*) \text{ s.t } r_{k-1} \in E(\mathcal{G}_{|C_i^*}) \text{ and } r_k \notin E(\mathcal{G}_{|C_i^*})\right] \tag{E.10}$$

Using the strong homogeneity of the clusters, one has:

$$= \sum_{i\in[K]} \mathbb{P}\left[\exists k \in \text{Steps}(C_i^*) \text{ s.t } u_k \leq -|\bar{\alpha}_i \max_{e\in E(\mathcal{G}_{|C_i^*})} w(e) - \min_{r\in\mathcal{R}_k} w(r)|\right] \tag{E.11}$$

$$\leq \sum_{i\in[K]} \mathbb{P}\left[\exists k \in \text{Steps}(C_i^*) \text{ s.t } u_k \leq -|\bar{\alpha}_i \max_{e\in E(\mathcal{G}_{|C_i^*})} w(e) - \min_{e\in E(\mathcal{G}_{|C_i^*})} w(e)|\right] \tag{E.12}$$

By setting $t_{k,i} = \frac{\epsilon'}{2\Delta u_{\mathcal{G}}}(\bar{\alpha}_i \max_{e\in E(\mathcal{G}_{|C_i^*})}(w(e)) - \min_{e\in E(\mathcal{G}_{|C_i^*})}(w(e))) + \ln(|\mathcal{R}_k|)$, one gets:

$$= \sum_{i\in[K]} \mathbb{P}\left[\exists k \in \text{Steps}(C_i^*) \text{ s.t } A_k(t_{k,i})\right] \tag{E.13}$$

Since for all $i \in [K]$, and $k \in \text{Steps}(C_i^*)$, $|\mathcal{R}_k| \leq |E|$, and using a union bound, one gets:

$$\leq \sum_{i\in[K]} \sum_{k\in\text{Steps}(C_i^*)} \mathbb{P}\quad [A_k(t_{k,i})] \leq \sum_{i\in[K]} \sum_{k\in\text{Steps}(C_i^*)} \exp\ (-t_{i,k}) \tag{E.14}$$

$$\leq \sum_{i=1}^{K} (|C_i^*| - 1) \exp\left(-\frac{\epsilon}{2\Delta u_{\mathcal{G}}(|V|-1)}\left(\bar{\alpha}_i \max_{e\in E(\mathcal{G}_{|C_i^*})} w(e) - \min_{e\in E(\mathcal{G}_{|C_i^*})} w(e)\right) + \ln(|E|)\right) \tag{E.15}$$

□

Finally, Theorem E.3.5 states that given a tree $\mathcal{T}$ under the strong homogeneity condition, if the subtree associated to a cluster respects Definition 5.4.3, then it still holds after applying the Weight-Release mechanism to this tree.

Theorem E.3.5. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K strongly homogeneous clusters $C_1^*, \ldots, C_K^*$ and $T = PAMST(\mathcal{G}, u_{\mathcal{G}}, w, \epsilon)$, $\mathcal{T} = (T, w_{|T})$ and $\mathcal{T}' = \mathcal{M}_{w.r}(T, w_{|T}, s, \tau, p)$ with $s \ll p, \tau$. Given some cluster C_i^*, and $j \neq i$ s.t $e^{(ij)} \in Cut_{\mathcal{G}}(\mathcal{T})$, if $H_{\mathcal{T}_{|C_i^*}}(e^{(ij)})$ is verified, then $H_{\mathcal{T}'_{|C_i^*}}(e^{(ij)})$ is verified with probability at least*

$$1 - \frac{\mathbb{V}(\varphi)}{\mathbb{V}(\varphi) + \mathbb{E}(\varphi)^2} \tag{E.16}$$

with the following notations :

- $\varphi = \max_{j\in[|C_i^*|-1]}(Y_j)^2 - \min_{j\in[|C_i^*|-1]} Z_j \times X^{out}$
- $Y_j \underset{iid}{\sim} Lap\left(\frac{\max_{e\in E(\mathcal{T})} w(e)+\tau}{p}, \frac{s}{p}\right)$
- $Z_j \underset{iid}{\sim} Lap\left(\frac{\min_{e\in E(\mathcal{T})} w(e)+\tau}{p}, \frac{s}{p}\right)$
- $X^{out} \sim Lap\left(\frac{w(e^{(ij)})+\tau}{p}, \frac{s}{p}\right)$,

The proof of Theorem E.3.5 uses the result from S. Kotz on the Laplace distribution and generalizations (2001):

Theorem E.3.6. *Let $n \in \mathbb{N}$, $(X_i)_{i\in[n]} \underset{iid}{\sim} Lap(\theta, s)$, denoting $X_{r:n}$ the order statistic of rank r one has for all $k \in \mathbb{N}$,*

$$\mathbb{E}\left[(X_{r:n}-\theta)^k\right] = s^k \frac{n!\Gamma(k+1)}{(r-1)!(n-r)!} \underbrace{\left((-1)^k \sum_{j=0}^{n-r} a_{j,r,k} + \sum_{j=0}^{r-1} b_{j,r,k}\right)}_{\alpha(n,k)} \tag{E.17}$$

Here is the proof of Theorem E.3.5.

Proof. Let $\tau > 0$ and $p > 1$, according to the weight-release mechanism, all the randomized edge weights $w'(e)$ with $e \in E(\mathcal{T}')$ are sampled from independents Laplace distributions $Lap(\frac{w(e)+\tau}{p}, \frac{s}{p})$. Given some cluster C_i^*, and $j \neq i$ s.t $e^{(ij)} \in Cut_{\mathcal{G}}(\mathcal{T})$, $H_{\mathcal{T}_{|C_i^*}}(e^{(ij)})$ is verified. Finding the probability that $H_{\mathcal{T}'_{|C_i^*}}(e^{(ij)})$ is verified is equivalent to find the probability $\mathbb{P}\left[\frac{\underset{e\in E(C_i^*)}{(\max X_e)^2}}{\underset{e\in E(C_i^*)}{\min X_e}} < X^{out}\right]$ with $X_e \underset{indep}{\sim} Lap(\frac{w(e)+\tau}{p}, \frac{s}{p})$ and $X^{out} \sim Lap(\frac{w(e^{(ij)})+\tau}{p}, \frac{s}{p})$. Denoting with $Y_i \underset{iid}{\sim} Lap(\theta_{\max}, \delta)$, $Z_i \underset{iid}{\sim} Lap(\theta_{\min}, \delta)$ and $X^{out} \sim Lap(\theta_{(ij)}, \delta)$, one can lower bounded this probability by $\mathbb{P}\left[\frac{\underset{i\in[|C_i^*|-1]}{(\max Y_i)^2}}{\underset{i\in[|C_i^*|-1]}{\min Z_i}} < X^{out}\right]$. Choosing τ big enough s.t $\underset{i\in[|C_i^*|-1]}{\min Z_i} < 0$ is negligible, one has:

$$\mathbb{P}\left[\frac{\underset{i\in[|C_i^*|-1]}{(\max Y_i)^2}}{\underset{i\in[|C_i^*|-1]}{\min Z_i}} < X^{out}\right] = \mathbb{P}\left[\underbrace{\underset{i\in[|C_i^*|-1]}{(\max Y_i)^2} - \underset{i\in[|C_i^*|-1]}{\min Z_i} \times X^{out}}_{\varphi} < 0\right]. \tag{E.18}$$

Moreover since τ, $p \gg s$, one has $\mathbb{E}(\varphi) \leq 0$. Therefore,

$$\mathbb{P}[\varphi < 0] = \mathbb{P}\left[\varphi - \mathbb{E}(\varphi) < \underbrace{-\mathbb{E}(\varphi)}_{\geq 0}\right] \tag{E.19}$$

$$= 1 - \mathbb{P}[\varphi - \mathbb{E}(\varphi) > -\mathbb{E}(\varphi)] \tag{E.20}$$

Using the one-sided Chebytchev inequality, one gets:

$$\geq 1 - \frac{\mathbb{V}(\varphi)}{\mathbb{V}(\varphi) + \mathbb{E}(\varphi)^2} = 1 - \frac{\mathbb{V}(\varphi)}{\mathbb{E}(\varphi^2)} \tag{E.21}$$

Moreover, by giving an analytic form to $\mathbb{E}(\varphi)$ and $\mathbb{V}(\varphi)$, and using Theorem E.3.6 one gets the more precise result:

$$1 - \frac{\mathbb{V}(\varphi)}{\mathbb{E}(\varphi^2)} = 1 - \frac{\Lambda_1 + (\theta_{(ij)}^2 + \delta)\Lambda_2 - (\Lambda_3^2 + \theta_{(ij)}^2\Lambda_4^2)}{\Lambda_1 + (\theta_{(ij)}^2 + \delta)\Lambda_2 + 2\Lambda_3\Lambda_4} \tag{E.22}$$

with the following notations:

- $\delta = \frac{s}{p}$, $\theta_{\min} = \frac{\min_{e \in E(\mathcal{T})} w(e) + \tau}{p}$
- $\theta_{\max} = \frac{\max_{e \in E(\mathcal{T})} w(e) + \tau}{p}$, $\theta_{(ij)} = \frac{w(e^{(ij)}) + \tau}{p}$
- $\Lambda_1 = 24\delta^4(|V|-1)\alpha(|V|-1,4) + 12\theta_{\max}\delta^3(|V|-1)\alpha(|V|-1,3) + 12\theta_{\max}^2\delta^2(|V|-1)\alpha(|V|-1,2) + 4\theta_{\max}^3\delta(|V|-1)\alpha(|V|-1,1) + \theta_{\max}^4$
- $\Lambda_2 = 2\delta^2(|V|-1)\alpha(1,2) + 2\theta_{\min}\delta(|V|-1)\alpha(1,1) + \theta_{\min}^2$
- $\Lambda_3 = 2\delta^2(|V|-1)\alpha(|V|-1,2) + 2\theta_{\max}\delta(|V|-1)\alpha(|V|-1,1) + \theta_{\max}^2$
- $\Lambda_4 = \delta(|V|-1)\alpha(1,1) + \theta_{\min}$

□

E.4 Experiments

So far we have exhibited the trade-off between clustering accuracy and privacy. We experimentally illustrate it now with some qualitative results. Let us discuss hereafter the quantitative performances of our algorithm.

E.4.1 Synthetic datasets

We have performed experiments on two classical synthetic graph datasets for clustering with non-convex shapes: two concentric circles and two moons, both in their noisy versions. For the sake of readability and for visualization purposes, both graph datasets are embedded into a two dimensional Euclidean space. Each dataset contains $N = 100$ data nodes that are represented by a point of two coordinates. Both graphs have been built with respect to the strong homogeneity condition: edge weights within clusters are between $w_{min} = 0.1$ and $w_{max} = 0.3$ while edges between clusters have a weight strictly above $w_{max}^2/w_{min} = 0.9$. In practice, the complete graph has trimmed from its irrelevant edges (i.e. not respecting the strong homogeneity condition). Hence, those graphs are not necessarily Euclidean since close nodes in the visual representation may not be connected in the graph. Finally, weights are normalized between 0 and 1.

Figures E.3 and E.4 (best viewed in color) show for each dataset (a) the original homogeneous graph $\mathcal{G}$ built by respecting the homogeneity condition, (b) the clustering partition[3] of DBMSTClu with the used underlying MST, the clustering partitions for PTClust with $\mu = 0.1$ obtained respectively with different privacy degrees[4] : (c) $\epsilon = 0.5$, (d) $\epsilon = 0.7$ and (e) $\epsilon = 1.0$. The utility function $u_{\mathcal{G}}$ corresponds to the graph weight. Each experiment is carried out independently and the tree topology obtained by PAMST will eventually be different. This explains why the edge between clusters may not be the same when the experiment is repeated with a different level of privacy. However, this will marginally affect the overall quality of the clustering.

As expected, DBMSTClu recovers automatically the right partition and the results are shown here for comparison with PTClust. For PTClust, the true MST is replaced with a private approximate MST obtained for suitable τ and p ensuring final weights between 0 and 1.

[3] For the sake of clarity, the edges in those Figures are represented based on the original weights and not on the privately released weights.

[4] Note that, although the range of ϵ is in $\mathbb{R}_+^\star$, it is usually chosen in practice in $(0, 1]$ [Dwork and Roth, 2013, Chap 1&2].

When the privacy degree is moderate ($\epsilon \in \{1.0, 0.7\}$), it appears that the clustering result is slightly affected. More precisely, in Figures E.3c and E.3d the two main clusters are recovered while one point is isolated as a singleton. This is due to the randomization involved in determining the edge weights for the topology returned by PAMST. In Figure E.4c, the clustering is identical to the one from DBMSTClu in Figure E.4b. In Figure E.3d, the clustering is very similar to the DBMSTClu one, with the exception of an isolated singleton. However, as expected from our theoretical results, when ϵ is decreasing, the clustering quality deteriorates, as DBMSTClu is sensitive to severe changes in the MST (cf. Figure E.3e, E.4e).

E.4.2 NYC "Taxi & Limousine Commission Trip Record" dataset

Both in private and non-private settings, another successful experiment has been conducted on the real NYC "Taxi & Limousine Commission Trip Record"[5] dataset taking the yellow and green taxis.

Experimental setup

From the dataset is built a graph, taking locations for vertices, trips for edges, and setting the weights by the number of trips between the locations. Some preprocessing is made on the graph. Edges with strictly less than 140 trips are removed in order to sparsify the graph. Self-loops are removed since they are not supported by the clustering algorithm. Two points belonging to the same airport are also merged because the goal is to distinguish airport places from Manhattan and it was considered as noise. Finally, only the biggest connected component is kept since it is more relevant to perform the clustering on it. As a result, the considered connected graph contains $N = 162$ nodes and $m = 236$ edges. For our clustering algorithm, the weights on the edges should represent a dissimilarity between two vertices. So the following softmax-like transformation taken from the R package[6], representing a dissimilarity function, is made on the weights of the graph:

$$\forall i \in [m],\ w_i \leftarrow f(w_i - w_{min}) \in (0, 1] \tag{E.23}$$

where $w_{min} = 140$ is the minimum weight in the graph and f is defined such that for all weight w:

$$f(w) = \frac{1}{1 + \exp\left(\frac{w-\mu}{\sigma \times \pi/2}\right)} \tag{E.24}$$

where μ is the average weight of the edges before, namely the average number of trips on the edges between the vertices and σ the standard deviation.

In this framework, the graph topology is public, and the private information is carried by the weights. In fact, what an individual would want to be kept private is rather if he/she participated or not to a trip (thus keeping his/her location/moves private).

[5] http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.
[6] https://www.rdocumentation.org/packages/DMwR/versions/0.4.1/topics/SoftMax

Results

Figure E.5 shows an exact MST that has been given as input to DBMSTClu algorithm. The results of the latter are demonstrated in Figure E.6. Finally, Figures E.7a, E.7b and E.7c exhibit the visualization results for PTClust algorithm with $\epsilon \in \{1.0, 0.7, 0.5\}$.

From only the number of trips made by the yellow and green taxis between two locations (but not the GPS coordinates), the private and non-private clustering algorithms (respectively PTClust and DBMSTClu) separate Manhattan's island from the two neighboring airports. Even if the softmax-like function helps a lot, it is really important to emphasize that geographical information have never been used to obtain these clustering partitions, which is an unusual, but meaningful way of considering this kind of datasets.

E.5 Conclusion

In this Appendix, we introduced PTClust, a novel graph clustering algorithm able to recover arbitrarily-shaped clusters while preserving differential privacy on the weights of the graph. It is based on the release of a private approximate Minimum Spanning Tree (MST) of the graph of the dataset, by performing suitable cuts to reveal the clusters.

To the best of our knowledge, this is the first differential private graph-based clustering algorithm adapted to non-convex clusters.

The theoretical analysis exhibited a trade-off between the degree of privacy and the accuracy of the clustering result. This work suits to applications where privacy is a critical issue and it could pave the way to metagenomics and genes classification using individual gene maps while protecting patient privacy. Future work will be devoted to deeply investigate these applications.

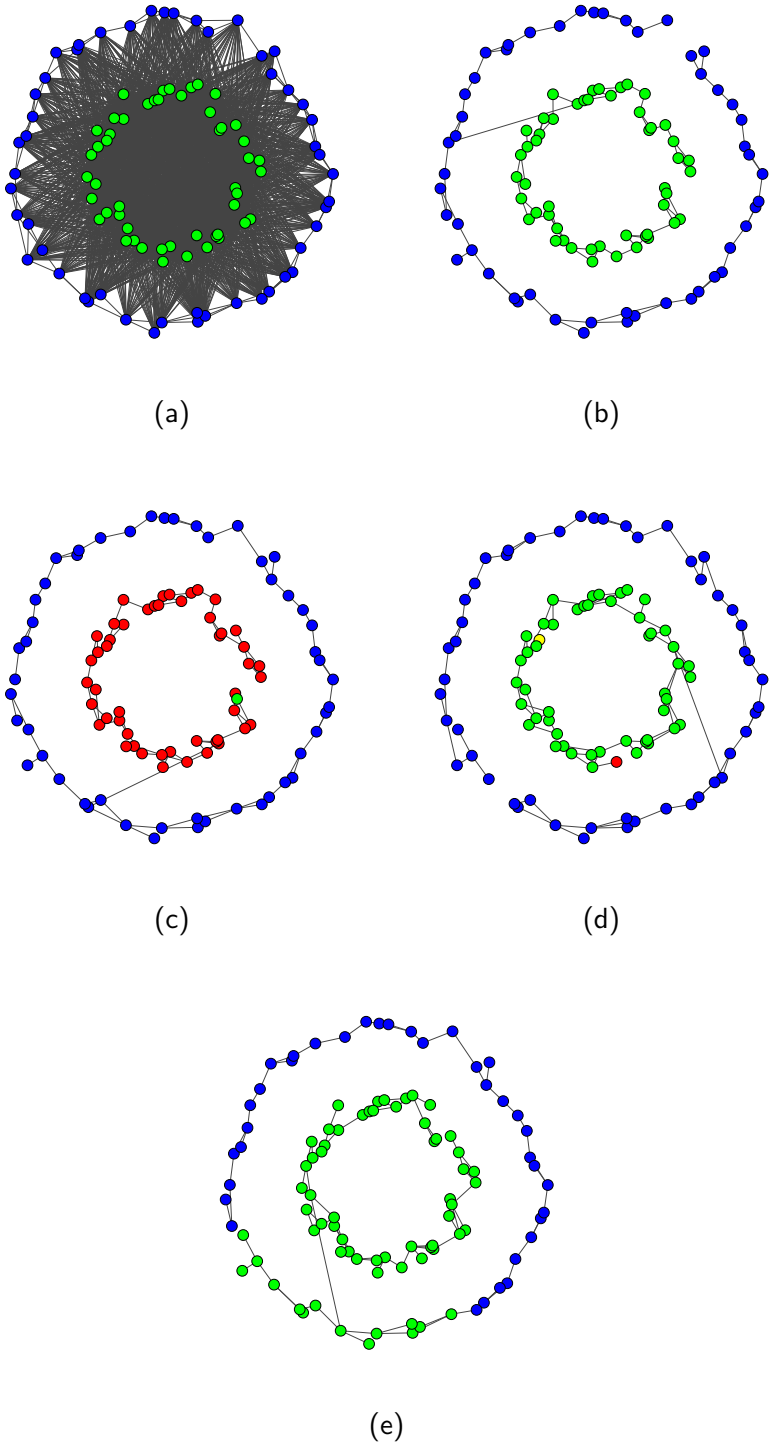


FIGURE E.3: Circles experiments for $N = 100$. Figure E.3a represents the homogeneous graph. Figure E.3b shows the results of the DBMSTClu algorithm. Remaining Figures illustrate results of PT-Clust for parameters $w_{min} = 0.1$, $w_{max} = 0.3$, $\mu = 0.1$ for all and respectively $\epsilon = 1.0$, $\epsilon = 0.7$ and $\epsilon = 0.5$ for Figures E.3c, E.3d and E.3e.

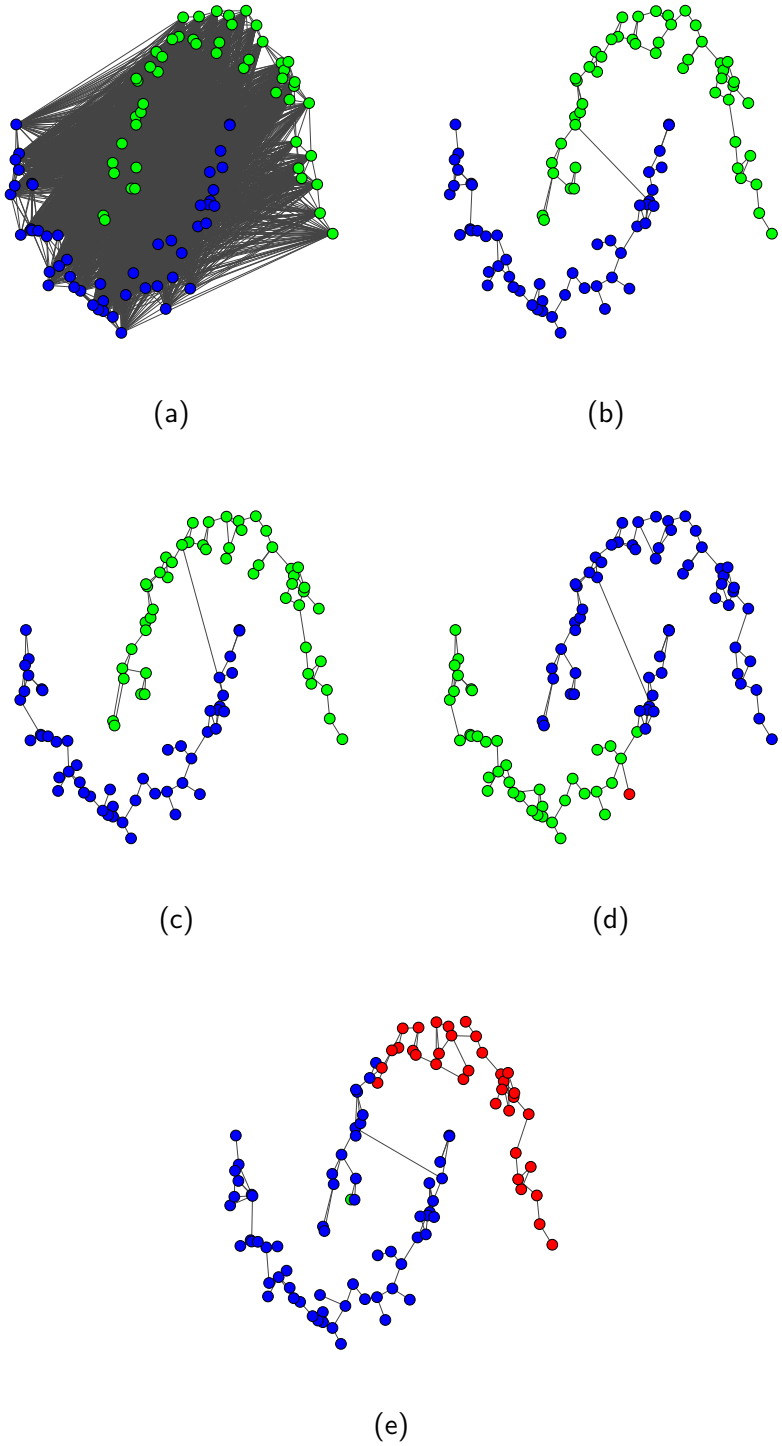


FIGURE E.4: Moons experiments for $N = 100$. Figure E.4a represents the homogeneous graph. Figure E.4b shows the results of the DBMSTClu algorithm. Remaining Figures illustrate results of PT-Clust for parameters $w_{min} = 0.1$, $w_{max} = 0.3$, $\mu = 0.1$ for all and respectively $\epsilon = 1.0$, $\epsilon = 0.7$ and $\epsilon = 0.5$ for Figures E.4c, E.4d and E.4e.

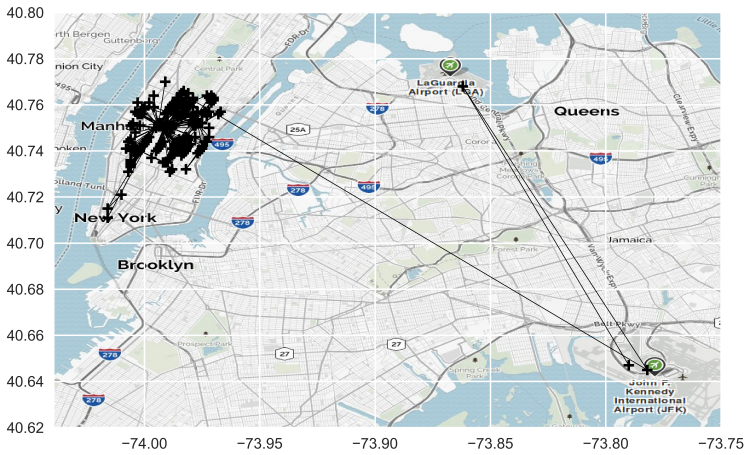


FIGURE E.5: An exact MST before DBMSTClu cuts on the NYC dataset.

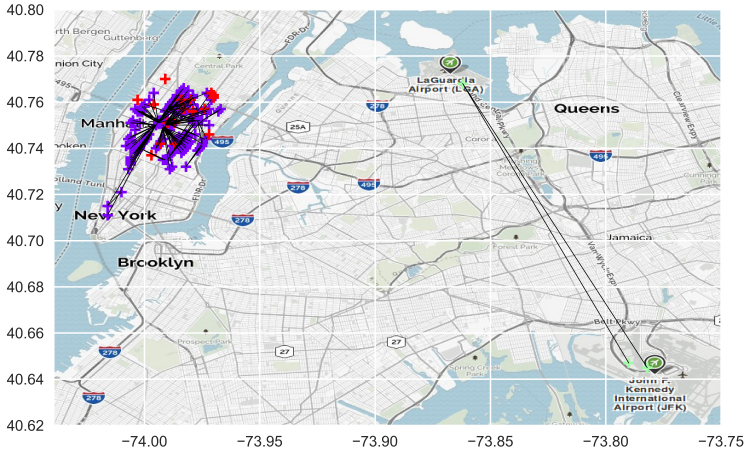
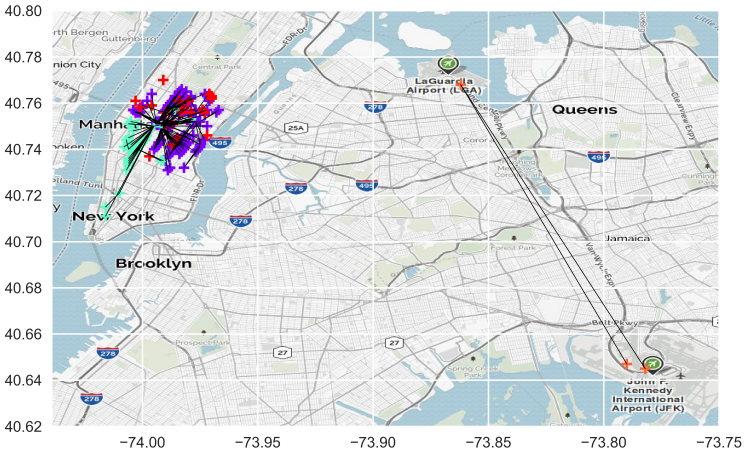
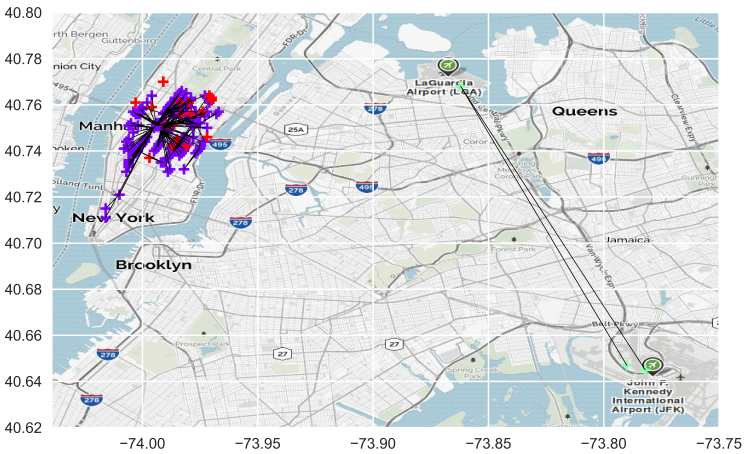


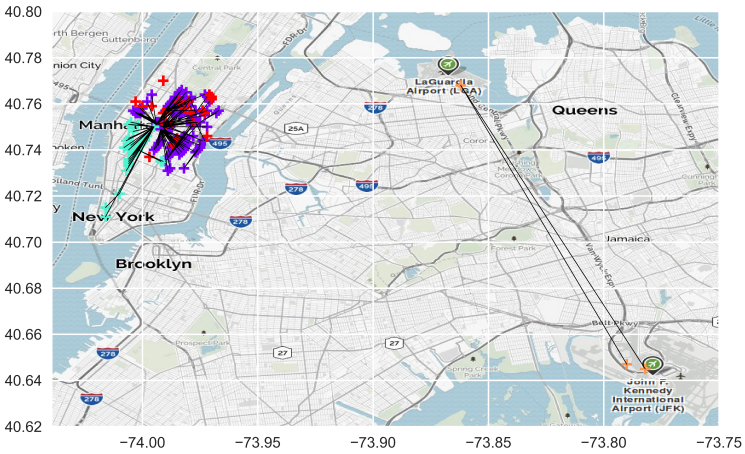
FIGURE E.6: DBMSTClu results on the NYC dataset.



(a) PTClust, $\epsilon = 1.0$



(b) PTClust, $\epsilon = 0.7$



(c) PTClust, $\epsilon = 0.5$

FIGURE E.7: NYC taxis experiments, PTClust results.

Appendix F

Publications

In this Appendix, we enumerate publications made during this PhD thesis.

The joint work described in Chapter 3 and Appendix C has been published at the Artificial Intelligence and Statistics (AISTATS) conference 2017:

- Mariusz Bojarski, Anna Choromanska, Krzysztof Choromanski, Francois Fagan, Cédric Gouy-Pailler, Anne Morvan, Nouri Sakr, Tamás Sarlós, Jamal Atif, *Structured adaptive and random spinners for fast machine learning computations*, Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS'17), 54, pp.1020-1029, Fort Lauderdale, FL, USA, 20-22 Apr.

The work explained in Chapter 4 and Appendix D has been published at the International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2018:

- Anne Morvan, Antoine Souloumiac, Cédric Gouy-Pailler, Jamal Atif, *Streaming Binary Sketching based on Subspace Tracking and Diagonal Uniformization*, Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018), Calgary, Alberta, Canada, 15-20 Apr.

The contribution from Chapter 5 has been published at SIAM Data Mining conference (SDM) 2018. Extended work with further theoretical guarantees and application to privacy-preserving clustering (see Appendix E) has been published at the International conference on Uncertainty in Artificial Intelligence (UAI) 2018.

- Anne Morvan, Krzysztof Choromanski, Cédric Gouy-Pailler, Jamal Atif, *Graph sketching-based Space-efficient Data Clustering*, Proceedings of the SIAM International Conference on DATA MINING (SDM'18), pp.10-18, San Diego, CA, USA, 3-4 May.
- Rafaël Pinot, Anne Morvan, Florian Yger, Cédric Gouy-Pailler, Jamal Atif, *Graph-based Clustering under Differential Privacy*, Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI 2018), Monterey, CA, USA, 6-10 Aug.

All these publications can be found on the personal webpage:

https://annemorvan.github.io/

with some code available on:

https://github.com/annemorvan

Appendix G

Résumé de la thèse en français

Contents

This Appendix written in French gives in few pages an overview of the motivation of this thesis and its main contributions.

Note au lecteur: Cette annexe présente un résumé en français de cette thèse qui a été rédigée en anglais. La motivation du sujet, les grandes lignes de chacune des contributions sont exposées ainsi que quelques résultats expérimentaux à titre d'illustration. Nous invitons cependant le lecteur à lire le corps de ce manuscrit en anglais pour une vision plus détaillée:

- des modèles utilisés,
- des concepts invoqués,
- des protocoles d'expérience,
- des résultats expérimentaux,
- et de leur analyse.

G.1 Introduction

G.1.1 Contexte et motivation

Ère du *Big Data* et le besoin d'un traitement adapté

Aujourd'hui, tous les composants électroniques sont conçus pour produire un volume gigantesque de données brutes dans l'espoir de les transformer en de l'information de valeur. Cette dernière permet de prendre des décisions stratégiques à travers des algorithmes d'apprentissage artificiel. Un nombre important d'applications émergeant aujourd'hui produisent et/ou utilisent des données en quantité massive. Par exemple, on notera:

- les observations des capteurs de l'Internet des Objets,
- les cookies liés au flux de clics des web-utilisateurs,
- les larges ensembles de pages web,
- les données multimédia,
- les transactions financières,
- etc.

Ces données amassées représentent une source de revenu colossale pour les entreprises lorsqu'elles sont analysées. Ainsi, d'importantes quantités de données doivent être traitées. Dans la pratique, à cause de l'énorme volume de données à gérer, des difficultés apparaissent et font échouer les méthodes d'analyse classiques. Ces défis sont les suivants:

1. Les bases de données comportent un grand nombre d'instances, encore appelées enregistrements.

   On distinguera deux cas:

   - D'un côté, les données peuvent ne pas tenir entièrement en mémoire.
   - De l'autre, même s'il est possible de centraliser ces données, leur nombre est tel que les algorithmes classiques ne peuvent plus y être appliqués. Par exemple, un algorithme de complexité temporelle $O(N^2)$, pour N le

nombre d'instances dans la base de données, est impraticable à partir de quelques milliers voire centaines de milliers de points.

2. Ces données sont potentiellement en *grande dimension*.

   Cela signifie que chaque enregistrement est décrit par un grand ensemble d'attributs. Cela peut être problématique à plus d'un titre. En plus d'augmenter le coût en complexité, les algorithmes d'analyse peuvent également souffrir de ce que l'on appelle la *malédiction de la dimension*. Cette expression traduit le fait qu'il est très difficile d'apprendre dans un espace de grande dimension puisque un plus grand nombre de données est nécessaire pour éviter le sur-apprentissage.

   Plus important, dans un espace de grande dimension, tous les points sont éloignés les uns des autres ce qui rend plus ardue la tâche de trouver des similarités ou des différences entre eux.

3. Les données peuvent également être intrinsèquement sous la forme d'un *flux* (potentiellement infini).

   Ainsi, les données sont collectées au vol au fur et à mesure de leur calcul (par exemple par les capteurs de l'Internet des objets) et doivent être traitées directement. De plus, même si les données ne sont pas par nature un flux, comme tout juste expliqué, elles peuvent être stockées sur des sources distribuées dans une quantité telle qu'il est alors impossible de les charger dans une mémoire centrale. Le seul moyen de les lire peut-être est sous la forme d'un flux.

   Pour ces deux cas de figure, un algorithme en ligne, *online* ou *streaming* (on utilisera indifféremment ces termes dans la suite) est nécessaire pour procéder à l'analyse de données, soit parce que les données sont par essence sous la forme d'un flux, ou parce que le seul moyen de respecter les contraintes d'espace est de les traiter comme un flux.

   La particularité des algorithmes online est de ne pouvoir lire qu'une seule donnée à la fois. Une décision doit être prise sans avoir accès aux données suivantes, en ayant à disposition qu'un résumé compact des données précédemment vues. La taille de cette représentation compressée est souvent une fonction sous-linéaire de la taille du dataset en entrée.

4. Enfin, la vaste quantité de données disponibles rend leur étiquetage pénible.

   Ainsi, la plupart du temps, l'analyse sur de telles données doit être effectuée de manière *non supervisée*. Sans étiquette, l'information à extraire est la *distance* (ou à l'opposé la similarité) entre les individus, soit également la *structure* des données.

Problèmes d'apprentissage artificiel non surpervisé

En particulier, la *recherche des plus proches voisins* et le *clustering* sont les deux problèmes d'apprentissage artificiel *non supervisé* considérés dans cette thèse.

En effet, la *recherche des plus proches voisins* et le *clustering* sont deux outils fondamentaux d'analyse *exploratoire* des données. Ils ont en commun la capacité à segmenter des données en groupes avec des propriétés similaires en appliquant une transformation sur les données qui *préserve approximativement la distance ou la structure* de celles-ci. De plus, les algorithmes de recherche des plus proches voisins et

de clustering consituent de fait des règles de classification en apprentissage des plus évidentes. En pratique, ces méthodes aident à:

- identifier des clients avec un comportement similaire pour de la segmentation de marché,
- exhiber des communautés dans les réseaux sociaux,
- ou à partir du génome séquencé d'un nouvel organisme, trouver les gènes qui sont similaires à ceux déjà répertoriés dans les bases de données,

pour ne citer que quelques applications concrètes.

Par ailleurs, ces méthodes sont très utiles pour le cas de figure majoritaire où l'étiquetage des données est trop coûteux. En effet, à partir de quelques données manuellement étiquetées, ces méthodes d'apprentissage non supervisé permettent d'étendre les propriétés d'un point à tous les autres contenus dans le même cluster ou voisinage. L'étiquetage des données est ensuite facilité s'il est fait directement sur des clusters de données ou des voisinages de points pré-sélectionnés.

Représentation compacte et approximation

Pour proposer des algorithmes efficaces en temps et en espace pour la recherche des plus proches voisins et le clustering, le traitement des données est facilité en travaillant plutôt directement sur des *représentations compactes* des données, qui maintiennent un résumé du flux au fur et à mesure que les données sont collectées. Ces représentations compactes sont obtenues par des méthodes de *réduction de dimension* et/ou d'*échantillonnage*. Parmi elles, on peut citer les *sketches*, particulièrement adaptés à des données en flux. De manière générale, ces représentations compactes permettent:

- évidemment de réduire le coût spatial des algorithmes de traitement,
- mais également la complexité temporelle des opérations effectuées,

ce qui de fait augmente fortement le débit de traitement.

La compression des données a cependant un coût, celui de l'*approximation* du résultat puisque celle-ci engendre une perte d'information. Mais dans bien d'applications, une approximation est suffisante: par exemple, quand on veut connaître l'ordre de grandeur du nombre de vues pour une page parmi le corpus Wikipédia qui en contient environ 35 millions, le chiffre exact n'est pas plus utile.

> Ainsi, cette thèse se concentre sur la conception d'algorithmes efficaces en temps et en espace pour répondre à des problèmes d'apprentissage non supervisé en s'appuyant sur des représentations compactes des données qui préservent approximativement les propriétés d'intérêt comme la distance ou la structure des données.

Compromis Pour ce faire, il est important de se rappeler que construire un algorithme autour d'une bonne représentation compacte des données est un compromis entre :

- le coût spatial de la structure de données,
- le débit de traitement des données,

- et la précision du résultat de l'algorithme,

le tout étant paramétré en fonction des besoins et des contraintes de l'utilisateur.

Nous présentons dans cette thèse, après revue de l'état de l'art - à consulter directement dans le corps de ce manuscrit - trois contributions au domaine.

G.1.2 Contributions

La revue de l'état de l'art a permis d'identifier trois types d'approches qui préservent approximativement la distance et la structure des données pour l'apprentissage non supervisé:

- les méthodes *indépendantes* des données,
- ou au contraire *adaptées* aux données,
- et enfin fondées sur les *graphes*.

Les méthodes *indépendantes* des données pour la réduction de dimension sont évidemment les moins coûteuses et permettent un traitement directement en ligne. La théorie des projections aléatoires pour la préservation des distances garantie une certaine précision du résultat.

Toutefois, cette précision peut ne pas être suffisante pour certaines applications. Ainsi, la construction des représentations compactes peut être *adaptée aux données*: la transformation de réduction de dimension étant apprise, la distance entre les points est mieux préservée. Les algorithmes de cette famille sont plus gourmands en temps et en espace et plus difficiles à adapter en ligne: ils doivent dépendre des données, mais dans le cadre streaming, ne peuvent pas toutes les stocker pour apprendre les paramètres de leur modèle!

Enfin, une approche *graphe* permet non seulement de conserver les distances mais aussi la *structure* des données en représentant le dataset comme un graphe, ce graphe exprimant les dissimilarités ou les similarités entre les points sous la forme de poids sur les arêtes les reliant. La difficulté est alors de rendre ce graphe le plus parcimonieux possible pour en réduire le coût de stockage.

Ces trois familles de méthodes ont toutes leurs avantages et leurs inconvénients. Les contributions dans cette thèse reflètent cette taxonomie puisque chacune d'elles appartient à une de ces différentes catégories.

Des matrices aléatoires structurées, une approche pour de l'apprentissage à grande échelle

En ce qui concerne la recherche des plus proches voisins de manière efficace, l'approche la moins coûteuse est *indépendante* des données. Il s'agit des méthodes de type LSH pour *Locality-Sensitive Hashing* [Indyk and Motwani, 1998, Charikar, 2002, Terasawa and Tanaka, 2007, Sundaram et al., 2013, Andoni et al., 2015a]. Brièvement, elles s'appuient sur des projections aléatoires pour réduire la dimension des données et faciliter la recherche par similarité en l'effectuant directement dans le nouvel espace de redesription de plus petite dimension. Dans le cadre du Cross-polytope LSH [Terasawa and Tanaka, 2007] qui est l'approche état de l'art pour le LSH, de nombreux calculs consistent en le produit d'une matrice aléatoire avec la donnée d'entrée en grande dimension (il s'agit de la procédure de *hashing*). Pour en réduire le coût, qui devient vite prohibitif en grande dimension, Andoni et al. [2015a] proposent de remplacer cette matrice aléatoire par trois blocs de transformées de Hadamard aléatoirisés. Les gains en temps de calcul et en coût de stockage de la matrice sont

significatifs. Les résultats expérimentaux montrent par ailleurs que la qualité des fonctions de hachage reste importante par rapport à celles avec une matrice aléatoire.

Nous proposons ici pour la première fois, un nouveau cadre théorique que l'on nomme *Pivoteurs structurés* (*Structured spinners* en anglais) pour expliquer cette performance. De manière générale, ce cadre propose de compresser par structuration des matrices apprises ou aléatoires gaussiennes dans de nombreux problèmes d'apprentissage s'appuyant sur des produits entre matrices et vecteurs. Cette structuration incorpore des rotations et implique des matrices structurées classiques comme les matrices circulantes, de Toeplitz, de Hankel, de Hadamard etc. Des expériences dans de nombreux domaines d'application comme la recherche approximative des plus proches voisins, l'approximation de noyaux, l'optimisation convexe avec les sketches de Newton ou les réseaux de neurones confirment l'utilité des *Pivoteurs structurés* dont l'usage n'entraîne presque aucune perte de précision en comparaison avec leurs homologues non structurés.

Apprentissage de codes compacts binaires de flux de données massives via le hashing hypercubique pour la recherche des plus proches voisins

En deuxième lieu, pour les applications qui le nécessitent, une méthode originale a été conçue pour *apprendre* - la méthode est donc adaptée aux données - en *une seule passe* sur le flux des codes compacts binaires de données en grande dimension.

De la famille des méthodes de hachage (ou *hashing* en anglais) hypercubique [Gong et al., 2013, Kong and Li, 2012, Leng et al., 2015a], elle s'appuie sur une méthode reconnue de réduction de dimension pour préserver la distance entre les points, l'Analyse en Composantes Principales (ACP). L'état de l'art fait le constat que la qualité de la fonction de hachage est améliorée si on applique à la suite de l'ACP une rotation qui tend à uniformiser la variance.

Nous mettons donc en place une version online de ce type d'approche. Nous apportons également une preuve de l'optimalité de cette rotation en posant quelques conditions. En plus de certaines garanties théoriques, la qualité des codes binaires obtenus est ensuite mesurée dans le cadre de la recherche approximative des plus proches voisins.

Clustering de données massives à partir d'un arbre couvrant minimum

Enfin pour cette dernière contribution, nous proposons un nouvel algorithme de clustering nommé DBMSTClu d'approche de type *graphe* pour préserver la structure des données. Un ensemble de données peut en effet être représenté par un graphe de dissimilarité mais pour les applications à grande échelle, ce dernier est trop large pour tenir entièrement en mémoire. En conséquence, l'information contenue dans le dataset est compressée sous la forme d'un arbre couvrant minimum du graphe de dissimilarité. En s'appuyant uniquement sur cet arbre, l'algorithme permet de respecter les contraintes de complexité en temps et en espace linéaire en le nombre de données. DBMSTClu est une solution sans paramètre, efficace en terme de coût de stockage et temps computationnel, capable de détecter automatiquement des clusters de forme arbitraire dans des données massives. Pour ce faire, DBMSTClu coupe des arêtes selon un critère à maximiser.

Pour obtenir efficacement un arbre couvrant minimum, nous proposons d'utiliser une technique récente de sketching de *graphe* [Ahn et al., 2012a]. Une application au clustering préservant la vie privée différentielle est également démontrée.

Des questions restées ouvertes et des perspectives de recherche future concluent enfin ce travail.

G.2 Des matrices aléatoires structurées, une approche pour de l'apprentissage à grande échelle

G.2.1 Principe

Cette première contribution propose une technique de structuration pour accélérer les schémas fondés sur la méthode *indépendante* des données LSH pour la recherche des plus proches voisins.

Les méthodes de type LSH [Indyk and Motwani, 1998, Charikar, 2002, Terasawa and Tanaka, 2007, Sundaram et al., 2013, Andoni et al., 2015a] consistent à réduire la dimension des données de grande dimension en entrée avec des projections aléatoires. C'est ensuite sur ces vecteurs de redescription plus petits que la recherche par similarité est effectuée, et ceci beaucoup plus rapidement. Quand les dimensions de la matrice de projection et des vecteurs sont élevées, le calcul de ce produit matrice-vecteur peut être très coûteux.

Pour remédier au problème, il est suggéré de remplacer les matrices aléatoires par des structurées choisies parmi la famille du modèle présenté des *Pivoteurs structurés* (*Structured spinners*). Les matrices de la famille des *Pivoteurs structurés* sont construites comme des produits de trois blocs de matrices structurées qui incorporent des rotations et suivent certaines conditions. Les matrices structurées pour constituer ces blocs peuvent être par exemple les matrices circulantes, de Toeplitz, de Hankel, de Hadamard etc.

Utiliser ces matrices conduit à la fois à la réduction du coût spatial et l'accélération du calcul du produit matrice-vecteur grâce à des transformées de Fourier ou de Hadamard rapides qui sont appliquées à la place du produit matrice-vecteur classique. Par exemple, dans le cas du matrice carrée, on passe d'un coût temporel de $O(n^2)$ à $O(n \log n)$ pour n la dimension de la matrice et du vecteur.

Des garanties théoriques sont fournies pour caractériser la capacité du modèle structuré en référence à son homologue non structuré.

Ces dernières s'appuient essentiellement sur des théorèmes de concentration comme le Théorème Central Limite de type Berry-Esseen [Bentkus, 2003] ou l'inégalité d'Azuma [Azuma, 1967]. Elles démontrent qu'il n'y a presque aucune perte de précision dans les applications considérées, ce qui est confirmé par la partie expérimentale.

En particulier, ces garanties théoriques sont les premières à expliquer la performance de la matrice structurée $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ proposée par [Andoni et al., 2015a] pour la variante état de l'art de LSH, le Cross-polytope LSH [Terasawa and Tanaka, 2007].

Comme sous-produit, la technique proposée est généralisable à beaucoup d'applications d'apprentissage s'appuyant sur des produits matrice-vecteur où les paramètres de la matrice sont aléatoires[1] ou appris (voir Annexe E). Quelques exemples incluent l'approximation de noyaux, l'optimisation convexe avec les Newton sketches, la quantification de vecteurs ou les réseaux de neurones.

[1] Typiquement, chaque coefficient de la matrice est tiré aléatoirement d'une distribution gaussienne.

G.2.2 Expérience

Dans cette expérience, on mesure la qualité de la fonction de hachage dans le Cross-polytope LSH [Terasawa and Tanaka, 2007] en comparant les probabilités de *collision*. Il s'agit de vérifier que:

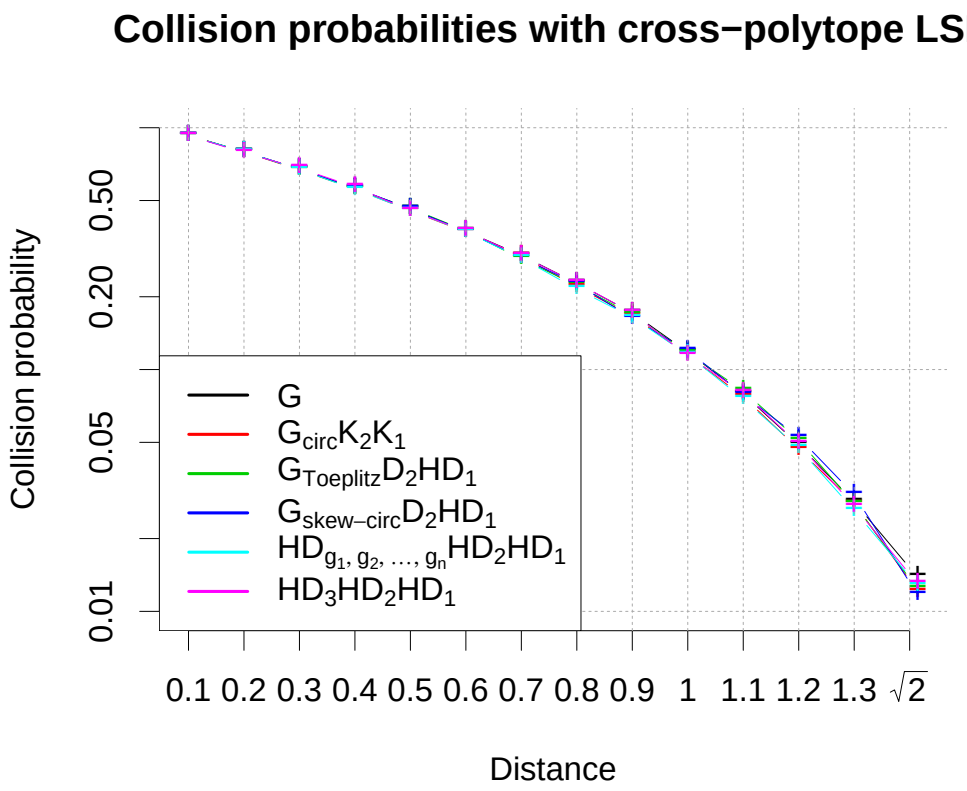
- les données similaires ont une forte probabilité d'avoir le même *hash* (c'est le résultat de la fonction de hachage) et une faible probabilité d'en avoir un différent,
- tandis que pour des données très différentes, la probabilité d'avoir un hash identique ou proche est être faible et la probabilité d'avoir des hashes différents, élevée.

Les données initiales sont 20000 points de dimension 256 aléatoirement tirés de manière à avoir approximativement autant de données par intervalle de distance entre 0 et $\sqrt{2}$ car les vecteurs ont été normalisés. La Figure G.1 reporte les résultats (moyennés sur 100 lancers) pour une seule fonction de hachage en comparant une matrice aléatoire gaussienne $\mathbf{G}$ de taille 256×64 et cinq autres types de matrices de la famille des *Pivoteurs structurés* (avec un ordre descendant du nombre de paramètres):

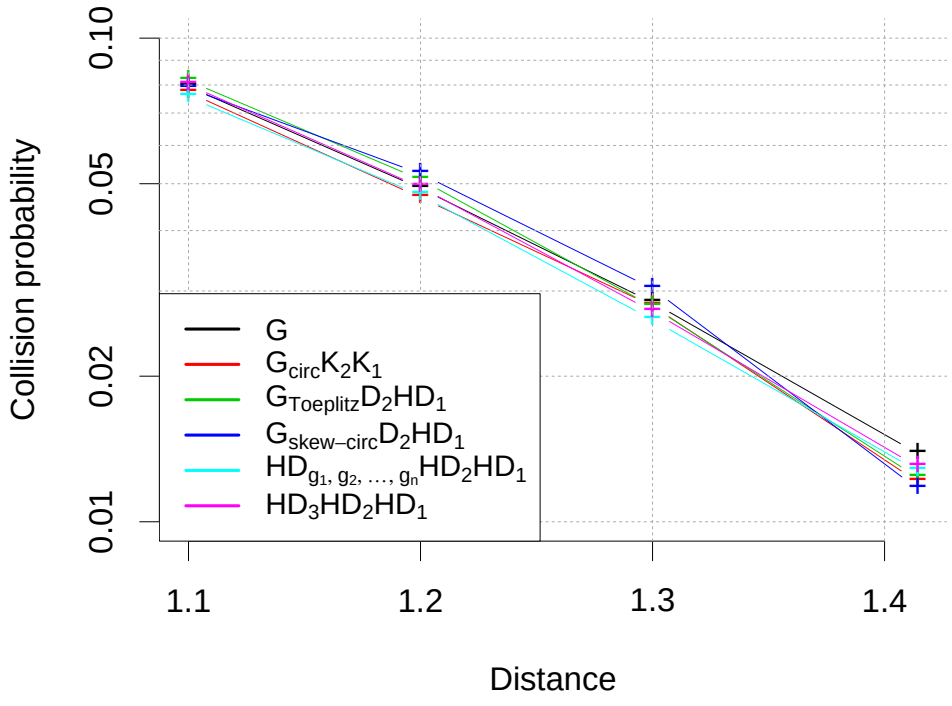
- $\mathbf{G}_{circ}\mathbf{K}_2\mathbf{K}_1$,
- $\mathbf{G}_{Toeplitz}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$,
- $\mathbf{G}_{skew-circ}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$,
- $\mathbf{H}\mathbf{D}_{g_1,\ldots,g_n}\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$,
- et $\mathbf{H}\mathbf{D}_3\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$,

où $\mathbf{K}_i$, $\mathbf{G}_{Toeplitz}$, et $\mathbf{G}_{skew-circ}$ sont respectivement une matrice de Kronecker avec des entrées discrètes, une matrice de Toeplitz gaussienne et une matrice *skew-circulant* gaussienne. On remarque que toutes ces matrices ont bien une grande probabilité de collision pour des distances faibles et une faible probabilité pour des distances élevées. Toutes les matrices considérées donnent des résultats presque identiques. Ceci confirme la théorie que l'usage des *Pivoteurs structurés* ne conduit pas à une perte significative dans la précision du résultat.

D'autres expériences ont été conduites pour les différentes applications nommées plus haut et leurs résultats sont présentés en Annexe C.



(a)



(b)

FIGURE G.1: Cross-polytope LSH - (a) Probabilités de collision pour des distances comprises entre 0 et $\sqrt{2}$. (b) Un zoom sur les distances les plus élevées permet de distinguer les courbes qui sont presque superposées.

G.3 Apprentissage de codes compacts binaires de flux de données massives via le hashing hypercubique pour la recherche des plus proches voisins

G.3.1 Principe

La seconde contribution *adaptée aux données* permet d'apprendre des représentations compactes binaires *en ligne* préservant la similarité pour des données massives en grande dimension.

Le but est de pouvoir effectuer une recherche par similarité efficace. A partir d'une longueur de code désirée c et des points en grande dimension, nous proposons un algorithme streaming qui fournit un code binaire de c bits préservant les distances entre les points dans l'espace d'origine de grande dimension.

La méthode s'inspire de la famille de hachage hypercubique dont font partie par exemple les méthodes offline ITerative Quantization (ITQ) [Gong et al., 2013], Isotropic Hashing (IsoHash) [Kong and Li, 2012] et la méthode Online Sketching Hashing (OSH) [Leng et al., 2015a]. Elle s'appuie sur une projection des données sur les c premières composantes principales de la matrice de covariance. Sur les données projetées est ensuite appliquée une rotation $\mathbf{R} \in \mathbb{R}^{c\times c}$ de manière à équilibrer la variance sur les différentes directions. Pour obtenir un code binaire, la fonction signe est appliquée coefficient par coefficient, tel que: un coefficient négatif est mis à -1 et un coefficient positif ou nul à $+1$.

Les c composantes principales sont mises à jour en ligne au moyen de l'algorithme existant OPAST [Abed-Meraim et al., 2000]. Puis, une rotation $\mathbf{R}$ est calculée comme le produit de $c-1$ rotations de Givens $\mathbf{G}_r(i_r, j_r, \theta_r)$ pour $r \in [c-1]$ et i_r, j_r, θ_r étant les paramètres déterminant une matrice de Givens (cf. Definition 4.4.1, p. 81). Soit $\mathbf{\Sigma_Y}$ la matrice de covariance des données après la rotation par $\mathbf{R}$, chacune des rotations de Givens $\mathbf{G}_r(i_r, j_r, \theta_r)$ permet de forcer la valeur du coefficient diagonal i_r de $\mathbf{\Sigma_Y}$ à une valeur désirée τ [Golub and van der Vorst, 2000] en préservant la trace. Le lecteur est invité à se référer au Chapitre 4 pour une description plus détaillée du modèle.

IsoHash [Kong and Li, 2012] propose également une manière de définir une rotation uniformisant la variance mais c'est seulement dans ce travail qu'une première tentative de justification de l'optimalité de telles rotations est présentée.

L'algorithme calculant la rotation $\mathbf{R}$, appelé UnifDiag, possède des garanties de convergence (contrairement à ITQ) et constitue de similaires ou meilleures performances que l'état de l'art du hachage hypercubique concernant la complexité temporelle et la qualité des codes courts binaires.

G.3.2 Expériences

La qualité des codes binaires compacts obtenus est mesurée dans le cadre de la recherche des plus proches voisins. Les plus proches voisins, obtenus en comparant les codes binaires de taille c, sont comparés avec les vrais voisins, issus de la comparaison des distances euclidiennes entre les points de l'espace de dimension initiale d.

Le *Mean Average Precision* (MAP) est utilisé pour mesurer la précision du résultat en tenant compte du nombre de vrais voisins correctement retournés et de leur rang. Dans le contexte streaming, l'algorithme UnifDiag est comparé avec différentes méthodes de l'état de l'art en ligne (se référer à la Section 2.3 et au Chapitre 4 pour

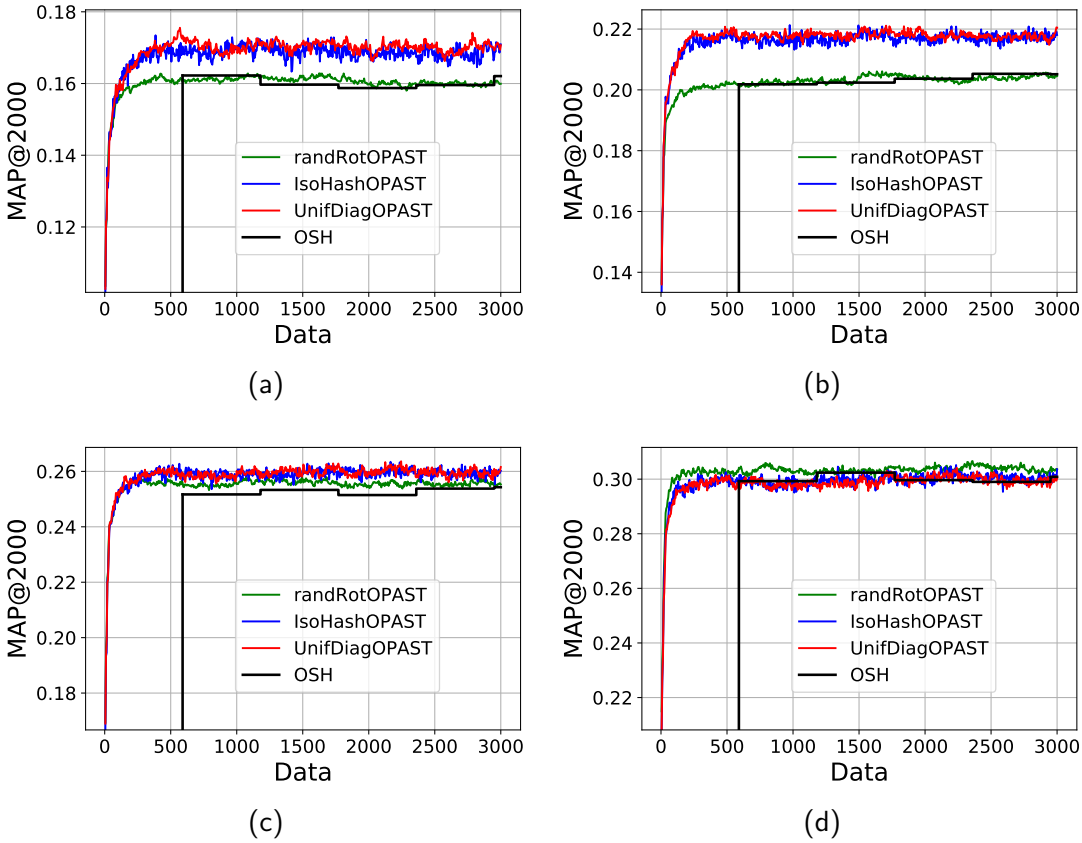


FIGURE G.2: MAP@2000 dans le cadre streaming pour différentes tailles de code et le dataset CIFAR: (a) $c = 8$, (b) $c = 16$, (c) $c = 32$, (d) $c = 64$.

une description du fonctionnement de ces méthodes). Les tests sont conduits sur deux bases de données: CIFAR-10[2] et GIST1M[3].

- CIFAR-10 (CIFAR) contient 60000 images colorées de taille 32×32, distribuées de manière égale en 10 classes. Des descripteurs GIST de dimension 960 sont extraits de ces données.
- GIST1M (GIST) contient 60000 descripteurs GIST de dimension 960.

1000 requêtes sont tirées aléatoirement et les 59000 points restant sont utilisés comme base d'apprentissage. La vérité terrain déterminant les voisins des non-voisins pour chacun des points-requête dépend d'un seuil calculé choisi arbitrairement. Après application de la fonction de hashing binaire sur les points de l'ensemble de requête et d'apprentissage, le MAP à 2000 est calculé sur les 2000 voisins retournés à partir du tri des distances de Hamming évaluées sur les codes binaires.

Les Figures G.2 et G.3 reportent les résultats respectivement pour les datasets CIFAR et GIST pour différentes longueurs de code. On remarque que pour des codes courts, notre méthode fonctionne aussi bien - sinon mieux - que l'état de l'art pour des méthodes de hashing binaire en streaming, tout en étant plus simple conceptuellement et avec une complexité temporelle plus faible.

[2] http://www.cs.toronto.edu/~kriz/cifar.html
[3] http://corpus-texmex.irisa.fr/

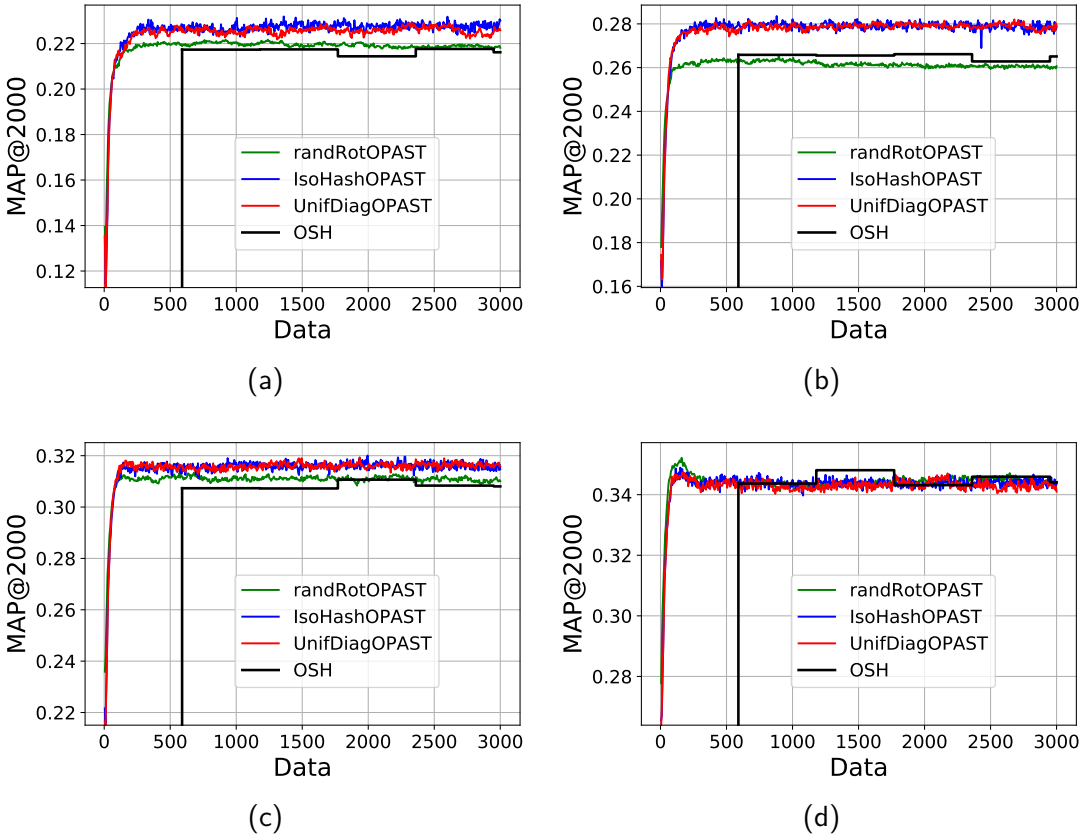


FIGURE G.3: MAP@2000 dans le cadre streaming pour différentes tailles de code et le dataset GIST: (a) $c = 8$, (b) $c = 16$, (c) $c = 32$, (d) $c = 64$.

G.4 Clustering de données massives à partir d'un arbre couvrant minimum

G.4.1 Principe

DBMSTClu est un nouvel algorithme sans paramètre qui retrouve automatiquement des clusters de données de forme arbitraire à partir d'un arbre couvrant minimum du graphe de dissimilarité des données.

Optimalité des approches à base d'arbre couvrant minimum Pour justifier cette approche, nous démontrons tout d'abord l'optimalité en général des méthodes de clustering s'appuyant sur un arbre couvrant minimum, en définissant soigneusement la notion de *cluster*.

Critère de coupe A partir d'un arbre couvrant minimum, l'algorithme DBMSTClu coupe des arêtes bien choisies faisant apparaître des composantes connexes qui seront considérées ensuite comme des clusters. Le critère de coupe, appelé DBCVI pour *Density-Based Clustering Validation Index* s'appuie sur le concept de *séparation* et de *dispersion* des clusters.

La séparation est le poids minimum des arêtes incidentes au cluster qui ont été précédemment coupées.

La dispersion est le poids maximum des arêtes internes au cluster.

Intuitivement, la séparation mesure à quel point deux clusters sont éloignés et la dispersion, le niveau de densité d'un cluster. Par exemple, une dispersion faible traduit un cluster dense avec peu de "vide" entre les points. Naturellement, le but de cet algorithme de clustering est de maximiser la séparation entre les clusters, tout en pénalisant les clusters à haute dispersion.

Etapes de l'algorithme Pour maximiser le critère DBCVI construit à partir de la séparation et de la dispersion pour chaque cluster, l'algorithme conçu est glouton. Cela signifie qu'à chaque étape, le DBCVI est calculé pour chacune des coupes possibles. Si l'algorithme trouve une arête qui, une fois coupée, maximise le critère de manière à augmenter (ou égaliser) la valeur de ce dernier par rapport à la partition précédente, l'arête est coupée. On continue ce processus jusqu'à ce que plus aucune amélioration du DBCVI ne soit possible. La Figure G.4 résume cette procédure.

Des garanties théoriques sur la qualité de la partition obtenue sont apportées et les résultats expérimentaux montrent l'intérêt de la méthode.

Notes sur l'implémentation A noter qu'un effort d'implémentation a été effectué pour assurer une complexité linéaire en temps et en espace en le nombre de noeuds. En effet, une implémentation naïve de l'algorithme a un complexité quadratique. L'implémentation efficace s'appuie ici deux observations:

1. Une coupe ne modifie la séparation et la dispersion associées à d'autres arêtes que si elles étaient précédemment dans le même cluster que l'arête de coupe. Ceci permet d'économiser un certain nombre de calculs. A une nouvelle étape, il n'est donc pas nécessaire de recalculer le DBCVI associé à une arête si elle est présente dans un cluster qui n'a pas été touché à l'étape précédente par une coupe.

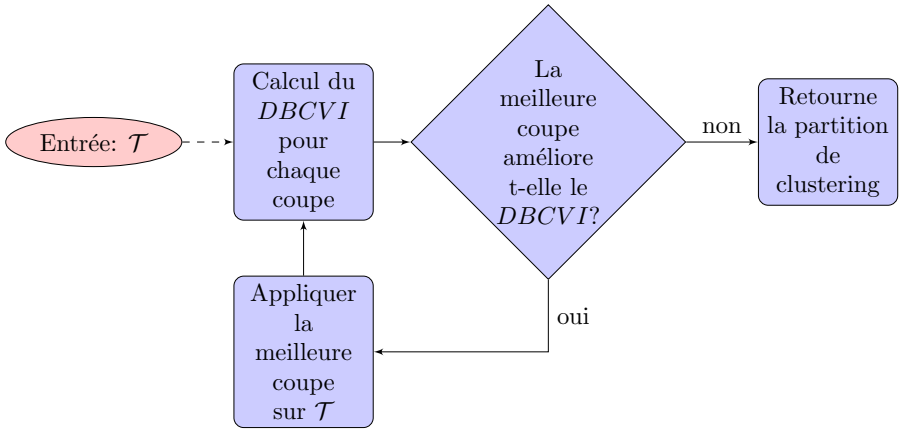


FIGURE G.4: Schéma basique de fonctionnement de l'algorithme DBMSTClu où $\mathcal{T}$ désigne un arbre couvrant minimum (approché ou non). Initialement, le DBCVI est égal à une valeur arbitraire la plus faible possible pour forcer une première coupe.

2. Lorsque l'on connaît la dispersion et la séparation associée à une arête, il est possible de déduire ces deux quantités pour des arêtes avoisinantes grâce à une relation de récursion.

La mise en place d'un parcours en profondeur d'abord double, à gauche et à droite d'une arête considérée pour une coupe, permet de ne parcourir qu'une seule fois l'arbre en entier.

Recouvrer efficacement un arbre couvrant minimum Pour répondre au besoin d'extraire efficacement un arbre couvrant minimum du graphe de dissimilarité, sans avoir à stocker en mémoire vive le graphe en entier s'il est trop large, nous proposons de traiter le graphe de dissimilarité sous la forme d'un flux d'arêtes pour en créer une représentation compacte, dite *sketch*, à partir de la méthode de Ahn et al. [2012a]. Cette dernière technique nécessite un espace mémoire en $O(N \operatorname{polylog} N)$ au lieu du $O(N^2)$ initial où N est le nombre de noeuds. Un arbre couvrant minimum *approché* peut être recouvré en $O(N \operatorname{polylog} N)$ de temps.

Application à la vie privée différentielle Enfin, nous présentons une application de cet algorithme dans le cadre du respect de la vie privée différentielle (Annexe E) où les poids des arêtes correspondent à l'information sensible à préserver.

G.4.2 Expériences

Sécurité de l'usage du sketching de graphe

Dans cette expérience, on montre qu'il n'y a pas de préjudice à la qualité de la partition de clustering résultante, en prenant comme entrée de DBMSTClu à la place d'un arbre couvrant minimum exact, une version approchée, obtenue par la technique de sketching du graphe de dissimilarité. Notre méthode est comparée avec:

- SEMST [Asano et al., 1988, Xu et al., 2002] une méthode naïve qui pour un nombre de clusters souhaité K, effectue $K-1$ coupes parmi les arêtes de poids le plus élevé dans l'arbre couvrant minimum,
- DBSCAN [Ester et al., 1996], une méthode classique de clustering recouvrant des clusters non convexes, qui n'est pas de type graphe et nécessite deux paramètres à optimiser.

Les Figures G.5, G.6 et G.7 présentent les résultats obtenus pour des données en dimension 20. Les deux premières dimensions modélisent la forme des clusters (boules, cercles ou bananes bruités) tandis que les 18 dimensions restantes sont du bruit. Les données ont été projetées sur les 2 premières dimensions par soucis de visualisation. Le Tableau G.1 donne les résultats quantitatifs associés.

Pour une analyse complète des résultats se référer au Chapitre 5.

Ce qui peut être retenu ici c'est que non seulement la qualité du clustering n'est pas détériorée dans les cas présents par l'usage d'un arbre couvrant minimum approché, mais l'algorithme semble être aussi capable de détecter des anomalies représentées par les points qui sont - visiblement sur les Figures G.6 et G.7 - dans des clusters singletons.

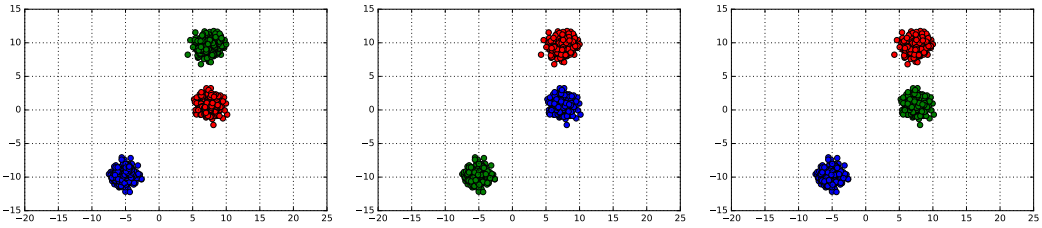


FIGURE G.5: Trois boules: SEMST, DBSCAN ($\epsilon = 1.4$, $minPts = 5$), DBMSTClu avec un arbre couvrant minimum approché.

	Silhouette coefficient			Adjusted Rand Index			DBCVI		
SEMST	**0.84**	**0.16**	−0.12	**1**	0	0	**0.84**	0.001	0.06
DBSCAN	**0.84**	0.02	**0.26**	**1**	**0.99**	**0.99**	**0.84**	−0.26	**0.15**
DBMSTClu	**0.84**	−0.26	**0.26**	**1**	**0.99**	**0.99**	**0.84**	**0.18**	**0.15**

TABLE G.1: Coefficients de Silhouette, Adjusted Rand Index et DBCVI pour les boules, les cercles et bananes bruités avec SEMST, DBSCAN et DBMSTClu.

Passage à l'échelle de l'algorithme

Nous invitons le lecteur à lire le Chapitre 5 en anglais pour découvrir les détails de l'implémentation de l'algorithme DBMSTClu qui permet de garantir une complexité temporelle linéaire en le nombre de noeuds.

Le coût temporel de l'algorithme est véritablement avantageux pour un grand nombre de clusters. Par exemple, sur le jeu de données réelles des champignons[4] qui contient 8124 enregistrements correspondant à 23 espèces de champignons, le temps nécessaire pour retrouver les 23 clusters est de 3.36s pour DBSMTClu tandis que DBSCAN nécessite 9.00s.

[4] https://archive.ics.uci.edu/ml/datasets/mushroom

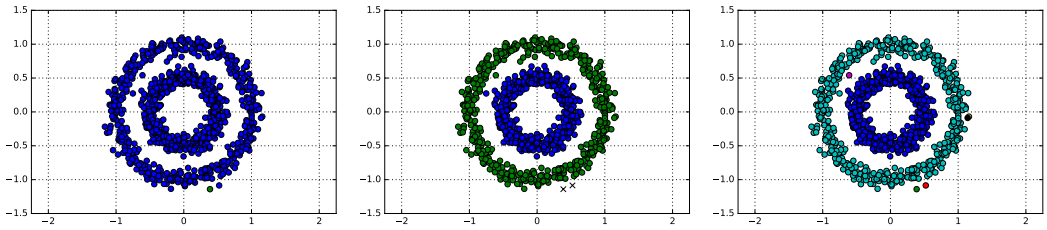


FIGURE G.6: Cercles bruités: SEMST, DBSCAN ($\epsilon = 0.15$, $minPts = 5$), DBMSTClu avec un arbre couvrant minimum approché.

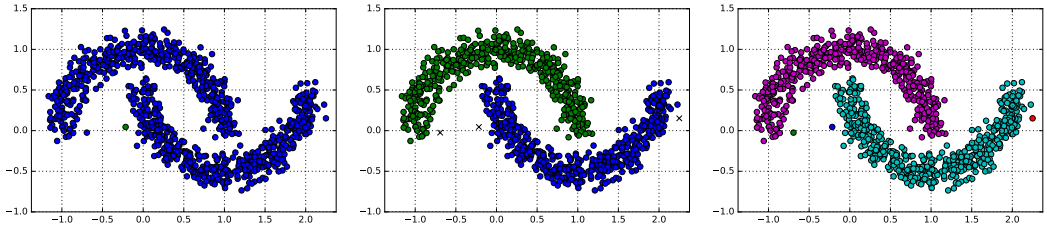


FIGURE G.7: Bananes bruitées: SEMST, DBSCAN ($\epsilon = 0.16$, $minPts = 5$), DBMSTClu avec un arbre couvrant minimum approché.

Pour montrer le passage à l'échelle de l'algorithme sur des gros graphes, des expériences ont été faites en utilisant le modèle stochastique par blocs. K clusters de taille égale sont créés pour un nombre de noeuds donné N. Les temps de calcul associés pour retrouver exactement les K clusters sont donnés dans la Figure G.8 et la Table G.2. En particulier, la Figure G.8 montre bien la complexité temporelle linéaire en N. Des graphes d'un million de noeuds avec 100 clusters sont facilement recouvrés.

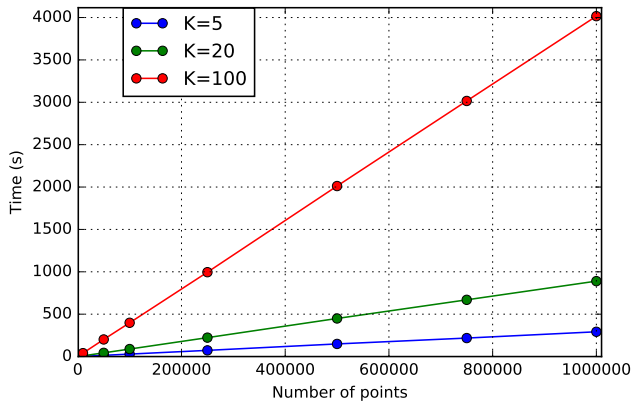


FIGURE G.8: Temps d'exécution de DBMSTClu avec $N \in \{1K, 10K, 50K, 100K, 250K, 500K, 750K, 1M\}$.

$K \backslash N$	1000	10000	50000	100000	250000	500000	750000	1000000
5	0.34	2.96	14.37	28.91	73.04	148.85	218.11	292.25
20	0.95	8.73	43.71	88.51	223.18	449.37	669.29	889.88
100	4.36	40.25	201.76	398.41	995.42	2011.79	3015.61	4016.13
"100/5"	12.82	13.60	14.04	13.78	13.63	13.52	13.83	13.74

TABLE G.2: Valeurs numériques pour le temps d'exécution de DB-MSTClu (en s) en faisant varier N et K (moyenné sur 5 lancers). La dernière ligne montre le ratio de temps d'exécution entre le cas $K = 100$ et $K = 5$.

G.5 Conclusion

G.5.1 Rappel des contributions et inscription dans le sujet

Rappelons que les objectifs de la thèse sont de concevoir des algorithmes efficaces d'*approximation* en temps et en espace pour deux problèmes fondamentaux de l'apprentissage non supervisé: la recherche des plus proches voisins et le clustering lorsqu'il faut faire face à des flux de données massives.

Ces algorithmes d'approximation, s'appuient sur des résumés minimalistes pour faciliter le traitement de ces flux de données massives et doivent satisfaire au mieux le compromis entre:

1. la précision du résultat,
2. le coût en espace mémoire,
3. et le débit des données traitées (streaming).

La première contribution (Section G.2) appartenant à la catégorie des méthodes *indépendantes* des données offre entre autres un cadre théorique à la version courante la plus rapide du Cross-polytope LSH [Terasawa and Tanaka, 2007] pour la recherche des plus proches voisins. Cette dernière consiste à construire les fonctions de hachage de la méthode Cross-polytope LSH en remplaçant les matrices aléatoires gaussiennes par la matrice structurée $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ où $\mathbf{H}$ est la transformée de Hadamard et les $\mathbf{D}_i$ pour $i \in \{1, 2, 3\}$ sont des matrices diagonales aléatoires indépendantes prenant pour valeur de coefficients $+1$ ou -1. Ce cadre théorique est plus général et s'applique à la fois à d'autres matrices structurées et d'autres applications d'apprentissage. Les applications concernées impliquent des produits matrice-vecteur où la matrice est apprise ou aléatoire. Le coût en espace et en temps de ces produits peut être considérablement réduit (typiquement en passant de $O(n^2)$ à $O(n \log n)$), tout en préservant une certaine précision du résultat de l'application. Ce travail respecte donc parfaitement le cadre imposé.

La deuxième contribution (Section G.3) correspond également en tout point au cahier des charges en proposant d'*apprendre* (méthode *adaptée* aux données) des codes binaires compacts de données en grande dimension.

La dernière contribution concernant la méthode de clustering DBMSTCLu (Section G.4) respecte la contrainte mémoire et temporelle puisque l'algorithme ne s'appuie que sur un arbre couvrant minimum, extrait à partir d'un sketch du graphe de dissimilarité, pour effectuer des coupes pertinentes dont résultent les clusters. Ceci assure une complexité spatiale et temporelle linéaire en le nombre de noeuds dans le graphe. De plus, la précision du résultat est garantie par deux choses:

1. la *théorie* proposée,
   - d'une part, qui justifie l'optimalité d'une méthode de clustering fondée sur un arbre couvrant minimum,
   - d'autre part, qui donne des résultats théoriques spécifiques à l'algorithme DBMSTClu sur le bon recouvrement de la partition de clustering,
2. les *expériences* présentées avec un arbre couvrant minimum approché à la place d'un exact comme entrée de l'algorithme. Cet arbre couvrant approché est:
   - soit retrouvé à partir du sketch du graphe des données considérées,

- soit délivré par un algorithme respectant la vie privée différentielle.

Néanmoins, il peut être objecté que l'algorithme de clustering DBMSTClu n'est pas intrinsèquement *streaming*. En effet, si le sketch du graphe est bien mis à jour au fur et à mesure qu'une nouvelle arête est lue du flux, il n'en est pas de même pour l'arbre couvrant minimum et la partition de clustering sous-jacente. En effet, supposons que l'on fige le flux et le sketch, puis que l'on construit à partir de ce sketch un arbre couvrant minimum et la partition de clustering sous-jacente grâce à DBMSTClu. Alors, à l'arrivée d'une nouvelle arête du flux, il n'y a pas d'autre moyen pour l'instant que de reconstruire l'arbre couvrant minimum et la partition de clustering.

G.5.2 Perspectives

Cette partie est mieux développée dans le Chapitre 6 en anglais. Nous n'en disons ici que les grandes lignes.

Une perspective d'amélioration évidente serait donc d'adapter complètement en ligne l'algorithme DBMSTClu. A noter toutefois que la conception d'un algorithme en ligne pour construire un arbre couvrant minimum avec un coût de stockage limité est un sujet de recherche en soi.

Par ailleurs, un travail supplémentaire pourrait être effectué pour déterminer en ligne une rotation optimale, pas nécessairement uniformisant la variance, pour la méthode de hashing hypercubique présentée en Section G.3. Sinon, n'ayant pas unicité de la rotation uniformisant la variance, serait-il possible de déterminer parmi celles-ci une optimale? Selon quel critère?

Enfin, d'autres développements possibles seraient de travailler sur l'aspect implémentation des méthodes dans des frameworks connus pour le streaming (ex: Hadoop, Spark, Redis) ce qui permettrait le déploiement de la méthode pour des applications plus concrètes comme l'analyse de données génomiques.

G.6 Publications

Ce travail de thèse a donné lieu, au moment de l'écriture de ce document, à quatre publications dans des conférences internationales. Elles sont les suivantes:

- Le travail en collaboration décrit en Section G.2 à propos du sketching de matrices via structuration a été publié à la Conférence Internationale d'Intelligence Artificielle et de Statistiques (AISTATS) 2017.

  Mariusz Bojarski, Anna Choromanska, Krzysztof Choromanski, Francois Fagan, Cédric Gouy-Pailler, Anne Morvan, Nouri Sakr, Tamás Sarlós, Jamal Atif.

  Structured adaptive and random spinners for fast machine learning computations

  Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS'17), 54, pp.1020-1029, Fort Lauderdale, FL, USA, 20-22 Apr.

- La contribution de la Section G.3 sur l'apprentissage de codes binaires compacts pour des données en grande dimension a été publiée à la Conférence Internationale de l'Acoutisque, de la Parole et du Traitement du Signal (ICASSP) 2018.

  Anne Morvan, Antoine Souloumiac, Cédric Gouy-Pailler, Jamal Atif.

  Streaming Binary Sketching based on Subspace Tracking and Diagonal Uniformization

  Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018), Calgary, Alberta, Canada, 15-20 Apr.

- L'approche sketching de graphe pour du clustering de flux de données massives a été publiée à la Conférence SIAM Data Mining (SDM) 2018. Une extension de ce travail présentant des garanties théoriques supplémentaires avec une application au clustering préservant la vie privée différentielle a été publiée à la Conférence Internationale de l'Incertitude en Intelligence Artificielle (UAI) 2018.

  Anne Morvan, Krzysztof Choromanski, Cédric Gouy-Pailler, Jamal Atif.

  Graph sketching-based Space-efficient Data Clustering

  Proceedings of the SIAM International Conference on DATA MINING (SDM'18), pp.10-18, San Diego, CA, USA, 3-4 May.

  Rafaël Pinot, Anne Morvan, Florian Yger, Cédric Gouy-Pailler, Jamal Atif.

  Graph-based Clustering under Differential Privacy

  Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI 2018), Monterey, CA, USA, 6-10 Aug.

Bibliography

K. Abed-Meraim, A. Chkeif, and Y. Hua. Fast Orthonormal PAST Algorithm. *IEEE Signal Processing Letters*, (3):60 – 62, 2000. 32, 33, 81, 182

D. Achlioptas. Database-friendly Random Projections: Johnson-Lindenstrauss with Binary Coins. *J. Comput. Syst. Sci.*, 66(4):671–687, June 2003. ISSN 0022-0000. 25

C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A Framework for Clustering Evolving Data Streams. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 81–92. VLDB Endowment, 2003. ISBN 0-12-722442-4. 37, 38

R. Agrawal and A. Swami. A One-Pass Space-Efficient Algorithm for Finding Quantiles. In *IN PROC. 7TH INTL. CONF. MANAGEMENT OF DATA (COMAD-95)*, 1995. 9

K. J. Ahn, S. Guha, and A. McGregor. Analyzing Graph Structure via Linear Measurements. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 459–467, Philadelphia, PA, USA, 2012a. Society for Industrial and Applied Mathematics. 41, 45, 46, 49, 51, 96, 115, 116, 125, 178, 186

K. J. Ahn, S. Guha, and A. McGregor. Graph Sketches: Sparsification, Spanners, and Subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '12, pages 5–14, New York, NY, USA, 2012b. ACM. 45

K. J. Ahn, S. Guha, and A. McGregor. *Spectral Sparsification in Dynamic Graph Streams*, pages 1–10. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. 45

N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proceedings of the 38th STOC*, pages 557–563. ACM, 2006. 6, 25, 54, 57, 58

N. Ailon and E. Liberty. An almost optimal unrestricted fast Johnson-Lindenstrauss transform. In *SODA*, 2011. 54

N. Ailon, R. Jaiswal, and C. Monteleoni. Streaming k-means approximation. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 10–18. Curran Associates, Inc., 2009. 36, 37

N. Ailon, Y. Chen, and H. Xu. Breaking the Small Cluster Barrier of Graph Clustering. *CoRR*, abs/1302.4549, 2013. 41

N. Alon, Y. Matias, and M. Szegedy. The Space Complexity of Approximating the Frequency Moments. In *Proceedings of the Twenty-eighth Annual ACM Symposium*

on Theory of Computing, STOC '96, pages 20–29, New York, NY, USA, 1996. ACM. 9

A. Andoni and P. Indyk. Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Commun. ACM*, (1):117–122, 2008. 22, 74

A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt. Practical and Optimal LSH for Angular Distance. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 1225–1233, Cambridge, MA, USA, 2015a. MIT Press. 5, 6, 11, 22, 23, 24, 51, 123, 129, 177, 179

A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt. Practical and Optimal LSH for Angular Distance. In *NIPS*, pages 1225–1233, 2015b. 54, 55, 56, 67, 68, 69, 70, 74

D. Arthur and S. Vassilvitskii. K-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. 36

T. Asano, B. Bhattacharya, M. Keil, and F. Yao. Clustering Algorithms Based on Minimum and Maximum Spanning Trees. In *Proceedings of the Fourth Annual Symposium on Computational Geometry*, SCG '88, pages 252–257, New York, NY, USA, 1988. ACM. 43, 44, 154, 187

F. Aurenhammer. Voronoi Diagrams&Mdash;a Survey of a Fundamental Geometric Data Structure. *ACM Comput. Surv.*, 23(3):345–405, Sept. 1991. ISSN 0360-0300. 5

K. Azuma. Weighted sums of certain dependent random variables. *Tohoku Math. J. (2)*, 19(3):357–367, 1967. doi: 10.2748/tmj/1178243286. 11, 57, 151, 179

B. Bandyopadhyay, D. Fuhry, A. Chakrabarti, and S. Parthasarathy. Topological Graph Sketching for Incremental and Scalable Analytics. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1231–1240, New York, NY, USA, 2016. ACM. 45

A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013. 33, 35

V. Bentkus. On the dependence of the Berry–Esseen bound on dimension. *Journal of Statistical Planning and Inference*, 113(2):385–402, 2003. 11, 65, 179

J. L. Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM*, 18(9):509–517, Sept. 1975. 5, 37

S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. Tsourakakis. Space- and Time-Efficient Algorithm for Maintaining Dense Subgraphs on One-Pass Dynamic Streams. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 173–182, New York, NY, USA, 2015. ACM. 45

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 3

A. Blum, K. Ligett, and A. Roth. A Learning Theory Approach to Non-interactive Database Privacy. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 609–618, New York, NY, USA, 2008. ACM. 154, 157

C. Boutsidis, D. Garber, Z. S. Karnin, and E. Liberty. Online Principal Components Analysis. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 887–901, 2015. 32

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787. 142

R. S. Boyer and J. S. Moore. *MJRTY - A Fast Majority Vote Algorithm*, pages 105–117. Springer Netherlands, Dordrecht, 1991. 9

R. P. Brent, J. H. Osborn, and W. D. Smith. Bounds on determinants of perturbed diagonal matrices. *arXiv:1401.7084*, 2014. 141

F. Cakir and S. Sclaroff. Adaptive Hashing for Fast Similarity Search. In *ICCV*, December 2015. 26, 33

F. Cakir, K. He, S. Bargal, and S. Sclaroff. MIHash: Online Hashing With Mutual Information. In *ICCV*, Oct 2017. 26, 33

F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *In 2006 SIAM Conference on Data Mining*, pages 328–339, 2006. 37, 38, 39

M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002. 5, 11, 22, 23, 51, 54, 150, 177, 179

L. Chen, T. Yu, and R. Chirkova. WaveCluster with Differential Privacy. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1011–1020, New York, NY, USA, 2015. ACM. 154, 157

X. Chen, I. King, and M. R. Lyu. FROSH: FasteR Online Sketching Hashing. In *UAI*, 2017. 28, 74

Y. Chen, S. Sanghavi, and H. Xu. Clustering Sparse Graphs. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2204–2212. Curran Associates, Inc., 2012a. 41

Y. Chen, S. Sanghavi, and H. Xu. Clustering Sparse Graphs. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2204–2212. Curran Associates, Inc., 2012b. 41

Y. Chen, A. Jalali, S. Sanghavi, and H. Xu. Clustering Partially Observed Graphs via Convex Optimization. *J. Mach. Learn. Res.*, 15(1):2213–2238, Jan. 2014a. 41

Y. Chen, S. H. Lim, and H. Xu. Weighted Graph Clustering with Non-uniform Uncertainties. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1566–II–1574. JMLR.org, 2014b. 41

A. Choromanska, K. Choromanski, M. Bojarski, T. Jebara, S. Kumar, and Y. LeCun. Binary embeddings with structured hashed projections. In *ICML*, 2016. 135, 136

K. Choromanski and V. Sindhwani. Recycling Randomness with Structure for Sublinear time Kernel Expansions. In *ICML*, 2016. 56, 57, 135, 136, 141, 142

A. Condon and R. M. Karp. Algorithms for Graph Partitioning on the Planted Partition Model. *Random Struct. Algorithms*, 18(2):116–140, Mar. 2001. 41

J. W. Cooley and J. W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19(90):297–301, 1965. 131

G. Cormode and D. Firmani. A unifying framework for l_0-sampling algorithms. *Distributed and Parallel Databases*, 32(3):315–335, 2014. Special issue on Data Summarization on Big Data. 47, 116

G. Cormode and S. Muthukrishnan. An Improved Data Stream Summary: The Count-min Sketch and Its Applications. *J. Algorithms*, 55(1):58–75, Apr. 2005. ISSN 0196-6774. 9

G. Cormode and S. Muthukrishnan. Approximating Data with the Count-Min Sketch. *IEEE Software*, 29(1):64–69, 2012. 9

G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches. *Found. Trends databases*, 4:1–294, Jan. 2012. 9

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online Passive-Aggressive Algorithms. *J. Mach. Learn. Res.*, 7:551–585, Dec. 2006. 26

B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. Balcan, and L. Song. Scalable Kernel Methods via Doubly Stochastic Gradients. In *NIPS*, 2014. 137

A. Dasgupta, R. Kumar, and T. Sarlos. A Sparse Johnson: Lindenstrauss Transform. 2010. 54

S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *STOC*, 2008. 135

S. Dasgupta and A. Gupta. An elementary proof of the Johnson-Lindenstrauss Lemma. Technical report, UC Berkeley, 1999. 25

P. J. Davis. *Circulant Matrices*. Wiley, 1970. 131

W. H. E. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1(1):7–24, Dec 1984. 7

D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977. 7

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. 7

M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. D. Freitas. Predicting Parameters in Deep Learning. In *NIPS*. 2013. 137

T.-T. Do, A.-D. Doan, and N.-M. Cheung. Learning to Hash with Binary Deep Neural Network. In *ECCV*, 2016. 26

M. Durand and P. Flajolet. Loglog Counting of Large Cardinalities. In *In ESA*, pages 605–617, 2003. 9

C. Dwork. A Firm Foundation for Private Data Analysis. *Commun. ACM*, 54(1): 86–95, Jan. 2011. 154, 157

C. Dwork and A. Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407, 2013. 156, 164

C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *Eurocrypt*, volume 4004, pages 486–503. Springer, 2006a. 156

C. Dwork, F. McSherry, K. Nissim, and A.Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*, pages 265–284. Springer Berlin Heidelberg, 2006b. 154, 155, 156

M. Ester, H. peter Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996. 8, 40, 96, 117, 187

T. Falkowski, A. Barth, and M. Spiliopoulou. DENGRAPH: A Density-based Community Detection Algorithm. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, WI '07, pages 112–115, Washington, DC, USA, 2007. IEEE Computer Society. 41

C. Feng, Q. Hu, and S. Liao. Random Feature Mapping with Signed Circulant Matrix Projection. In *IJCAI*, 2015. 136

B. J. Fino and V. R. Algazi. Unified Matrix Treatment of the Fast Walsh-Hadamard Transform. *IEEE Trans. Comput.*, 25(11):1142–1146, Nov. 1976. 129

P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182 – 209, 1985. 9

P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier. HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. In P. Jacquet, editor, *AofA: Analysis of Algorithms*, volume DMTCS Proceedings vol. AH, 2007 Conference on Analysis of Algorithms (AofA 07) of *DMTCS Proceedings*, pages 137–156, Juan les Pins, France, June 2007. Discrete Mathematics and Theoretical Computer Science. 9

P. Frankl and H. Maehara. The Johnson-Lindenstrauss Lemma and the Sphericity of Some Graphs. *J. Comb. Theory Ser. A*, 44(3):355–362, June 1987. 24

M. L. Fredman and D. E. Willard. Trans-dichotomous Algorithms for Minimum Spanning Trees and Shortest Paths. *J. Comput. Syst. Sci.*, 48(3):533–551, June 1994. ISSN 0022-0000. 45

H. N. Gabow, Z. Galil, T. Spencer, and R. E. Tarjan. Efficient Algorithms for Finding Minimum Spanning Trees in Undirected and Directed Graphs. *Combinatorica*, 6 (2):109–122, Jan. 1986. ISSN 0209-9683. 45

A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-615-7. 5

G. H. Golub and H. A. van der Vorst. Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, (1–2):35 – 65, 2000. Numerical Analysis 2000. Vol. III: Linear Algebra. 32, 82, 182

Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (12):2916–2929, 2013. 6, 17, 26, 28, 29, 74, 76, 126, 178, 182

I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. Book in preparation for MIT Press, 2016. 136

K. Grauman and B. Kulis. Kernelized Locality-Sensitive Hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1092–1104, 2011. 22

O. Grygorash, Y. Zhou, and Z. Jorgensen. Minimum Spanning Tree Based Clustering Algorithms. In *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, pages 73–81, 2006. 43, 44, 154

S. Guha, R. Rastogi, and K. Shim. Cure: An Efficient Clustering Algorithm for Large Databases. *Inf. Syst.*, 26(1):35–58, 2001. 37

S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering Data Streams: Theory and Practice. *IEEE Trans. on Knowl. and Data Eng.*, 15(3): 515–528, Mar. 2003. 36, 37

I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.*, 3:1157–1182, Mar. 2003. ISSN 1532-4435. 3

S. Har-Peled, P. Indyk, and R. Motwani. Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality. *Theory of Computing*, 8(14):321–350, 2012. 54

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001. 3

M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate Estimation of the Degree Distribution of Private Networks. In *2009 Ninth IEEE International Conference on Data Mining*, pages 169–178, Dec 2009. 155

M. R. Henzinger, P. Raghavan, and S. Rajagopalan. External Memory Algorithms. chapter Computing on Data Streams, pages 107–118. American Mathematical Society, Boston, MA, USA, 1999. ISBN 0-8218-1184-3. 9

J. Herault and B. Ans. Neuronal network with modifiable synapses: decoding of composite sensory messages under unsupervised and permanent learning. 299:525–8, 02 1984. 3

A. Hinrichs and J. Vybíral. Johnson-Lindenstrauss lemma for circulant matrices. *Random Struct. Algorithms*, 39(3):391–398, 2011. 54, 129

S.-S. Ho and S. Ruan. Preserving Privacy for Interesting Location Pattern Mining from Trajectory Data. *Trans. Data Privacy*, 6(1):87–106, Apr. 2013. 154, 157

P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, June 1983. 41

H. Hotelling. Analysis of a complex of statistical variables into principal components. *J. Educ. Psych.*, 24, 1933. 3, 27

L.-K. Huang, Q. Yang, and W.-S. Zheng. Online Hashing. In *IJCAI*, pages 1422–1428, 2013. 26, 33

P.-S. Huang, H. Avron, T. Sainath, V. Sindhwani, and B. Ramabhadran. Kernel Methods match Deep Neural Networks on TIMIT. In *ICASSP*, 2014. 135, 137

Z. Huang and P. Peng. Dynamic Graph Stream Algorithms in $o(n)$ Space. *CoRR*, abs/1605.00089, 2016. 45

M. Huhtanen and A. Perämäki. Factoring Matrices into the Product of Circulant and Diagonal Matrices. *Journal of Fourier Analysis and Applications*, 21(5):1018–1033, 2015. 137

P. Indyk and R. Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM. 5, 11, 18, 21, 25, 33, 177, 179

A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988. 7, 96

W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, volume 26 of *Contemporary Mathematics*, pages 189–206. American Mathematical Society, 1984. 24, 54

I. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986. 30

H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311, 2010. 28, 74

T. Kailath and A. Sayed. *Fast Reliable Algorithms for Matrices with Structure*. Society for Industrial and Applied Mathematics, 1999. 131, 132

D. M. Kane and J. Nelson. A Sparser Johnson-Lindenstrauss Transform. 2010. 25

D. R. Karger, P. N. Klein, and R. E. Tarjan. A Randomized Linear-time Algorithm to Find Minimum Spanning Trees. *J. ACM*, 42(2):321–328, Mar. 1995. ISSN 0004-5411. 45

Z. Karnin and E. Liberty. Online PCA with Spectral Bounds. In *COLT*, 2015. 32

S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith. Analyzing Graphs with Node Differential Privacy. In *Proceedings of the 10th Theory of Cryptography Conference on Theory of Cryptography*, TCC'13, pages 457–476, Berlin, Heidelberg, 2013. Springer-Verlag. 155

L. Kaufman and P. J. Rousseeuw. Clustering by means of medoids. *Statistical Data Analysis Based on the L_1–Norm and Related Methods*, pages 405–416, 1987. 7, 96

W. Kong and W.-j. Li. Isotropic Hashing. In *NIPS*, pages 1646–1654. 2012. 26, 28, 29, 30, 74, 76, 81, 86, 126, 178, 182

H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, pages 3270–3278, 2015. 26, 34

Q. Le, T. Sarlós, and A. Smola. Fastfood-computing hilbert space expansions in loglinear time. In *ICML*, 2013. 135, 137

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 136

Y. Lee. Spherical Hashing. In *CVPR*, pages 2957–2964, 2012. 26

C. Leng, J. Wu, J. Cheng, X. Bai, and H. Lu. Online sketching hashing. In *CVPR*, pages 2503–2511, 2015a. 6, 26, 28, 32, 74, 75, 77, 84, 89, 149, 178, 182

C. Leng, J. Wu, J. Cheng, X. Zhang, and H. Lu. Hashing for Distributed Data. In *ICML*, pages 1642–1650, 2015b. 26

J. Li. Restructuring of Deep Neural Network Acoustic Models with Singular Value Decomposition. In *Interspeech*, 2013. 137

E. Liberty. Simple and Deterministic Matrix Sketching. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 581–588, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2174-7. 30, 31, 32

E. Liberty, N. Ailon, and A. Singer. Dense fast random projections and lean Walsh transforms. In *RANDOM*, 2008. 54

V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *CVPR*, pages 2475–2483, June 2015. 26

W. Liu, J. Wang, and S. fu Chang. Hashing with graphs. In *ICML*, 2011. 26

W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012. 26

W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete Graph Hashing. In *NIPS*, pages 3419–3427. 2014. 26

S. Lloyd. Least Squares Quantization in PCM. *IEEE Trans. Inf. Theor.*, 28(2): 129–137, Sept. 1982. 7, 35, 96

U. Luxburg. A Tutorial on Spectral Clustering. *Statistics and Computing*, 17(4): 395–416, Dec. 2007. 41

U. v. Luxburg and O. Bousquet. Distance–Based Classification with Lipschitz Functions. *J. Mach. Learn. Res.*, 5:669–695, Dec. 2004. ISSN 1532-4435. 3

Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe LSH: Efficient Indexing for High-dimensional Similarity Search. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 950–961. VLDB Endowment, 2007. ISBN 978-1-59593-649-3. 22, 23

Q. Ma, S. Muthukrishnan, and M. Sandler. *Frugal Streaming for Estimating Quantiles*, pages 77–96. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. 9

J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional Kernel Networks. In *NIPS*, 2014. 137

J. Matoušek. On Variants of the Johnson-Lindenstrauss Lemma. *Random Struct. Algorithms*, 33(2):142–156, Sept. 2008. 25

A. McGregor. Graph Stream Algorithms: A Survey. *SIGMOD Rec.*, 43(1):9–20, May 2014. 9, 45

F. McSherry. Privacy Integrated Queries, booktitle = Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (SIGMOD). Association for Computing Machinery, Inc., June 2009. 154, 157

F. McSherry and K. Talwar. Mechanism Design via Differential Privacy. In *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, Providence, RI, October 2007. IEEE. 156

M. Moczulski, M. Denil, J. Appleyard, and N. de Freitas. ACDC: A Structured Efficient Linear Layer. In *ICLR*, 2016. 137

D. Moulavi, P. A. Jaskowiak, R. J. G. B. Campello, A. Zimek, and J. Sander. Density-Based Clustering Validation. In *Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014*, pages 839–847, 2014. 101

Y. Mülle, C. Clifton, and K. Böhm. Privacy-Integrated Graph Clustering Through Differential Privacy. In *EDBT/ICDT Workshops*, 2015. 157

J. Munro and M. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12(3):315 – 323, 1980. 9

S. Muthukrishnan. Data Streams: Algorithms and Applications. *Found. Trends Theor. Comput. Sci.*, 1(2):117–236, Aug. 2005. 9, 96, 115

M. C. Nascimento and A. C. de Carvalho. Spectral methods for graph clustering - A survey. *European Journal of Operational Research*, 211(2):221–231, 2011. 96

National Security Agency. FIPS 180-1. Secure Hash Standard. Technical report, US Department of Commerce, Apr. 1995. 19

A. Y. Ng, M. I. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, 2002. 43

H. H. Nguyen, A. Imine, and M. Rusinowitch. Detecting Communities Under Differential Privacy. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, WPES '16, pages 83–93, New York, NY, USA, 2016. ACM. 157

K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing - STOC*. ACM Press, 2007. 154, 157

M. Norouzi and D. J. Fleet. Cartesian K-Means. In *CVPR*, pages 3017–3024, June 2013. 26

S. Oymak and B. Hassibi. Finding Dense Clusters via "Low Rank + Sparse" Decomposition. *CoRR*, abs/1104.5186, 2011. 41

V. Y. Pan. *Structured Matrices and Polynomials: Unified Superfast Algorithms*. Springer-Verlag, Berlin, Heidelberg, 2001. 131

K. Pearson. On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine*, 2:559–572, 1901. 3, 27

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 117

M. Pilanci and M. J. Wainwright. Randomized sketches of convex programs with sharp guarantees. In *ISIT*, 2014. 135, 136

M. Pilanci and M. J. Wainwright. Newton Sketch: A Linear-time Optimization Algorithm with Linear-Quadratic Convergence. *CoRR*, abs/1505.02250, 2015. 135, 136, 142, 143

R. Pinot. Minimum spanning tree release under differential privacy constraints. *ArXiv e-prints*, Jan. 2018. xvii, 125, 155, 157, 158

M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, pages 1509–1517. 2009. 22

A. Rahimi and B. Recht. Random Features for Large-Scale Kernel Machines. In *NIPS*, 2007. 135

W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971. 120

R. Raziperchikolaei and M. Á. Carreira-Perpiñán. Optimizing affinity-based binary hashing using auxiliary coordinates. In *NIPS*, pages 640–648, 2016. 26

K. Rohe, S. Chatterjee, and B. Yu. Spectral clustering and the high-dimensional stochastic blockmodel. *Ann. Statist.*, 39(4):1878–1915, 08 2011. 41

P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987. 101, 120

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290:2323–2326, 2000. 3

T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran. Low-rank matrix factorization for Deep Neural Network training with high-dimensional output targets. In *ICASSP*, 2013. 137

S. E. Schaeffer. Survey: Graph Clustering. *Comput. Sci. Rev.*, 1(1):27–64, Aug. 2007. 41, 154

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear Component Analysis As a Kernel Eigenvalue Problem. *Neural Comput.*, 10(5):1299–1319, July 1998. 3

A. Sealfon. Shortest Paths and Distances with Differential Privacy. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems - PODS*. ACM Press, 2016. 155

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug 2000. 43

R. Sibson. SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973. 7

V. Sindhwani, T. N. Sainath, and S. Kumar. Structured Transforms for Small-Footprint Deep Learning. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3088–3096, 2015. 137

N. Sundaram, A. Turmukhametova, N. Satish, T. Mostak, P. Indyk, S. Madden, and P. Dubey. Streaming Similarity Search over One Billion Tweets Using Parallel Locality-sensitive Hashing. *Proc. VLDB Endow.*, 6(14):1930–1941, Sept. 2013. ISSN 2150-8097. 11, 23, 177, 179

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319, 2000. 3

K. Terasawa and Y. Tanaka. Spherical LSH for Approximate Nearest Neighbor Search on Unit Hypersphere. In *WADS*, pages 27–38, 2007. 5, 11, 22, 23, 51, 54, 55, 74, 123, 177, 179, 180, 190

C. F. van Loan. The Ubiquitous Kronecker Product. *J. Comput. Appl. Math.*, 123 (1-2):85–100, Nov. 2000. ISSN 0377-0427. 133

J. S. Vitter. Random Sampling with a Reservoir. *ACM Trans. Math. Softw.*, 11(1): 37–57, Mar. 1985. 9, 37

J. Vybíral. A variant of the Johnson-Lindenstrauss lemma for circulant matrices. *Journal of Functional Analysis*, 260(4):1096–1105, 2011. 54, 129

J. Wang, S. Kumar, and S.-F. Chang. Semi-Supervised Hashing for Large-Scale Search. *IEEE Trans. Pattern Anal. Mach. Intell.*, (12):2393–2406, 2012. 26

J. Wang, W. Liu, S. Kumar, and S.-F. Chang. Learning to Hash for Indexing Big Data - A Survey. *Proceedings of the IEEE*, (1):34–57, 2016. 52

J. Wang, T. Zhang, j. song, N. Sebe, and H. T. Shen. A Survey on Learning to Hash. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, PP(99):1–1, 2018. 26, 74

Y. Weiss, A. Torralba, and R. Fergus. Spectral Hashing. In *NIPS*, pages 1753–1760, 2008. 26

E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance Metric Learning, with Application to Clustering with Side-information. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, pages 521–528, Cambridge, MA, USA, 2002. MIT Press. 34

Y. Xu, V. Olman, and D. Xu. Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees. 18(4):536–545, 2002. 43, 44, 187

Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. Smola, L. Song, and Z. Wang. Deep Fried Convnets. In *ICCV*, 2015. 137

F. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant Binary Embedding. In *ICML*, 2014. 26, 74

C. T. Zahn. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Trans. Comput.*, 20(1):68–86, Jan. 1971. 43, 44, 117, 154

T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, pages 103–114, New York, NY, USA, 1996. ACM. 38

X. Zhang, F. X. Yu, R. Guo, S. Kumar, S. Wang, and S.-F. Chang. Fast Orthogonal Projection Based on Kronecker Product. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2929–2937, 2015. 129, 133

Résumé

Cette thèse étudie deux tâches fondamentales d'apprentissage non supervisé : la recherche des plus proches voisins et le clustering de données massives en grande dimension pour respecter d'importantes contraintes de temps et d'espace.

Tout d'abord, un nouveau cadre théorique permet de réduire le coût spatial et d'augmenter le débit de traitement du Cross-polytope LSH pour la recherche du plus proche voisin presque sans aucune perte de précision.

Ensuite, une méthode est conçue pour apprendre en une seule passe sur des données en grande dimension des codes compacts binaires. En plus de garanties théoriques, la qualité des représentations obtenues est mesurée dans le cadre de la recherche approximative des plus proches voisins.

Puis, un algorithme de clustering sans paramètre et efficace en termes de coût de stockage est développé en s'appuyant sur l'extraction d'un arbre couvrant minimal approché du graphe de dissimilarité compressé auquel des coupes bien choisies sont effectuées.

Mots Clés

Apprentissage non supervisé, Recherche des plus proches voisins, Clustering, Approximation, Flux, Réduction de dimension, Hachage, Résumés minimalistes.

Abstract

This thesis focuses on how to perform efficiently unsupervised machine learning such as the fundamentally linked nearest neighbor search and clustering task, under time and space constraints for high-dimensional datasets.

First, a new theoretical framework reduces the space cost and increases the rate of flow of data-independent Cross-polytope LSH for the approximative nearest neighbor search with almost no loss of accuracy.

Second, a novel streaming data-dependent method is designed to learn compact binary codes from high-dimensional data points in only one pass. Besides some theoretical guarantees, the quality of the obtained embeddings is evaluated on the approximate nearest neighbors search task.

Finally, a space-efficient parameter-free clustering algorithm is conceived, based on the recovery of an approximate minimum spanning tree of the sketched data dissimilarity graph on which suitable cuts are performed.

Keywords

Unsupervised learning, Nearest neighbors search, Clustering, Approximation, Streaming, Dimensionality reduction, Hashing, Sketching.