UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

Compressive Clustering of High-Dimensional Datasets by 1-bit Sketching

Dissertation presented by
Vincent SCHELLEKENS

for obtaining the Master's degree in
Electrical Engineering

Supervisor
Laurent JACQUES

Readers
David BOL, Christophe DE VLEESCHOUWER

Academic year 2016-2017

Abstract

Machine learning algorithms, such as the k-means clustering problem, typically require several passes on a dataset of learning examples, which becomes prohibitive when the amount of examples becomes very large. Inspired by the field of Compressed Sensing, recent work has proposed a new technique to reduce the size of very large dataset by summarizing it into a single object called the *sketch*. To construct this summary, the sketch passes random projections of the learning examples through a complex exponential before a pooling step, i.e. a summation over the set of learning examples. The resulting sketch has a size that is independent of the size of the dataset and has been used successfully in Gaussian Mixture Model estimation and clustering problems. This work considers the possibility to define a new sketching operator that replaces the complex exponential with the universal quantization function. The idea of this modification of the sketch is inspired by 1-bit embeddings of signals with the universal quantization, that preserve the local distances of high-dimensional vectors. This new sketch has the advantage of being a step closer to the hardware implementation of a sensor capturing the sketch of a dataset directly. However this modification of the sketch operator presents new challenges for the centroid retrieval algorithms due to the discontinuous nature of the universal quantization operation, and raises questions regarding the selection of the parameters of this new sketch.

New algorithms to find the centroids from the quantized sketch are proposed. A theoretical analysis of generalized sketching operators that rely on any periodic function as signature function allows to obtain new insights that highlight the consequences of changing the sketch signature function. Eventually those new insights lead to empirical rules for tuning the parameters of the sketch relying on universal quantization.

Experimental results show that the quantized sketch, combined with adapted algorithms and sketch parameters, allows for accurate clustering results. However, the required sketch size in the quantized case is at least 20 times larger than when the traditional sketch is used. The reason for this increase of the sketch dimension is discussed and potential solutions are proposed.

Contents

Chapter 1

Introduction

Throughout history, mankind has always been collecting information, written at first, and later in the form of analog and digital signals generated by *sensors* that record physical quantities about the surrounding world. In recent years, with an explosion in the number of sensors (e.g. for Internet of Things applications), and more generally of information producers and consumers, this amount of recorded information is reaching unprecedented peaks. The buzzword "big data" is a common name for datasets that are very large in size, and therefore costly to store, access and process. Those enormous quantities of data play in favor of *machine learning algorithms* that exploit large datasets to learn useful information -a model- from them, and become thus more accurate when more data -more learning examples- are available.

However, the training phase of machine learning, i.e. discovering a model from the data, asserting its performances on a test set and comparing it to other models, is well known to be very demanding in terms of computational power. On very large datasets, machine learning algorithms can keep running for days given the sheer amount of examples to process. Paradoxally, the output of those machine learning algorithms that treat tons of learning examples is usually a very "simple" model, in the sense that it can be expressed very compactly. Indeed, a machine learning model is often represented in the form of a small amount of coefficients; e.g. in k-means clustering, the problem we will focus on, those coefficient are the positions centroids defining the clusters, that are far less numerous than the amount of learning examples. It seems thus to be a waste to invest a lot of acquisition ressources to capture huge amounts of data to, at the end, produce "only" a simple model. Moreover, large datasets often are very redundant, first because many examples are very similar to each other, but also because the features of a single example can be themselves redundant, especially in high-dimensional dataset (for examples, neighbouring pixels in an image are -most of the time- very similar).

To account for this discrepancy in the size of the dataset and the true information content hidden inside it (the model we want to learn), the redundancy inside datasets are often reduced with one of the many existing dimensionality reduction tools. But since the number of training examples is not reduced, this technique scales poorly with the size of very large datasets: the redundancy accross the different learning examples is still present. Recent work has proposed another approach in dataset reduction consisting in mapping the dataset to a single object called the *sketch*, whose size is independent of the number of training examples [1],[2]. Given the sketch is computed beforehand, the resulting training algorithms possess an execution time that is independent of the number of training examples, and are for this reason a promising alternative to traditional learning techniques in the context of very large datasets. Those algorithms have proven to compete favorably with state-of-the art traditional learning algorithms both in terms of learning speed and accuracy of the results.

However, in this compressive learning setting, the whole dataset still needs to be acquired at the sensing stage. Similarities of sketched learning with the field of Compressed Sensing raises the question: why not acquiring, from the start, only the information needed for learning (i.e. the sketch)? In other words, is it possible to design sensors that record *only the mimimum amount of information* needed to train *accurate* machine learning models?

This work addresses this question in the context of the clustering problem. Clustering aims at grouping training examples in an unsupervised way by producing centroids that are representatives of the different classes in the data. The goal of this work will be to obtain the centroids representing a high-dimensional dataset from a compressed representation of this dataset in the form of a sketch. Unlike previous work, we focus on a sketch that is hardware-friendly, based on the universal quantization operator, in the optic of designing sensors that would be able to directly acquire the sketch of the learning examples. This would make way to designing acquisition devices to acquire the bare minimum information content for clustering applications, e.g. in low-power cameras or hyperspectral sensors.

Notations and conventions

In this work, lower case symbols (e.g. $n, m...$) usually denote scalar quantities, bold symbols (e.g. $\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\alpha}...$) denote (column) vectors, and matrices are represented by upper case letters (e.g. $A, \Omega, \Sigma...$). Oftentimes, we refer to a matrix and to the set whose elements are the columns of this matrix with the same symbol. By abuse of notation, we often write probability distributions (e.g. $\mathcal{P}, \mathcal{Q}, \Lambda, \mathcal{U}$ (the uniform distribution),...) and their probability density function (pdf) with the same symbol. When a vector or matrix is passed as an argument to a scalar function (e.g. $f(t), Q_\Delta(t), e^t...$), we mean by abuse of notation that this function is applied component-wise to each element of this vector/matrix. The Hadamard (or "component-wise") product on vectors/matrices is noted $\circ$, the convolution between functions is noted $*$. The complex number is noted i (i and j are summation indexes). The asymptotic notations $\mathcal{O}$ and Ω mean that, respectively, $f(n) = \mathcal{O}(g(n)) \iff \exists\, N, c \text{ s.t. } f(n) \leq cg(n)\ \forall n \geq N$ and $f(n) = \Omega(g(n)) \iff \exists\, N, c \text{ s.t. } f(n) \geq cg(n)\ \forall n \geq N$.

Structure of the thesis

The rest of this master thesis is organized as follows:

- In **Chapter 2**, the preliminary concepts related to the field of machine learning (and in particular, clustering) and Compressed Sensing are introduced. Then, the Compressive K-Means (`CKM`) algorithm [1] for compressive clustering is presented in detail along with important interpretations of what it does exactly, and how it connects to other work.

- Starting from the `CKM` algorithm and inspired by quantized embeddings presented in Chapter 2, **Chapter 3** proposes Quantized Compressive K-Means (`QCKM`), a novel compressive clustering algorithm based on 1-bit quantized measurements of the dataset. The advantages and limitations of `QCKM` are discussed, and solutions to its main challenges are proposed, one of them being the main inspiration for the next chapter.

- **Chapter 4** presents an unified framework for compressive clustering (or more generally, probability distribution matching) by sketching with general periodic signature functions f. This framework allows to draw connexions between `CKM` and `QCKM` that can each be seen as particular cases of this generalized sketched learning procedure. After discussing how the results and interpretations of `CKM` from Chapter 2 can be generalized, we then particularize those results to get new insights about `QCKM` proposed in Chapter 3, and

propose adapted choices of parameters that should enhance its performances. Based on those insights, we conclude this chapter by a complete overview of the `QCKM` algorithm and discuss the different modelization and truncation errors that are present in the scheme.

- The `QCKM` algorithm from Chapter 3 (with parameters choices derived from Chapter 4) is instantiated and experimental results asserting its performances are given in **Chapter 5**. In the light of empirical observations from this chapter, technical promises/requirements of `QCKM` are discussed.
- **Chapter 6** concludes this work by summarizing the main results and observations of the previous chapters, and discusses questions that remain open at this point along with directions for future work.

Chapter 2

State of the art

This chapter presents the core concepts of this master thesis, and introduces the notations that will be used as consistently as possibly in the sections that follow. In a "top-down" style, we will first present the fields that are related to this thesis (that is, in turn, machine learning and Compressed Sensing), and then focus our attention on the concepts that are relevant for this work. Thus, the first section of this chapter will explain the clustering problem -a machine learning problem- and present several state of the art algorithms to solve this problem such as the popular `k-means++`. This clustering problem is the **task** that we want to solve in this work. In the second section, the Compressed Sensing (CS) framework will be presented. The field of CS will provide us essential **tools** for this work, such as the Orthogonal Matching Pursuit algorithm and the notion of (1-bit universal quantization) embeddings. Finally, in the third section, the fields of machine learning and Compressed Sensing meet together in the form of "compressive learning" (i.e. solving machine learning problems in the compressed domain, the **procedure** of solving machine learning tasks with tools from CS), and we present the `CKM` algorithm for *compressive clustering* along with personal insights and interpretations.

2.1 The task: the clustering problem

2.1.1 Machine learning

Nowadays, *machine learning* is a very popular field in computer science and applied mathematics. As its name implies, the main idea of machine learning is to get computers (the "machine") to execute a complicated task (such a pattern and face recognition in images, or spam classification in a mailbox) by letting the computer discover by itself from a dataset of examples (the "learning") how to do this task instead of programming the task explicitly. The dataset of examples X is a set of N examples usually represented by vectors $\boldsymbol{x}_i \in \mathbb{R}^n$ living in a n-dimensional space : $X = \{\boldsymbol{x}_i \mid i = 1, \dots, N\} \subset \mathbb{R}^n$. Machine learning problems are called *supervised* if the training examples $\boldsymbol{x}_i$ are associated with a (known) output value y_i (that can be continuous for a regression problem or discrete for a classification problem, y_i then being called the "label"), and the goal is then usually to predict this output value on new, unseen examples. On the other side, *unsupervised* learning problems aim at extracting some form of knowledge from learning examples that have no (or an unknown) output value.

The extracted knowledge in an unsupervised learning problem can take many forms, but usually the goal is to find some structure hidden in the unlabeled data. In our context, the learning examples are assumed to be drawn independently from some distribution with associated probability density function $\mathcal{P}$, in other words $\boldsymbol{x}_i \overset{\text{iid}}{\sim} \mathcal{P}$[1]. This distribution $\mathcal{P}$ then completely

[1]By abuse of notation, we will denote the probability distribution and the probability density function of a random variable with the same symbol.

characterises the "structure" in the data.

2.1.2 Clustering

The most notable unsupervised machine learning problem is called the *clustering problem*. In its general formulation clustering aims at finding, in the unlabeled dataset $X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$, K subsets of X (groups of examples) called *clusters* (or sometimes *classes*), such that data samples in the same clusters are "similar" to each other, and "dissimilar" to the ones in different clusters. Usually, the number K of clusters is chosen beforehand; we will come back to this non-trivial choice at the end of this section. The measure of similarity between points can equivalently be associated with a measure of distance $d(\boldsymbol{x}, \boldsymbol{y})$, where the similarity between points decreases with the distance between those points. Different cluster definition exists but the most common one is to represent C_k, the cluster k, by a single vector called the *centroid* $\boldsymbol{c}_k$. The cluster C_k is then defined as the set of examples that are closer to the centroid $\boldsymbol{c}_k$ than to any other centroid:

$$C_k = \{\boldsymbol{x}_i \mid \forall k' \neq k,\ d(\boldsymbol{x}_i, \boldsymbol{c}_k) \leq d(\boldsymbol{x}_i, \boldsymbol{c}_{k'})\}. \tag{2.1}$$

The most popular choice for the distance $d(\boldsymbol{x}, \boldsymbol{y})$ is the Euclidean or ℓ_2-distance $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2$. We can then define the K-means clustering problem[2], oftentimes refered to as "the" clustering problem: given a number of clusters K, the goal is to find the set of centroids C^* that are optimal in the sense that they minimize the Sum of Squared Errors (SSE), where the "errors" are the distances between the data samples and their respective (i.e. closest) centroid.

Definition 1. ***K-means clustering problem.*** *Given $X = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\} \subset \mathbb{R}^n$ a dataset and K a number of clusters, K-means clustering requires finding the optimal centroids $C^* = \{\boldsymbol{c}_1^*, ..., \boldsymbol{c}_K^*\} \subset \mathbb{R}^n$ satifying*

$$C^* = \arg \min_{C=\{\boldsymbol{c}_1,...,\boldsymbol{c}_K\}} \sum_{i=1}^{N} \min_k \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2 = \arg\min_C \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in C_k} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2. \tag{2.2}$$

As expressed in last expression, minimizing the SSE objective function (also called the potential) is equivalent to choosing a partition of the data that minimizes the sum of variances of the clusters multiplied by the cluster cardinality. Making this interpretation requires the centroids to be equal to the mean of their clusters, as it is always the case for an optimal solution of the centroids, hence the name *K-means clustering*. A visual example of clustering is given Figure 2.1.

2.1.3 Clustering algorithms

Solving the K-means clustering problem as stated in the form (2.2) exactly has been proven to be NP-hard [3] and is thus intractable. However, there exist several heuristics that try to solve (2.2) approximatively, while other algorithms rely on other formulations of the clustering problem. We present here some of the most commonly used clustering algorithms, that are also relevant to this work, with their advantages and drawbacks.

k-means and k-means++

The most popular clustering algorithm is the `k-means` algorithm, also named the Lloyd-Max algorithm after it was proposed by Lloyd [4]. The `k-means` algorithm, detailed as algorithm 1, is

[2]Although the particular clustering problem defined as (2.2) is called *K-means clustering*, it is important to avoid confusion with the `k-means` algorithm, one algorithm -amongst others- to solve the K-means clustering problem.

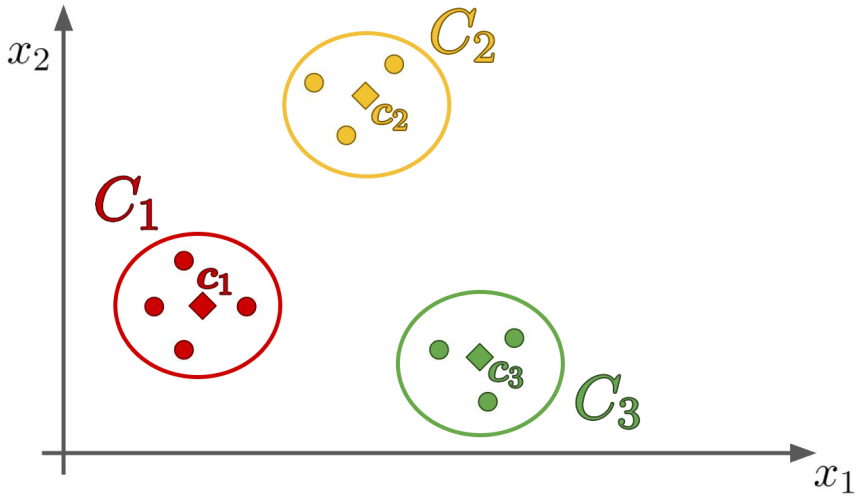


Figure 2.1: Example of clusters C_k of training examples (circles) defined by their centroids $\boldsymbol{c}_k$ (squares) in two dimensions.

a local search algorithm: starting from an initial set of centroids C, it improves C iteratively by alternatively i) assigning points to the clusters, then ii) moving the centroid to the mean of its cluster. The algorithm iterates until no further improvements can be made; since `k-means` decreases the objective function at each iteration, it is guaranteed to converge to a (local) minimum. As most local search algorithms, `k-means` is sensible to initialization: the initial set of centroids that is fed to `k-means` is a critical parameter, bad initialization can result in a local minima with unsatisfying centroids (i.e. whose SSE is much higher than the one obtained with C^*, because some clusters are not "discovered" at all for example).

Algorithm 1: `k-means` algorithm with generic initialization method `init`.

Input : A dataset $X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\} \subset \mathbb{R}^n$, a number of centroids K, an initialization strategy `init` *(e.g. `init = range` or `init = sample`)*

Output : A set of centroids C that (locally) minimizes the SSE (the potential)

Initialization $C = \{\boldsymbol{c}_1, ..., \boldsymbol{c}_K\} \leftarrow$ `init` ;
while *not stuck in local minima (stabilization)* **do**
  for k = *1,...,K* **do**
    $C_k \leftarrow \{\boldsymbol{x}_i \in X \mid \forall k' \neq k, \ \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2 \leq \|\boldsymbol{x}_i - \boldsymbol{c}_{k'}\|_2\}$ *(group examples in clusters)*;
  end
  for k = *1,...,K* **do**
    $\boldsymbol{c}_k \leftarrow \frac{1}{|C_k|} \sum_{\boldsymbol{x}_i \in C_k} \boldsymbol{x}_i$ *(replace centroid by its cluster mean)*;
  end
end

Different initialization strategies `init` exist for `k-means`, the two simplest being:

- `range`: selects the initial centroids uniformly at random in a box $\subset \mathbb{R}^n$ bounded by the minimum and maximum values taken by the training examples. In other words, $\boldsymbol{c}_i \overset{\text{iid}}{\sim} \mathcal{U}([\boldsymbol{l}, \boldsymbol{u}])$ with the (tight) lower and upper bound vectors $\boldsymbol{l} \leq \boldsymbol{x}_j \leq \boldsymbol{u}$, $\forall \boldsymbol{x}_j \in X$. Note that this strategy requires only access to the two n-dimensional vectors $\boldsymbol{l}$ and $\boldsymbol{u}$.
- `sample`: sample elements of X uniformly at random[3], i.e. choose $\boldsymbol{c}_i = \boldsymbol{x}_j \in X$ with

[3]To be completely rigorous, all initial centroids have to be different therefore sampling should be performed

probability $p_j = \frac{1}{N}$. Note that this strategy requires access to the entire dataset X.

Although those initializations methods are fast, they lack guarantees on the accuracy of the potential SSE_{kmeans} obtained by `k-means` with respect to the optimal potential SSE^* obtained by exact solution of (2.2), in short, $\frac{\text{SSE}_{kmeans}}{\text{SSE}^*}$ is unbounded. Therefore, the most widely used initialization strategy is the one used in `k-means++` [5], a version of `k-means` with a specific initialization, described as algorithm 2. In `k-means++`, the initial centroids are selected at random from the dataset like in `sample`, but this time with a different probability for each sample. This probability is increasing with the distance to the centroids already chosen (see the definition of p_i in algorithm 2). With this initialization scheme, the selected initial centroids are more often in different clusters, with as consequence an increased stability of the algorithm to initialization. Note that like `sample`, the `k-means++` initialization strategy requires access to the whole dataset X.

Remark. In the litterature as well as in this work, "`k-means++`" does sometimes refer to algorithm 2 (i.e. initialization *and* finding the centroids), and sometimes only to lines 1-6 (i.e. initialization phase only). On the other hand, oftentimes the name "`k-means`" is used instead of `k-means++`, implicitly assuming a particular initialization strategy. We will try to be as specific as possible but usually the meaning of those names should be either clear from the context or irrelevant.

Algorithm 2: `k-means++`: the `k-means` with a specific initialization `init`.

Input : A dataset $X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\} \subset \mathbb{R}^n$, a number of centroids K
Output : A set of centroids C that (locally) minimizes the SSE

1 *Initialization* $C = \emptyset$;
2 $\boldsymbol{c}_1 \leftarrow$ random sample from X *(first centroid at random, such as in* `sample`*)*;
3 **for** k = *2,...,*K **do**
4 $\quad D(\boldsymbol{x}_i) \leftarrow \min_{j \in \{1...k-1\}} \|\boldsymbol{x}_i - \boldsymbol{c}_j\|$ *(Define the distance to closest exisiting centroid)*;
5 $\quad \boldsymbol{c}_k \leftarrow \boldsymbol{x}_i$ with probability $p_i = \frac{D(\boldsymbol{x}_i)^2}{\sum_{i'} D(\boldsymbol{x}_{i'})^2}$ *(Promote isolated new centroids)*;
6 **end**
7 Run lines 2 to 9 of algorithm 1 *(Run the usual* `k-means` *algorithm)*;

As hinted at before, the main advantage of `k-means++` over `k-means` with simple initializations is that we have a guarantee on the mean[4] potential (SSE) obtained [5]:

$$\mathbb{E}[\text{SSE}_{kmeans++}] \leq 8(\ln K + 2)\text{SSE}^*, \tag{2.3}$$

with SSE^* the optimal value, at the cost of a slightly slower initialization of `k-means` (see the complexity analysis below). Due to its simplicity, `k-means`/`k-means++` has also the advantage of being easy to implement. It is worthwhile to mention that for its advantages, `k-means++` is the default initialization strategy of Matlab's `kmeans()` method.

The main disadvantage of `k-means` and (although to a lesser extent) `k-means++` is their instability with respect to the initialization. Even with `k-means++`, the resulting clusters can be very different accross multiple runs of the algorithms with different initial centroids. Also, the time complexity of `k-means` without taking into account the initialization is $\mathcal{O}(nNKI)$ where I is the number of iterations needed before convergence [6]. The number of iterations I is highly dependent on the initialization, but as illustration, it has be shown that `k-means` is superpolynomial in the worst case with $I_{worst} = 2^{\Omega(\sqrt{n})}$ when `range` or `sample` are used [7]. For `k-means++`, the runtime of initialization alone is $\mathcal{O}(nNK^2)$.

without replacement.

[4]The mean is taken over the initial centroids.

Remark. Usually, `k-means` is not considered to be *particularly* slow, since it is has a polynomial -and even linear- runtime in the sample size N, which is often considered acceptable. However we will later assume that N is *very* large (e.g. in the context of "big data"), making `k-meanss` runtime prohibitive given the *multiple passes* on the dataset it requires.

Another problem, related to the complexity, arises when the dataset X is too large to store in memory or, even worse, can be accessed only once. In the first case, chuncks of X will have to be -painfully slowly- loaded in memory one by one multiple times, and in the second case, `k-means` is not even applicable. As we shall see at the end of this chapter, compressive clustering does not suffer from those issues.

Spectral clustering

One of the disadvantages of `k-means` is that, since the cluster is represented by a single centroid, it does not account for the shape of the cluster and is therefore limited to simple cluster shapes. Another family of clustering methods that are more suitable to general cluster shapes are the *spectral clustering* algorithms that do not solve (2.2) but solve a different problem inspired by graph theory. The idea behind spectral clustering is to account for the particular geometry of the dataset by creating a graph $G = (V, E)$ in which a node $i \in V$ is associated with a training sample $\boldsymbol{x}_i \in X$ and an edge $e = (i, j) \in E$ conveys information about the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Possibly, weigths w_{ij} are associated to the edges (i, j), and -weighted or not- the adjacency amtrix of the graph is W. There are several ways to built the graph depending on the way the edges represent $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ the distance between their respective samples [8]:

- ϵ-neighborhood graph : there is an unweighted edge $e = (i, j)$ if $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq \epsilon$, where ϵ is a scale parameter that determines the distance between points in the same cluster and thus the size of clusters.
- k-nearest neighbor graph : there is an unweighted edge $e = (i, j)$ if $\boldsymbol{x}_i$ is one of the k nearest neighbors of $\boldsymbol{x}_j$ and/or[5] if $\boldsymbol{x}_j$ is one of the k nearest neighbors of $\boldsymbol{x}_i$, where k is a scale parameter that determines the size of clusters.
- fully connected graph : there is an weighted edge $e = (i, j)$ between each pair of nodes (i, j) with a weight $w_{ij} \geq 0$ measuring the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. This similarity should encode local neighbourhoods, an example being the Gaussian similarity function $w_{ij} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$, where σ is a scale parameter that determines how quickly similarity decays with distance and thus the size of the clusters.

Having defined the graph associated with the data, we need to define what clusters are in spectral clustering since we no longer rely on the definition based on centroids (2.1). As before, the clusters[6] are subsets of the examples, or in graph-theoretic terms, subgraphs of G, such that the *cut* between each subgraph and its complement (i.e. sum of the weigth of the edges leaving the subgraph) is as small as possible (this means that the sum of the weigths in the graph between the clusters are as small as possible).

Definition 2. ***Spectral clustering problem [8].*** *Given $G = (V, E)$ the graph associated with a dataset $X = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\} \subset \mathbb{R}^n$ a dataset and K a number of clusters, spectral*

[5] The nearest neighbor relation is not symetric, therefore choosing to add an edge if i **and** j are both one of the k nearest neighbors of the other (resulting in what is called the *mutual k-nearest neighbor graph*) is not the same as choosing to add an edge if i **or** j are one of the k nearest neighbors of one another (resulting in what is called simply "the" *k-nearest neighbor graph*).

[6] The clusters defined in this context are also known as *communities* in the field of graph theory.

clustering requires finding the optimal clustering $C_1^*, C_2^*, \cdots, C_K^* \subset V$ *satifying*

$$(C_1^*, C_2^*, \cdots, C_K^*) = \arg \min_{(C_1, C_2, \cdots, C_K)} \sum_{i=1}^{K} \frac{\text{cut}(C_i, V \setminus C_i)}{\alpha_i} \tag{2.4}$$

where the cut is defined as the sum of weighted edges between two sets of nodes

$$\text{cut}(A, B) = \sum_{\substack{i \in A \\ j \in B}} w_{ij}, \tag{2.5}$$

and α_i *is a measure of the size of* C_i *(to promote large clusters), for example* $\alpha_i = |C_i|$.

As in K-means clustering, solving the spectral clustering problem (2.4) exactly is NP-hard [9], but the spectral clustering algorithm solves it approximatively by relaxing it. The spectral clustering algorithm is called *spectral* because it uses the eigenvectors of the Laplacian matrix L of the graph G (built in one of the three ways described just above). The property spectral clustering is based upon is the fact that (assuming here an unweighted graph) given[7] one eigenvector $\boldsymbol{v}$ of L, the components v_i and v_j associated with the nodes i and j are equal *iff* i and j are in the same connected component of G. Moreover, in the ideal case where there are no edges between the desired clusters, finding clusters in X becomes equivalent to finding connected components in G. Based on this, if we build a feature vector associated to a data sample $\boldsymbol{x}_i$ by taking the i-th elements of the k first eigenvectors of L and compare it to the vector constructed in the same way for another point j, those feature vectors should be very similar when $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are in the same cluster. Then, it is possible to launch a traditional clustering such as `k-means++` on those feature vectors to retrieve the connected components hence the clusters.

Algorithm 3: The spectral clustering algorithm.

Input : W the adjacency matrix of the graph associated with the dataset $X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\} \subset \mathbb{R}^n$, a number of centroids K

Output : A set of centroids C that locally minimizes the SSE

1 $D \leftarrow$ diagonal matrix, $D_{ii} = \sum_j W_{ij}$ *(Build the degree matrix D)*;
2 $L = D - W$ *(Build L, the graph Laplacian)* ;
3 $\boldsymbol{v}_1, ..., \boldsymbol{v}_k = \text{eigv}(L)$ *(Compute the k eigenvectors of L associated with smallest eigenvalues)*;
4 $\boldsymbol{u}_i = [(\boldsymbol{v}_1)_i, (\boldsymbol{v}_2)_i, ..., (\boldsymbol{v}_k)_i]^T \in \mathbb{R}^k$;
5 *Run* `k-means++` *on the dataset* $U = \{\boldsymbol{u}_1, ..., \boldsymbol{u}_n\}$;

Note that with spectral clustering, it is still necessary to use an "external" clustering algorithm (usually `k-means++`): the "spectral" part of the algorithm can thus be seen as a pre-processing step on the dataset X, converting it in a new dataset U. The advantage of this pre-processing are that the whole geometry (the connected components or communities in general) of the dataset is embedded in the eigenvectors. On top of that the "new dataset" originating from the eigenvectors is guaranteed to be easily solved by `k-means++` since the shape of the clusters in this new representation of the data are much more easy than in the original space.

The main advantage of spectral clustering is its ability to represent complex cluster shapes, if the parameters of the graph (see next paragraph) are well-chosen. However, this comes at the cost of computing W, the adjacency matrix of the graph, which is $\mathcal{O}(N^2)$ given the number of entries of W which becomes prohibitive for large N.

[7] And assuming that $\boldsymbol{v}$ is not the first eigenvector, that trivially is $\mathbf{1}_N$.

Conclusion

The common feature of all clustering algorithm is that the user must define the *scale* of the clusters. This is done for the most part by choosing K, the number of clusters, and in the case of spectral clustering, the parameters of the graph associated with the data (ϵ, the similarity function...). Indeed, the quality of the clusters cannot be compared for different values of K for example, since the "best" clustering would be to pick $K = N$ and we would obtain an optimal SSE of 0 since all the clusters contain only one single training example. This task is often not trivial and require some insight about the clusters we would like to extract. As we will see, this aspect of clustering also naturally applies to compressive clustering, where we have to choose the number of clusters K, but also to provide a frequency distribution Λ that implicitly defines the way distance between the training examples are evaluated; this will be developped in Chapter 4.

Our goal will thus be to solve the clustering problem in a *compressive framework* where we do not have access to the whole dataset but a comrpessed representation of it. We first define the Compressed Sensing tools that we will use to compress this dataset.

2.2 The tools: Compressed Sensing and embeddings

2.2.1 Compressed Sensing

Remark. In this section, the notations we use are usual notations in Compressed Sensing, with no particular link with the notations in the rest of this work. For example, here, $f(t)$ does refer to a generic signal acquired by a sensor, and not to the signature function used for sketching data, ψ_j does refer to a waveform or atom and not to the characterstic function of a pdf, and so on. In all the sections except this one, we will be as consistent as possible.

Compressed Sensing: why? The limits of "uncompressed sensing"

Nearly all sensors that aim at capturing a signal work in the following way : the continuous target signal $f(t)$ is sampled, that is, aquired at different and equally spaced time instants $f[n] = f(t = n/f_s)$. The rate of acquisition f_s, called *sampling frequency* of the sensor, must satisfy the famous Shannon-Nyquist theorem: $f_s > 2f_{max}$ with f_{max} the maximal frequency present in the signal, in order to guarantee perfect reconstruction of the original signal from the samples.

Although this sensing mechanism (we could call it "uncompressed sensing") has been extensensively used, it is not necessarily optimal, in the sense that it produces far more samples than necessary. Why? As a first clue, it is good to keep in mind that the Shannon-Nyquist sampling theorem is a *sufficient condition* for perfect recovery from the samples, but not a necessary one. The second observation, linked to the first, is that in the large majority of "useful signals" (signals that convey information), the "information rate" is much lower than the "signal rate", i.e. f_{max}.

Example 1. Consider a simple communication by Frequency-Shift Keying (FSK[8]): the transmitter sends a bitstream of information bits b_m by sending over a channel $f(t) = \cos((2\pi f_0 + b_m 2\pi\Delta f)t)$ during the time interval $t = [mT, (m+1)T]$. In realistic applications, the "information rate", $1/T$, is much smaller than the maximal frequency in the signal sent[9] $f_{max} \simeq f_0 + \Delta f$. A traditional sensor would need to acquire $2f_{max} >> 1/T$ samples per second to produce $1/T$ "information

[8] Actual FSK demodulation does not work in this way, but the point of the example is still valid.

[9] To be perfectly rigourous, one should compute the power spectral density of $f(t)$ since the bits can be considered random, but we are making approximations for this example.

samples" per second.

Although we made observations for continuous-time signals, the same applies to finite-length discrete-time (or -space) signals such as images. Those signals are vectors $\boldsymbol{x} \in \mathbb{R}^n$, and sensors must acquire those full n samples, while in the end (after compression or other signal processing tasks) only a few information bits are kept. The theory of Compressed Sensing (CS) thus aims at sensing signals with a number of samples that are directly related to the "information rate" in the signal [10].

A few words about sparsity

Before diving into CS, it is necessary to formalize the "information rate" (or "information samples") through the notion of *sparsity* : indeed, the sparsity of a signal is a measure of the information content in this signal, usually much lower than its length.

Definition 3. ***Sparsity.***
The signal $\boldsymbol{\alpha} \in \mathbb{R}^d$ *is* ***K-sparse*** *if*[a]

$$\|\boldsymbol{\alpha}\|_0 := |\text{supp}(\boldsymbol{\alpha})| = |\{i \mid \alpha_i \neq 0\}| \leq K, \tag{2.6}$$

meaning that it has at most K *non-zero entries.*
In a broader sense, for some basis[b] $\Psi = [\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, ..., \boldsymbol{\psi}_d] \in \mathbb{R}^{n \times d}$*, a signal* $\boldsymbol{x} = \Psi\boldsymbol{\alpha}$ *is also called sparse in* Ψ *if* $\boldsymbol{\alpha}$ *is (K-)sparse, i.e. the coefficients of* $\boldsymbol{x}$ *in the basis* Ψ *have few non-zero entries and* $\boldsymbol{x}$ *is thus described by the combination of only a few basis elements* $\boldsymbol{\psi}_j$*.*

[a] Note that $\|\cdot\|_0$ not a true norm as its notation suggests.
[b] e.g. the Fourier basis, the wavelet basis, but also redundant dictionnaries and other transforms.

Sometimes, the signal $\boldsymbol{x}$ is not strictly sparse expressed in some basis Ψ but the associated coefficients $\boldsymbol{\alpha}$ have only few entries that are *significantly* different from zero, and most coefficients can be discarded without perceptual loss of information (when ordered, the magnitude of the coeffients decay quicly, e.g. their decrease follows a power law). The signal $\boldsymbol{x}$ is then called *compressible* meaning that it is well approximated by the sparse signal $\boldsymbol{x}^K = \Psi\boldsymbol{\alpha}^K$ where $\boldsymbol{\alpha}^K$ keeps the K largest coefficients of $\boldsymbol{\alpha}$ and sets the others to zero.

The core of CS : sparse signal reconstruction from few linear measurements

We now assume that we want to acquire a signal $\boldsymbol{x} = \Psi\boldsymbol{\alpha}$ that is K-sparse in a basis Ψ. In the Compressed Sensing framework, $\boldsymbol{x} \in \mathbb{R}^n$ is acquired through a few linear measurements $y_i = \langle \boldsymbol{x}, \boldsymbol{\phi}_i \rangle$. Those m measurements are assembled in an observation vector

$$\boldsymbol{y} = \Phi\boldsymbol{x} = \Phi\Psi\boldsymbol{\alpha} \tag{2.7}$$

where $\Phi = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_m]^T \in \mathbb{R}^{m \times n}$ contains the waveforms ("sensing vectors") $\boldsymbol{\phi}_i$ that are correlated with $\boldsymbol{x}$ at the sensing stage. The goal is then to reconstruct $\boldsymbol{x}$ from the $m \ll n$ measurements in $\boldsymbol{y}$. Solving (2.7) for $\boldsymbol{x}$ directly is problematic since it is an undetermined linear system with infinitely many solutions. However, knowing that (by assumption!) $\boldsymbol{x}$ must be sparse in Ψ, the Compressed Sensing reconstruction idea is then to select the sparsest vector $\boldsymbol{x}$ amongst this infinite set of solution $\{\boldsymbol{x} \mid \boldsymbol{y} = \Phi\boldsymbol{x}\}$. More precisely, signal reconstruction from compressive measurements is achieved by solving

$$\boldsymbol{x}^* = \Psi\boldsymbol{\alpha}^* \quad \text{with} \quad \boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_0 \quad \text{s.t.} \quad \boldsymbol{y} = \Phi\Psi\boldsymbol{\alpha}. \tag{2.8}$$

Constructing the sensing matrix and the RIP property

Before discussing how this optimization problem can be solved in practice, it is worth noting that the solution to (2.8) will yield a good reconstruction only if the sensing matrix Φ is well-chosen. First, let us define the Restricted Isometry Property (RIP) of a matrix A:

Definition 4. ***Restricted Isometry Property:***
The matrix A satisfies the RIP with isometry constand δ_K if

$$(1-\delta)\|\boldsymbol{x}\|_2^2 \leq \|A\boldsymbol{x}\|_2^2 \leq (1+\delta)\|\boldsymbol{x}\|_2^2 \quad \forall \boldsymbol{x} \in \Sigma_K := \{\boldsymbol{x} |\ \|\boldsymbol{x}\|_0 \leq K\} \tag{2.9}$$

where Σ_K stands for the set of K-sparse signals. In words, if A satisfies the RIP with a small constant δ, A approximatively preserves the Euclidian norm of sparse vectors.

The notion of sparsity enables to obtain guarantees on $\boldsymbol{x}^*$ the reconstruction obtained through (2.8)[10]. In particular, we can obtain a guarantee on the reconstruction error with respect to $\boldsymbol{x}$ the true original signal, well approximated by a K-sparse signal $\boldsymbol{x}^K$ that keeps the K largest entries of $\boldsymbol{x}$, in the form of the following theorem [11]:

Theorem 1. *If Φ satisfies the RIP with $\delta_{2K} < 1$ and $\boldsymbol{x}$ is K-sparse, then $\boldsymbol{x}^*$ the solution to (2.8) with $\Psi = I$ is exact, i.e. $\boldsymbol{x}^* = \boldsymbol{x}$.*

Proof. First observe that $\boldsymbol{x}^*$ must be K-sparse: if it has strictly more than K non-zero coefficients, it can't be the optimal solution to (2.8) since there exists at least one strictly better solution, that is, $\boldsymbol{x}$ itself. Since $\boldsymbol{x}$ and $\boldsymbol{x}^*$ are K-sparse, $\boldsymbol{x} - \boldsymbol{x}^*$ is at most 2K-sparse (if their supports are disjoint). Then, because Φ satisfies the RIP for 2K-sparse vectors, we can apply the property to $\boldsymbol{x} - \boldsymbol{x}^*$ and write

$$(1-\delta_{2K})\|\boldsymbol{x}-\boldsymbol{x}^*\|_2^2 \leq \|\Phi(\boldsymbol{x}-\boldsymbol{x}^*)\|_2^2 \leq (1+\delta_{2K})\|\boldsymbol{x}-\boldsymbol{x}^*\|_2^2 \tag{2.10}$$

and since $\boldsymbol{y} = \Phi\boldsymbol{x}$ by definition of $\boldsymbol{y}$ and $\boldsymbol{y} = \Phi\boldsymbol{x}^*$ by contraints on $\boldsymbol{x}^*$, this is equivalent to

$$\frac{1}{1+\delta_{2K}} \underbrace{\|\boldsymbol{y}-\boldsymbol{y}\|_2^2}_{0} \leq \|\boldsymbol{x}-\boldsymbol{x}^*\|_2^2 \leq \frac{1}{1-\delta_{2K}} \underbrace{\|\boldsymbol{y}-\boldsymbol{y}\|_2^2}_{0}, \tag{2.11}$$

meaning that $\|\boldsymbol{x}-\boldsymbol{x}^*\|_2^2 = 0$, therefore $\boldsymbol{x}^* = \boldsymbol{x}$. □

It is thus possible to reconstruct sparse signals if the sensing matrix satisfies the RIP property. But how can we ensure in practice that the matrix Φ in the sensor we build satisfies the RIP, with the smallest amount of rows (the amount of measurements of the signal, m) possible? It turns out that deterministic contructions of RIP matrices require large amouts of measurements m [12], but surprisingly, *random matrices* allow to obtain RIP matrices with small amounts of m, with high probability. For example, if the elements of Φ are drawn from a Gaussian distribution $\Phi_{i,j} \overset{\text{iid}}{\sim} \mathcal{N}(0, 1/m)$, it turns out that if the number of rows satisfies $m \geq cK\log(n/K)$ for some constant c, Φ satisfies the RIP with very high probability [12]. The key observation here is that the number of measurements m (the rate of the sensor) scales almost linearly (up to logarithmic factors) with the sparsity of the measured signal K.

Before moving on to the reconstruction algorithms, we conclude this section about sensing with the fact that there exist some practical sensor designs that capture signals through random matrices, such as the single-pixel camera [13] or the CMOS compressed imager by random convolution [14]. Acquiring the compressed signals without ever recording the whole signal is thus practically feasible with known hardware components.

[10] For this result we focus wlog on signals that are sparse in the direct domain, i.e. we assume that $\Psi = I$.

OMP and OMPR algorithms

Sadly, once the compressed signal $\boldsymbol{y}$ has been acquired, solving (2.8) exactly turns out to be NP-hard, making it intractable in practice [15]. To make up for this problem, two general approaches have been pursued: *relaxation* of the original problem, and *iterative, greedy algorithms* that solve it approximatively.

One relaxation approach is Basic Pursuit (BP, [16]), that replaces the number of nonzero coefficients $\|\boldsymbol{\alpha}\|_0$ in (2.8) by the ℓ_1-norm $\|\boldsymbol{\alpha}\|_1$, which transforms it into a convex problem that can be solved exaclty. The motivation for this approach lies in the fact that minima to the ℓ_1-norm in this problem will almost always lie on the edges of the ℓ_1-sphere, thus indeed promoting sparser solutions as was the objective. In fact for this relaxed problem we obtain a new guarantee that replaces Theorem 1: if $\Phi\Psi$ satisfies the RIP with $\delta_{2K} \leq \sqrt{2} - 1$, then $\boldsymbol{x}^*$ the solution to the relaxed version of (2.8) satisfies

$$\|\boldsymbol{x}^* - \boldsymbol{x}\|_2 \leq C_0 \|\boldsymbol{x} - \boldsymbol{x}^K\|_1 / \sqrt{K} \tag{2.12}$$

for some constant C_0, and $\boldsymbol{x}^K$ the best K-sparse approximation of $\boldsymbol{x}$. In particular, if $\boldsymbol{x}$ is exactly K-sparse, $\boldsymbol{x}^* = \boldsymbol{x}$ (the reconstruction of the relaxed problem is exact) [17].

Matching Pursuit (MP, [18]) and its variants Orthogonal Matching Pursuit (OMP, [19]) and Orthogonal Matching Pursuit with Replacement (OMPR, [20]) are iterative greedy algorithm to reconstruct $\boldsymbol{x}$, based on a different formulation of the reconstruction problem. Indeed, they solve approximatively[11]

$$\boldsymbol{x}^* = \Psi\boldsymbol{\alpha}^* \quad \text{with} \quad \boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\boldsymbol{y} - A\boldsymbol{\alpha}\|_2^2 \quad \text{s.t. } \|\boldsymbol{\alpha}\|_0 \leq K. \tag{2.13}$$

where instead of minimizing the sparsity K in the reconstructed signal, the sparsity is imposed *a priori* as a constraint on the signal, and we thus search for the K-sparse signal that *matches* (hence the name) the observation vector $\boldsymbol{y}$ as best as possible. In other words, this formulation approximates $\boldsymbol{y}$ as best as possible as a linear combination of K columns of A. The idea is to built the support Γ of the reconstructed vector (i.e. the set of columns of A, also called *atoms*, that are taken in the K-term linear combination) iteratively by adding at each iteration the index that reduces the most the *residual* $\boldsymbol{r}$ and stops when the desired sparsity level K is atteined. The residual $\boldsymbol{r}$ measures the mismatch (distance) between the observation vector $\boldsymbol{y}$ and the one that would be observed with the current best reconstruction given the currently chosen support Γ. The `OMPR` algorithm is described as algorithm 4, in the same style as in [2].

Note that if the number of iterations of algorithm 4 is $T = K$, step 3 never occurs and algorithm 4 reduces to the `OMP` algorithm. In `OMPR`, a typical choice of iterations is $T = 2K$. As mentioned above, `OMP(R)` is a greedy algorithm: it does not "look back" (too much) on the greedy choices it makes, the indexes of the support. It is therefore not guaranteed to find the optimal solution, although the replacement step in `OMPR` when $t > K$ allows to eventually climb out of a local optimum, but does still not guarantee that *the* optimal solution will be found. The `OMPR` algorithm will be the inspiration of the main algorithm in compressive clustering, namely `CLOMPR`.

2.2.2 Quantized embeddings

In this subsection, 1-bit (universal) quantizion embeddings are reviewed, as they are the main inspiration for the modifications we propose on the existing `CKM` algorithm that we will discuss

[11] We introduce the notation $A = \Phi\Psi$.

Algorithm 4: `OMPR` algorithm for reconstruction of a sparse vector $\boldsymbol{\alpha}'$ from measurements $\boldsymbol{y} = A\boldsymbol{\alpha}$ by greedy construction of the support Γ (finds a local optimum for (2.13))

Input : measurement matrix $A = [\boldsymbol{a}_1, ..., \boldsymbol{a}_d]$ with normalized columns $\boldsymbol{a}_i$, an observation vector $\boldsymbol{y} = A\boldsymbol{\alpha}$ ($\boldsymbol{\alpha}$ is unkown), a desired sparsity K, a number of iterations $T \geq K$.

Output : α', a greedy (K-sparse) locally optimal solution to (2.13).

Initialization $\boldsymbol{r} \leftarrow \boldsymbol{y}$, $\Gamma \leftarrow \emptyset$ *(Initialize the residual and the support set)* ;
for *t = 1,...,T* **do**
  Step 1 : find the column of A with highest correlation with residual:
  $i^* = \arg\max_{i \notin \Gamma} |\langle \boldsymbol{a}_i, \boldsymbol{r} \rangle|$
  Step 2 : add it to the support:
  $\Gamma = \Gamma \cup \{i^*\}$
  Step 3 : Reduce support by Hard Thresholding to keep sparsity ("Replacement" step):
  if $|\Gamma| > K$ **then**
    $\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{|\Gamma|}} \|\boldsymbol{y} - A_\Gamma \boldsymbol{\beta}\|_2$ *(A_Γ are the columns of A indexed by Γ)*
    $\Gamma \leftarrow$ set of K indexes corresponding to K largest magnitude values of $\boldsymbol{\beta}$
  end
  Step 4 : Find the optimal coefficients given the support (by projection):
  $\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{|\Gamma|}} \|\boldsymbol{y} - A_\Gamma \boldsymbol{\beta}\|_2$
  $\boldsymbol{r} = \boldsymbol{y} - A_\Gamma \boldsymbol{\beta}$ *(Update residual, go back to step 1)*
end
$\boldsymbol{\alpha}' = \mathbf{0}_d$
$\boldsymbol{\alpha}'_\Gamma \leftarrow \boldsymbol{\beta}$ *($\boldsymbol{\alpha}'$ is K-sparse with support Γ, and takes coefficent values $\boldsymbol{\beta}$)*

later. We begin by a general definition: an embedding is a mapping of signals from a high-dimensional space to a lower-dimensional one that preserves the distances. More formally, [21] defines an embedding as:

Definition 5. ***Embedding:*** *The function $f : \mathcal{S} \mapsto \mathcal{W}$ is a (g, δ, ϵ) embedding of the signal set $\mathcal{S}$ into the set $\mathcal{W}$ if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}$:*

$$(1 - \delta) g\left(d_{\mathcal{S}}(\boldsymbol{x}, \boldsymbol{y})\right) - \epsilon \leq d_{\mathcal{W}}(f(\boldsymbol{x}), f(\boldsymbol{y})) \leq (1 + \delta) g\left(d_{\mathcal{S}}(\boldsymbol{x}, \boldsymbol{y})\right) + \epsilon \tag{2.14}$$

where $g : \mathbb{R} \mapsto \mathbb{R}$ is an invertible distance map.

Thus, a (g, δ, ϵ) embedding maps (or embeds) vectors of $\mathcal{S}$ to $\mathcal{W}$ (typically a space of lower dimension than $\mathcal{S}$) such that the distances of those vectors in $\mathcal{W}$ are equal to a mapping g of their distances in $\mathcal{S}$, with a multiplicative error δ and an additive error ϵ.

Example 2. If $A \in \mathbb{R}^{m \times n}$ satisfies the RIP with constant δ, by definition $f(\boldsymbol{x}) = A\boldsymbol{x}$ is a $(I, \delta, 0)$ embedding (I is the identity function) of $\mathcal{S} = \Sigma_K$ into $\mathcal{W} = \mathbb{R}^m$ and with $d_{\mathcal{S}}$ and $d_{\mathcal{W}}$ being the Euclidean or ℓ_2-distance.

Although $f(\boldsymbol{x}) = A\boldsymbol{x}$ is an embedding that allows *dimensionality reduction*, to obtain a true compression or *rate reduction* and to be stored on a digital system, the low-dimensional signals must be quantized, e.g. with a uniform quantizer. In this case, the quantization introduces an additive error ϵ that depends on the number of bits assigned per dimension. The extreme case of 1-bit CS, where only the sign of the random measurements are recorded $f(\boldsymbol{x}) = \text{sign}(A\boldsymbol{x})$, the signal amplitude is lost but the embedding preserves (with an additive error) the angle between vectors in the input space [22].

1-bit universal quantization embeddings

Besides the 1-bit quantization embedding that preserves angles, another 1-bit embedding has been proposed, that preserves the distances up to a given radius [21], [23]. The difference here is that the random measurements $A\boldsymbol{x}$ are quantized not with a uniform (scalar) quantizer $\bar{q}_\Delta(t) = \lfloor t/\Delta \rfloor$ with quantization step Δ (Figure 2.2, left), but with a *universal quantizer* (Figure 2.2, right). The universal quantizer $q_\Delta(t)$ can be seen as the Least Significant Bit of the uniform quantizer, thus $q_\Delta(t) = \bar{q}_\Delta(t) \bmod 2$.

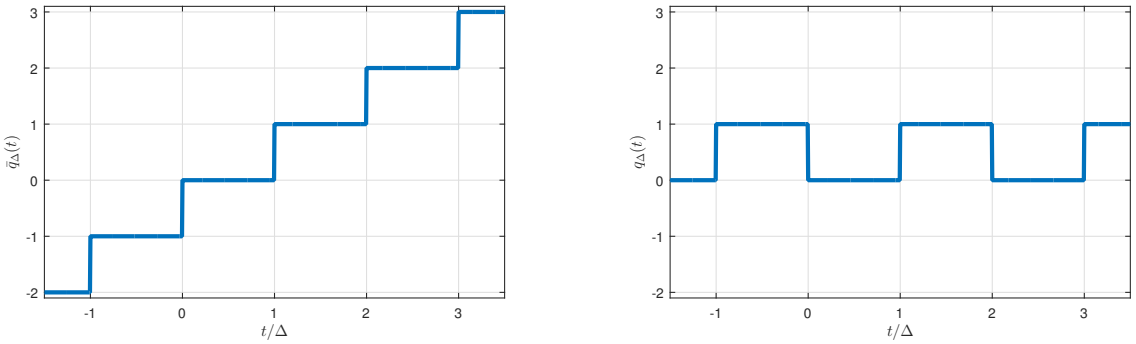


Figure 2.2: Comparison of the usual uniform quantizer $\bar{q}_\Delta(t)$ (left) and the universal quantizer $q_\Delta(t)$ (right) as a function of t/Δ.

Remark. In order to simplify the mathematical expressions that we encounter later, we define the *centered universal quantization* function $Q_\Delta(t)$ with quantization step Δ that corresponds to

$$Q_\Delta(t) = 2q_\Delta(t) - 1 = \begin{cases} -1 & \text{if } t \in [2k\Delta, (2k+1)\Delta[\text{ for some } k \in \mathbb{Z} \\ +1 & \text{otherwise.} \end{cases}, \tag{2.15}$$

or more shortly, $Q_\Delta(t)$ maps to $\{-1, +1\}$ whereas $q_\Delta(t)$ maps to $\{0, 1\}$. In the following sections, we will find it useful to re-formulate this function as an infinite linear combination of Heaviside step functions $u(t)$:

$$Q_\Delta(t) = 2\sum_{k\in\mathbb{Z}} (-1)^k u(t - k\Delta). \tag{2.16}$$

Since later on we will almost exclusively work with $Q_\Delta(t)$ instead of $q_\Delta(t)$, we will usually refer to the centered universal quantization simply as "the universal quantization" for convenience.

It turns out that combined with a random dithering $\boldsymbol{\xi}$ (a shift on the random projections), the quantized embedding using the universal quantizer preserves the *local* distances, as stated by Theorem 2 [23].

Theorem 2. ***Embedding by universal quantization.***
Let A be an $\mathbb{R}^{m\times n}$ matrix whose entries are $A_{ij} \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$, $\boldsymbol{\xi}$ a random dither with uniformly distributed entries $\xi_j \overset{\text{iid}}{\sim} \mathcal{U}([0, \Delta])$. Then, $f(\boldsymbol{x}) = q_\Delta(A\boldsymbol{x} + \boldsymbol{\xi})$ is a $(g, \tau, 0)$ embedding of the space $\mathbb{R}^n$ (with the Euclidian distance $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2$ as associated distance) into $\{0, 1\}^m$ (with the Hamming distance $d_H(\boldsymbol{y}_1, \boldsymbol{y}_2)$ as associated distance) with very high probability on the choice of A and $\boldsymbol{\xi}$. This means that

$$g(\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2) - \tau \leq d_H(f(\boldsymbol{x}_1), f(\boldsymbol{x}_2)) \leq g(\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2) + \tau \tag{2.17}$$

where we have $\tau \propto 1/\sqrt{m}$ and the distance map $g(d)$

$$g(d) = \frac{1}{2} - \sum_{i=0}^{+\infty} \frac{e^{-\left(\frac{\pi(2i+1)\sigma d}{\sqrt{2}\Delta}\right)^2}}{(\pi(i+1/2))^2} \tag{2.18}$$

which is represented graphically Figure 2.3.

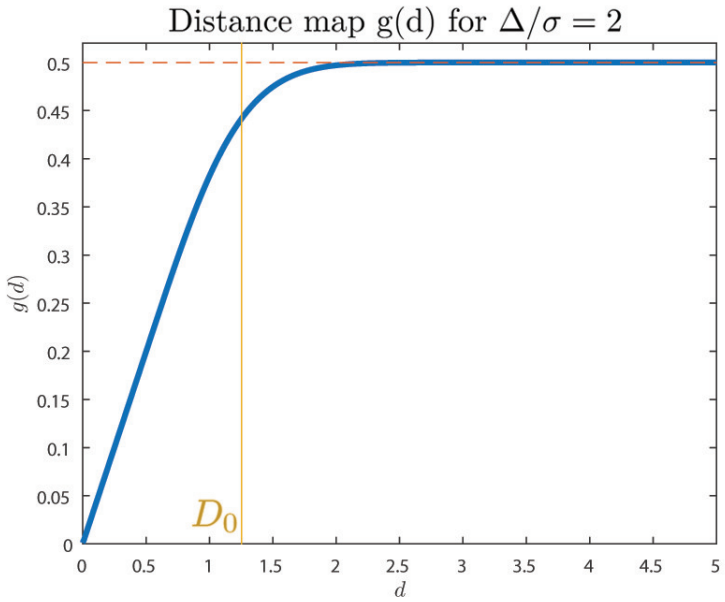


Figure 2.3: The function $g(d)$ (in blue), that represents the mapping of distances in $\mathbb{R}^n$ into $\{0,1\}^m$ achieved by an embedding $f(\boldsymbol{x}) = q_\Delta(A\boldsymbol{x} + \boldsymbol{\xi})$, for $\Delta/\sigma = 2$. On the left of the yellow separation, $d < D_0$ and the map is approximatively linear (it is a good representation of the original distance d). On the right, $d > D_0$ and all distances map to 0.5 and it becomes almost impossible to inverse the relation accurately (to obtain a good estimation of d from the embedded distance $g(d)$).

The interesting property of the universal quantization embedding $f(\boldsymbol{x}) = q_\Delta(A\boldsymbol{x} + \boldsymbol{\xi})$ is the fact that it preserves the Euclidean distance of signals in the input space *up to a critical distance threshold* D_0. In this range $d < D_0$, the distance map $g(d)$ is almost linear (see Figure 2.3). If the distance of the signals d is greater than D_0, the map value is approximatively $\frac{1}{2}$ wathever the value of d, and the embedded distance contains no information besides the fact that the original sigals were separated by a distance greater than D_0. The threshold distance D_0 depends on the ratio $\frac{\Delta}{\sigma}$, since

$$D_0 = \sqrt{\frac{\pi}{8}}\frac{\Delta}{\sigma}. \tag{2.19}$$

Increasing $\frac{\Delta}{\sigma}$ will thus increase the "range of sight" of the embedding f. But because of the additive error τ, at a fixed number of measurements m, the ambiguity on the reconstructed distance increases with $\frac{\Delta}{\sigma}$. More precisely, the ambiguity, i.e. the maximal difference between $d_{\mathcal{S}}$ the true distance in the input space $\mathcal{S}$ and $\hat{d}_{\mathcal{S}}$ the reconstructed distance using the compressive measurements, is bounded by

$$|d_{\mathcal{S}} - \hat{d}_{\mathcal{S}}| \;\leq\; c\frac{\tau}{g'(\hat{d}_{\mathcal{S}})} \;\simeq\; c\frac{\tau}{\sqrt{2/\pi}}\frac{\Delta}{\sigma} \quad \text{for } d \leq D_0 \tag{2.20}$$

for some constant c [23]. A trade-off for chosing the value $\frac{\Delta}{\sigma}$, illustrated Figure 2.4, thus appears: $\frac{\Delta}{\sigma}$ should be large enough to represent correcltly the range of distances of interest, but as small

as possible to reduce the ambiguity.

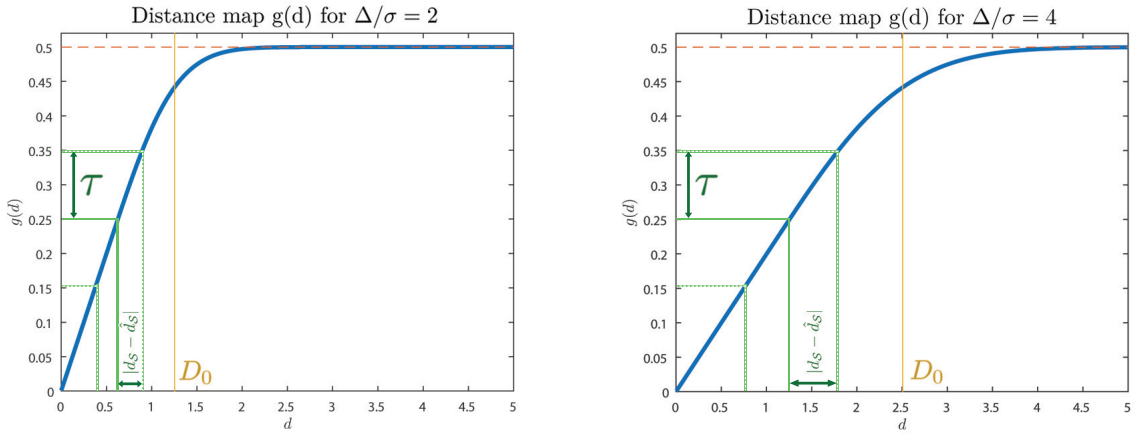


Figure 2.4: Comparison of the distance map $g(d)$ induced by a universal quantization embedding. The range of distances D_0 captured increases with Δ/σ : we have $D_0 \simeq 1.25$ for $\Delta/\sigma = 2$ (left) but $D_0 \simeq 2.5$ for $\Delta/\sigma = 4$ (right). However, for a constant additive error τ (for this example $\tau = 0.1$) the ambiguity $|d_{\mathcal{S}} - \hat{d}_{\mathcal{S}}|$ increases with Δ/σ : we have $|d_{\mathcal{S}} - \hat{d}_{\mathcal{S}}| \simeq 0.25$ for $\Delta/\sigma = 2$ (left) but $|d_{\mathcal{S}} - \hat{d}_{\mathcal{S}}| \simeq 0.5$ for $\Delta/\sigma = 4$ (right).

The advantage of universal embeddings if that if $\frac{\Delta}{\sigma}$ is chosen properly (assuming thus that in the application of interest, it is only necessary to represent distances accurately up to a certain radius), they are efficient in the number of measurements m (here also equal to the number of bits) required to represent the signals with a certain ambiguity since no bits are "wasted" to encode distance ranges that are not necessary given the target application. Intuitively, the whole bit budget is spent to represent only the distances of interest (up to D_0) with the best accuracy possible.

Remark. In clustering applications, the information that we want to encode about pairs of high-dimensional signals is "those signals belong to the same cluster since they are similar" or "those signals belong to different clusters since they are dissimilar". This can be translated, in the view of 1-bit universal embeddings, as, "those signals are in the same cluster since their distance is less than D_0" or "those signals are in different clusters since thetheir distance is larger than D_0". This suggests that it should be possible to build the high-dimensional clusters from 1-bit universal quantization embeddings of signals. As we will see in the next section, this is not exactly what we will do since we will consider an additional *pooling of the data* step in our compressive clustering framework (to obtain memory and storage requirements independent of the amount of learning examples for example) but this encouraging fact will nonethless be a motivation for the quantized sketch we propose in Chapter 3.

2.3 The procedure: compressive clustering by sketching

In this section we present compressive clustering by sketching, a clustering strategy based on Compressed Sensing ideas, the method at the very center of this thesis. Before diving into compressive clustering, we present the general idea of compressive signal processing, i.e., processing the compressed signals instead of the "full-length" signals.

2.3.1 Working in the compressed domain and compressed learning

As discussed above, the goal of Compressed Sensing is usually to *reconstruct* a sparse signal from a potentially very small set of linear -usually random- projections. However, in many applications the goal is not to extract the signal itself but *some information* about the signal by running some signal processing task on the signal. Typically, this information is the presence/absence of a signal at all in a noisy environment (detection problem), a class label for the signal (classification problem), or more generally some function of the signal (estimation problem), and even filtering out some interfering signal (filtering problem). Thus, spending a lot of computational ressources to reconstruct the whole signal in order to extract a small amount of information seems to be a waste. In fact, it is possible to design algorithm to solve those problems (detection, classification...) directly on the compressed signals, without ever reconstructing the high-dimensional sparse signal [24]. Those signal processing methods are thus working in the compressed domain without reconstructing the sparse signals; this idea is illustrated Figure 2.5.

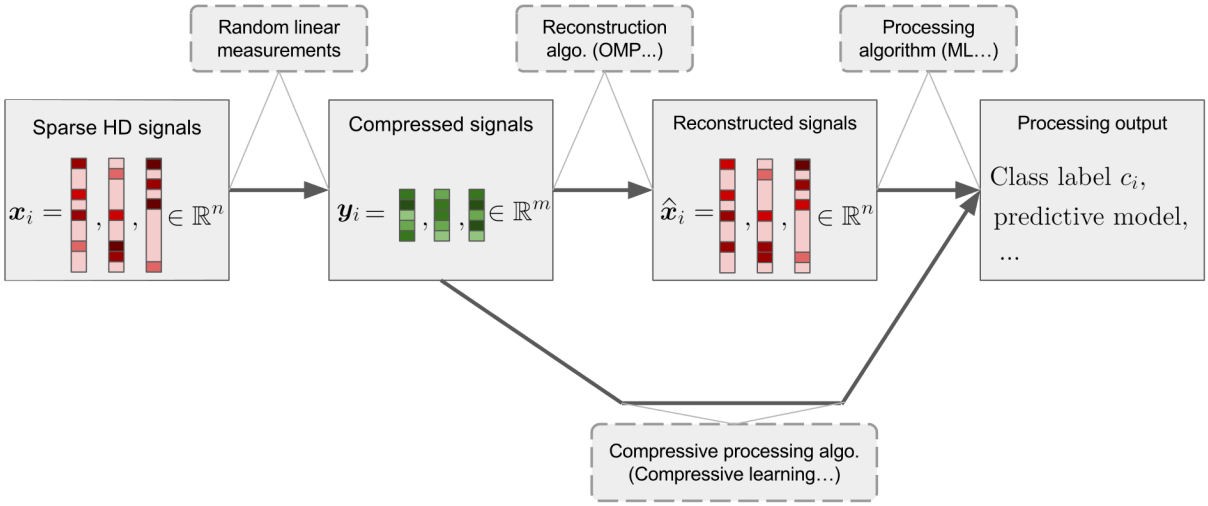


Figure 2.5: Schematic view of signal processing steps when Compressed Sensing is used to sample sparse vectors. The upper branch represents the usual method, since traditional CS theory focuses only on reconstruction of signal regardless of their subsequent use in practical applications. The lower branch corresponds to processing directly in the compressed domain. This alternative approach ("compressive processing algorithm") can require far less samples m than what is needed for an accurate reconstruction of the whole signal ("reconstruction algorithm").

More importantly, for the case of detection for example (but it is also true for the other problems we presented), the required number of samples to obtain a reliable detection (with a small probability of error) is significantly smaller than what is needed for reconstructing the whole signal accurately [25]. Intuitively, the fact that signal reconstruction produces a richer output than, e.g., signal detection (one single bit of information) explains why more measurements are needed in the case of full reconstruction.

Since it is possible to perform signal processing tasks in the compressed domain for a single signal (i.e. *compressed processing*), it seems straighforward to extend this approach to perform machine learning in the compressed domain for a (data)set of signals (i.e. *compressed or compressive learning*).

There are two main ways to perform compressive learning on a dataset X, as summarized Figure 2.6. The first one, on the top right in Figure 2.6 (an extension of Figure 2.5 to multiple

signals instead of a single one), is rather straightforward: the training data examples are compressed one by one independently, forming a compressed dataset Y with as many elements as in the original dataset, but with a smaller dimension. The other approach, bottom left in Figure 2.6, is to compress the *whole dataset* at once, into one single compressed object $\boldsymbol{z}$ (that will be called a sketch in the next section) representing the information associated with all the elements of the original dataset. As we will see in the next section, compressive clustering is a compressed learning algorithm that uses this second option.

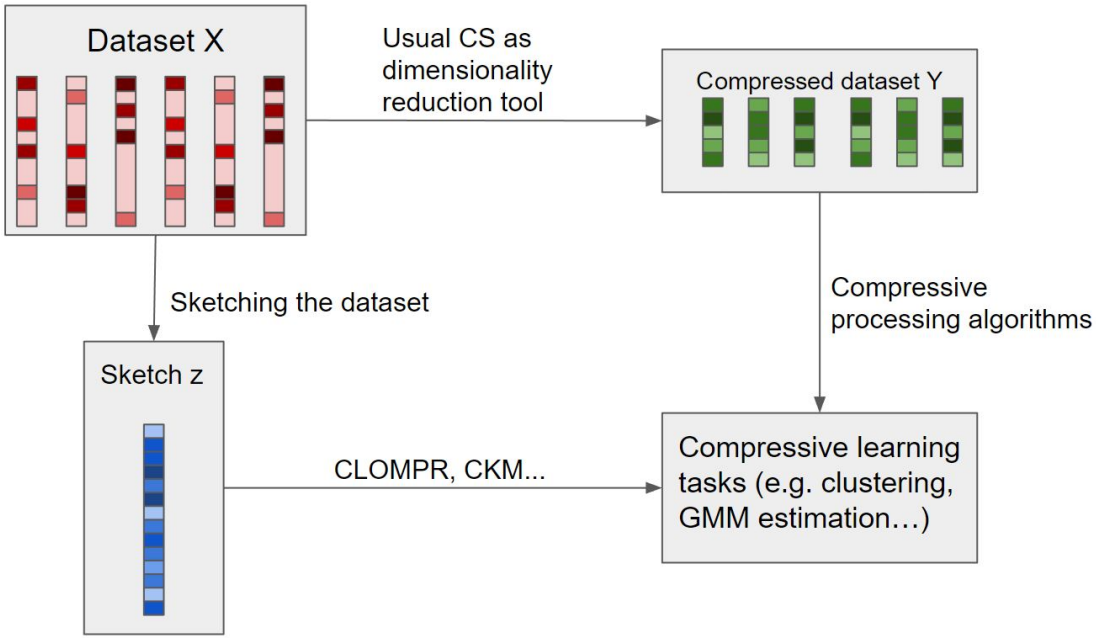


Figure 2.6: The two ways compress a dataset for learning. In the top branch, the training examples are compressed individually through random projections, and then fed to a compressive learning algorithm. The bottom branch is the method we focus on, where the training examples are projected randomly, then a **pooling** (sum of the elements) is applied to create a unique vector, the sketch, to represent the whole dataset. Image inspired by [2], Figure 1.

2.3.2 Clustering by sketching: from sketching pdfs to a clustering algorithm

In this subsection, the core concepts of this thesis will be presented. We will start from the abstract definition of the sketch of a probability density function, and then see how we can particularize this definition to the sketch of a dataset, and how it is used to solve compressive learning problems such as compressive clustering in particular. Before presenting the theoretical concepts of CKM and discussing its different interpretations, we give in Figure 2.7 briefly an overview of how the algorithm works in practice. A small subset of the dataset is used to design a sketching operator adapted to the dataset, then this operator is used to compute the sketch of the dataset. Clustering is achieved by accessing only this sketch.

Sketch of a probability density function and a few words on mixture models

The idea of *sketching* whole a dataset X, summarizing it into a single vector (the sketch) has been proposed in [2] for compressive estimation of Gaussian Mixture Models (GMM), and has been extended in [1] for performing Compressive K-Means clustering (a method called CKM and that is actually the main inspiration for this work). Although those two goals may seem different, they can be interpreted as two particular cases of a more general problem where we are trying to match two *probability density functions* (pdfs) $\mathcal{P}$ and $\mathcal{Q}$ in a compressive manner, using

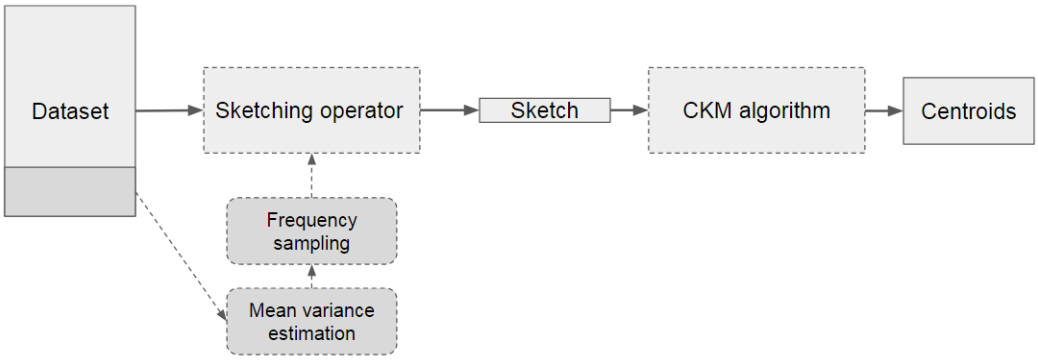


Figure 2.7: The practical stages of centroid retrieval with the CKM algorithm. First, a small sample of the dataset is acquired (dark grey portion) and is used to estimate the mean variance $\overline{\sigma}^2$ in the clusters. This information is used to construct the frequency sampling pattern Λ (either (G), (Gr) or (Ar)) that defines the sketching operator. Once the sketch of the dataset is computed, the CKM algorithm is used to extract the centroids, without needing access to the original dataset.

a particular choice of measure of distance between pdfs. Before explaining more precisely what this all means, and how the sketch of the sketch *of a dataset is defined*, we start with the general definition of the sketch of a pdf $\mathcal{P}(\boldsymbol{x})$.

Definition 6. ***Sketch of a probability density function:***
The sketching operator $\mathcal{A}$ associated with m frequencies $\Omega = [\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m] \in \mathbb{R}^{n\times m}$ in n dimensions is a linear compressive operator on the probability density function $\mathcal{P}(\boldsymbol{x}) : \mathbb{R}^n \mapsto \mathbb{R}_+ \in E$ (E is the set of continuous nonnegative measures with unit L^1 norm, i.e., the set of probability density functions):

$$\mathcal{A} : E \mapsto \mathbb{C}^m : \mathcal{P} \mapsto \mathcal{A}\mathcal{P} := \left[\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{P}} e^{-\mathrm{i}\boldsymbol{\omega}_j^T\boldsymbol{x}}\right]_{j=1}^{m} = [\psi_{\mathcal{P}}(\boldsymbol{\omega}_j)]_{j=1}^{m}, \tag{2.21}$$

where the second equality is obtained by definition of $\psi_{\mathcal{P}}$, the characteristic function[a] of $\mathcal{P}$

$$\psi_{\mathcal{P}}(\boldsymbol{\omega}) = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{P}} e^{-\mathrm{i}\boldsymbol{\omega}^T\boldsymbol{x}} = \int_{\mathbb{R}^n} \mathcal{P}(\boldsymbol{x}) e^{-\mathrm{i}\boldsymbol{\omega}^T\boldsymbol{x}} \mathrm{d}\boldsymbol{x}. \tag{2.22}$$

[a]Usually in probability theory, the name "characteristic function" usually denotes $\phi_{\mathcal{P}}(\boldsymbol{t}) = \mathbb{E}_{\boldsymbol{x}\simeq\mathcal{P}}\, e^{\mathrm{i}\boldsymbol{t}^T\boldsymbol{x}}$ instead of the function $\psi_{\mathcal{P}}(\boldsymbol{\omega})$ defined here. However, those two definitions of "the" characteristic function are linked by a simple change of variables : $\psi_{\mathcal{P}}(\boldsymbol{\omega}) = \phi_{\mathcal{P}}(\boldsymbol{t} = -\boldsymbol{\omega})$ so the difference has little importance. The definition here allows to interpret the characteristic function as the (direct) Fourier transform of the probability desnity function $\mathcal{P}$.

Thus, the j-th component of the sketch $\mathcal{A}\mathcal{P}$ corresponds to a sample of the characteristic function of $\mathcal{P}$ at $\boldsymbol{\omega} = \boldsymbol{\omega}_j$, or said differently, to the projection of $\mathcal{P}(\boldsymbol{x})$ on the complex exponential $e^{-\mathrm{i}\boldsymbol{\omega}_j^T\boldsymbol{x}}$. The sketch $\mathcal{A}\mathcal{P}$ is used as a *compressed representation* of the probability distribution $\mathcal{P}$ (an infinite-dimensional vector). It is also possible to interpret the sketching operator as a vector composed of *generalized moments* [26] of the probability distribution $\mathcal{P}$:

$$\mathcal{A} : \mathcal{P} \mapsto \mathcal{A}\mathcal{P} := \frac{1}{\sqrt{m}} \left[\int_{\mathbb{R}^n} M_1 \mathrm{d}\mathcal{P}, \cdots, \int_{\mathbb{R}^n} M_m \mathrm{d}\mathcal{P}\right]^T \quad \text{with} \quad M_j(\boldsymbol{x}) = \sqrt{m}\, e^{-\mathrm{i}\boldsymbol{\omega}_j^T\boldsymbol{x}}. \tag{2.23}$$

As will become clear when we will define the problem of probability density function matching from the sketch, this allows to interpret (2.28) as a particular case of the Generalized Method of

Moments (GeMM, [26]) where a parametric distribution is estimated by matching its generalized moments with empirical generalized moments of the data. We will come back later on those interpretations and on the way the frequencies $\boldsymbol{\omega}_j$ are chosen.

The probability distibutions that we are interested in here are mixture models [27]: the set of distributions whose probability density function is a convex combination[12] of L elementary densities belonging to a parametric family of probability density functions $\mathcal{G} := \{\mathcal{P}_{\boldsymbol{\theta}} \,|\, \boldsymbol{\theta} \in \Theta\} \subset E$. This means that $\mathcal{P}$ is of the form

$$\mathcal{P} = \sum_{l=1}^{L} \alpha_l \mathcal{P}_{\boldsymbol{\theta}_l} \in \mathcal{G}_L \quad \text{with} \quad \mathcal{P}_{\boldsymbol{\theta}_l} \in \mathcal{G},\ \alpha_l \geq 0,\ \sum_l \alpha_l = 1, \tag{2.24}$$

and we call $\{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_L\}$ the set of parameters of $\mathcal{P}$. Before addressing how we can define the sketch of a dataset, we give simple examples to better understand the concept of mixture model.

Example 3. As a first example, consider mixtures of $L = 3$ one-dimensional uniform probability distributions $\mathcal{G}_3$, with the elementary distribution family $\mathcal{G} = \{\mathcal{U}([a, b]) \mid (a, b) \in \mathbb{R}^2,\ a < b\}$. One element of $\mathcal{G}_3$ is represented Figure 2.8 on the left, with the set of parameters $\{(a, b)_i\} = \{(-2, 2), (0, 1), (3, 4)\}$ and the weights $\{\alpha_i\} = \{0.2, 0.2, 0.6\}$.
As a second example, we consider the mixture of Gaussian distributions, thus here $\mathcal{G} = \{\mathcal{N}(\mu, \sigma^2) \mid (\mu, \sigma^2) \in \mathbb{R}^2,\ \sigma^2 > 0\}$. One example of Gaussian mixture is shown Figure 2.8 on the right, with the set of parameters $\{(\mu, \sigma^2)_i\} = \{(-3, 0.04), (3, 0.04), (-1, 4)\}$ and the weights $\{\alpha_i\} = \{1/6, 1/6, 4/6\}$.

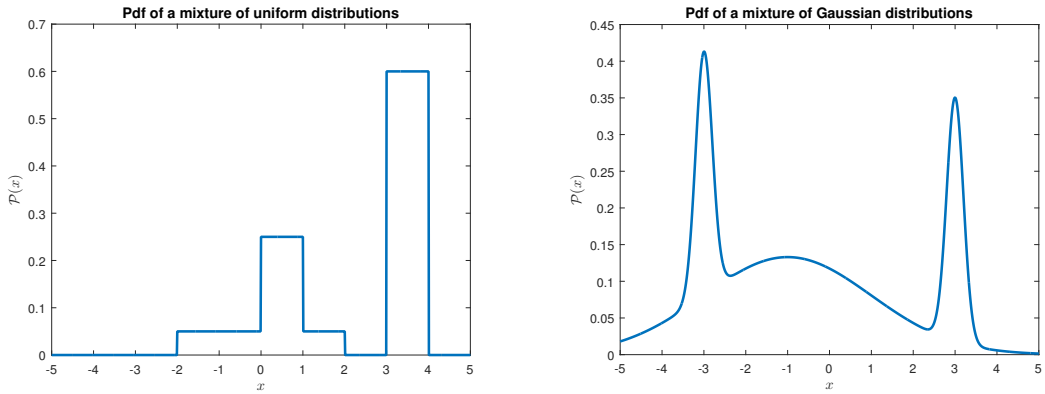


Figure 2.8: Examples of mixtures of uniform (left) and Gaussian (right) elementary distributions.

The reason why mixture models are so important for our work lies in the modelization choice of the dataset. Signals that are aquired $\boldsymbol{x}_i$ (elements of the dataset) are assumed to be *realizations of a random variable* $\boldsymbol{x}$. In this framework, the signals $\boldsymbol{x}_i$ can belong to one of L different *classes* $c(\boldsymbol{x}_i) \in \{1, ..., L\}$ where $c(\boldsymbol{x}_i)$ denotes the class index of $\boldsymbol{x}_i$. If $T_{\boldsymbol{x}}(\boldsymbol{x})$ denotes the probability density function of the random variable $\boldsymbol{x}$, we can write using the total probability rule

$$\mathcal{P}(\boldsymbol{x}) := T_{\boldsymbol{x}}(\boldsymbol{x}) = \sum_{l=1}^{L} \underbrace{\mathbb{P}\left[c(\boldsymbol{x}) = l\right]}_{\alpha_l} \underbrace{T_{\boldsymbol{x}|c(\boldsymbol{x})=l}(\boldsymbol{x}|c(\boldsymbol{x}) = l)}_{\mathcal{P}_{\boldsymbol{\theta}_l}(\boldsymbol{x})}, \tag{2.25}$$

which gives sense to the parameters of a mixture model in our setting of multiple classes: α_l is the prior probability of class l and $\mathcal{P}_{\boldsymbol{\theta}_l}(\boldsymbol{x})$ is the probability density function of $\boldsymbol{x}$ conditioned on the fact that the signal $\boldsymbol{x}$ is of class l.

[12] A linear combination whose non-negative coeffients have a sum equal to 1.

Sketch of a dataset

To use the sketch of a probability distribution as we defined just above, one needs obviously to know the probability density function of the random variable we are sensing. In practice however, when receiving a dataset $X = [\boldsymbol{x}_i]_{i=1}^N$ of signals $\boldsymbol{x}_i$ that are realisations of a random variable $\boldsymbol{x}$, we do not have access to the true pdf $\mathcal{P}(\boldsymbol{x})$. However, we can construct an *empirical probability density* of the sketch, noted $\hat{\mathcal{P}}_{X,\mathbf{1}_N/N}(\boldsymbol{x})$ or sometimes $\hat{\mathcal{P}}$ for conciseness, as

$$\hat{\mathcal{P}}_{X,\mathbf{1}_N/N}(\boldsymbol{x}) = \sum_{i=1}^{N} \frac{1}{N}\, \delta_{\boldsymbol{x}_i}(\boldsymbol{x}), \tag{2.26}$$

where $\delta_{\boldsymbol{x}_i}(\boldsymbol{x}) := \delta(\boldsymbol{x} - \boldsymbol{x}_i)$ is the Dirac delta function at $\boldsymbol{x}_i$. Note that (2.26) is a mixture density as in (2.24) with uniform weigths (the weights vector is $\mathbf{1}_N/N$) and delta distributions as elementary distributions. We are now ready to introduce the sketch $\mathrm{Sk}\,(Y, \boldsymbol{\beta})$ of a dataset Y of L points, where we allow non-uniform weigths β_l on the points $\boldsymbol{y}_l$, associated with m frequencies $\Omega = [\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m]$:

$$\mathrm{Sk}\,(Y, \boldsymbol{\beta}) := \mathcal{A}\hat{\mathcal{P}}_{Y,\boldsymbol{\beta}} = \left[\sum_{l=1}^{L} \beta_l e^{-\mathrm{i}\boldsymbol{\omega}_j^T \boldsymbol{y}_l}\right]_{j=1}^{m} = \exp(-\mathrm{i}\Omega^T Y)\boldsymbol{\beta} \qquad \in \mathbb{C}^m. \tag{2.27}$$

The sketch $\mathrm{Sk}\,(Y, \boldsymbol{\beta})$ can be seen as a *pooling* over the dataset Y of the complex exponential of the points of the dataset projected on the different frequencies $\boldsymbol{\omega}_j$ (with weights given by $\boldsymbol{\beta}$). The resulting vector is a summary of the dataset and can be used *a posteriori* to retrieve information about it, as we will now see.

How to use the sketch to match distributions, GMM estimation and CKM

Finally, the *probability distribution reconstruction principle* based on the sketch can be detailed, first in a general setting and then in the two particular cases of GMM estimation [2] and of Compressive K-Means [1]. In the general setting, one wishes to reconstruct the unknown probability density function $\mathcal{P}$ characterizing the samples $\boldsymbol{x}_i$ of a dataset X that is composed of K classes, thus $\mathcal{P} \in \mathcal{G}_K$ (see (2.24)). The criterion for reconstruction is to select a reconstruction pdf $\mathcal{Q}$ such that the empirical sketch of the dataset $\hat{\boldsymbol{z}} := \mathcal{A}\hat{\mathcal{P}}_{X,\mathbf{1}_N/N} = \mathrm{Sk}\,(X, \mathbf{1}_N/N)$ and the sketch of the reconstructed pdf $\mathcal{A}\mathcal{Q}$ are close to each other. Thus, the reconstructed distribution $\mathcal{Q}^*$ is selected by sketch matching with the empirical sketch of the data

$$\mathcal{Q}^* = \arg\min_{\mathcal{Q}} \|\hat{\boldsymbol{z}} - \mathcal{A}\mathcal{Q}\|_2^2 \quad \text{s.t.} \quad \mathcal{Q} \in \mathcal{G}_K, \tag{2.28}$$

where the choice of the seach space $\mathcal{G}_K$ ensures that $\mathcal{Q}$ is a mixture model of K classes. The general setup being defined, we will discuss the two particularizations of this setup that are instantiated when the space of classes distributions is defined $\mathcal{G} := \{\mathcal{P}_{\boldsymbol{\theta}} \mid \boldsymbol{\theta} \in \Theta\}$.

- Gaussian Mixture Estimation [2]: each class is assumed to be normally distributed[13], which means that $\mathcal{G} = \{\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) \mid \boldsymbol{\theta} = (\boldsymbol{\mu}, \Sigma),\ \boldsymbol{\mu} \in \mathbb{R}^n,\ \Sigma = \Sigma^T \in \mathbb{R}^{n\times n} \succeq 0\}$. Thus $\mathcal{Q}$ will be of the form $\mathcal{Q} = \sum_{k=1}^{K} \alpha_k\, \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k)$. In short, the reconstructed distribution is a mixture of Gaussians with different weights, means and covariance matrices.

- Compressive K-Means [1]: each class is allowed to be at one single location $\boldsymbol{c}$ (the "centroid"), the density is thus a Dirac function: $\mathcal{G} = \{\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \delta_{\boldsymbol{c}}(\boldsymbol{x}) \mid \boldsymbol{\theta} = \boldsymbol{c} \in \mathbb{R}^n\}$. This means that $\mathcal{Q}$ will be of the form $\mathcal{Q} = \sum_{k=1}^{K} \alpha_k\, \delta_{\boldsymbol{c}_k}$. In short, the reconstructed distribution is a mixture of Dirac deltas with different weights and locations.

[13] In n dimensions, the normal distribution density is $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) = ((2\pi)^n |\Sigma|)^{-1/2} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right)$ (remember that we are abusing notations a little bit, writing the distribution and its probability density function with the same symbol).

	Usual CS	Compressive mixture estimation
Signal of interest	Vectors $\boldsymbol{\alpha} \in \mathbb{R}^d$	Distributions $\mathcal{P} \in E$
K-Sparse model	$\boldsymbol{\alpha} \in \Sigma_K = \{\boldsymbol{\alpha}' = \sum_{k=1}^K \alpha'_k \boldsymbol{e}_{l_k}\}$	$\mathcal{P} \in \mathcal{G}_K = \{\mathcal{P}' = \sum_{k=1}^K \alpha_k \mathcal{P}_{\boldsymbol{\theta}_k}\}$
Linear sensing object	$A \in \mathbb{R}^{m \times n}$	$\mathcal{A} : E \mapsto \mathbb{C}^m$
Measurement	$\boldsymbol{y} = A\boldsymbol{x} \ \in \mathbb{R}^m$	$\boldsymbol{z} = \mathcal{A}\mathcal{P} \in \mathbb{C}^m$
Reconstruction procedure	$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}' \in \Sigma_K} \|\boldsymbol{y} - A\boldsymbol{x}'\|_2^2$	$\mathcal{Q}^* = \arg\min_{\mathcal{Q} \in \mathcal{G}_K} \|\boldsymbol{z} - \mathcal{A}\mathcal{Q}\|_2^2$

Table 2.1: Mapping of the "usual CS objects" to "pdf estimation through sketching objects", based on a table in [2]. The basis vector $\boldsymbol{e}_{l_k} \in \mathbb{R}^d$ is a vector with only zeroes as entries except a 1 for the l_k-th entry.

Having discussed the general framework, we focus mostly on CKM since we are interested, for this work, in the clustering problem. The probability density function matching problem (2.28) can in this case be re-written to obtain *the compressed clustering problem*, an optimization problem to solve to obtain the clusters centroids $C = \{\boldsymbol{c}_1, \cdots, \boldsymbol{c}_K\}$ and, as a by-product, the relative importance (or "class priors") of the clusters $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_K]^T$.

Definition 7. ***Compressive K-Means clustering problem.***
The estimated cluster centroids $C^* = \{\boldsymbol{c}_1^*, \cdots, \boldsymbol{c}_K^*\}$ and weights $\boldsymbol{\alpha}^*$ are solution to

$$(C^*, \boldsymbol{\alpha}^*) = \arg\min_{C,\boldsymbol{\alpha}} \|\mathrm{Sk}(X, \mathbf{1}_N/N) - \mathrm{Sk}(C, \boldsymbol{\alpha})\|_2^2 = \arg\min_{C,\boldsymbol{\alpha}} \left\| \hat{\boldsymbol{z}} - \mathcal{A}\left(\sum_{k=1}^K \alpha_k \delta_{\boldsymbol{c}_k}\right) \right\|_2^2 \quad (2.29)$$

with a sketch operator (2.27) whose frequencies are drawn $\boldsymbol{\omega}_j \overset{\text{iid}}{\sim} \Lambda$ for a distribution Λ to be specified (see later).

Note that this problem (and more generally, the problem (2.28)) is a highly non-convex optimization problem [2], an indication that solving it exactly will probably prove to be difficult. As we will soon see when discussing the `CLOMPR` algorithm and its variant `CKM`, an approximate solution can be found using a greedy algorithm inspired by the `OMPR` algorithm (algorithm 4) from the field of Compressed Sensing. Before discussing how (2.29) can be approximatively solved, we discuss the relation between probability density function matching through sketching and CS to understand why the `OMPR` is a good inspiration for solving the sketch matching problem.

Link with Compressed Sensing

Interestingly, the problem of sketch matching can be seen as a Compressed Sensing estimation problem, where the sparse signal of interest is the infinite-dimensional propability density function underlying the acquired samples. The measured and reconstructed densities are "sparse" in the sense that they are expressed as a combination of a small number K of elementary densities $\mathcal{P}_{\boldsymbol{\theta}_k} \in \mathcal{G}$ (the "atoms"), just as a sparse vector is a combination of few basis vectors $\boldsymbol{e}_{l_k}$. As in usual compressive sensing, the measurement operator (the sketch $\mathcal{A}$) is made up of m linear measurements, that is, correlation with well-chosen measurement vectors -or rather here, functions- $e^{-\mathrm{i}\boldsymbol{\omega}_j^T \boldsymbol{x}}$. The optimal reconstructed density is then the sparse density that matches the compressive measurements the best. More formally, the correspondance between usual Compressed Sensing of vectors and compressive density estimations is described table 2.1.

The sketch can thus be seen as an *embedding* of the set of probability distributions ("infinite-dimensional vectors") into the set of m-dimensional complex vectors. Just as in classical CS estimation where the measurement operator A is typically random (e.g. $A_{i,j} \overset{\text{iid}}{\sim} \mathcal{N}(0, 1/m)$), the

sketching operator is also random since the sensing functions frequencies are random $\boldsymbol{\omega}_j \overset{\text{iid}}{\sim} \Lambda$. In CS, the randomness is used to obtain the RIP on A with very high probability given m is high enough with respect to the sparsity of the signals, and the RIP in turn guarantees bounds for the error of the reconstructed vector $\boldsymbol{x}^*$. [2] generalizes the RIP and reconstruction error guarantees of CS to sketching operators $\mathcal{A}$, but those early results are pessimistic (the "compressive advantages" seem to be nullified). However, the authors believe those bounds can be improved.

CLOMPR algorithm and CKM algorithm

We are now looking to solve (2.28) in practice. As we said before, this problem is highly non-convex, and there are currently no (tractable) tools to solve it exactly. Given the strong relationship between this problem and sparse reconstructions (see table 2.1, especially the last line), the authors of [2] propose to adapt the `OMPR` algorithm (algorithm 4) for sparse reconstruction to the compressive mixture estimation framework, calling it Compressive Learning Orthogonal Matching Pursuit with Replacement (CLOMPR). The `CLOMPR` algorithm thus solves (2.28) approximatively by selecting, at each iteration, in a greedy manner the probability density function $\mathcal{P}_{\boldsymbol{\theta}_k}$ to add to the support of $\mathcal{Q}$ that reduces the most the residual.

`CLOMPR` is an algorithm for *generic* mixture models. To stay to the point, we will not describe `CLOMPR` here extensively but we will instead present a particular case of `CLOMPR`: `CKM` for Compressive K-Means [1]. `CKM`, detailed as algorithm 5 is nothing but `CLOMPR` applied to the CKM setting, where the elementary densities are Dirac deltas. Note that the algorithm only access the dataset X through its sketch $\hat{\boldsymbol{z}}$ and the bounds $\boldsymbol{l}$ and $\boldsymbol{u}$ such that $\boldsymbol{l} \leq \boldsymbol{x}_i \leq \boldsymbol{u} \quad \forall\, \boldsymbol{x}_i \in X$.

The idea and structure of `CKM` are the same as `OMPR` (with $T = 2K$ iterations), but there are some significant differences that were introduced either by `CLOMPR` or by its particularization to mixtures of deltas, i.e. `CKM`:

- **Non-negativity** (added for CLOMPR): the weights $\boldsymbol{\alpha}$ of the centroids (associated with the prior probabilities of those clusters) must be positive, in constrast to the free coefficients in `OMPR`. Therefore, in step 1 the amplitude of the correlation is replaced by its real part (if the correlation is negative, it means that only a negative weight can reduce the objective function). Moreover in steps 3, 4 and 5 the weights are constrained to be positive.

- **Continuous dictionary** (added for CLOMPR): in `OMPR`, the support is a finite set of indices $\subset \{1, \cdots, d\}$ that can be exhaustively searched in step 1, but for `CLOMPR` the support is a set of centroids $\subset \mathbb{R}^n$. Consequently, the maximization at steps 1 and 5 are approximated by local optima searches with a quasi-Newton method (we explain this in greater detail below). The main point is that step 1 and 5 are not solved exactly but local optima are reached instead.

- **Additional cost reduction** (added for CLOMPR): there is an additional global minimization step where the centroids are allowed to move if the cost function is reduced. Remember that this minimization step is not a global minimization but a local minimization initialized with the current centroids.

- **Additional constraints** (added for CKM): in steps 1 and 5, the centroids are constrained to be in the box that bounds the whole dataset $\boldsymbol{l} \leq \boldsymbol{x}_i \leq \boldsymbol{u} \quad \forall\, \boldsymbol{x}_i \in X$.

The optimization problems to solve in step 1 and 5 are non-convex with respect to the position of the centroids ($\boldsymbol{c}$ for step 1 and $C = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_{\min(t,K)}]$ for step 5). Thus, as mentioned, a local optimum is computed using the L-BFGS method, a quasi-Newton minimization method for non-linear problems. This method is presented in Appendix A but the property that is relevant

Algorithm 5: `CKM` : `CLOMPR` for K-means clustering.

input : Empirical sketch $\hat{\boldsymbol{z}} = \mathrm{Sk}\left(X, \mathbf{1}_N/N\right)$, frequencies Ω, number of centroids K, bounds $\boldsymbol{l}$, $\boldsymbol{u}$

output : A set of centroids C and weights $\boldsymbol{\alpha}$ that locally minimizes the sketch mismatch $\|\hat{\boldsymbol{z}} - \mathrm{Sk}\left(C, \boldsymbol{\alpha}\right)\|_2^2 = \|\hat{\boldsymbol{z}} - \sum_k \alpha_k \mathcal{A}\delta_{\boldsymbol{c}_k}\|_2^2$

Initialization $\hat{\boldsymbol{r}} \leftarrow \hat{\boldsymbol{z}}$, $C \leftarrow \emptyset$ *(Initialize the residual and the support (centroids))* ;

for *t = 1,...,2K* **do**

  Step 1 : find the new centroid $\boldsymbol{c}$ with highest correlation with residual:

  $\boldsymbol{c} = \arg\max_{\boldsymbol{c}} \Re < \frac{\mathcal{A}\delta_{\boldsymbol{c}}}{\|\mathcal{A}\delta_{\boldsymbol{c}}\|}, \hat{\boldsymbol{r}} > \quad \text{s.t.} \quad \boldsymbol{l} \leq \boldsymbol{c} \leq \boldsymbol{u}$

  Step 2 : add it to the support:

  $C = C \cup \{\boldsymbol{c}\}$

  Step 3 : Reduce support by Hard Thresholding to keep sparsity ("Replacement" step):

  if $|C| > K$ **then**

    $\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}_+^{|C|}} \left\| \hat{\boldsymbol{z}} - \sum_{k=1}^{|C|} \beta_k \frac{\mathcal{A}\delta_{\boldsymbol{c}_k}}{\|\mathcal{A}\delta_{\boldsymbol{c}_k}\|} \right\|_2$

    $C \leftarrow$ set of K centroids $\boldsymbol{c}_k$ corresponding to K largest magnitude values of $\boldsymbol{\beta}$

  end

  Step 4 : Find the optimal coefficients given the support (by projection):

  $\boldsymbol{\alpha} = \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}_+^{|C|}} \left\| \hat{\boldsymbol{z}} - \sum_{k=1}^{|C|} \alpha_k \mathcal{A}\delta_{\boldsymbol{c}_k} \right\|_2$

  Step 5 : Global gradient descent

  $(C, \boldsymbol{\alpha}) \leftarrow \arg\min_{C, \boldsymbol{\alpha}} \left\| \hat{\boldsymbol{z}} - \sum_{k=1}^{|C|} \alpha_k \mathcal{A}\delta_{\boldsymbol{c}_k} \right\|_2 \quad \text{s.t.} \quad \boldsymbol{l} \leq \boldsymbol{c} \leq \boldsymbol{u}$

  $\hat{\boldsymbol{r}} = \hat{\boldsymbol{z}} - \sum_{k=1}^{|C|} \alpha_k \mathcal{A}\delta_{\boldsymbol{c}_k}$ *(Update residual, go back to step 1)*

end

here is that it requires access to i) the function to minimize, ii) the gradient of this function with respect to the decision variables and finally iii) an initial search point for the decision variables.

Appendix B.1 details how the first two requirements i) and ii) are computed to be fed to an external solver implementing L-BFGS. The key fact is that is that the gradients ii) require computing the term $\boldsymbol{\nabla}_{\boldsymbol{c}}\mathcal{A}\delta_{\boldsymbol{c}}$, and we will see in Chapter 3 that this term is problematic in the context of the `QCKM` method we propose. The initial point iii) is selected uniformly at random in the box $\boldsymbol{l} \leq \boldsymbol{c} \leq \boldsymbol{u}$ (since once the sketch is computed we "forget" the dataset X, this is the only information we have about the location of the examples) for step 1 and at the current location of the centroids in step 5.

Selecting the frequency pattern Λ and the number of frequencies m

So far we have defined what the sketching operator is and described how to perform clustering using the sketch of a dataset in practice with the `CKM` algorithm. We still need to present how the frequencies $\boldsymbol{\omega}_j$ of the sketch are chosen, and how many of those frequencies are required for accuracte clustering.

The distribution Λ from which the frequencies are drawn $\boldsymbol{\omega}_j \overset{\text{iid}}{\sim} \Lambda$ for Compressive K-Means is, in fact, adapted to perform the Gaussian Mixture Model estimation (a different problem), but it has been shown that it performs well for `CKM` as well [1]. The design of the frequency distribution is a based on a heuristic criterion for the synthetic case where the goal probability density function to estimate is a single known Gaussian (this is in practice not the case but we will see how to use this criterion in real applications). Starting from this oracle density to estimate, three heuristic choices for Λ are presented: the Gaussian distribution `(G)`, Gaussian radius distribution `(Gr)` and the adapted radius distribution `(Ar)`.

Gaussian heuristic (G). Assumeing the Gaussian probability density function to estimate is known

$$\mathcal{P}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})}, \tag{2.30}$$

we can then compute $\psi_{\mathcal{P}}(\boldsymbol{\omega})$, the characteristic function associated with $\mathcal{P}$, and its modulus $|\psi_{\mathcal{P}}(\boldsymbol{\omega})|$:

$$\psi_{\mathcal{P}}(\boldsymbol{\omega}) = e^{-\mathrm{i}\boldsymbol{\omega}^T \boldsymbol{\mu}} e^{-\frac{1}{2}\boldsymbol{\omega}^T \Sigma \boldsymbol{\omega}} \quad \Rightarrow \quad |\psi_{\mathcal{P}}(\boldsymbol{\omega})| = e^{-\frac{1}{2}\boldsymbol{\omega}^T \Sigma \boldsymbol{\omega}} \propto \mathcal{N}(\boldsymbol{\omega}; 0, \Sigma^{-1}) \tag{2.31}$$

it seems thus an inuitive choice to choose a Gaussian distribution $\Lambda^{\mathtt{G}}(\boldsymbol{\omega}) = \mathcal{N}(0, \Sigma^{-1})$. Since the sketch is a sampling of the characteristic function, choosing the frequencies $\boldsymbol{\omega} \sim \Lambda^{\mathtt{G}}(\boldsymbol{\omega})$ corresponds to sampling frequencies of the characteristic function with probabilities proportional to the amplitude of the characteristic function, hopefully producing more significant values for the sketch.

Gaussian radius heuristic (Gr). Unfortunately, the -intuitively sound- choice $\Lambda^{\mathtt{G}}(\boldsymbol{\omega})$ does not account for the *curse of dimensionality*. Simply put, the curse of dimensionality is the fact that for increasing dimensions, the n-dimensional space becomes exponentially difficult to fill (for a simple example, an n-dimensional cube has 2^n vertexes). In this context, the curse of dimensionality manifests in the fact that points drawn from high-dimensional Gaussiants concentrate on an ellipsoidal "shell", so $\Lambda^{\mathtt{G}}(\boldsymbol{\omega})$ will almost only sample high frequencies. Sadly, the sketch coefficients associated with high frequencies do bring little information about the pdf

we are sensing, and will almost always be close to zero.

To circumvent this problem and have precise control over the amplitude of the sketch, $e^{-\frac{1}{2}\boldsymbol{\omega}^T\Sigma\boldsymbol{\omega}}$, the Gaussian radius heuristic proposes not to sample $\boldsymbol{\omega}$ directly but rather to sample the radius of the high-dimensional Gaussian $R = \sqrt{\boldsymbol{\omega}^T\Sigma\boldsymbol{\omega}}$. More precisely, it chooses the frequencies as[14]

$$\boldsymbol{\omega} = R\Sigma^{-1/2}\boldsymbol{\varphi} \quad \text{with} \quad \boldsymbol{\varphi} \sim \mathcal{U}(S_{n-1}), \quad R \sim p_R(R) \tag{2.32}$$

where $\boldsymbol{\varphi}$ is a random direction sampled uniformly on the ℓ_2 unit sphere S_{n-1}, and R is a radius sampled from some radius distribution $p_R(R)$ to specify. Note that for this particular form of $\boldsymbol{\omega} = R\Sigma^{-1/2}\boldsymbol{\varphi}$, we have indeed that $\boldsymbol{\omega}^T\Sigma\boldsymbol{\omega} = R^2$ and the amplitude of the characteristic function (2.31) reduces to

$$\begin{aligned} \psi_{\mathcal{P}}(R\Sigma^{-1/2}\boldsymbol{\varphi}) &= e^{-\mathrm{i}R\overbrace{\boldsymbol{\varphi}^T\Sigma^{-1/2}\boldsymbol{\mu}}^{\mu_0}} \cdot e^{-\frac{1}{2}R^2\cdot\overbrace{1}^{\sigma_0^2}} = \psi_{\mathcal{N}(\mu_0,\sigma_0^2)}(R) \\ \Rightarrow |\psi_{\mathcal{P}}(R\Sigma^{-1/2}\boldsymbol{\varphi})| &= e^{-\frac{1}{2}R^2} \propto \mathcal{N}(R; 0, \sigma_0^2 = 1) \end{aligned} \tag{2.33}$$

and, unlike the n-dimensional intuitive Gaussian for $\boldsymbol{\omega}$ we had in (2.31), we have a *one-dimensional* intuitive Gaussian distribution for R, hence eliminating the curse of dimensionality that affects high-dimensional Gaussians. To summarize, the Gaussian radius heuristic $\Lambda^{\texttt{Gr}}(\boldsymbol{\omega})$ corresponds to sampling $\boldsymbol{\omega}$ as (2.32) with

$$p_R(R) = \mathcal{N}(R; 0, 1). \tag{2.34}$$

Adapted radius heuristic (Ar). In the two precedent heuristics, we sought to sample frequencies that would produce a sketch with entries whose amplitude were significant. Actually, the relevance of this criterion to produce a "good" sketch is debatable: what a "good" sketch is should be determined by what the sketch is used for (i.e. discriminating probability density functions). In fact, the Gaussian radius density has the highest sampling density at low radius R, where *all* probability density functions are close to 1. The adapted radius heuristic aims therefore at sampling the radius R where the one-dimensional characteristic function (2.33), noted $\psi_{\mathcal{N}(\mu_0,\sigma_0^2)}(R)$, is sufficiently different from characteristic functions of other Gaussians, i.e. with other parameters μ and σ^2. The (absolute) rate of change of

$$\psi_{\mathcal{N}(\mu=\mu_0,\sigma^2=\sigma_0^2)}(R) = e^{-\mathrm{i}R\mu}e^{-\frac{1}{2}R^2\sigma^2} \tag{2.35}$$

with respect to the parameters (μ, σ^2) is mathematically represented by the norm of the gradient

$$\begin{aligned} \|\boldsymbol{\nabla}_{(\mu,\sigma^2)}\psi_{\mathcal{N}(\mu,\sigma^2)}(R)\|_2^2 &= \|\boldsymbol{\nabla}_{(\mu,\sigma^2)}e^{-\mathrm{i}R\mu}e^{-\frac{1}{2}R^2\sigma^2}\|_2^2 \\ &= |\nabla_\mu\psi_{\mathcal{N}(\mu,\sigma^2)}(R)|^2 + |\nabla_{\sigma^2}\psi_{\mathcal{N}(\mu,\sigma^2)}(R)|^2 \\ &= \left(R\,e^{-\frac{1}{2}R^2\sigma^2}\right)^2 + \left(\frac{R^2}{2}e^{-\frac{1}{2}R^2\sigma^2}\right)^2 = \left(R^2 + \frac{R^4}{4}\right)e^{-R^2\sigma^2}. \end{aligned} \tag{2.36}$$

We want to sample the radius R where the norm of the gradient is significant around the parameters $(\mu_0 = \boldsymbol{\varphi}^T\Sigma^{-1/2}\boldsymbol{\mu}, \sigma_0^2 = 1)$. The third heuristic $\Lambda^{\texttt{Ar}}(\boldsymbol{\omega})$ called adapted radius heuristic thus selects the frequencies $\boldsymbol{\omega}$ as (2.32) with

[14] $\Sigma^{-1/2}$ is the inverse of $\Sigma^{1/2}$, the Cholesky factorization of $\Sigma = \Sigma^{1/2}(\Sigma^{1/2})^T = (\Sigma^{1/2})^T\Sigma^{1/2}$ (the second equality coming from the fact that the covariance Σ must be symetric). The Cholesky factorization exists since the covariance Σ must be positive definite.

$$p_R(R) \propto \left(R^2 + \frac{R^4}{4}\right)^{1/2} e^{-\frac{1}{2}R^2}. \tag{2.37}$$

In practice in GMM, the probability distribution of interest is not a single Gaussian with known parameters $(\boldsymbol{\mu}, \Sigma)$ but a mixture of Gaussians whose true parameters $(\boldsymbol{\mu}_k, \Sigma_k)$ are unknown. To be still able to use the three heuristics discussed above, the covariances of the Gaussians have to be estimated. To do this, before sketching, a small subset of the dataset is acquired to estimate[15] $\overline{\sigma}^2$, the mean variance across all clusters and all dimensions, then the different clusters are assumed to have a covariance $\overline{\sigma}^2 I$. This procedure is inspired by Distilled Sensing (DS) [28], a sensing paradigm where some of the acquisition resources (measurement bits) are first spent to acquire very crude measurements to identify the support of a sparse signal, then the remaining of the resources are used to acquire this sparse signal with measurements that are *adapted* to this support.

Finally, as will be clear with the experimental results from Chapter 5, the number of frequencies m is a crucial parameter. Intuitively, larger values of m do contain more information about the characteristic function thus about the distribution of the dataset, but how many frequencies are "enough" for e.g. clustering applications? Empirical results from [1] show that there seems to be a rule that m should grow proportionally with the number of clusters and with the dimension of the problem, i.e. $m = \mathcal{O}(Kn)$, in order to have clustering results that compete with state of the art clustering algorithms such as `k-means++`. For GMM estimation, under some assumtions this requirement $m = \mathcal{O}(Kn)$ is proven ([2], Corollary 3). The main advantage of compressive clustering is that the sketch dimension m is independent of the dataset cardinality (number of examples) N, just like the in CS the number of measurements is independent of the dimension of the input space.

2.3.3 Additional interpretations and connexions to other work

Although we now have completed our description of clustering by sketching, there are some interesting interpretations of the sketch. Connexions can be made with Reproducing Kernel Hilbert Spaces (RKHS), where we will formalize the notion of distance between probability density functions captured by the sketch difference, and see that the selection of the frequency distribution Λ is related to an implicit kernel design. There is also an interesting link between the sketch and Random Fourier Features (RFF) for fast kernel approximations.

Reproducing Kernel Hilbert Spaces, embeddings of probability density functions and implicit kernel design

What is a kernel and the kernel trick. Before diving into RKHS, we introduce what a kernel is. In machine learning, the notion of *kernel* is mostly known for its applications in Support Vector Machines (SVM), a supervised binary classification problem. SVM classification relies on finding a separating hyperplane, where all the positive examples (training examples $\boldsymbol{x}_i \in X$ with class $y_i = +1$) are on one side of the hyperplane and all the negative examples (training examlpes $\boldsymbol{x}_i \in X$ with class $y_i = -1$) are on the other side of the hyperplane. Because in most datasets X the examples are not linearly separable, nonlinear Support Vector Machines first map the dataset to a higher-dimensional space where they are (hopefully) linearly separable through a "feature map" $\varphi : \boldsymbol{x}_i \mapsto \varphi(\boldsymbol{x}_i)$ as represented Figure 2.9 ($\varphi(\boldsymbol{x}_i)$ can be interpreted as "features" of the example $\boldsymbol{x}_i$).

[15] For conciseness, we do not detail this procedure here, it can be found in [2].

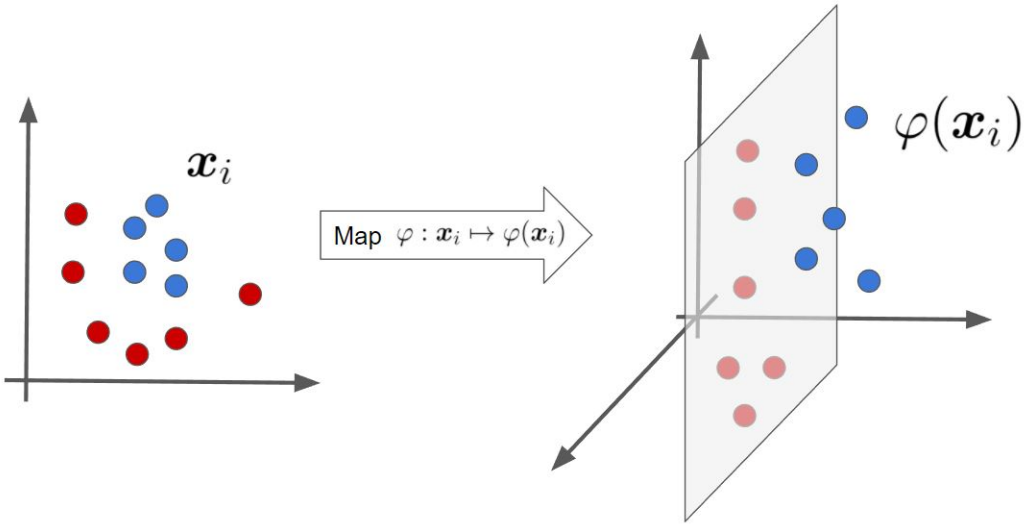


Figure 2.9: Schematic view of SVM classification. On the left, the dataset X in the original space (here, $\mathbb{R}^2$) with positive (red) and negative (blue) examples $\boldsymbol{x}_i$ that are not linearly separable. On the right, the feature vectors $\varphi(\boldsymbol{x}_i)$ in a higher-dimensional feature space (here $\mathbb{R}^3$) where they are linearly separable. The SVM classifier is then defined by the separable hyperplane in this high-dimensional space, that corresponds to a nonlinear boundary in the original space.

In practice mapping all the examples through φ and working with the feature vectors is very costly (we have to work in a potentially very high-dimensional[16] space), but this is where the *kernel trick* comes in handy. The kernel trick is based on the observation that finding the SVM classifier (the hyperplane in the high-dimensional feature space) does not require the feature vectors $\varphi(\boldsymbol{x}_i)$ but only their inner products $\langle \varphi(\boldsymbol{x}_i), \varphi(\boldsymbol{x}_j) \rangle$. Therefore, it is possible to perform SVM classification based only on the evaluation of the kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, that is, the inner product $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \varphi(\boldsymbol{x}_i), \varphi(\boldsymbol{x}_j) \rangle$, without even having access to a closed-form expression of the map φ. The kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ can be seen as a *similarity measure* between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as it is their scalar product in an high-dimensional feature space. Formally, we define a (in all generality, complex) positive definite kernel ([29], definition 1):

Definition 8. ***Positive definite kernel:***
A positive definite (pd) kernel is a symetric function $\kappa : X \times X \mapsto \mathbb{C}$ *if* $\forall M \in \mathbb{N}, \boldsymbol{x}_i \in X, c_i \in \mathbb{C}$ *we have*

$$\sum_{i,j=1}^{M} c_i c_j^* \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0. \tag{2.38}$$

This means, in particular, that the kernel value (the similarity) is always positive. A kernel $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is called a *translation-invariant kernel* if it can be written as a function of the difference between its arguments only: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = K(\boldsymbol{x}_i - \boldsymbol{x}_j)$, where K is a positive definite function iff κ is a pd kernel. We hereafter focus on translation-invariant kernels.

RKHS and embeddings of probability density functions. To make the connexion between the notion of kernel and the sketch of a probability density function, we need to introduce the concept of Reproducing Kernel Hilbert Spaces, or RKHS in short ([27], definition 1):

[16] For example, a Gaussian kernel maps examples to a feature space of infinite dimension!

Definition 9. ***Reproducing Kernel Hilbert Space:***
A kernel $\kappa : X \times X \mapsto \mathbb{R} : (\boldsymbol{x}, \boldsymbol{y}) \mapsto \kappa(\boldsymbol{x}, \boldsymbol{y})$ *is a reproducing kernel of the Hilbert space* $\mathcal{H}$ *iff we have*

$$\text{(i)} \quad \forall \boldsymbol{y} \in X,\ \kappa(\cdot, \boldsymbol{y}) \in \mathcal{H}$$

and

$$\text{(ii)} \quad \forall \boldsymbol{y} \in X,\ \forall f \in \mathcal{H}\ \langle f, \kappa(\cdot, \boldsymbol{y}) \rangle_{\mathcal{H}} = f(\boldsymbol{y}).$$

Then, $\mathcal{H}$ *is called a reproducing kernel Hilbert space.*

It turns out that for every RKHS $\mathcal{H}$, the associated reproducing kernel κ is symetric and positive definite [27] (it is thus a pd kernel). In fact, the Moore-Aronszajn theorem [30] states the converse proposition: each positive definite kernel κ defines a unique RKHS $\mathcal{H}$: there is thus a bijective relationship between the pd kernels and their Reproducible Kernel Hilbert Spaces. The key fact to keep in mind is that any positive definite kernel is associated with a unique Hilbert space of functions.

The concept of RKHS can be used to, given a RKHS $\mathcal{H}$ and associated kernel κ, define an embedding φ of the probability density functions $\mathcal{P} \in E$ (by E we denote the set of probability density functions) into the RKHS $\mathcal{H}$ as [27]

$$\varphi : E \mapsto \mathcal{H} : \mathcal{P} \mapsto \varphi(\mathcal{P}) := \int_X \kappa(\cdot, \boldsymbol{x}) \mathcal{P}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{P}} [\kappa(\cdot, \boldsymbol{x})]\,. \tag{2.39}$$

Because of this last expression, [2] calls the embedding $\varphi(\mathcal{P})$ the Mean Map. Note that in our case, we focus on $X = \mathbb{R}^n$. In the idea of usual embeddings of vectors, we will compare two probability distributions $\mathcal{P}$ and $\mathcal{Q}$ by comparing their embeddings $\varphi(\mathcal{P})$ and $\varphi(\mathcal{Q})$, as represented Figure 2.10.

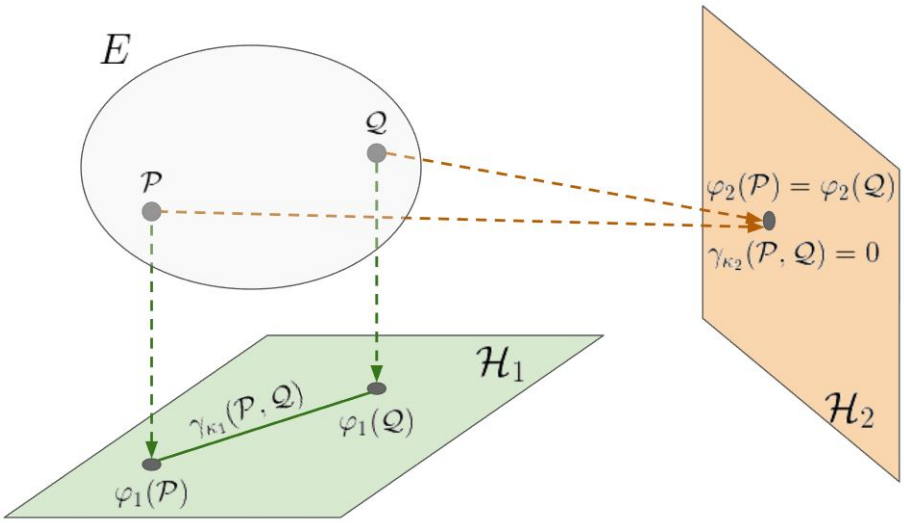


Figure 2.10: Schematic view of the Maximum Mean Discrepancy (pseudo-)metric induced by the Mean Map for two different kernels, κ_1 (characteristic) and κ_2 (not characteristic). The two probability distributions of interest, $\mathcal{P}, \mathcal{Q} \in E$ are embedded into $\mathcal{H}_1$, the RKHS associated with κ_1 (the green plane). Their distance in this space is the Maximum Mean Discrepancy metric γ_{κ_1}. If a different kernel κ_2 is used, a different RKHS $\mathcal{H}_2$ is associated with it (the orange plane) with a different (pseudo-)metric γ_{κ_2}. Since κ_2 is not characteristic γ_{κ_2} is not a true metric.

By comparing the Mean Map of different probability distributions $\mathcal{P}$ and $\mathcal{Q}$, we obtain the **pseudometric** known as Maximum Mean Discrepancy

$$\gamma_\kappa(\mathcal{P}, \mathcal{Q}) = \|\varphi(\mathcal{P}) - \varphi(\mathcal{Q})\|_{\mathcal{H}} \tag{2.40}$$

and it is thus in general **not** a metric since the mean map is in general not injective, i.e., it is possible to have $\mathcal{P} \neq \mathcal{Q}$ but $\gamma_\kappa(\mathcal{P}, \mathcal{Q}) = 0$ (see for example the kernel κ_2 in Figure 2.10: two different pdfs or points in E could map to the same point in the space $\mathcal{H}_2$). A kernel κ inducing an injective Mean Map and hence for which γ_κ is a true metric, in other words a kernel for which $\mathcal{P} = \mathcal{Q} \Leftrightarrow \gamma_\kappa(\mathcal{P}, \mathcal{Q}) = 0$, is called a *characteristic kernel* ([31], definition 6). In what follows, having a characteristic kernel will thus be a desirable property since we to obtain a true metric for the probability density functions.

How the sketch approximates the MMD metric. We are not far from the link between the MMD abd the sketch of a probability distribution, but we need one additional theorem ([32], Theorem 1.4.3):

Theorem 3. ***Bochner's theorem:*** *for a continuous function* $K(\boldsymbol{u})$

$$\begin{aligned} &K(\boldsymbol{u}) : \mathbb{R}^n \mapsto \mathbb{C} \text{ is a positive definite kernel } \iff \\ &K(\boldsymbol{u}) = \int_{\mathbb{R}^n} e^{-\mathrm{i}\boldsymbol{\omega}^T \boldsymbol{u}} \Lambda(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \text{ for a finite nonnegative measure } \Lambda \end{aligned} \tag{2.41}$$

where the latter expression means that the translation-invariant kernel $K(\boldsymbol{u})$ *is the Fourier transform of a finite*[a] *measure* Λ *on* $\mathbb{R}^n$.

[a] This means that $\int_{\mathbb{R}^n} \mathrm{d}\Lambda(\boldsymbol{\omega}) < \infty$.

Thus, every positive definite kernel is the Fourier transform of a nonnegative measure Λ, that can in particular be normalized to a probability density function (this implies a kernel normalization). A translation-invariant positive definite kernel is thus the Fourier transform of a probability density function $\Lambda(\boldsymbol{\omega})$ on the frequencies $\boldsymbol{\omega} \in \mathbb{R}^n$. Moreover, a condition on Λ guarantees that κ is characteristic[17], thus defining a true metric γ_κ ([31], Theorem 9):

Theorem 4. ***Characterization of characteristic translation-invariant kernels:***

$$\begin{aligned} &\text{A positive definite kernel } \kappa(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{x} - \boldsymbol{y}) \text{ is characteristic } \iff \\ &\mathrm{supp}(\Lambda) = \mathbb{R}^n, \text{ i.e. } \forall \boldsymbol{\omega} \in \mathbb{R}^n,\ \Lambda(\boldsymbol{\omega}) > 0 \end{aligned} \tag{2.42}$$

with Λ *defined in Theorem 3.*

As last step before coming to the sketch, we need a corollary of Theorem 3 that re-writes the γ_κ metric (that we will from now on also write, by abuse of notation, γ_Λ since there is a bijective relation between κ and Λ) as a function of the frequency distribution Λ related to κ ([31], Corollary 4):

$$\gamma_\Lambda(\mathcal{P}, \mathcal{Q}) := \gamma_\kappa(\mathcal{P}, \mathcal{Q}) = \sqrt{\int_{\mathbb{R}^n} |\psi_{\mathcal{P}}(\boldsymbol{\omega}) - \psi_{\mathcal{Q}}(\boldsymbol{\omega})|^2 \Lambda(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}}, \tag{2.43}$$

where $\psi_{\mathcal{P}}(\boldsymbol{\omega})$ is the characteristic function of the pdf $\mathcal{P}$. The interesting link between the sketch operator $\mathcal{A}$ and the frequencies $\boldsymbol{\omega}_j \overset{\text{iid}}{\sim} \Lambda$ is that the sketch distance $\|\mathcal{A}\mathcal{P} - \mathcal{A}\mathcal{Q}\|_2$ is an approximation of the γ_κ metric [2]:

[17] In fact, Theorem 4 implies that the third frequency sampling heuristic $\Lambda^{\mathtt{Ar}}$ defines a kernel that is not characteristic since $\Lambda^{\mathtt{Ar}}(\mathbf{0}) = 0 \Rightarrow \mathrm{supp}(\Lambda^{\mathtt{Ar}}) = \mathbb{R}^n \setminus \{\mathbf{0}\}$. However this frequency distribution produces good results in practice, this can be intuitively explained by the fact that for all characteristic functions, the value af $\boldsymbol{\omega} = \mathbf{0}$ is 1.

$$\frac{1}{m}\|\mathcal{A}\mathcal{P} - \mathcal{A}\mathcal{Q}\|_2^2 = \frac{1}{m}\sum_{j=1}^{m} |\psi_{\mathcal{P}}(\boldsymbol{\omega}_j) - \psi_{\mathcal{Q}}(\boldsymbol{\omega}_j)|^2 \simeq \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim\Lambda} |\psi_{\mathcal{P}}(\boldsymbol{\omega}) - \psi_{\mathcal{Q}}(\boldsymbol{\omega})|^2 = \gamma_\Lambda^2(\mathcal{P}, \mathcal{Q}), \quad (2.44)$$

where the approximation holds for, by law of large numbers, when m is large enough (the empirical mean concentrates around the true mean). This link betwen the sketch $\mathcal{A}$ and the RKHS $\mathcal{H}$ with kernel κ being the Fourier transform of the frequency sampling pattern of the sketch is represented Figure 2.11. The key fact is here that the sketch distance is an approximation of the MMD metric γ_Λ defined of the probability distributions that are sensed by the sketch. The choice of Λ determines the particular nature of the metric used to discriminate the pdfs and is thus a crucial parameter in the sketch design.

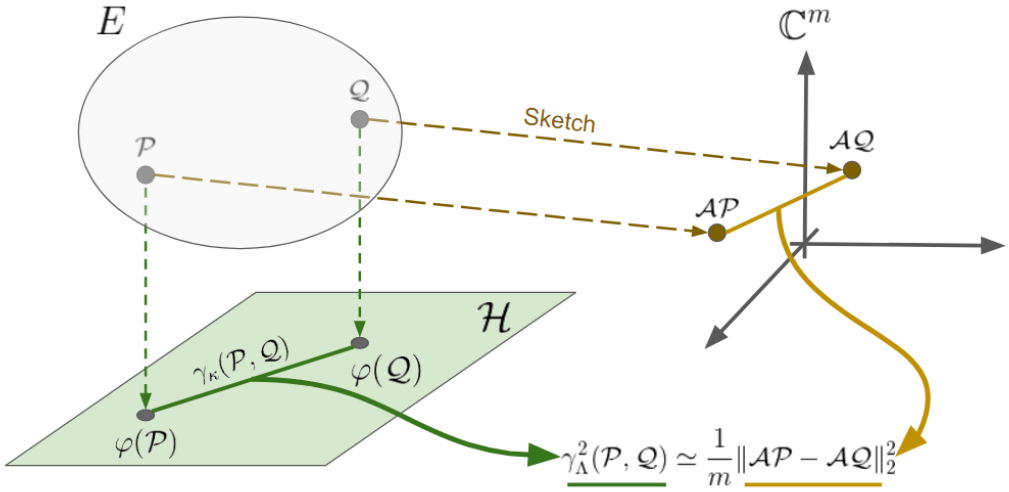


Figure 2.11: This figure represents how the sketch approximates the γ_κ metric. We want to compare the two probability density functions $\mathcal{P}$ and $\mathcal{Q} \in E$ (top left). The way we acquire those distributions is by the sketh operator $\mathcal{A}$, we thus only have access to the m-dimensional vectors $\mathcal{A}\mathcal{P}$ and $\mathcal{A}\mathcal{Q}$ (on the right). However, for m large enough, the distance between those vectors approximates the γ_κ metric in the RKHS $\mathcal{H}$ (bottom left) defined by the frequency selecting pattern Λ (since κ is the Fourier transform of Λ).

Remark. As we will see, when designing the frequency distribution of a sketch, we are not really interested in the high-frequency content and it is desirable to have $\Lambda(\boldsymbol{\omega}) \to 0$ as $\|\boldsymbol{\omega}\| \to \infty$. In this optic, Theorem 4 is "bad news" since it forbids us to have, e.g., $\Lambda(\boldsymbol{\omega}) = 0$ for $\|\boldsymbol{\omega}\| \geq R_0$ for some radius R_0, which would be a nice property for the frequency sampling pattern.

Since in the sketching framework we are selecting Λ (that defines a kernel through Bochner's theorem) based on a small subset of the dataset rather than -such as in traditional machine learning approached- choosing the kernel κ based on a computationally-intensive cross-validation procedure (basically, testing out a lot of different kernels and picking the best), the selection of the frequency distribution can be seen as an "implicit kernel design", and it turn out to be competitive with kernel design by cross validation in terms of performances while being much more faster [2]. In chapter 4, we come back to the kernel to make an interesting interpretation of the objective function of compressive clustering in regard of the kernel. We now start from the rather abstract connexions between RKHS and the sketch to explain more intuitively what this means, in particular in the case of `CKM` where $\mathcal{P}$ and $\mathcal{Q}$ are mixtures of Dirac deltas.

The filter interpretation

At this point, the MMD metric might seem a little bit abstract. In addition to this, it might seem surprising, in the `CKM` algorithm, to try to reconstruct (or approximate) the sketch of the data distribution by a mixture of K Dirac delta functions although it is obvious that the training examples can take more than K different values, so the mixture of K deltas is clearly "wrong" given the dataset. In this subsection we bring a new interpretation ("the filter interpretation") of the sketch that brings answers to those questions.

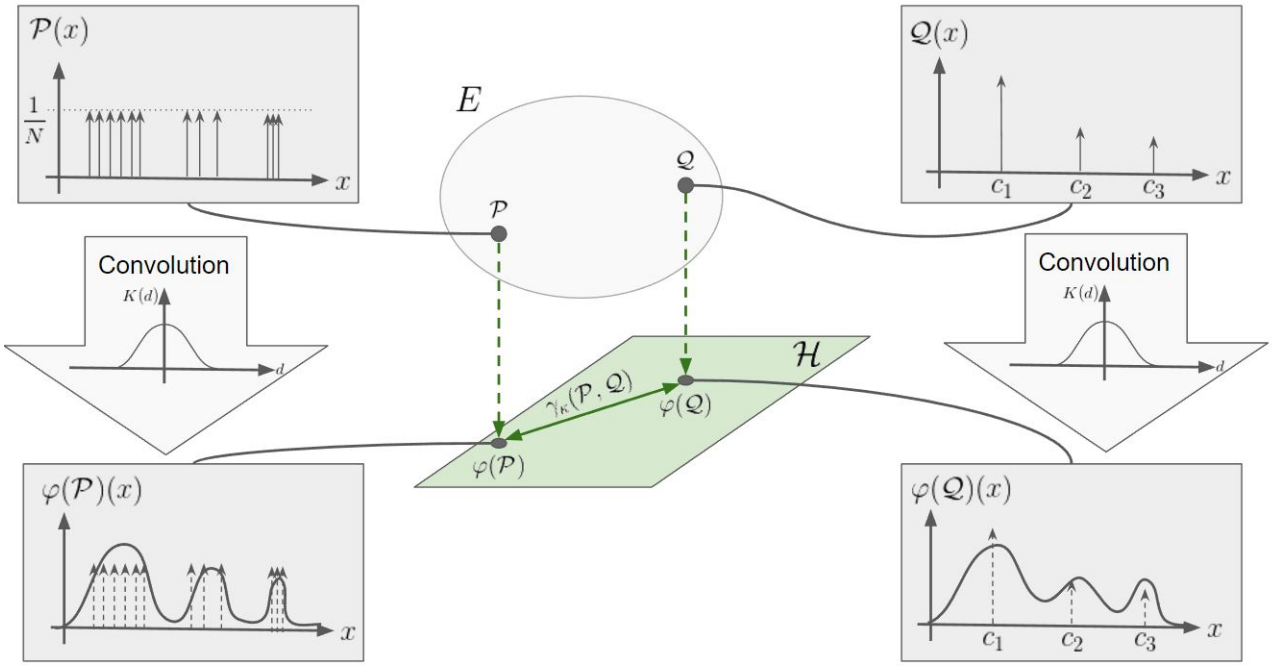


Figure 2.12: The interpretation of the Mean Map as a filtering of the empirical distribution of the dataset $\mathcal{P}$ and the distribution of weighted centroids $\mathcal{Q}$ (here there are three centroids, and we have one-dimensional data). The Maximum Mean Discrepancy metric compares the low-frequency content of the probability density functions (or the convolution of those distributions with the kernel $K(d)$).

The filter interpretation is summarized Figure 2.12. In fact, the Mean Map $\varphi(\mathcal{P})$ associated with a translation-invariant kernel $\kappa(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{x} - \boldsymbol{y})$ corresponds to the *convolution* of $\mathcal{P}$ with the kernel K (note that K is symetric):

$$\varphi(\mathcal{P})(\boldsymbol{x}) = \int_{\mathbb{R}^n} K(\boldsymbol{x} - \boldsymbol{y})\mathcal{P}(\boldsymbol{y})\mathrm{d}\boldsymbol{y} = K(\boldsymbol{x}) * \mathcal{P}(\boldsymbol{x}), \tag{2.45}$$

meaning that, in other words, the mean map is a *filtering* of the input probability distribution $\mathcal{P}$ with a filter whose frequency response is given by $\Lambda(\boldsymbol{\omega})$. Given the shape of $\boldsymbol{\omega}$, we use low-pass filters to modify the probability distributions, then compare the filtered versions. For mixture of deltas in particular, we can hope that the filtered probability distributions are comparable even though the probability distributions are quite different (see Figure 2.12, bottom). With a good filter (i.e., a good choice of Λ) the filtered empirical probability distribution should approximate the true probability distribution of the data, and the filtered mixture of clusters (deltas) should model this distribution approximatively.

Random Fourier Features

As seen above, the RKHS approch to matching probability distributions compares the probability distrubtions through the MMD metric γ_κ for a kernel κ, and we have argued that the sketch distance *approximates* this metric γ_κ. The nature of this approximation is best understood in the light of Random Fourier Features, a method for approximating kernels.

Random Fourier Features [33] is method that maps input vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ to m-dimensional vectors $\boldsymbol{z}(\boldsymbol{x}), \boldsymbol{z}(\boldsymbol{y}) \in \mathbb{R}^m$ for *approximating* translation-invariant kernels $\kappa(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{x} - \boldsymbol{y})$. Building on Bochner's theorem (Theorem 3), which can be re-written as

$$K(\boldsymbol{x} - \boldsymbol{y}) = \int_{\mathbb{R}^n} \Lambda(\boldsymbol{\omega}) e^{-\mathrm{i}\boldsymbol{\omega}^T(\boldsymbol{x}-\boldsymbol{y})} \mathrm{d}\boldsymbol{\omega} = \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim\Lambda} \left[e^{-\mathrm{i}\boldsymbol{\omega}^T\boldsymbol{x}} \left(e^{-\mathrm{i}\boldsymbol{\omega}^T\boldsymbol{y}} \right)^* \right] \tag{2.46}$$

and focusing on real kernels, the Random Fourier Feature vector $\boldsymbol{z}$ is defined as

$$\boldsymbol{z}(\boldsymbol{x}) = \sqrt{\frac{2}{m}} \left[\cos(\boldsymbol{\omega}_1^T \boldsymbol{x} + \xi_1), \cdots, \cos(\boldsymbol{\omega}_m^T \boldsymbol{x} + \xi_m) \right]^T \quad \text{where} \quad \boldsymbol{\omega}_j \overset{\text{iid}}{\sim} \Lambda(\boldsymbol{\omega}), \;\; \xi_j \overset{\text{iid}}{\sim} \mathcal{U}([0, 2\pi]). \tag{2.47}$$

The approximation of the kernel is achieved through the scalar product of the Fourier features of the different input vectors

$$K(\boldsymbol{x} - \boldsymbol{y}) \simeq \boldsymbol{z}(\boldsymbol{x})^T \boldsymbol{z}(\boldsymbol{y}) \tag{2.48}$$

where $\boldsymbol{z}(\boldsymbol{x})^T \boldsymbol{z}(\boldsymbol{y})$ can be seen as an unbiased estimator of the kernel $K(\boldsymbol{x} - \boldsymbol{y})$. The probability of committing a fixed error $\mathbb{P}\left[|K(\boldsymbol{x} - \boldsymbol{y}) - \boldsymbol{z}(\boldsymbol{x})^T \boldsymbol{z}(\boldsymbol{y})| \geq \epsilon \right]$ decreases exponentially with the dimension of the features m , i.e., the estimator converges uniformly [33]. The Random Fourier Features idea is to use the features $\boldsymbol{z}(\boldsymbol{x})$ instead of the much higher dimensional feature map $\varphi(\boldsymbol{x})$, allowing to with feature vectors in a much lower dimension (as we saw, this was unnecessary for SVM learning for example since we then do not need the feature vectors).

The sketch of a dataset can be seen as the pooling (sum) of the Random Fourier Features of the whole dataset (except we are replacing the cosine function by the complex exponential). Another difference between sketching algorithms such as `CKM` and RFF it the fact that RFF are sampling the frequencies from the inverse Fourier transform of a data-independent kernel imposed a priori (that is the main drawback of RFF compared to other and better performing kernel approximation approaches such as the Nyström method [34]) while the sketch operator chooses the frequency distrubtion conditionally on a small observation of the data. In Chapter 4, we come back to the connexions with RFF when we study the true objectife function of `CKM` and its variants.

Chapter 3

Clustering with a 1-bit quantized sketch

In this chapter we present the main goal of this thesis: performing a version of compressive clustering such as in [1] but using the universal quantization function (also called "a square wave") as non-linear periodic signature function for constructing the sketch, instead of the complex exponential. We present the `QCKM` algorithm that solves this problem, and technical questions about this algorithm motivates the study of generalized sketches in Chapter 4.

3.1 Motivation and challenges

3.1.1 Introducing the quantized sketch and motivations

Definition

Recall that the *sketch* Sk of a generic dataset $Y = [\boldsymbol{y}_1, ..., \boldsymbol{y}_L]$ where each column corresponds to one example, is given by a pooling over the L examples of the complex exponential of the examples at different frequencies given by $\Omega = [\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m]$:

$$\mathrm{Sk}\left(Y, \boldsymbol{\beta}\right) = \left[\sum_{l=1}^{L} \beta_l e^{-\mathrm{i}\boldsymbol{\omega}_j^T \boldsymbol{y}_l}\right]_{j=1}^{m} = \exp(-\mathrm{i}\Omega^T Y)\boldsymbol{\beta} \qquad \in \mathbb{C}^m, \tag{3.1}$$

with associated weigths $\boldsymbol{\beta} = [\beta_1, ..., \beta_L]^T$, $\beta_i \geq 0$ and $\sum_i \beta_i = 1$. The objective of this thesis is to replace the complex exponential $e^{-\mathrm{i}t}$ "signature function" by the universal quantization $Q_\Delta(t)$ defined as (2.15) with some quantization step Δ. We also introduce, inspired by [21] a random dither $\boldsymbol{\xi}$ with components $\xi_j \sim \mathcal{U}(0, \Delta)$. The quantized sketch Sk_Q then becomes

$$\mathrm{Sk}_Q\left(Y, \boldsymbol{\beta}\right) = \left[\sum_{l=1}^{L} \beta_l Q_\Delta(\boldsymbol{\omega}_j^T \boldsymbol{y}_l + \xi_j)\right]_{j=1}^{m} = Q_\Delta(\Omega^T Y + \boldsymbol{\xi}\mathbf{1}_L^T)\boldsymbol{\beta} \qquad \in [-1, 1]^m, \tag{3.2}$$

where we note that the sketch values are now real and in the interval $[-1, +1]$. The sketch is thus a *pooling -a sum- over the whole dataset of the 1-bit universal quantization of random projections of the high-dimensional signals* (with an additional dithering). In contrast to Compressive K-Means (CKM) that uses the complex exponential, we name our new clustering approach Quantized Compressive K-Means (QCKM). Defore diving into the implications of doing such a thing we will first discuss why it could be a good idea.

Quantized sketch: why?

In the compressed k-means (CKM) approach [1], clustering can be performed in a time that is independent of the size of the dataset (the amount of data), given that the sketch is already

known. Actually, for very large datasets, computing the sketch becomes the task that is the most computationally demanding, but with the advantage that computing the sketch is a heavily parallelizable task. However, the current approach of CKM still assumes that the whole dataset is at one point acquired (although the sketch can be computed on-line, i.e. it is possible to aquire one example at a time and to forget it once its contribution to the sketch has been computed). Keeping the motivations that led to the theory of CS in mind, this seems to be a huge waste of data acquisition at the sensing step. We are thus investigation the feasability of a sensor (like [13] and [14]) that would, instead of acquiring the signals $\boldsymbol{x}_i$, compute their contribution to the sketch directly (in other words, we want to short-cut the "data acquisition" step).

Using the existing sketch, a sensor computing the usual sketch of a signal $\boldsymbol{x}$ should thus compute $e^{-\mathrm{i}\Omega^T\boldsymbol{x}}$ with $\Omega \in \mathbb{R}^{n\times m}$, where the operation $e^{-\mathrm{i}t}$ is complicated to perform directly in hardware. On the other hand, computing $Q_\Delta(t)$ should be much easier to implement in an hardware sensor. Although this is beyond the scope of this work, ADC technologies such as pipelined (or "folding") ADCs [35], [36] and self-reset ADCs [37],[38] could here be an interesting starting point. We thus believe that computing $Q_\Delta(\Omega^T\boldsymbol{x} + \boldsymbol{\xi})$ instead of $e^{-\mathrm{i}\Omega^T\boldsymbol{x}}$ is much easier to do in hardware, which is the main motivation to introduce the new sketch Sk_Q. The actual design of such an efficient sensor is beyond the scope of this work but an important open question to keep in mind. We believe that changing the signature function $e^{-\mathrm{i}t}$ by another will not nullify the possibility to use the sketch to retrieve the centroids, as long as *the signature function is periodic*, such as $Q_\Delta(t)$.

Could it work?

At first, the (quantized) sketch of a dataset might seem rather strange, and the reader might wonder how the sketch could convey the information needed to perform clustering. We give here a reassuring geometric intuitive interpretation of the information captured by the sketch (quantized or not) by numerical illustrations on a toy example.

In Figure 3.1, the objective function to find the first centroid (step 1) of both the `CKM` algorithm and the `QCKM` algorithms are compared for different sketch sizes m, for a Gaussian radius frequency sampling adapted to each case (we use the sampnling (Gr) and (Gr-h), see later). The first row ($m = 1$) shows how we can interpret the contribution of one single coefficient j of the sketch to the objective: it attracts the centroid to positions (the blue region) where more data samples are located when projected on the waveform $w(\boldsymbol{c}) = (\mathcal{A}\delta_{\boldsymbol{c}})_j$, and we obtain respectively a sinusoidal and a squared wave-shaped objective function.

When adding coefficients to the sketch (the three following rows), different sketch coeffiencts, that is, correlations with different waveforms are combined to refine the objective function. To obtain a "good" objective function fast enough (for m small enough), selecting the frequencies is in fact a crucial step of the sketching procedure. The frequencies must not be too small, nor too high, in order to locate the clusters accurately for acceptable values of m: the frequencies must, in fact, be *adapted to the size of the clusters.*

The reassuring observation is that the objective function of `QCKM` contains enough information to find the good centroids (for $m = 1000$). From Figure 3.1 we can also already guess, as we will formally observe in Chapter 5, that the number of frequencies m to obtain a good estimation of the centroids is, at least for the current choices of freqeuncy distributions, higher for `QCKM` than for `CKM`. Indeed for $m = 100$ the minima of `CKM`s objective function are very close to the true centroids, unlike `QCKM`s objective function. A final comment to be made is the fact that the objective function of `CKM` is very smooth while the one of `QCKM` is a "stairsteps-shaped" objective function.

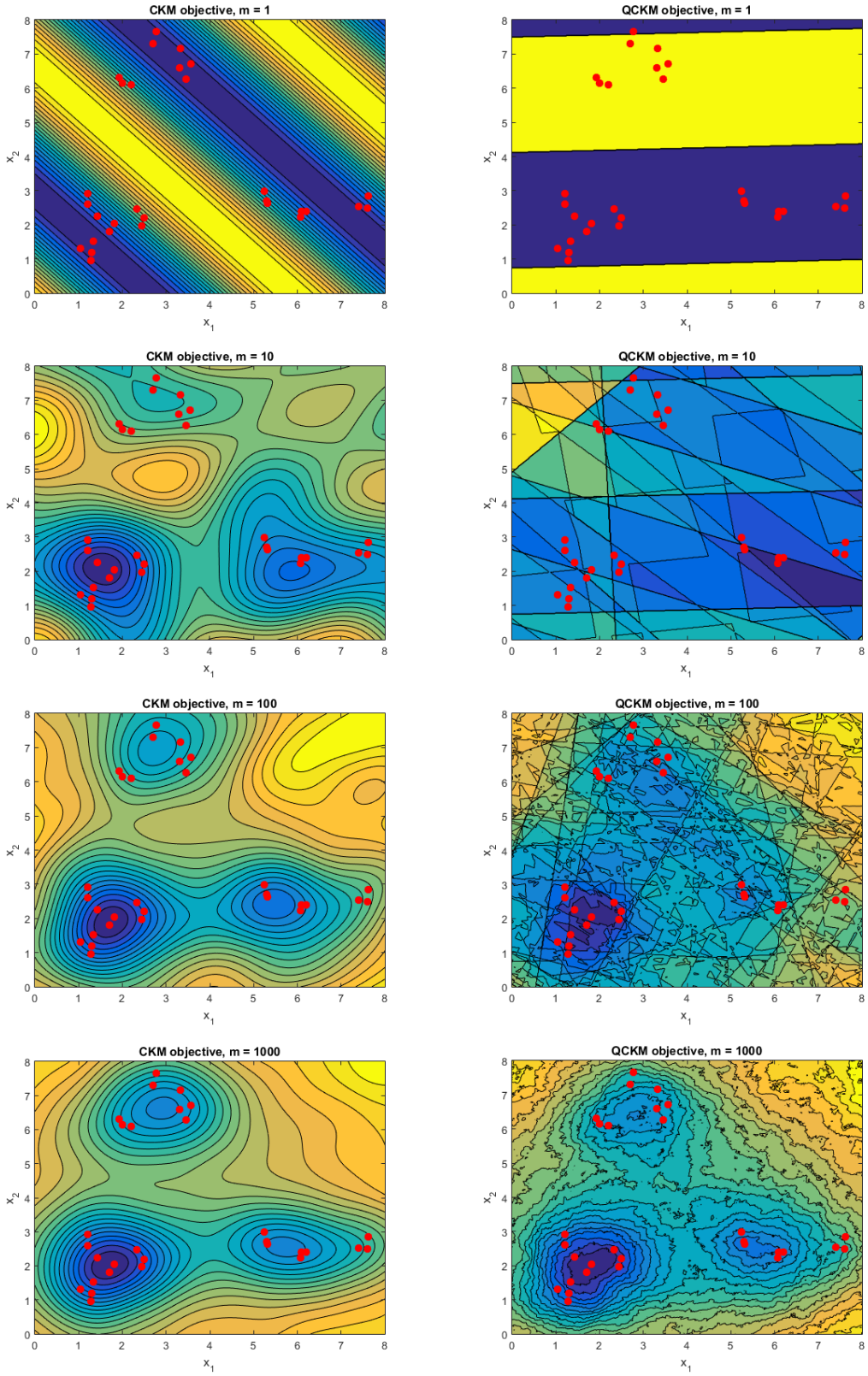


Figure 3.1: Evolution of the objective function to find the first centroid on a 2-dimensional toy example with `CKM` (left) and `QCKM` (right) and for different amounts of frequencies, from top to bottom: $m = 1, 10, 100, 1000$. The minima of the objective functions are the blue regions, and the big red dots represents the dataset X.

3.1.2 Quantized sketching in practice with QCKM and its challenges

Although we have shown, at least intuitively for now, that provided its dimension m is high enough the quantized 1-bit sketch contains the information necessary to find centroids, we have not yet addressed *how* the centroids can be found in practice. Adapting the CKM algorithm (algorithm 5) to our quantized sketch raises some technical questions:

- How should one choose the *parameters* of the sketch $\mathrm{Sk}_Q(Y, \boldsymbol{\beta})$ as defined (3.2)? In particular, what quantization step Δ must be chosen? How should the random "frequencies" Ω be drawn? How many frequencies m shoud be used?
- How should one adapt the CKM algorithm to account for the discontinuous nature of the quantizer? As mentioned when we presented it in Chapter 2, CKM relies on solving optimization problems in a continuous space, that are solved with quasi-Newton methods that require the gradient of the objective function. Unfortunately in QCKM, the quantized sketch version of CKM, the objective function becomes discontinuous with respect to the decision variables, and the gradient of this function is therefore ill-defined.

Leaving aside those questions for the next section, the QCKM algorithm is presented below, indexed as algorithm 6. The differences with CKM are highlighted. Note in particular that the norm of the quantized vector of a centroid $\boldsymbol{c}$ is always $\|\mathrm{Sk}_Q(\boldsymbol{c}, 1)\| = \sqrt{\sum_{j=1}^m (\pm 1)^2} = \sqrt{m}$ so we drop it from the problem.

3.2 Solutions to the challenges

3.2.1 Parameters of the sketch

We answer here how the quantized sketch should be constructed, in particular, how to choose the parameters Δ, Ω and m. The choice of the two latter parameters are in fact coming from results in Chapters 4 and 5, but for the sake of completeness of this section we already present the conclusions here.

Quantization stepsize Δ

How should the quantizer stepsize Δ be chosen, given that the sketch entry j records the pooling of the values $Q_\Delta(\boldsymbol{\omega}_j^T \boldsymbol{x}_i + \xi_j)$? Actually, it is interesting to note that, from the definition of universal quantization, $Q_\Delta(t) = Q_1(\frac{t}{\Delta})$, therefore for $\Delta \neq \Delta'$,

$$Q_\Delta(\boldsymbol{\omega}_j^T \boldsymbol{x}_i + \xi_j) = Q_1\left(\frac{\boldsymbol{\omega}_j^T \boldsymbol{x}_i}{\Delta} + \frac{\xi_j}{\Delta}\right) = Q_{\Delta'}\left(\frac{\Delta'}{\Delta}(\boldsymbol{\omega}_j^T \boldsymbol{x}_i) + \frac{\Delta'}{\Delta}\xi_j\right) = Q_{\Delta'}\left(\boldsymbol{\omega}_j'^T \boldsymbol{x}_i + \xi_j'\right) \quad (3.3)$$

with $\xi_j \sim \mathcal{U}([0, \Delta])$, $\xi_j' \sim \mathcal{U}([0, \Delta'])$ and $\boldsymbol{\omega}_j' = \frac{\Delta'}{\Delta}\boldsymbol{\omega}_j$. What (3.3) means is that changing the stepsize Δ does not modify the sketch if the frequencies $\boldsymbol{\omega}_j$ are scaled proportionally. This is of course reminiscent of [23] where the embedding properties only depend on $\frac{\Delta}{\sigma}$ and not on Δ alone. Note that it is also possible to obtain the same sketch for a different Δ' by scaling the dataset $\boldsymbol{x}_i' = \frac{\Delta'}{\Delta}\boldsymbol{x}_i$ instead of scaling the frequencies. In short, choosing Δ arbitrarily has no impact on future discussions as long as we keep in mind that we do possibly need to rescale the frequencies or the dataset, and the whole "decision" can be put in this rescaling.

Without loss of generality, we will therefore choose $\Delta = \pi$, resulting in a signature function $Q_\Delta(t)$ that has the same period ($T = 2\pi$) as $e^{-\mathrm{i}t}$, the signature function in CKM. This will also simplify notations in the Fourier series expressions in Chapter 4.

Algorithm 6: `QCKM` : `CKM` adapted to 1-bit sketches relying on universal quantization.

input : Empirical **quantized** sketch $\hat{\boldsymbol{z}} = \mathrm{Sk}_Q\left(X, \mathbf{1}_N/N\right)$, frequencies Ω, number of centroids K, bounds $\boldsymbol{l}$, $\boldsymbol{u}$

output : A set of centroids C and weights $\boldsymbol{\alpha}$ that locally minimizes the **quantized** sketch mismatch $\|\hat{\boldsymbol{z}} - \mathrm{Sk}_Q\left(C, \boldsymbol{\alpha}\right)\|_2^2$

Initialization $\hat{\boldsymbol{r}} \leftarrow \hat{\boldsymbol{z}}$, $C \leftarrow \emptyset$ *(Initialize the residual and the support (centroids))* ;

for *t = 1,...,2K* **do**

  Step 1 : find the new centroid $\boldsymbol{c}$ with highest correlation with residual **by solving approximatively the discontinuous problem**:

  $\boldsymbol{c} \simeq \arg\max_{\boldsymbol{c}} < \mathrm{Sk}_Q\left(\boldsymbol{c}, 1\right), \hat{\boldsymbol{r}} >$ s.t. $\boldsymbol{l} \leq \boldsymbol{c} \leq \boldsymbol{u}$ ***(The sketch is always real)***

  Step 2 : add it to the support:

  $C = C \cup \{\boldsymbol{c}\}$

  Step 3 : Reduce support by Hard Thresholding to keep sparsity ("Replacement" step):

  if $|C| > K$ **then**

    $\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}_+^{|C|}} \left\| \hat{\boldsymbol{z}} - \sum_{k=1}^{|C|} \beta_k \mathrm{Sk}_Q\left(\boldsymbol{c}_k, 1\right) \right\|_2$

    $C \leftarrow$ set of K centroids $\boldsymbol{c}_k$ corresponding to K largest magnitude values of $\boldsymbol{\beta}$

  end

  Step 4 : Find the optimal coefficients given the support (by projection):

  $\boldsymbol{\alpha} = \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}_+^{|C|}} \left\| \hat{\boldsymbol{z}} - \sum_{k=1}^{|C|} \alpha_k \mathrm{Sk}_Q\left(\boldsymbol{c}_k, 1\right) \right\|_2$

  Step 5 : Global gradient descent **by solving approximatively the discontinuous problem**:

  $(C, \boldsymbol{\alpha}) \simeq \arg\min_{C, \boldsymbol{\alpha}} \left\| \hat{\boldsymbol{z}} - \sum_{k=1}^{|C|} \alpha_k \mathrm{Sk}_Q\left(\boldsymbol{c}_k, 1\right) \right\|_2$ s.t. $\boldsymbol{l} \leq \boldsymbol{c} \leq \boldsymbol{u}$

  $\hat{\boldsymbol{r}} = \hat{\boldsymbol{z}} - \sum_{k=1}^{|C|} \alpha_k \mathrm{Sk}_Q\left(\boldsymbol{c}_k, 1\right)$ *(Update residual, go back to step 1)*

end

Drawing the random projections vectors or "frequencies" Ω

With $\Delta = \pi$, the "sensing functions" used in `CKM` and `QCKM` are, for the same frequencies Ω, not too different (we replace the complex exponential by a real square wave, but the frequency of those sensing functions is the same). We show in Chapter 5 that re-using the frequencies Ω from `CKM` in `QCKM` does indeed yield decent results, but is it possible to do better?

In `CKM`, the frequencies $\Omega = [\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_m]$ are drawn from a distribution $\boldsymbol{\omega}_j \overset{\text{iid}}{\sim} \Lambda$ where $\Lambda(\boldsymbol{\omega})$ is one of the three heuristic that matches, in "distilled sensing-like" approach, some small sample of the dataset. Using the Fourier series decomposition of the sketch, we transpose in Chapter 4 the heuristic choices of $\Lambda(\boldsymbol{\omega})$ in `CKM` to `QCKM`, obtain new frequency densities $\Lambda_Q(\boldsymbol{\omega})$ adapted to the universal quantization signature function. In Chapter 5 we show that the performances of `QCKM` with $\Lambda_Q(\boldsymbol{\omega})$ are improved compared to using $\Lambda(\boldsymbol{\omega})$.

Number of frequencies m

Empirically, it has been shown that `CKM` requires $m = \mathcal{O}(nK)$ frequencies to produce clustering results that compete with the state-of-the-art `k-means++` algorithm [1]. In particular m is independent of N, the number of training examples, which makes `CKM` a particularly appealing strategy for very large datasets. In this work we show -empirically again- in Chapter 5 that for `QCKM`, the rule $m = \mathcal{O}(nK)$ still hold, but with a higher hidden constant: more precisely we observe the rule $m_{\texttt{QCKM}} \gtrsim 20...50\, m_{\texttt{CKM}}$ in order to obtain similar clustering performances.

3.2.2 From CKM to QCKM: optimization of discontinuous objective functions

As illustrated Figure 3.1, in QCKM the objective functions of the optimization problems to solve become discontinuous and therefore non-differentiable and L-BFGS can't be used as optimization routine. To circumvent this problem, we propose two strategies that approximate the objective function by a differentiable approximation, hence allowing us to compute a gradient and to solve this approximated problem with the L-BFGS method.

Strategy 1: Smoothing the universal quantization operator with a filter

The first strategy porposed consists in *smoothing* the discontinuous function. The only part of the gradients of CKM that we need to modify (those gradients are computed Appendix B.1) is the Jacobian of the sketch of a single centroid $\boldsymbol{c}$: we need $\boldsymbol{\nabla}_{\boldsymbol{c}}\mathrm{Sk}_Q\left(\boldsymbol{c},1\right) = \boldsymbol{\nabla}_{\boldsymbol{c}} f(\Omega^T\boldsymbol{c}+\boldsymbol{\xi})$ with $f(t) = Q_\Delta(t)$. With the chain rule it comes

$$\boldsymbol{\nabla}_{\boldsymbol{c}}\left(\mathrm{Sk}_Q\left(\boldsymbol{c},1\right)\right)_j = \boldsymbol{\nabla}_{\boldsymbol{c}} f(\boldsymbol{\omega}_j^T\boldsymbol{c}+\xi_j) = \left.\frac{\partial f(t)}{\partial t}\right|_{t=\boldsymbol{\omega}_j^T\boldsymbol{c}+\xi_j} \cdot \boldsymbol{\nabla}_{\boldsymbol{c}}\left(\boldsymbol{\omega}_j^T\boldsymbol{c}+\xi_j\right) = \boldsymbol{\omega}_j \left.\frac{\partial f(t)}{\partial t}\right|_{t=\boldsymbol{\omega}_j^T\boldsymbol{c}+\xi_j} \tag{3.4}$$

where we have a problem since $f(t)$ is discontinuous hence technically not differentiable. In fact, we can express $\frac{\partial f(t)}{\partial t}$ theoretically with Dirac delta functions using the expression of $f(t) = Q_\Delta(t)$ as a combination of step functions (2.16):

$$f(t) = 2\sum_{k\in\mathbb{Z}}(-1)^k u(t-k\Delta) \Rightarrow \frac{\partial f(t)}{\partial t} = 2\sum_{k\in\mathbb{Z}}(-1)^k \delta(t-k\Delta) \tag{3.5}$$

but this expression is of course not applicable in a practical algorithm on a computer. We propose thus to smoothen $f(t)$ by convolution with a function $g(t)$ to obtain $\hat{f}(t) = f(t) * g(t)$, and we can then replace

$$\frac{\partial f(t)}{\partial t} \simeq \frac{\partial \hat{f}(t)}{\partial t} = \frac{\partial}{\partial t}\left(f(t)*g(t)\right) = \frac{\partial f(t)}{\partial t} * g(t) = 2\sum_{k\in\mathbb{Z}}(-1)^k g(t-k\Delta). \tag{3.6}$$

The function $g(t)$ should satify $\int_{\mathbb{R}} g(t)\mathrm{d}t = 1$ in order to leave the DC component of $f(t)$ untouched (if $f(t)$ is a constant, $\hat{f}(t)$ should yield the same constant). We can also see this strategy as computing a moving average of $f(t)$ before differentiation, with weights given by $g(t)$.

For example, we can chose $g(t)$ to be 0 everywhere except in $t \in [-\epsilon, +\epsilon]$ where it is constant; this is equivalent to replace the disconinuity of by a linear slope of steepness ϵ. The parameter ϵ can in theory be chosen arbitrarily small but in practice, it should be chosen large enough in order to avoid numerical instabilities.

Remark. Since the second strategy that we present hereafter is more interesting to study from a theoretical point of view (see Chapter 4), we will focus on the second strategy in the following chapters. However, this first strategy has also been implemented in Matlab and yields decent results, so it is a viable strategy that could be considered in real applications, even though we won't come back to it in the rest of this work.

Strategy 2 : Approximating the universal quantization as a Fourier series

We present here another way to construct $\hat{f}(t)$, a smooth approximation of $f(t) = Q_\Delta(t)$ to replace $\frac{\partial f(t)}{\partial t}$ by $\frac{\partial \hat{f}(t)}{\partial t}$. Since $f(t)$ is periodic with period $T = 2\Delta$, it can be written as a Fourier series with coefficients given by Proposition 1.

Proposition 1. ***The Fourier series coefficients of the universal quantization function.*** *The universal quantization function $f(t) = Q_\Delta(t)$, of period $T = 2\Delta$ can be written in a Fourier series*

$$f(t) = \sum_{k=-\infty}^{+\infty} F_k e^{\mathrm{i}\frac{2\pi}{2\Delta}kt} = \sum_{\substack{k=-\infty \\ k\ odd}}^{+\infty} \frac{2\mathrm{i}}{k\pi}\, e^{\mathrm{i}\frac{\pi}{\Delta}kt} \tag{3.7}$$

where the Fourier series coefficients coefficients are independent of Δ and given by

$$F_k = \begin{cases} 0 & \text{if } k \text{ is even,} \\ \frac{2\mathrm{i}}{k\pi} & \text{if } k \text{ is odd.} \end{cases} \tag{3.8}$$

Proof. By definition, the Fourier coefficients are (with $T = 2\Delta$)

$$\begin{aligned} F_k &= \frac{1}{T}\int_0^T f(t)e^{-\mathrm{i}k\frac{2\pi}{T}t}\mathrm{d}t = \frac{1}{T}\int_0^\Delta (-1)e^{-\mathrm{i}k\frac{2\pi}{T}t}\mathrm{d}t + \frac{1}{T}\int_\Delta^{2\Delta} (+1)e^{-\mathrm{i}k\frac{2\pi}{T}t}\mathrm{d}t \\ &= \frac{1}{T}\left[\frac{+1}{\mathrm{i}k\frac{2\pi}{T}}e^{-\mathrm{i}k\frac{2\pi}{2\Delta}t}\right]_0^\Delta + \frac{1}{T}\left[\frac{-1}{\mathrm{i}k\frac{2\pi}{T}}e^{-\mathrm{i}k\frac{2\pi}{2\Delta}t}\right]_\Delta^{2\Delta} \\ &= \frac{1}{\mathrm{i}k2\pi}\left(e^{-\mathrm{i}k\pi} - 1\right) - \frac{1}{\mathrm{i}k2\pi}\left(1 - e^{-\mathrm{i}k\pi}\right) \\ &= \frac{\mathrm{i}}{k\pi}\left(1 - (-1)^k\right) = \begin{cases} 0 & \text{if k is even,} \\ \frac{2\mathrm{i}}{k\pi} & \text{if k is odd.} \end{cases} \end{aligned} \tag{3.9}$$

□

We thus construct the approximation $\hat{f}(t)$ as a truncated Fourier series, where we truncate the infinite summation on k to K_{coef} coefficients:

$$\hat{f}(t) = \sum_{k=-K_{coef}}^{K_{coef}} F_k e^{\mathrm{i}\frac{2\pi}{2\Delta}kt} \tag{3.10}$$

with F_k the coefficients in Proposition 1. We leave an extensive discussion of this strategy for the next chapter where we replace the signature function of the sketch by *any periodic function*, i.e. any function that can be represented as a Fourier series. Empirical results of strategy 2 are then presented in Chapter 5.

Chapter 4

Generalized sketches using any periodic signature function

As seen in Chapter 2, it is possible with the `CKM` algorithm to obtain the centroids defining the clustering of a dataset from a nonlinear sketch of this dataset. This sketch is the result of pooling the complex exponential of the examples $\boldsymbol{x}_i$ (vectors in $\mathbb{R}^n$) at m different frequencies $\boldsymbol{\omega}_j$ over the whole dataset X containing N examples:

$$\mathrm{Sk}_\Omega\left(X, \mathbf{1}_N/N\right) = \left[\frac{1}{N}\sum_{i=1}^{N} e^{-\mathrm{i}\boldsymbol{\omega}_j^T \boldsymbol{x}_i}\right]_{j=1}^{m} = \frac{1}{N} e^{-\mathrm{i}\Omega^T X}\mathbf{1}_N \qquad \in \mathbb{C}^m \tag{4.1}$$

where $\Omega = [\boldsymbol{\omega}_1 \cdots \boldsymbol{\omega}_m] \in \mathbb{R}^{n\times m}$ is the frequencies matrix and $X = [\boldsymbol{x}_1 \cdots \boldsymbol{x}_N] \in \mathbb{R}^{n\times N}$ is the dataset. Note that we added a subscript Ω to the sketch symbol to avoid confusion in the rest of this chapter. In Chapter 3, we suggested to modify this sketch by replacing the signature function of the data, the complex exponential $e^{-\mathrm{i}t}$, by the universal quantization function $Q_\Delta(t)$, arguing that the key ingredient of a signature function is its periodicity.

In this context, we will in this chapter study what happens when the signature function is *any* periodic function $f(t)$ of period T. Then, in the second section of this chapter, we will apply the results found in the general case to the universal quantization function to get new insights about the quantized sketch scenario. This will allow us in particular to propose new heuristics to select the frequencies of the sketch, heuristics that are adapted to the modified signature function. Finally, in the third section of this chapter, we will list and characterize the different error terms that appear in the whole compressed learning scheme.

4.1 Theoretical results for a general periodic signature function

4.1.1 Generalized sketching operator

Changing the sketching operator changes the way the information about the probability density function of interest $\mathcal{P}$ is captured. In this section we show that the generalized sketch captures more high-frequency content information about $\mathcal{P}$ than the usual sketch, and that the sketch is not, as before, a sampling of the Fourier transform of $\mathcal{P}$ but a generalized transorm with arbitrary (periodic) basis functions.

Sketch from the data First, we are going to see what changes in the sketch that we compute from the data. When we replace the complex exponential by a generic periodic function $f(t) : \mathbb{R} \to \mathbb{C}$ of period T, we obtain a new sketch:

$$\overline{\mathrm{Sk}}_{\Omega,\boldsymbol{\xi}}\left(X, \mathbf{1}_N/N\right) = \left[\frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{\omega}_j^T \boldsymbol{x}_i + \xi_j)\right]_{j=1}^{m} = \frac{1}{N} f(\Omega^T X + \boldsymbol{\xi}\mathbf{1}_N^T)\mathbf{1}_N \qquad \in \mathbb{C}^m \tag{4.2}$$

where, by abuse of notation, f is applied component-wise in the outer right handside. In this case, we have added a random *dither* $\boldsymbol{\xi}$ with *iid* entries uniformly distributed between 0 and the period of f, i.e. $\xi_j \overset{\text{iid}}{\sim} \mathcal{U}([0,T])$, for reasons that will become clear later on.

Since $f(t)$ is a periodic function, it can be expanded as a Fourier series with coefficients $F_k \in \mathbb{C}$:

$$f(t) = \sum_{k=-\infty}^{\infty} F_k e^{\mathrm{i}k\frac{2\pi}{T}t} \quad \text{where} \quad F_k = \frac{1}{2\pi}\int_0^T f(t) e^{-\mathrm{i}k\frac{2\pi}{T}t}\mathrm{d}t. \tag{4.3}$$

Keeping this expansion in mind, the sketching operator using $f(t)$ as signature function can be re-written (as a reminder, $\circ$ is the Hadamard product or "componenent-wise" product between vectors or matrices):

$$\begin{aligned}
\overline{\mathrm{Sk}}_{\Omega,\boldsymbol{\xi}}\left(X, \mathbf{1}_N/N\right) &= \left[\frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{\omega}_j^T \boldsymbol{x}_i + \xi_j)\right]_{j=1}^{m} = \left[\frac{1}{N}\sum_{i=1}^{N}\sum_{k=-\infty}^{+\infty} F_k e^{\mathrm{i}k\frac{2\pi}{T}(\boldsymbol{\omega}_j^T\boldsymbol{x}_i + \xi_j)}\right]_{j=1}^{m} \\
&= \sum_{k=-\infty}^{+\infty} F_k \left[\frac{1}{N}\sum_{i=1}^{N} e^{\mathrm{i}k\frac{2\pi}{T}\boldsymbol{\omega}_j^T\boldsymbol{x}_i} e^{\mathrm{i}k\frac{2\pi}{T}\xi_j}\right]_{j=1}^{m} \\
&= \sum_{k=-\infty}^{+\infty} F_k \left[\frac{1}{N}\sum_{i=1}^{N} e^{\mathrm{i}k\frac{2\pi}{T}\boldsymbol{\omega}_j^T\boldsymbol{x}_i}\right]_{j=1}^{m} \circ \left[e^{\mathrm{i}k\frac{2\pi}{T}\xi_j}\right]_{j=1}^{m} \\
&= \sum_{k=-\infty}^{+\infty} F_k \,\mathrm{Sk}_{-k\frac{2\pi}{T}\Omega}\left(X, \mathbf{1}_N/N\right) \circ e^{\mathrm{i}k\frac{2\pi}{T}\boldsymbol{\xi}}
\end{aligned} \tag{4.4}$$

The last line relates the new sketch operator $\overline{\mathrm{Sk}}_{\Omega,\boldsymbol{\xi}}\left(X, \mathbf{1}_N/N\right)$ that uses a general periodic function f to the usual sketch $\mathrm{Sk}_{\Omega}\left(X, \mathbf{1}_N/N\right)$ that uses the complex exponential. This new sketch consists in a sum of the usual sketches over multiples of the frequency set Ω, weighted by the Fourier coefficients F_k, and with an additional argument shift of $k\frac{2\pi}{T}\xi_j$ for each component j.

Sketching a probability distribution The previous paragraph shows how to compute the sketch of a set of points, but what really interests us for a theoretical analysis is the sketching operator $\mathcal{A}_\Omega$ defined on a probability distribution $\mathcal{P}$ (that is *approximated* by the sketch of the dataset of realisations of this distribution). As a reminder, the usual sketching operator $\mathcal{A}_\Omega\mathcal{P}$ linear with respect to probability distributions $\mathcal{P}$ is a sampling of the characteristic function $\psi_{\mathcal{P}}(\boldsymbol{\omega})$, so we have

$$\mathcal{A}_\Omega\mathcal{P} = \left[\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{P}} \; e^{-\mathrm{i}\boldsymbol{\omega}_j^T\boldsymbol{x}}\right]_{j=1}^{m} = \left[\int_{\mathbb{R}^n} e^{-\mathrm{i}\boldsymbol{\omega}_j^T\boldsymbol{x}}\,\mathcal{P}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\right]_{j=1}^{m} = \left[\psi_{\mathcal{P}}(\boldsymbol{\omega}_j)\right]_{j=1}^{m} \tag{4.5}$$

where the frequencies are drawn from some distribution $\boldsymbol{\omega}_j \overset{\text{iid}}{\sim} \Lambda$. In our generalization context, we replace $e^{-\mathrm{i}t}$ by $f(t)$, and add a dither term, so the new operator becomes

$$\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P} = \left[\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{P}} \; f(\boldsymbol{\omega}_j^T\boldsymbol{x} + \xi_j)\right]_{j=1}^{m} = \left[\int_{\mathbb{R}^n} f(\boldsymbol{\omega}_j^T\boldsymbol{x} + \xi_j)\,\mathcal{P}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\right]_{j=1}^{m}.$$

Similarly to what we did with the data sketch, we can re-write this sketch as a function of the usual sketching operator by writing $f(t)$ in its Fourier series form:

$$
\begin{aligned}
\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P} &= \left[\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{P}} \sum_{k=-\infty}^{+\infty} F_k e^{\mathrm{i}k\frac{2\pi}{T}(\boldsymbol{\omega}_j^T\boldsymbol{x}+\xi_j)}\right]_{j=1}^{m} \\
&= \left[\sum_{k=-\infty}^{+\infty} F_k e^{\mathrm{i}k\frac{2\pi}{T}\xi_j} \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{P}} e^{\mathrm{i}k\frac{2\pi}{T}\boldsymbol{\omega}_j^T\boldsymbol{x}}\right]_{j=1}^{m} \\
&= \sum_{k=-\infty}^{+\infty} F_k \left[e^{\mathrm{i}k\frac{2\pi}{T}\xi_j}\, \psi_{\mathcal{P}}\left(-k\frac{2\pi}{T}\boldsymbol{\omega}_j\right)\right]_{j=1}^{m} \\
&= \sum_{k=-\infty}^{+\infty} F_k \left[e^{\mathrm{i}k\frac{2\pi}{T}\xi_j}\right]_{j=1}^{m} \circ \mathcal{A}_{-k\frac{2\pi}{T}\Omega}.
\end{aligned}
$$

In words, the generalized sketching operator $\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P}$ is computing:

1. the usual sketches $\mathcal{A}_\Omega\mathcal{P}$ but where we changed the frequencies $\boldsymbol{\omega}_j \in \Omega$ by all different multiples of $-\frac{2\pi}{T}\boldsymbol{\omega}_j$,
2. then each component of those sketches are affected by a (random) phase shift due to the multiplication by $e^{\mathrm{i}k\frac{2\pi}{T}\xi_j}$ (different for each multiple k and component j),
3. finally, the phase-shifted sketches at all multiples of $-\frac{2\pi}{T}\boldsymbol{\omega}_j$ are summed over all multiples k with complex weights F_k, the Fourier coefficents of the signature function.

In short, we can interpret the new sketch as a weighted sum of the old sketches at different frequencies. This new sketch "captures" therefore the information about $\mathcal{P}$ at higher frequencies than the original sketch. Note that we still have a link between $\overline{\mathcal{A}}$ and $\overline{\mathrm{Sk}}_{\Omega,\boldsymbol{\xi}}(X, \mathbf{1}_N/N)$, since $\overline{\mathcal{A}}\hat{\mathcal{P}} = \overline{\mathrm{Sk}}_{\Omega,\boldsymbol{\xi}}(X, \mathbf{1}_N/N)$ if $\hat{\mathcal{P}}(\boldsymbol{x}) = \frac{1}{N}\sum_{\boldsymbol{x}_i\in X}\delta(\boldsymbol{x}-\boldsymbol{x}_i)$ is the empirical probability distribution function of the dataset X.

Coming back to the interpretation where the sketch of a probability density function $\mathcal{P}$ computes generalized moments of $\mathcal{P}$ [26], based on given functions M_j defining the moments, we had previously that

$$
\mathcal{A}_\Omega : \mathcal{P} \mapsto \mathcal{A}_\Omega\mathcal{P} := \frac{1}{\sqrt{m}}\left[\int_{\mathbb{R}^n} M_1 \mathrm{d}\mathcal{P}, \cdots, \int_{\mathbb{R}^n} M_m \mathrm{d}\mathcal{P}\right]^T \quad \text{with} \quad M_j(\boldsymbol{x}) = \sqrt{m}\, e^{-\mathrm{i}\boldsymbol{\omega}_j^T\boldsymbol{x}}, \tag{4.6}
$$

we can also now interpret our new sketching operator as a computation of different moments based on new functions $\overline{M}_j$, that we will try to match as in GeMM:

$$
\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}} : \mathcal{P} \mapsto \overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P} := \frac{1}{\sqrt{m}}\left[\int_{\mathbb{R}^n} \overline{M}_1 \mathrm{d}\mathcal{P}, \cdots, \int_{\mathbb{R}^n} \overline{M}_m \mathrm{d}\mathcal{P}\right]^T \quad \text{with} \quad \overline{M}_j(\boldsymbol{x}) = \sqrt{m}\, f(\boldsymbol{\omega}_j^T\boldsymbol{x} + \xi_j). \tag{4.7}
$$

Before we chose particular moments that corresponded to a sampling of the characteristic function of the random variable from which the learning example are realizations. Now we are replacing the characteristic function by something else. For example, in the context of QCKM, instead of computing samples of the Fourier transform of $\mathcal{P}$, we could say that we are sampling another transform of $\mathcal{P}$, a transform that uses square-shaped basis functions instead of sinusoidal ones; we will note this transform $\mathcal{S}$ (for **S**quare).

4.1.2 Generalized objective function in the optimization problem

In the logical continuation of the description of $\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P}$, the new way we are sensing a probability distribution $\mathcal{P}$, we observe how this affects the general "sketch matching" optimization problem to solve

$$\mathcal{Q}^* = \arg\min_{\mathcal{Q}} \|\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P} - \overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{Q}\|_2^2 \quad \text{s.t.} \quad \mathcal{Q} \in \mathcal{G}_K. \tag{4.8}$$

We first explain how the sketch distance in (4.8) captures the mismatch between a transform of the probability density functions that can be seen as a generalized characteristic function, then how the objective function is a weighted sum of the usual sketch distances. Finally we explain how the γ-metric between probability distributions is changed. In a later section we come back to the objective function applied to the particular case of clustering (where the pdfs are mixtures of deltas) in the light of the generalized sketch kernel.

Previously (in (2.44)), we have seen that the objective function, the ℓ_2-norm of the difference the sketches of probability distributions $\mathcal{P}$ and $\mathcal{Q}$, is (up to a factor m) the empirical *mean amplitude squared of the difference of probability characteristic functions $\mathcal{P}$ and $\mathcal{Q}$*. This empirical mean thus approximates, by law of large numbers, the true mean taken over the random variable $\boldsymbol{\omega}$ (that we called the γ distance):

$$\frac{1}{m}\|\mathcal{A}_\Omega\mathcal{P} - \mathcal{A}_\Omega\mathcal{Q}\|_2^2 = \frac{1}{m}\sum_{j=1}^{m} |\psi_\mathcal{P}(\boldsymbol{\omega}_j) - \psi_\mathcal{Q}(\boldsymbol{\omega}_j)|^2 \simeq \mathbb{E}_{\boldsymbol{\omega}\sim\Lambda}\left[|\psi_\mathcal{P}(\boldsymbol{\omega}) - \psi_\mathcal{Q}(\boldsymbol{\omega})|^2\right] \tag{4.9}$$

with ψ_ν is the characteristic function of the probability distribution function ν. In our case, we have $(\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P})_j = \sum_k F_k e^{\mathrm{i}k\frac{2\pi}{T}\xi_j}\ \psi_\mathcal{P}\left(-k\frac{2\pi}{T}\boldsymbol{\omega}_j\right)$, thus, the distance between the sketches of probability distrubtions $\mathcal{P}$ and $\mathcal{Q}$ gives (by comparing the empirical average with the true expectation)

$$\frac{1}{m}\|\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P} - \overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{Q}\|_2^2 = \frac{1}{m}\sum_{j=1}^{m} \left|\overline{\psi}_\mathcal{P}(\boldsymbol{\omega}_j;\xi_j) - \overline{\psi}_\mathcal{Q}(\boldsymbol{\omega}_j;\xi_j)\right|^2 \simeq \mathop{\mathbb{E}}_{\boldsymbol{\omega},\xi}\left[\left|\overline{\psi}_\mathcal{P}(\boldsymbol{\omega};\xi) - \overline{\psi}_\mathcal{Q}(\boldsymbol{\omega};\xi)\right|^2\right] \tag{4.10}$$

where we define the following ("generalized characteristic") function of ω with parameter ξ

$$\overline{\psi}_\mathcal{P}(\boldsymbol{\omega};\xi) = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{P}} f(\boldsymbol{\omega}^T\boldsymbol{x} + \xi) = \sum_k F_k e^{\mathrm{i}k\frac{2\pi}{T}\xi}\psi_\mathcal{P}(-k\frac{2\pi}{T}\boldsymbol{\omega}). \tag{4.11}$$

The main point here is that the sketch of the data defines the distance measure that we will use to compare the probability distributions $\mathcal{P}$ and $\mathcal{Q}$, so a new sketch means a different distance measure (we will come back to this when developping the generalized γ-distance). This shows clearly that we are generalizing the characteristic function $\psi_\mathcal{P}$ (i.e. the Fourier transform of $\mathcal{P}$) by replacing it with a more general function $\overline{\psi}_\mathcal{P}$ that projects $\mathcal{P}(\boldsymbol{x})$ on generic periodic basis functions $f(\boldsymbol{\omega}^T\boldsymbol{x}+\xi)$.

It is possible to show that the generalized sketch objective function is a weighted sum of the usual sketch objective function at different frequencies by developping (4.10) a little bit further. This will be useful to characterize the impact of truncating the Fourier series at the end of this chapter.

Proposition 2. ***The objective function of a generalized kernel.*** *Using the generalized*

sketching operator $\overline{\mathcal{A}}$ with a signature f that has Fourier coefficients F_k

$$\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P} = \left[\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\, f(\boldsymbol{\omega}_j^T\boldsymbol{x}+\xi_j)\right]_{j=1}^m = \left[\mathbb{E}_{x\sim\mathcal{P}}\sum_k F_k\left(e^{\mathrm{i}k\frac{2\pi}{T}(\boldsymbol{\omega}_j^T\boldsymbol{x}+\xi_j)}\right)\right]_{j=1}^m \tag{4.12}$$

we have

$$\|\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P} - \overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{Q}\|_2^2 \simeq \sum_k |F_k|^2 \|\mathcal{A}_{k\frac{2\pi}{T}\Omega}\mathcal{P} - \mathcal{A}_{k\frac{2\pi}{T}\Omega}\mathcal{Q}\|_2^2 \tag{4.13}$$

with the original sketch with frequencies multiplied by α

$$\mathcal{A}_{\alpha\Omega}\mathcal{P} = \left[\mathbb{E}_{x\sim\mathcal{P}}\left(e^{-\mathrm{i}\alpha\boldsymbol{\omega}_j^T\boldsymbol{x}}\right)\right]_{j=1}^m. \tag{4.14}$$

More precisely, both expressions in (4.13) approximate the same expression

$$m\sum_k |F_k|^2 \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim\Lambda}\left|\psi_{\mathcal{P}}\left(k\frac{2\pi}{T}\boldsymbol{\omega}\right) - \psi_{\mathcal{Q}}\left(k\frac{2\pi}{T}\boldsymbol{\omega}\right)\right|^2 \tag{4.15}$$

with ψ_ν the characteristic function of the distribution ν.

Proof. Similarly as what is written just before, the empirical mean approximates the true mean:

$$\begin{aligned}\frac{1}{m}\|\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P} - \overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{Q}\|_2^2 &= \frac{1}{m}\sum_{j=1}^m\left|\sum_k F_k\left(\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{P}}\left(e^{\mathrm{i}k\frac{2\pi}{T}(\boldsymbol{\omega}_j^T\boldsymbol{x}+\xi_j)}\right) - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{Q}}\left(e^{\mathrm{i}k\frac{2\pi}{T}(\boldsymbol{\omega}_j^T\boldsymbol{x}+\xi_j)}\right)\right)\right|^2 \\ &\simeq \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim\Lambda}\mathop{\mathbb{E}}_{\xi\sim\mathcal{U}(0,T)}\left|\sum_k \underbrace{F_k\left(\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{P}}\left(e^{\mathrm{i}k\frac{2\pi}{T}(\boldsymbol{\omega}^T\boldsymbol{x}+\xi)}\right) - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{Q}}\left(e^{\mathrm{i}k\frac{2\pi}{T}(\boldsymbol{\omega}^T\boldsymbol{x}+\xi)}\right)\right)}_{a_k e^{\mathrm{i}k\frac{2\pi}{T}\xi}}\right|^2.\end{aligned}$$

Now, we have the squared absolute value of the sum of complex numbers $a_k e^{\mathrm{i}k\frac{2\pi}{T}\xi}$ with a_k independent on ξ. By developing the sum we have

$$\left|\sum_k a_k e^{\mathrm{i}k\frac{2\pi}{T}\xi}\right|^2 = \left(\sum_k a_k e^{\mathrm{i}k\frac{2\pi}{T}\xi}\right)\left(\sum_{k'} a_{k'} e^{\mathrm{i}k'\frac{2\pi}{T}\xi}\right)^* = \sum_k\sum_{k'} a_k a_{k'}^* e^{\mathrm{i}(k-k')\frac{2\pi}{T}\xi},$$

and going back to the original expression:

$$\begin{aligned}\frac{1}{m}\|\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P} - \overline{\mathcal{A}}\mathcal{Q}_{\Omega,\boldsymbol{\xi}}\|_2^2 &\simeq \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim\Lambda}\mathop{\mathbb{E}}_{\xi\sim\mathcal{U}(0,T)}\left|\sum_k a_k e^{\mathrm{i}k\frac{2\pi}{T}\xi}\right|^2 = \mathop{\mathbb{E}}_{\boldsymbol{\omega}}\mathop{\mathbb{E}}_{\xi}\sum_k\sum_{k'} a_k a_{k'}^* e^{\mathrm{i}(k-k')\frac{2\pi}{T}\xi} \\ &= \mathop{\mathbb{E}}_{\boldsymbol{\omega}}\sum_k\sum_{k'} a_k a_{k'}^* \mathop{\mathbb{E}}_{\xi} e^{\mathrm{i}(k-k')\frac{2\pi}{T}\xi} \\ &= \mathop{\mathbb{E}}_{\boldsymbol{\omega}}\sum_k\sum_{k'} a_k a_{k'}^* \frac{1}{T}\underbrace{\int_0^T e^{\mathrm{i}\frac{2\pi}{T}\xi(k-k')}\mathrm{d}\xi}_{T\delta[k-k']} \\ &= \mathop{\mathbb{E}}_{\boldsymbol{\omega}}\sum_k |a_k|^2 = \sum_k |F_k|^2\cdot\mathop{\mathbb{E}}_{\boldsymbol{\omega}}\left|\psi_{\mathcal{P}}\left(k\frac{2\pi}{T}\boldsymbol{\omega}\right) - \psi_{\mathcal{Q}}\left(k\frac{2\pi}{T}\boldsymbol{\omega}\right)\right|^2 \\ &\simeq \frac{1}{m}\sum_k |F_k|^2\|\mathcal{A}_{k\frac{2\pi}{T}\Omega}\mathcal{P} - \mathcal{A}_{k\frac{2\pi}{T}\Omega}\mathcal{Q}\|_2^2\end{aligned}$$

where the last step comes from the fact that the initial sketch approximates the mean difference between the characteristic functions of the probability distributions $\mathcal{P}$ and $\mathcal{Q}$, i.e. $\frac{1}{m}\|\mathcal{A}_{\alpha\Omega}\mathcal{P} - \mathcal{A}_{\alpha\Omega}\mathcal{Q}\|_2^2 = \frac{1}{m}\sum_{j=1}^m |\psi_{\mathcal{P}}(\alpha\boldsymbol{\omega}_j) - \psi_{\mathcal{Q}}(\alpha\boldsymbol{\omega}_j)|^2 \simeq \mathbb{E}_{\boldsymbol{\omega}}|\psi_{\mathcal{P}}(\alpha\boldsymbol{\omega}) - \psi_{\mathcal{Q}}(\alpha\boldsymbol{\omega})|^2$. □

Proposition 2 interprets the general sketch matching objective function as a sum -weighted by the square amplitude of the Fourier coefficients- of the original sketch matching objective function at different frequencies. Once again, it appears that in this generalization, we are "mixing up" information at higher frequencies to the original sketch. Note that the dithering allows to get rid of the "crossed products" at different frequencies in the objective function.

The generalized embedding of probability distributions

We now define rigorously the distance metric between probability distributions achieved by a general sketching operator. In chapter 2, we saw that (yet another) interpretation of the objective function of general probability distribution matching approximates through an empirical mean the Maximum Mean Discrepancy (MMD) metric γ_Λ on the set of probability distributions:

$$\frac{1}{m}\|\mathcal{A}_\Omega\mathcal{P} - \mathcal{A}_\Omega\mathcal{Q}\|^2 = \frac{1}{m}\sum_{j=1}^{m}|\psi_\mathcal{P}(\boldsymbol{\omega}_j) - \psi_\mathcal{Q}(\boldsymbol{\omega}_j)|^2 \simeq \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim\Lambda}|\psi_\mathcal{P}(\boldsymbol{\omega}) - \psi_\mathcal{Q}(\boldsymbol{\omega})|^2 = \gamma_\Lambda^2(\mathcal{P},\mathcal{Q}) \quad (4.16)$$

meaning that minimizing the sketch distance between $\mathcal{P}$ and $\mathcal{Q}$ is equivalent to approximatively minimizing the γ_Λ distance between $\mathcal{P}$ and $\mathcal{Q}$, noted $\gamma_\Lambda(\mathcal{P},\mathcal{Q})$. From a CS point of view, $\mathcal{A}_\Omega\mathcal{P}$ is an embedding of the probability distributions $\mathcal{P}$ that preserves the γ_Λ-distances. The generalized sketch $\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P}$ is in turn an embedding of the probability distributions $\mathcal{P}$ that preserves the $\overline{\gamma_\Lambda}$-distances defined in the following proposition.

Proposition 3. ***The metric on probability distributions when sketching with a general signature function.***
The generalized sketch distance approximates a new MMD metric $\overline{\gamma_\Lambda}(\mathcal{P},\mathcal{Q})$*:*

$$\frac{1}{m}\|\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P} - \overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{Q}\|^2 \simeq \overline{\gamma_\Lambda}^2(\mathcal{P},\mathcal{Q}) \quad (4.17)$$

with the new metric $\overline{\gamma_\Lambda}(\mathcal{P},\mathcal{Q})$ *that can be written, when squared, as (equivalence 1)*

$$\overline{\gamma_\Lambda}^2(\mathcal{P},\mathcal{Q}) = \sum_{k=-\infty}^{+\infty}|F_k|^2\gamma_\Lambda^2(D_{2\pi k/T}\mathcal{P}, D_{2\pi k/T}\mathcal{Q}) \quad (4.18)$$

where we define the n-dimensional dilation operator $D_\alpha g(\cdot) := \frac{1}{|\alpha|^n}g(\frac{\cdot}{\alpha})$*. It can also be written as (equivalence 2)*

$$\overline{\gamma_\Lambda}^2(\mathcal{P},\mathcal{Q}) = \gamma_{\Lambda_f}^2(\mathcal{P},\mathcal{Q}) \quad (4.19)$$

with $\Lambda_f = \sum_k |F_k|^2 D_{2\pi k/T}\Lambda$.

Proof. See Appendix C.1. □

Proposition 3 formalizes our previous discussions: the new metric $\overline{\gamma_\Lambda}(\mathcal{P},\mathcal{Q})$ used to measure the difference between the probability density functions can be seen either as a weighted sum of the usual γ-distances (defined by the same frequency distribution Λ) between contracted versions of the probability density functions, or equivalently as using a new frequency distribution Λ_f to define γ, that captures higher frequencies. It is important to stress the fact that here, Λ is still the actual frequency distribution used to draw the frequencies $\boldsymbol{\omega}_j \overset{\text{iid}}{\sim} \Lambda$, and Λ_f is the *equivalent frequency distribution* that determines which frequency content of the pdfs is compared (Figure 4.2 can help visualize this fact, even though it illustrates primarily the equivalences for the dual notion of Λ, that is, the kernel).

To conclude this section about the optimization problem solved with the general sketch, we note that to solve this problem in practice with L-BFGS, we need the gradient of this objective function, the details are given Appendix B.2.

4.1.3 Kernel approximated by the general sketch

The developpements of proposition 3 that led us to two interpretations for $\overline{\gamma_\Lambda}(\mathcal{P}, \mathcal{Q})$ (measuring *distance between probability distributions*), can in fact be transposed (since the kernel and the frequency distribution function are related) to obtain two interpretations for $\overline{\kappa}_\Lambda(\boldsymbol{x}_1, \boldsymbol{x}_2)$ the kernel of the generalized signature function (measuring *similarity between points*).

As we said in Chapter 2, it is agrued in [2] that the design of the frequency sampling pattern Λ is associated with an implicit kernel design. Indeed, the sketch of the learning examples are Random Fourier Features whose scalar product approximates a translation-invariant kernel [33]. This kernel κ_Λ is defined as

$$\kappa_\Lambda(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim\Lambda}\left[e^{-\mathrm{i}\boldsymbol{\omega}^T\boldsymbol{x}_1}\left(e^{-\mathrm{i}\boldsymbol{\omega}^T\boldsymbol{x}_2}\right)^*\right] = \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim\Lambda}\left[e^{-\mathrm{i}\boldsymbol{\omega}^T(\boldsymbol{x}_1-\boldsymbol{x}_2)}\right] =: K_\Lambda(\boldsymbol{x}_1 - \boldsymbol{x}_2) \tag{4.20}$$

where we can write the kernel as $K_\Lambda(\boldsymbol{x}_1 - \boldsymbol{x}_2)$ since it is a translation-invariant kernel (i.e. it only depends on the difference between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and not on their actual values). As a reminder, for the usual sketch, the translation-invariant kernel $\kappa_\Lambda(\boldsymbol{x}_1, \boldsymbol{x}_2) = K_\Lambda(\boldsymbol{x}_1 - \boldsymbol{x}_2)$ is related to the frequency sampling distribution Λ through a Fourier transform (see Theorem 3 in chapter 2, Bochner's theorem):

$$K_\Lambda(\boldsymbol{u}) = \int_{\mathbb{R}^n} e^{-\mathrm{i}\boldsymbol{\omega}^T\boldsymbol{u}}\Lambda(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega}, \tag{4.21}$$

a visual representation of this relationship is given in Figure 4.1.

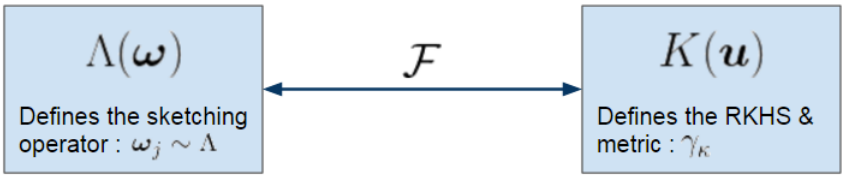


Figure 4.1: The relationship between the frequency sampling distribution Λ and the translation-invariant kernel $K(\boldsymbol{u})$ (related to the objective function, that is, to the metric γ_κ) in the case of usual sketching with complex exponentials: they are related by a Fourier transform $\mathcal{F}$ (Theorem 3).

Let us generalize this result for the kernel to a complex nonlinear periodic signature function $f(t)$ with period T with Fourier series coefficients F_k as in (4.3) instead of $e^{-\mathrm{i}t}$, and the addition of a dithering ξ. The kernel $\overline{\kappa}_\Lambda$ of this generalized case is

$$\overline{\kappa}_\Lambda(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mathop{\mathbb{E}}_{\boldsymbol{\omega},\xi}\left[f(\boldsymbol{\omega}^T\boldsymbol{x}_1 + \xi)f^*(\boldsymbol{\omega}^T\boldsymbol{x}_2 + \xi)\right] \tag{4.22}$$

where $\boldsymbol{\omega} \sim \Lambda(\boldsymbol{\omega})$ (we suppose the frequency sampling pattern is left unchanged) and $\xi \sim \mathcal{U}([0, T])$. We make the claim that this kernel can be written in two ways, that each have an interesting interpretation.

Proposition 4. ***The kernel when sketching with a general signature function.*** *The kernel as defined in (4.22) associated with a zero-mean function f that has F_k as Fourier series coefficents is translation-invariant,*

$$\overline{\kappa}_\Lambda(\boldsymbol{x}_1, \boldsymbol{x}_2) = \overline{K}_\Lambda(\boldsymbol{x}_1 - \boldsymbol{x}_2) \tag{4.23}$$

where $\overline{K}_\Lambda$ can moreover be written as (equivalence 1)

$$\overline{K}_\Lambda(\boldsymbol{x}_1 - \boldsymbol{x}_2) = \sum_{k=-\infty}^{\infty} |F_k|^2 K_\Lambda\left(-k\frac{2\pi}{T}(\boldsymbol{x}_1 - \boldsymbol{x}_2)\right) \tag{4.24}$$

or can also equivalently be written as (equivalence 2)

$$\overline{K}_\Lambda(\boldsymbol{x}_1 - \boldsymbol{x}_2) = K_{\Lambda_f}(\boldsymbol{x}_1 - \boldsymbol{x}_2) \quad \textit{with} \quad \Lambda_f(\boldsymbol{\omega}) = \sum_{k=-\infty}^{\infty} |F_k|^2 D_{2\pi k/T}\Lambda(\omega) \tag{4.25}$$

where we introduce the n-dimensional dilation operator $D_\alpha g(\cdot) := \frac{1}{|\alpha|^n} g(\frac{\cdot}{\alpha})$ that dilates a function g by a factor α while preserving its L^1-norm in $\mathbb{R}^n$.

Proof. See Appendix C.2 □

We can see Propostion 4 as the generalization of the relationship between the frequency sampling distribution Λ and the kernel $\overline{\kappa}$ depending on Λ. Previously, this relationship was directly Bochner's theorem, that is, the one is the Fourier transform of the other, as represented Figure 4.1. The main observation about propostion 4 is that the impact on the frequency distribution-kernel relationship of changing the signature function can be seen in two very different ways (illustrated Figure 4.2):

- **Interpretation 1**: the new kernel is a superposition (sum) of multiple copies of κ_Λ the original kernel defined by Λ, shrinked and weighted by the squared amplitude of the Fourier coefficients of f. This is represented by the top branch in Figure 4.2.
- **Interpretation 2**: the new kernel is a kernel that is defined through Bochner's theorem by a new probability density function Λ_f. Λ_f is itself a superposition of multiple copies of the original density function Λ, shrinked and weighted by the squared amplitude of the Fourier coefficients of f.

About the importance of dithering

The presence of a dither $\boldsymbol{\xi}$ is crucial in order to obtain Proposition 4. Supposing we don't use a dither vector, we will obtain, with very similar steps to the proof Appendix C.2, a kernel given by

$$\overline{\overline{\kappa}}_\Lambda(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_k \sum_l F_k F_l^* K_\Lambda\left(-\frac{2\pi}{T}(k\boldsymbol{x}_1 - l\boldsymbol{x}_2)\right). \tag{4.26}$$

The problem here is that this kernel without dithering is in general **not** translation invariant which is a desirable property for a kernel in clustering tasks (for example, taking the term of the sum coming from ($k = 0$, $l = 1$) are not translation invariant since they depend on $\boldsymbol{x}_2$ alone). In fact, it follows from this result very clearly that the only periodic functions allowing to obtain a translation invariant kernel without dithering are the ones with only one nonzero Fourier coefficient (the complex exponential from the Compressive K-means!), or two complex conjugate coeffients (a sinusoidal signature function).

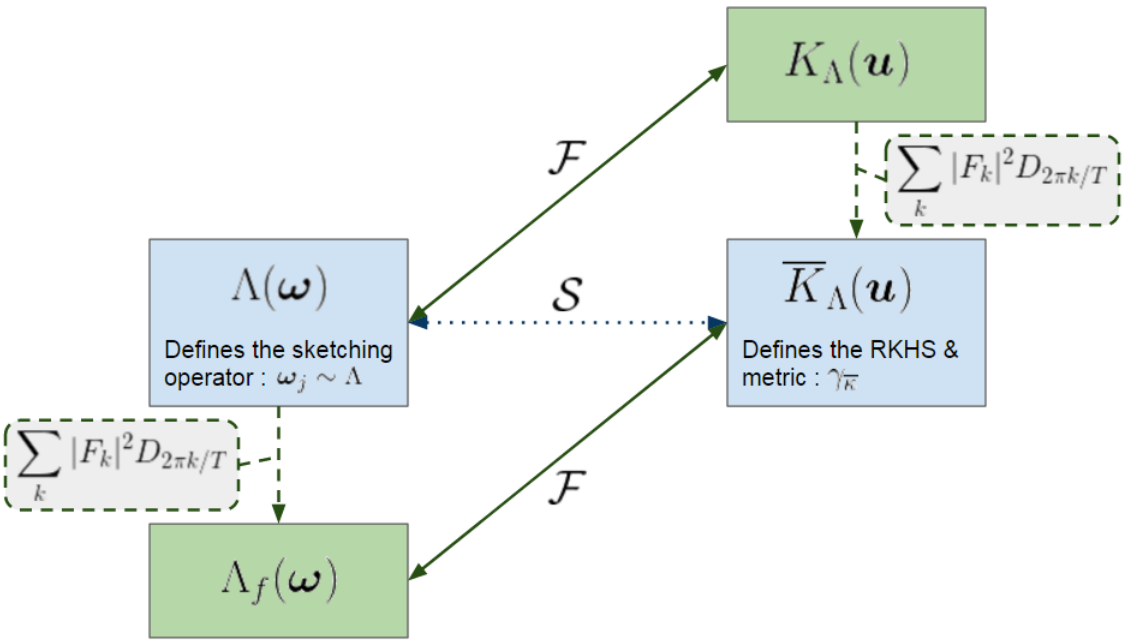


Figure 4.2: The relationship between the frequency sampling distribution Λ and the translation-invariant kernel $\overline{K}_\Lambda(\boldsymbol{u})$ in generalized sketching with a periodic function $f(t)$: they are not directly related by a Fourier transform but by a "square wave transform" $\mathcal{S}$, that is, a projection on square wave basis functions. By writing the signature function $f(t)$ as a Fourier series, two equivalences can however be made (green branches). The top branch is equivalence 1 : the kernel $\overline{K}_\Lambda$ is a weighted sum of shrinked versions of the kernel K_Λ that is the Fourier transform of Λ (through Bochner's theorem). The bottom branch is equivalence 2 : the kernel $\overline{K}_\Lambda$ is the Fourier transform of an equivalent frequency distribution Λ_f (through Bochner's theorem), a weighted sum of shrinked versions of Λ.

A new look on the objective function versus the SSE, from the perspective of kernels

Up to now, the role played by the kernel in the compressive clustering algorithms was never explicit. We will here re-develop the objective function that defines the centroids in the light of the kernel to make it appear explicitly in the clustering problem.

First, we do some priliminary observations. Remember that in general, a (translation-invariant) kernel $\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is a similarity measure between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$; κ is maximal when $\boldsymbol{x}_1 = \boldsymbol{x}_2$ and decreases when they become more distant. Also, it is worth noting that if $\boldsymbol{y}_1$ (resp. $\boldsymbol{y}_2$) is the sketch associated with frequencies $\Omega = [\boldsymbol{\omega}_j \overset{\text{iid}}{\sim} \Lambda]_{j=1}^m$ a single data sample $\boldsymbol{x}_1$ (resp. $\boldsymbol{x}_2$), we have for large m, since the empirical mean approaches the true mean,[1]

$$\begin{aligned}
\frac{1}{m} < \boldsymbol{y}_1, \boldsymbol{y}_2 > &= \frac{1}{m} f(\Omega^T \boldsymbol{x}_1 + \boldsymbol{\xi}) f^*(\Omega^T \boldsymbol{x}_2 + \boldsymbol{\xi}) \\
&= \frac{1}{m} \sum_{j=1}^m f(\boldsymbol{\omega}_j^T \boldsymbol{x}_1 + \xi_j) f^*(\boldsymbol{\omega}_j^T \boldsymbol{x}_2 + \xi_j) \\
&\simeq \mathop{\mathbb{E}}_{\boldsymbol{\omega} \sim \Lambda,\, \xi} \Big[f(\boldsymbol{\omega}^T \boldsymbol{x}_1 + \xi) f^*(\boldsymbol{\omega}^T \boldsymbol{x}_2 + \xi) \Big] = \overline{\kappa}_\Lambda(\boldsymbol{x}_1, \boldsymbol{x}_2).
\end{aligned} \tag{4.27}$$

We are now ready to develop $J(C, \boldsymbol{\alpha})$, the cost function of QCKM, on a dataset X where there are K clusters $C = [\boldsymbol{c}_k]_{k=1}^K$ with weights α_k to fit (assuming that the kernel is a real function):

[1] $\boldsymbol{a}^*$ is conjugate *transpose* of $\boldsymbol{a}$.

$$
\begin{aligned}
J(C,\boldsymbol{\alpha}) &= \frac{1}{m}\|\overline{\mathrm{Sk}}_{\Omega,\boldsymbol{\xi}}\left(X,\mathbf{1}_N/N\right) - \overline{\mathrm{Sk}}_{\Omega,\boldsymbol{\xi}}\left(C,\boldsymbol{\alpha}\right)\|_2^2 \\
&= \frac{1}{m}\|\overline{\mathrm{Sk}}_{\Omega,\boldsymbol{\xi}}\left(X,\mathbf{1}_N/N\right)\|_2^2 + \frac{1}{m}\|\overline{\mathrm{Sk}}_{\Omega,\boldsymbol{\xi}}\left(C,\boldsymbol{\alpha}\right)\|_2^2 - \frac{2}{m}\Re\left(\overline{\mathrm{Sk}}_{\Omega,\boldsymbol{\xi}}\left(X,\mathbf{1}_N/N\right)\overline{\mathrm{Sk}}_{\Omega,\boldsymbol{\xi}}\left(C,\boldsymbol{\alpha}\right)^*\right) \\
&= \frac{1}{m}\frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N} f(\Omega^T\boldsymbol{x}_i+\boldsymbol{\xi})f^*(\Omega^T\boldsymbol{x}_{i'}+\boldsymbol{\xi}) + \frac{1}{m}\sum_{k=1}^{K}\sum_{k'=1}^{K}\alpha_k\alpha_{k'}f(\Omega^T\boldsymbol{c}_k+\boldsymbol{\xi})f^*(\Omega^T\boldsymbol{c}_{k'}+\boldsymbol{\xi}) \\
&\qquad - \frac{2}{m}\Re\left(\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\alpha_k f(\Omega^T\boldsymbol{x}_i+\boldsymbol{\xi})f^*(\Omega^T\boldsymbol{c}_k+\boldsymbol{\xi})\right) \\
&\simeq \underbrace{\frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\overline{\kappa}_\Lambda(\boldsymbol{x}_i,\boldsymbol{x}_{i'})}_{\phi_X} + \underbrace{\sum_{k=1}^{K}\sum_{k'=1}^{K}\alpha_k\alpha_{k'}\,\overline{\kappa}_\Lambda(\boldsymbol{c}_k,\boldsymbol{c}_{k'})}_{\phi_C} - 2\underbrace{\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\alpha_k\overline{\kappa}_\Lambda(\boldsymbol{x}_i,\boldsymbol{c}_k)}_{\phi_{XC}}.
\end{aligned}
\tag{4.28}
$$

We have thus shown that the *approximated* objective function of QCKM (this is also transposable to CKM) is made up of three contributions $J(C,\boldsymbol{\alpha}) = \phi_X + \phi_C - 2\phi_{XC}$, that we can actually interpret as follows:

- a **cost** ϕ_X, corresponding to the average similarity between all the pairs of learning examples $\boldsymbol{x}_i$. Actually, this term depends solely on the dataset and not on the centroid positions or weights, so it is a constant for the optimization problem at hand and is irrelevant.
- a **cost** ϕ_C, corresponding to a *weighted* average of the similarity between all pairs of centroids. This term will promote centroids that are well separated. For centroids that are important (high weight α_j), this penalty will increase, so important centroids do repell other centroids stronger. Or equivalently, this term promotes smaller weights on the centroids if they are close to other centroids.
- a **gain** ϕ_{XC}, corresponding to the average similarity between the data samples and the centroids. This is the most interesting term since it corresponds to the "clustering" part of the objective function[2]. This term promotes centroids that lie close to the data, but instead of using the squared distance towards the centroids as objective function (as in "traditional clustering"), we use the kernel to evaluate the similarity between the examples and the centroids. Note also that in this term it becomes clear that if a centroid is close to a lot of training examples, it should receive a higher weight. Or equivalently, if a centroid has a small weight, it will be less attracted to the learning examples.

Given this interpretation, the objective function of QCKM makes a lot of sense intuitively, and the role of the kernel appears explicitly. It is important to note that therefore, sketch-based compressive clustering algorithms (thus in particular, `CKM` and `QCKM`) do not aim at solving the k-means clustering problem (2.2) (although [1] presents `CKM` as a heuristic to solve (2.2)), but in fact solve another clustering problem based on the cost $J(C,\boldsymbol{\alpha})$. Recall the general definition of a clustering problem from Section 2.1.2: grouping data into clusters such that points within clusters are "similar" and points in different clusters are "dissimilar": here similarity is measured through a kernel. The differences between $J(C,\boldsymbol{\alpha})$ the (Q)CKM objective function and the SSE the objective function of K-means clustering also appear more clearly.

- First, the SSE compares the examples and the centroids through the Euclidean distance, whereas J compares them through a similarity defined by a generic kernel.

[2]Thus, it is a good thing that it is weighted two times more than the other terms.

- Secondly, training examples only contribute to the SSE in pair with their closest centroid, while in J, the similarity of a training example with all the centroids does contribute (however, if the kernel is decaying fast enough with the distance, only the closest centroids will really contribute to the objective).
- Third difference, the SSE does not take into account the relative distance between the centroids. In J, one important (half as important as the "clustering" term) term is ϕ_C, the spacing of the centroids.
- Fourth difference (but this one was already clear), J has some additional variables compared to K-means clustering, namely the weights (relative importance) of the different centroids. The objective J will promote well-spaced important centroids. Those additional values are very helpful to detect "lost" centroids that are close to no cluster at all (a situation that can happens in practice from time to time, when the algorithm is stuck in a local optimum).

4.1.4 New distribution of the frequencies

Previously, the frequencies $\boldsymbol{\omega}$ were drawn iid from a distribution Λ, where Λ was choosen in order to sample the characteristic function ψ of the data "well", i.e. choosing Λ such that $|\psi(\boldsymbol{\omega})|$ takes large values. Obviously, for our generalized case, this distribution Λ has to be changed to account for the fact that we are now evaluating the distance differently.

In (4.10) we saw that we are not evaluating the difference between the probability distributions through the characteristic funcion $\psi_{\mathcal{P}}(\boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\, e^{-i\boldsymbol{\omega}^T\boldsymbol{x}}$ anymore but through a new function

$$\overline{\psi}_{\mathcal{P}}(\boldsymbol{\omega};\xi) = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{P}} f(\boldsymbol{\omega}^T\boldsymbol{x}+\xi) = \sum_k F_k e^{\mathrm{i}k\frac{2\pi}{T}\xi}\psi_{\mathcal{P}}\left(-k\frac{2\pi}{T}\boldsymbol{\omega}\right). \tag{4.29}$$

with ξ uniformly distributed in $[0,T]$ with T is the period of f, and we now suppose without loss of generality that $T = 2\pi$.

Gaussian with adapted heuristic distribution (G-h). First, as is done in [2], we suppose we have a single known Gaussian, i.e. $\mathcal{P}$ is $\mathcal{N}(\boldsymbol{\mu},\Sigma)$ with known parameters $\boldsymbol{\mu}$ and Σ[3]. If we want to proceed as before and choose the distribution Λ to sample $|\overline{\psi}_{\mathcal{P}}|$ essentially where it takes large values, we could go for the intuitive choice $\Lambda \propto |\overline{\psi}_{\mathcal{P}}|$. Although this approach has a good intuitive motivation, problems could arise because this function $|\overline{\psi}_{\mathcal{P}}|$ is unbounded when f is the universal quantization function:

$$\begin{aligned}
|\overline{\psi}_{\mathcal{P}}(\boldsymbol{\omega};\xi)| &= \left|\sum_{k=-\infty}^{+\infty} F_k\psi_{\mathcal{P}}(-k\boldsymbol{\omega})e^{\mathrm{i}k\xi}\right| = \left|\sum_{k=-\infty}^{+\infty} F_k e^{\mathrm{i}k\boldsymbol{\omega}^T\boldsymbol{\mu}}e^{-\frac{k^2}{2}\boldsymbol{\omega}^T\Sigma\boldsymbol{\omega}}e^{\mathrm{i}k\xi}\right| \\
&\leq \sum_{k=-\infty}^{+\infty} |F_k|\cdot 1\cdot|e^{-\frac{k^2}{2}\boldsymbol{\omega}^T\Sigma\boldsymbol{\omega}}|\cdot 1 \\
&= \sum_{\substack{k=-\infty\\ k\text{ odd}}}^{+\infty} \frac{2}{\pi|k|}e^{-\frac{k^2}{2}\boldsymbol{\omega}^T\Sigma\boldsymbol{\omega}} = \sum_{\substack{k=-\infty\\ k\text{ odd}}}^{-1} \frac{2}{\pi(-k)}e^{-\frac{k^2}{2}\boldsymbol{\omega}^T\Sigma\boldsymbol{\omega}} + \sum_{\substack{k=+1\\ k\text{ odd}}}^{+\infty} \frac{2}{\pi k}e^{-\frac{k^2}{2}\boldsymbol{\omega}^T\Sigma\boldsymbol{\omega}} \\
&= 2\sum_{\substack{k=1\\ k\text{ odd}}}^{+\infty} \frac{2}{\pi k}e^{-\frac{k^2}{2}\boldsymbol{\omega}^T\Sigma\boldsymbol{\omega}} = \frac{4}{\pi}\sum_{\substack{k=1\\ k\text{ odd}}}^{+\infty} \frac{1}{k}e^{-\frac{k^2}{2}\boldsymbol{\omega}^T\Sigma\boldsymbol{\omega}}
\end{aligned} \tag{4.30}$$

[3]Thus, we can write $\psi_{\mathcal{P}}(\boldsymbol{\omega}) = e^{-i\boldsymbol{\omega}^T\boldsymbol{\mu}}e^{-\frac{1}{2}\boldsymbol{\omega}^T\Sigma\boldsymbol{\omega}}$.

thus in particular $|\overline{\psi}_{\mathcal{P}}(\mathbf{0};\xi)| \leq \frac{4}{\pi}\sum_{k=1,k \text{ odd}}^{+\infty}\frac{1}{k}$ which diverges: sampling the frequencies as $\Lambda \propto |\overline{\psi}_{\mathcal{P}}|$ will potentially oversample the zero frequencies, exactly the point where the characteristic function does not contain any information since it is equal to 1 for all probability distributions. Furthermore, even without this problem at the origin, it is in practice hard to sample $\boldsymbol{\omega} \sim \Lambda \propto |\overline{\psi}_{\mathcal{P}}|$ because $|\overline{\psi}_{\mathcal{P}}|$ is not easily converted into a probability distribution (to normalize it we have to compute its integral over the whole domain $\mathbb{R}^m$ which is not trivial).

To circumvent those problem, we propose to select $\Lambda \propto \mathbb{E}_{\xi}|\overline{\psi}_{\mathcal{P}}(\boldsymbol{\omega};\xi)|^2$. Intuitively it makes sense: 1) we don't want our frequency distribution to depend on a particular realisation of ξ (that will be different for each frequency we will have to draw, so we will have one different frequency distribution for each frequency we have to draw) but rather on the average behavior of the function with ξ (and in this case, we have only one frequency distribution for all the frequencies), and 2) if $|\overline{\psi}_{\mathcal{P}}(\boldsymbol{\omega};\xi)|$ is significant, $|\overline{\psi}_{\mathcal{P}}(\boldsymbol{\omega};\xi)|^2$ significant too[4]. In this case we have

$$\begin{aligned}\mathop{\mathbb{E}}_{\xi}|\overline{\psi}_{\mathcal{P}}(\boldsymbol{\omega};\xi)|^2 &= \sum_k |F_k|^2 \left|\psi_{\mathcal{P}}(-k\boldsymbol{\omega})\right|^2 \\ &= \sum_k |F_k|^2 \left|e^{\mathrm{i}k\boldsymbol{\omega}^T\boldsymbol{\mu}}e^{-\frac{k^2}{2}\boldsymbol{\omega}^T\Sigma\boldsymbol{\omega}}\right|^2 \\ &= \sum_k |F_k|^2 e^{-\frac{2k^2}{2}\boldsymbol{\omega}^T\Sigma\boldsymbol{\omega}} = \sum_k |F_k|^2 e^{-\frac{1}{2}\boldsymbol{\omega}^T\left(2\Sigma/k^2\right)\boldsymbol{\omega}}\end{aligned} \tag{4.31}$$

so it turns out that the intuitive choice is here to take $\Lambda \propto \sum_k |F_k|^2 \mathcal{N}(0, k^2\Sigma^{-1}/2)$. To do this in practice, one idea is to select for each frequency that we have to draw a Fourier coefficient number r with probability $\mathbb{P}[r=k] = p_k = |F_k|^2/\sum_l |F_l|^2$ and then to draw $\boldsymbol{\omega} \sim \mathcal{N}(0, r^2\Sigma^{-1}/2)$.

Since this is the equivalent to the Gaussian frequency sampling (G) in [2], we will call this sampling of frequencies the Gaussian with adapted heuristic (G-h). Sadly, as was the case with (G), the (G-h) frequency sampling suffers from the curse of dimensionality and produces too many frequencies concentrated on a sphere away from the origin, where the information content of the sketch is small and the sketch takes small values.

Gaussian radius with adapted heuristic distribution (Gr-h). Similarly to the Gaussian radius (Gr) heuristic form [2], we therefore propose to select the frequencies as $\boldsymbol{\omega} = R\Sigma^{-1/2}\boldsymbol{\varphi}$ with R a (random) radius whose distribution is to be determined later, and $\boldsymbol{\varphi} \sim \mathcal{U}(S_{n-1})$ a uniform direction vector of unit norm. In this case, the function we want to sample expresses as

$$\mathop{\mathbb{E}}_{\xi}|\overline{\psi}_{\mathcal{P}}(\boldsymbol{\omega};\xi)|^2 = \sum_k |F_k|^2 e^{-\frac{1}{2}\boldsymbol{\omega}^T\left(2\Sigma/k^2\right)\boldsymbol{\omega}} = \sum_k |F_k|^2 e^{-\frac{1}{2}\frac{R^2}{k^2/2}} = \overline{\psi}_{\mathcal{P}}(R) \tag{4.32}$$

which is a function of R only. The intuitive choice is here to pick $R \sim p_R(R)$ with

$$p_R(R) \propto \overline{\psi}_{\mathcal{P}}(R) = \sum_k |F_k|^2 \mathcal{N}(0, k^2/2) \tag{4.33}$$

and we obtain thus a one-dimensional Gaussian distribution that no longer suffers from the curse of dimensionality. We denote this frequency sampling heuristic (Gr-h).

[4] Actually, empirical simulations have shown that taking the square root afterwards, i.e. choosing $\Lambda \propto \sqrt{\mathbb{E}_{\xi}|\overline{\psi}_{\mathcal{P}}(\boldsymbol{\omega};\xi)|^2}$ has no noticeable effect on the clustering performances.

Adapted radius with adapted heuristic distribution (Ar-h). Finally, similarly to the Adapted radius (Ar) density presented in [2] that samples the frequencies where they the best at differentiating different data probability density functions, we propose its equivalent in the generalized sketch case, that we will call (Ar-h). We propose here to sample the frequencies where the average (with respect to ξ) norm of the gradient with respect to the parameters of the Gaussian distribution is high. We have, given that $\overline{\psi}_{\mathcal{N}(\mu,\sigma^2)}(R) = \sum_k F_k e^{\mathrm{i}k\xi} e^{\mathrm{i}kR\mu} e^{-\frac{k^2}{2}R^2\sigma^2}$ and building on developements from Chapter 2,

$$\begin{aligned}\mathop{\mathbb{E}}_{\xi}\left\|\boldsymbol{\nabla}_{(\mu,\sigma^2)}\overline{\psi}_{\mathcal{N}(\mu,\sigma^2)}(R)\right\|_2^2 &= \mathop{\mathbb{E}}_{\xi}\left|\nabla_{\mu}\overline{\psi}_{\mathcal{N}(\mu,\sigma^2)}(R)\right|^2 + \mathop{\mathbb{E}}_{\xi}\left|\nabla_{\sigma^2}\overline{\psi}_{\mathcal{N}(\mu,\sigma^2)}(R)\right|^2 \\ &= \sum_k |F_k|^2 \left(k^2R^2 + \frac{k^4R^4}{4}\right) e^{-k^2R^2\sigma^2}\end{aligned} \tag{4.34}$$

and, when evaluating this gradient at the true parameters ($\mu = \mu_0, \sigma^2 = \sigma_0^2 = 1$), we obtain finally the distribution for the radius R:

$$p_R(R) \propto \sum_k |F_k|^2 \left(k^2R^2 + \frac{k^4R^4}{4}\right) e^{-k^2R^2} \tag{4.35}$$

and the resulting frequency distribution is denoted as the (Ar-h) distribution.

4.2 Particularisation to the universal quantization signature function

The previous section focused on general periodic functions $f(t)$. In this section, we re-visit the results of this section but in the particular case where the signature function is the universal quantization function $Q_\Delta(t)$. Recall from proposition 1 that $f(t) = Q_\Delta(t)$ can be written with Fourier coefficients $F_k = \frac{2\mathrm{i}}{k\pi}$ for k odd (the coefficients are 0 whenever k is even). For this particular case, the gradients to solve the optimization problem can be more easily computed than in the general case (see Appendix B.3). Interestingly, for the universal quantization sketch, the objective function of clustering can be linked to the distance map that characterizes the embedding induced by 1-but universal quantization of random projections of vectors [23].

4.2.1 Kernel approximated by the quantized sketch

From Proposition 4 and given the Fourier coefficients F_k and $T = 2\pi$ for the universal quantization, we have that (equivalence 1)

$$\overline{K}_\Lambda(\boldsymbol{x}_1 - \boldsymbol{x}_2) = \sum_{k=-\infty}^{\infty} |F_k|^2 K_\Lambda\left(-k\frac{2\pi}{T}(\boldsymbol{x}_1 - \boldsymbol{x}_2)\right) = \frac{4}{\pi^2}\sum_{\substack{k=-\infty \\ k \text{ odd}}}^{\infty} \frac{1}{k^2} K_\Lambda\left(-k(\boldsymbol{x}_1 - \boldsymbol{x}_2)\right) \tag{4.36}$$

with Λ the frequency distribution of the frequencies. Let us first suppose that the frequency drawing pattern we use is the Gaussian heuristic from `CKM` (G) with $\Sigma = \overline{\sigma}^2 I$, where $\overline{\sigma}^2$ is the variance of the dataset estimated from a small fraction of X. Thus, we have $\Lambda(\boldsymbol{\omega}) = \Lambda^{\mathtt{G}}(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}; \mathbf{0}, \overline{\sigma}^{-2} I)$, and for the complex exponential sketch kernel, applying Bochner's theorem yields a Gaussian kernel

$$K_{\Lambda^{\mathtt{G}}}(\boldsymbol{x}_1 - \boldsymbol{x}_2) = e^{-\frac{1}{2}\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}{\overline{\sigma}^2}}. \tag{4.37}$$

Combining this kernel with Proposition 4 (equivalence 1), we can obtain the kernel approximated by the universal quantization sketch when the drawing of the frequencies is Gaussian:

$$\overline{K}_{\Lambda^{\mathtt{G}}}(\boldsymbol{x}_1 - \boldsymbol{x}_2) = \frac{4}{\pi^2} \sum_{\substack{k=-\infty \\ k \text{ odd}}}^{\infty} \frac{1}{k^2} e^{-\frac{k^2}{2} \frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}{\overline{\sigma}^2}}, \tag{4.38}$$

and this kernel is represented Figure 4.3.

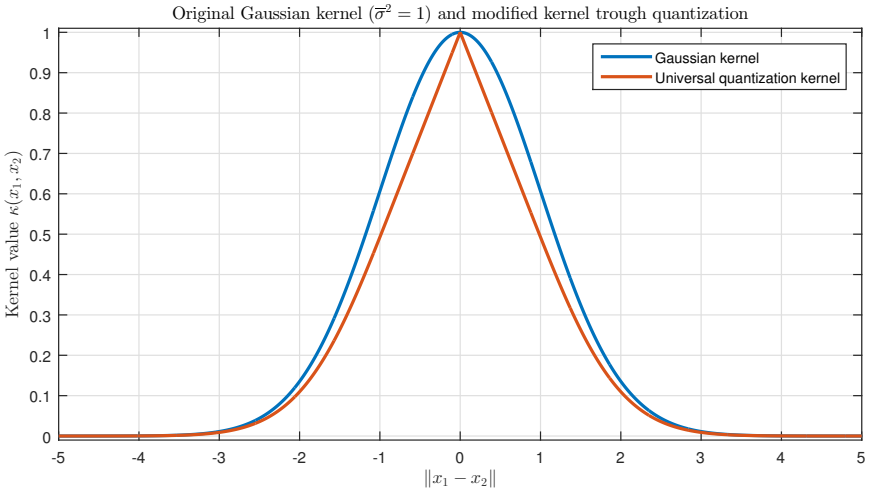


Figure 4.3: In blue, the original kernel approximated by the complex exponential sketch with normally distributed frequencies $K_{\Lambda^{\mathtt{G}}}(\boldsymbol{x}_1 - \boldsymbol{x}_2)$. In red, the sketch approximated by the universal quantization sketch with the same frequency distribution, $\overline{K}_{\Lambda^{\mathtt{G}}}(\boldsymbol{x}_1 - \boldsymbol{x}_2)$.

Interestingly, the kernel $\overline{K}_{\Lambda^{\mathtt{G}}}(\boldsymbol{x}_1 - \boldsymbol{x}_2)$ is related to $g(d)$, the distance map induced by the univeral quantization embedding $f(\boldsymbol{x}) = q_\pi(A\boldsymbol{x} + \boldsymbol{\xi})$ (the quantizer stepsize being equal to π) in Theorem 2, by

$$\overline{K}_{\Lambda^{\mathtt{G}}}(\boldsymbol{x}_1 - \boldsymbol{x}_2) = 1 - 2g(\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2) \quad \text{if} \quad \overline{\sigma} = \frac{1}{\sigma} \tag{4.39}$$

where we recall that σ^2 is the variance of the iid entries of A (the variance of the "frequencies") and $\overline{\sigma}^2$ is the (approximated) mean variance of the clusters (the variance of the "data", hence the *inverse* relationship with the variance of the frequencies). This means that in the quantized sketch, the kernel - a similarity measure - is proportional to minus the distance mapped by $g(d)$. In particular, going back to the objective function (4.28), we can re-write the clustering problem objective function to minimiez for the quantized sketch clustering problem when $\Lambda(\boldsymbol{\omega}) = \Lambda^{\mathtt{G}}(\boldsymbol{\omega})$:

$$J(C, \boldsymbol{\alpha}) \simeq -\sum_{k=1}^{K} \sum_{k'=1}^{K} \alpha_k \alpha_{k'} \, g(\|\boldsymbol{c}_k - \boldsymbol{c}_{k'}\|_2) + 2\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \alpha_k g(\|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2). \tag{4.40}$$

This allows us to see clearly the link between the sketch pooling universal quantization projections of the signals and previous work on universal quantized embeddings [23], [21]. Indeed, we can do (yet another) intepretation of the quantized clustering problem that is solved by `QCKM`: the centroids and weights are chosen in order to maximize the weigthed distances -passed through $g(d)$- between the centroids (first term) and at the same time to minimize the distances -passed through $g(d)$- between the learning examples and the centroids (second term). If we make the assumption that i) there is one centroid per cluster, ii) the size of the clusters are smaller than D_0 meaning that intra-cluster distances are mapped exactly by $g(d)$ and iii) the distances between the clusters are larger than D_0 meaning that inter-cluster distances are constant, then this

objective function becomes *almost* K-means's objective function, without squaring the distances and with weighted centroids:

$$J(C, \boldsymbol{\alpha}) \propto \sum_{i=1}^{N} \min_k \alpha_k \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2. \tag{4.41}$$

The connexions with [21] can be drawn even further by considering the general signature function $f(t)$ again, so we have a kernel

$$\overline{K}_{\Lambda^{\mathtt{G}}}(\boldsymbol{x}_1 - \boldsymbol{x}_2) = \sum_{k=-\infty}^{\infty} |F_k|^2 K_{\Lambda^{\mathtt{G}}}\left(-k\frac{2\pi}{T}(\boldsymbol{x}_1 - \boldsymbol{x}_2)\right) \tag{4.42}$$

for general Fourier coefficients F_k. Then, if we assume $T = 1$ (we do the same assumption as [21] in order to make the link more easily) the general kernel $\overline{K}_{\Lambda^{\mathtt{G}}}(\boldsymbol{x}_1 - \boldsymbol{x}_2)$ is related to the distance map $g(d)$ of the embedding defined by $\boldsymbol{y} = f(A\boldsymbol{x} + \boldsymbol{\xi})$ for A an m-by-n matrix with $A_{ij} \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$ and $\xi_j \overset{\text{iid}}{\sim} U([0, 1])$ by

$$\overline{K}_{\Lambda^{\mathtt{G}}}(\boldsymbol{x}_1 - \boldsymbol{x}_2) = \sum_k |F_k|^2 - \frac{1}{2} g(\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2) \quad \text{if} \quad \overline{\sigma} = \frac{1}{\sigma} \tag{4.43}$$

since the distance map $g(d)$ of $\boldsymbol{y} = f(A\boldsymbol{x} + \boldsymbol{\xi})$ is given by ([21], Section 4.2.1)

$$g(d) = 2\sum_k |F_k|^2 (1 - e^{-2(\pi\sigma d k)^2}). \tag{4.44}$$

In particular, the `CKM` algorithm (where the signature function $f(t) = e^{-\mathrm{i}t}$) with $\Lambda = \Lambda^{\mathtt{G}}$ uses implicitly a distance map that consists of a reversed Gaussian.

However, for the other frequency distribution heuristics, the kernel is not Gaussian anymore, and the distance map is different. In particular, when using the (G-h), (Gr-h) and (Ar-h) distributions, sampling heuristics that are *adapted* to the universal quantization sketch, since the frequency distribution determines the kernel through proposition 4, the kernel approximated by those new frequency sampling heuristics will also be changed.

4.2.2 Quantized frequency distribution

As we have sen with the equivalences, when we sample the quantized sketch at frequenices drawn from a frequency density $\Lambda(\boldsymbol{\omega})$, it is as we are sampling the original sketch with frequencies drawn from $\Lambda_f(\boldsymbol{\omega}) = \sum_{k=-\infty}^{\infty} |F_k|^2 D_{2\pi k/T} \Lambda(\omega)$. $\Lambda_f(\boldsymbol{\omega})$ can be interpreted as the equivalent frequency content that is used to compare the frequency distributions. In the particular case of a Gaussian sampling of the frequencies (G), we provide bounds for Λ_f to show how the frequency content is changed when the quantized sketch is used. We furthermore assume without loss of generality that $\Sigma = I$, thus $\Lambda(\boldsymbol{\omega}) = ce^{-\|\boldsymbol{\omega}\|^2/2}$ for a normalization constant c. Then given the particular Fourier series coefficients,

$$\Lambda_f(\boldsymbol{\omega}) \propto c \sum_{k\ odd} \frac{1}{|k|^2} \cdot \frac{1}{|k|^n} e^{-\|\boldsymbol{\omega}\|^2/2k^2} = 2c \sum_{\substack{k\ odd \\ k \geq 0}} \frac{1}{k^{n+2}} e^{-\|\boldsymbol{\omega}\|^2/2k^2}, \tag{4.45}$$

and since all terms in the sum are strictly positive we can bound the sum below by keeping only some terms corresponding to $k \geq \|\boldsymbol{\omega}\|$, where we assume $\|\boldsymbol{\omega}\| \gg 1$:

$$
\begin{aligned}
\Lambda_f(\boldsymbol{\omega}) &= c_1 \sum_{\substack{k \text{ odd} \\ k \geq 0}} \frac{1}{k^{n+2}} e^{-\|\boldsymbol{\omega}\|^2/2k^2} \\
&\geq c_1 \sum_{\substack{k \text{ odd} \\ k \geq \|\boldsymbol{\omega}\|}} \frac{1}{k^{n+2}} e^{-\|\boldsymbol{\omega}\|^2/2k^2} \\
&\geq c_1 \sum_{\substack{k \text{ odd} \\ k \geq \|\boldsymbol{\omega}\|}} \frac{1}{k^{n+2}} \underbrace{e^{-\|\boldsymbol{\omega}\|^2/2\|\boldsymbol{\omega}\|^2}}_{e^{-1/2}} \\
&= c_2 \sum_{\substack{k \text{ odd} \\ k \geq \|\boldsymbol{\omega}\|}} \frac{1}{k^{n+2}} \\
&\geq c_2 \frac{1}{\|\boldsymbol{\omega}\|^{n+2}}
\end{aligned}
\tag{4.46}
$$

where in the last step we only kept the smallest term in the sum. Thus, for large $\|\boldsymbol{\omega}\|$, the equivalent frequency distribution $\Lambda_f(\boldsymbol{\omega})$ is $\Omega\left(\|\boldsymbol{\omega}\|^{-n-2}\right)$. We stated previously that the high frequencies do not bring useful information to the sketch for clustering, and we have shown here that naturally, the equivalent frequency distribution with the quantized sketch is sampling high frequency with a probability density that goes as $\Omega\left(\|\boldsymbol{\omega}\|^{-n-2}\right)$. In short, the equivalent probability distribution oversamples high frequencies.

4.3 Characterization of the different error terms

To conclude the theoretic approach of this chapter, we summarize the different approximations made by `QCKM` (this discusion also applies to `CKM`), as represented Figure 4.4. The problem on top is the k-means clustering problem, that is, the initial problem we want to solve. From there, there are two interpretations of the approximation made by replacing this problem by the sketch matching problem.

- **Approximation (1)** is the *replacement* of the K-means clustering problem by another problem, that minimizes the $\gamma_{\overline{\kappa}}$-distance between two distributions $\mathcal{P}$ and $\mathcal{Q}$. The distribution $\mathcal{P}$ is the true distribution underlying the data, and the distribution $\mathcal{Q}$ is a mixture of deltas that we are matching with the data ($\mathcal{G}_K$ is the set of mixturs of K deltas). Characterizing this error term is not trivial due to the "heuristic nature" of approximation (1).
- Since in practice the true distribution $\mathcal{P}$ of the data is unknown, **approximation (2)** consists in replacing this distribution by the empirical distribution of the data, $\hat{\mathcal{P}}$. This approximation induces an estimation error on the objective that is $\mathcal{O}(1/\sqrt{N})$ ([27], Theorem 3). In short, the quality of this approximation increases with N.
- In **approximation (3)**, this problem is replaced by the sketch distance. This approximation relies on the law of large numbers and is therefore valid for large m. Note that *on average*, the sketch distance is equal to the *gamma* distance (on average over the drawing of the frequencies, approximation (3) induces no approximation error), and that increasing m *decreases the variance* of this approximation error. Characterizing this error term requires extensions of the "Information preserving guarantees" theorems from [2], theorems that are themselves still to be improved in future work by the authors.
- As discussed in this chapter, we can also interpret the `(Q)CKM` problem as a *different clustering problem* (instead of a pdf matching problem as done with approximation (1)).

Therefore, we can see **approximation (4)** as the replacement of the k-means clusterign problem with a clustering problem that compares the similarity of training examples through a translation-invariant kernel. Again, this new problem can be approximated by the sketch distance, by law of large numbers (just as approximation (3)).

- Finally, the optimization problem based on the sketch distance is not solved exactly but the centroids C^* are *a local optimum that is found by using the L-BFGS method and with approximating the gradient as a truncated Fourier series*, this is approximation (5). Note that approximpation (5) is the combination of two approximations: replacing the exact solution by a local minimum, and truncating the gradient in the local minimum search method. The approximation error commited by (5) is still an open question and will require an in-depth analysis of the properties of L-BFGS on our particular non-convex optimization problems.

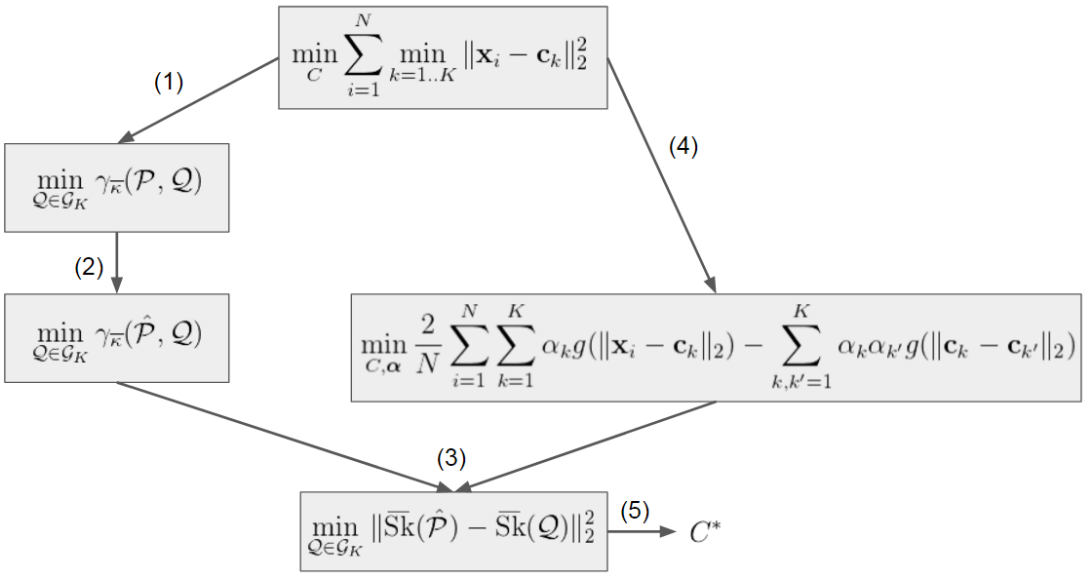


Figure 4.4: Schematic view of the different approximations that are done in `(Q)CKM`, with two different interpretations (left and right branches).

Chapter 5

Experimental results

Clustering with a general periodic signature function, and in particular the `QCKM` algorithm have been implemented in Matlab and tested on both synthetic Gaussian data and a real dataset originating from the MNIST handwritten digits image collection. This chapter presents and discusses the most significant experimental results obtained. The most important conclusion is that `QCKM` can reach performances that compete with `CKM` and `k-means++` as long as the sketch dimension m is high enough. Since we focused during our theoretical analysis on the Fourier series approximation of the universal quantization function to obtain the gradient (strategy 2), we also focused on this approximation in the experimental results. However we also implemented strategy 1 (smoothing the universal quantization function), even though we present no results for this strategy for the sake of conciseness.

5.1 Toy examples : mixture of Gaussians

5.1.1 Experiment 1: from `CKM` to `QCKM`

As a starting point, we will see how the clustering performances are affected by the signature function, and more particularly, when this signature function approaches the universal quantization. In this experiment, we run a variant of `QCKM` where the sketch (and the gradients) use as signature function $f(t)$ a truncated Fourier series of the universal quantization function $Q_\Delta(t)$

$$f(t) = \sum_{k=-K_{coef}}^{+K_{coef}} F_k e^{\mathrm{i}\frac{2\pi}{T}kt}, \tag{5.1}$$

with F_k defined in Proposition 1. The dataset for this experiment is 7000 examples generated from a mixture of 5 isotropic Gaussians in two dimensions, and we compare the following values of $K_{coef} = [1, 21, 41, 61, 81, 101]$. The results are averaged over 60 repetitions of the experiment and are represented Figure 5.1.

We can see that for $K_{coef} = 1$ the performances are similar to `CKM`, which is to be expected since the signature function is (almost) the same (the complex exmponential is replaced by a sine function). When the number of Fourier coefficients increase, the signature function ressembles less and less to the complex exponential and becomes closer to the universal quantization function. As we can observe in Figure 5.1, this causes the clustering performances to degrade with the amount of Fourier coefficients. This shows that the clustering performances worsen when the complex exponential is replaced by the universal quantization. We also observe that the performances of `QCKM` stabilize around $K_{coef} = 41..61$. We will use this as an argument to choose the number of Fourier coefficients to keep in the following experiments.

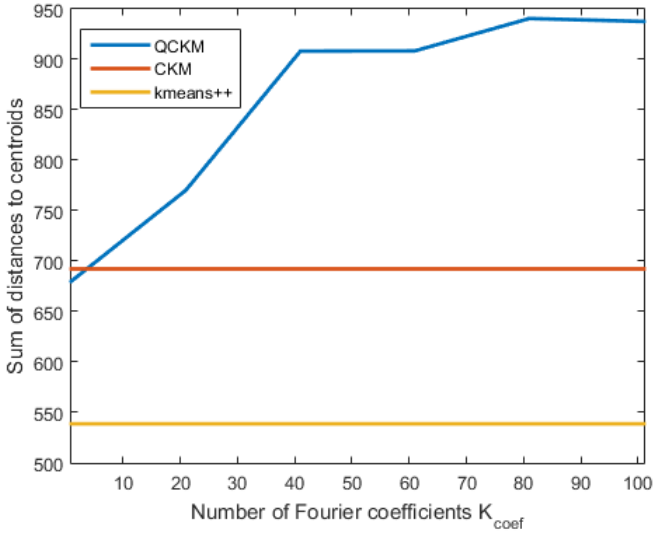


Figure 5.1: Evolution of the average distance of the learning examples to the centroids with the number of Fourier coefficients used to approximate the universal quantization function.

5.1.2 Experiment 2: comparing the different frequency drawing patterns

We now keep a fixed number of Fourier coefficients in the approximation of the gradient, $K_{coef} = 51$. In this second experiment, we compare the performances in terms of SSE of the `CKM` with the `QCKM` algorithm for different choices of a frequency density function Λ. The learning dataset is a set of $N = 7500$ points drawn from a mixture of 2 isotropic Gaussians in dimension $n = 10$ with the same covariance $\Sigma = \frac{1}{2\sqrt{10}} * I$ but with different means, respectively $+\mathbf{1}_{10}$ and $-\mathbf{1}_{10}$. We fit $K = 10$ centroids on this dataset using both `CKM` with the three heuristics proposed by [2], and `QCKM` with the three same heuristics, as well as their variants adapted to the quantized sketch. In this example, in order to reduce the variance of the results, we suppose that the covariance Σ of the data is known when drawing the frequencies, so we don't have to run an algorithm to estimate $\overline{\sigma}^2$ to approximate Σ as $\overline{\sigma}^2 I$, but we must keep in mind that in practice Σ is unkown. Different values of m, the number of frequencies to draw, are tested, and for each value of m, we report the average performance over 10 repetitions of the selection of the frequencies. The results are reported Figure 5.2 (for `CKM`) and 5.3 (for `QCKM`).

First, from Figure 5.2 we can conclude that the Gaussian heuristic performs very poorly on the given dataset. Since we are in dimension $n = 10$, this was to be expected since the "curse of dimensionality" is of application and the Gaussian heuristic oversamples the high frequencies (for the highest numbers of frequencies, the performances seem to improve, but only slightly). The two other heuristics outperform `k-means`, although the adapted radius (Ar) heuristic requires at least $m = 400$ while the Gaussian radius (Gr) requires only $m = 100$. We thus observe, suprisingly, that at low frequencies (Gr) outperforms (Ar) even though [2] reports that (Ar) performs better (but note that (Ar) is still slightly better at high frequencies). We suggest that the better performances of (Ar) compared to (Gr) applies primarily to GMM estimation and not for compressive clustering. We justify this claim intuitively by noting that the (Ar) heurisitics is built on a criterion that allows to discriminate Gaussian distributions and in particular, allows discrimate different variances. In compressive clustering, we are not interested in the information provided about the variance of the clusters, but only in their means, so the intuitive gradient justification for (Ar) is less relevant than it is in GMM estimation.

We now turn to our new algorithm, `QCKM`, whose performances are represented Figure 5.3.

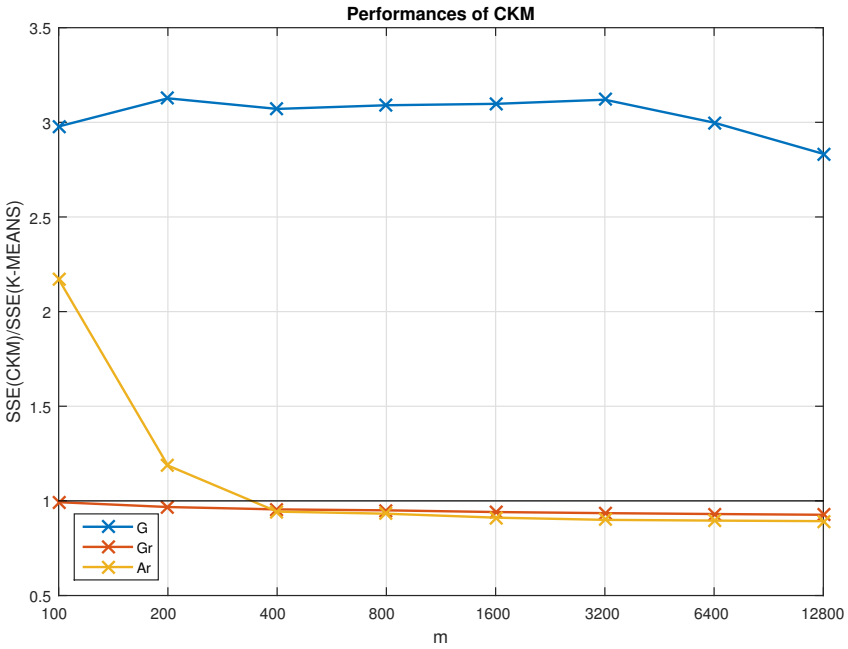


Figure 5.2: Average Sum of Squared Errors of `CKM` relative to the SSE obtained by `k-means++` for 10 experiments as a function of m in a logarithmic scale. When the value is less than 1 (black line), it means that `CKM` outperforms `k-means++`. The blue, orange and yellow line correspond to respectively the Gaussian heuristic (G), the Gaussian radius heuristic (Gr) and the adapted radius heuristic (Ar).

First, we observe that, when using the same frequencies as the ones adapted to `CKM` (plain lines), the same relative performances between the three heuristics appear: (G) does not perform well even at high frequencies, and (Gr) and (Ar) outperform `k-means++` given the number of frequencies is high enough, with again better performances for (Gr). However, the number of frequencies m to obtain accurate clustering results is much higher than needed by `CKM`. In general for all experiences (even the ones not presented in this report), we have observed that for `QCKM` there seems to be a factor of 20...50 in the number of frequencies m needed in order to have good (i.e. same accuracy as k-means) clustering results compared to `CKM` . We will try to explain this behavior when discussing the next experience. If we look at the modified frequency densities adapted to `QCKM` that we proposed in Chapter 4 (the dashed lines in Figure 5.3), called (G-h), (Gr-h) and (Ar-h), we observe that each modified heuristic outperforms its counterpart that is not adapted to the quantization. The proposed heuristics are thus indeed improving the clustering capabilities of `QCKM`. When using the adapted heuristics less frequencies are needed, but at high frequencies the performances are the same for all heuristics. Interestingly, the adapted Gaussian heuristic (G-h) outperforms the `CKM` performances of the Gaussian heuristic (G), Figure 5.2. We believe that this is due to the fact that the adapted heuristic has a tendency to sample more low frequencies. However, this advantage is not significant enough to make the Gaussian frequency distribution a viable choice in pratice.

5.1.3 Experiment 3: phase transition diagrams

The last experience suggests that `QCKM` can obtain good clustering performances when the number of frequencies m is increased. In this experiment we are going a little bit further by confirming whether or not the empirical rule $m = \mathcal{O}(nK)$ observed for `CKM` still holds for `QCKM`. In this experiment we use solely the Gaussian radius (Gr) frequency sampling distribution and its

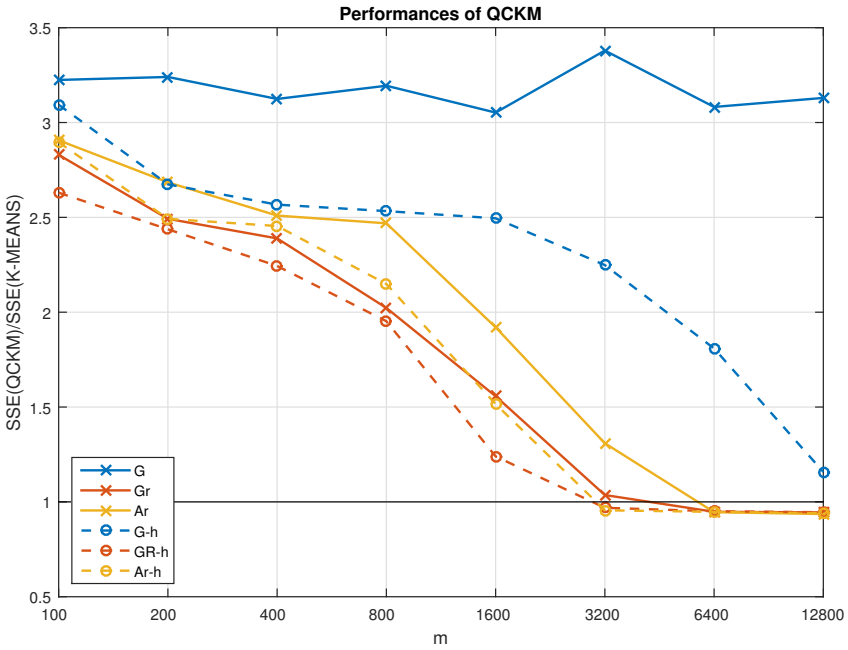


Figure 5.3: Average Sum of Squared Errors of `QCKM` relative to the SSE obtained by `k-means++` for 10 experiments as a function of m in a logarithmic scale. The blue, orange and yellow solid line correspond to respectively the Gaussian heuristic (G), the Gaussian radius heuristic (Gr) and the adapted radius heuristic (Ar) for `CKM`, while the dashed lines correspond to their variants adapted to `QCKM`, respectively (G-h), (Gr-h) and (Ar-h).

adptation (Gr-h) since they yielded the best results in experiment 2.

Experiment 3a: the performances as a function of m and n, the problem dimension

We first let the dimension if the learning examples, n, vary. In this experience, the $N = 10000$ learning examples are drawn from a mixture of two isotropic Gaussians with mean $\mathbf{1}_n$ and $-\mathbf{1}_n$. We chose a variance proportional to n in order to have a comparable distance between the clusters when n varies (because the distance between the two means of the Gaussians is $2\sqrt{n}$, therefore the width σ of the clusters should be proportional to $\sqrt{n}$); more precisely we choose $\Sigma = \frac{n}{20} I$. On this data, we fit again $K = 10$ centroids. We let n vary from 5 to 30 dimensions by steps of 5, and the number of frequencies we draw is $m = cnK$ for various values of c, in particular, $c = 0.2, 0.5, 1, 5, 10, 20$ and 50. For each value of n and m, we are taking the average over 10 repetitions of the experience for `CKM` and for `QCKM` with both the (Gr) and (Gr-h) frequency sampling patterns; the results are shown Figures 5.4 and 5.5.

The main observation from Figure 5.5 is the fact that the empirical rule $m = \mathcal{O}(n)$ still holds for `QCKM` (except a deviation at small n that is also present in `CKM`, probably linked to a lesser impact of the curse of dimensionality). By comparing those graphs with Figure 5.4, as we already observed, we conclude that `QCKM` has good performances given the number of frequencies is approximatively 20 (for the (Gr-h) frequency distribution) to 40 (for the (Gr) frequency distribution) times higher than in `CKM`. Using our adapted frequency sampling heuristic gives a reduction of the number of frequency to use by almost a factor 2.

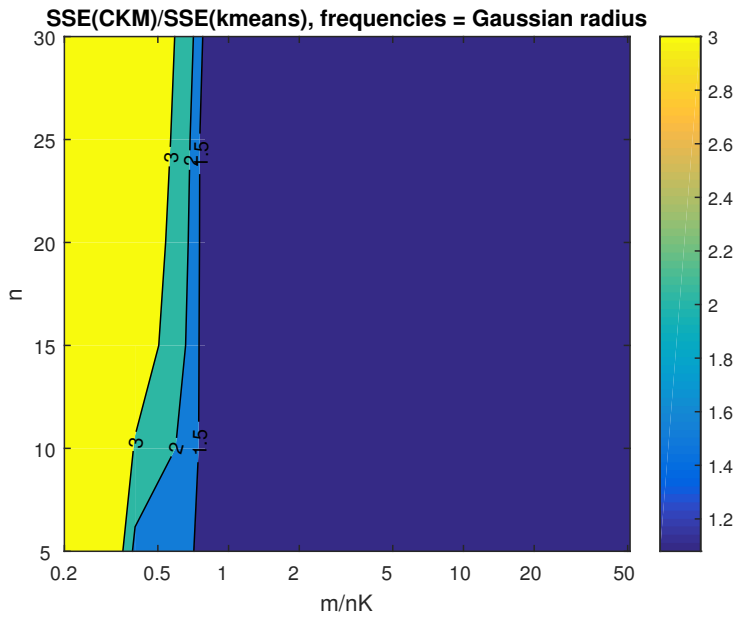
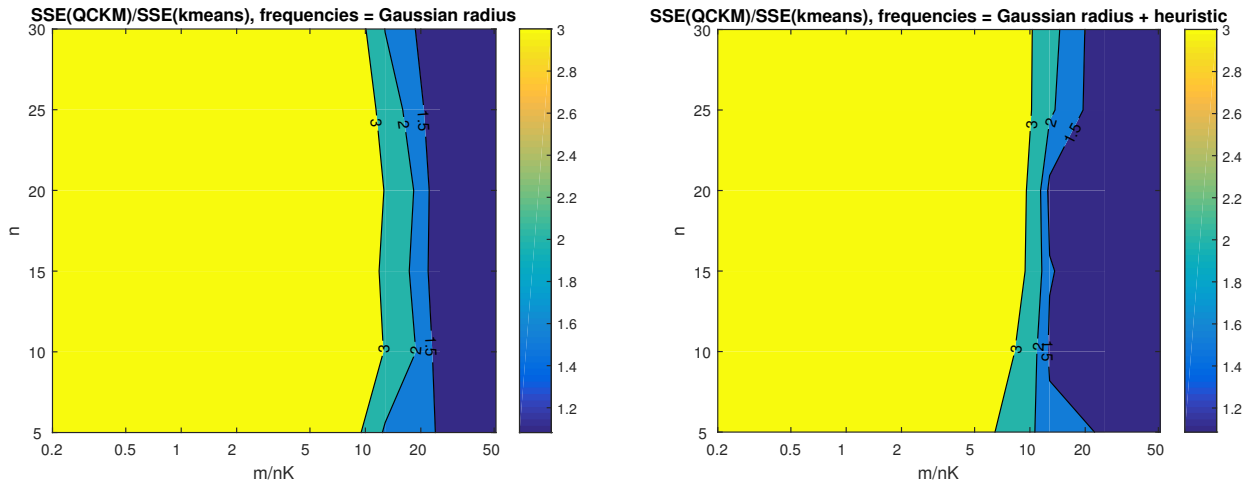


Figure 5.4: Phase transition diagram for the `CKM` perfomances : the color represents the average SSE obtained for `CKM` relative to the one obtained with `k-means++`. In the yellow region, the relative SSE is higher than 3 while in the dark blue region it is lower than 1.5. In between, there is a transition (we will call the isoline where the relative SSE = 2 the phase transition). Here the phase transition happens between $m = 0.5nK$ and $m = nK$.



(a) Sampling : (Gr)

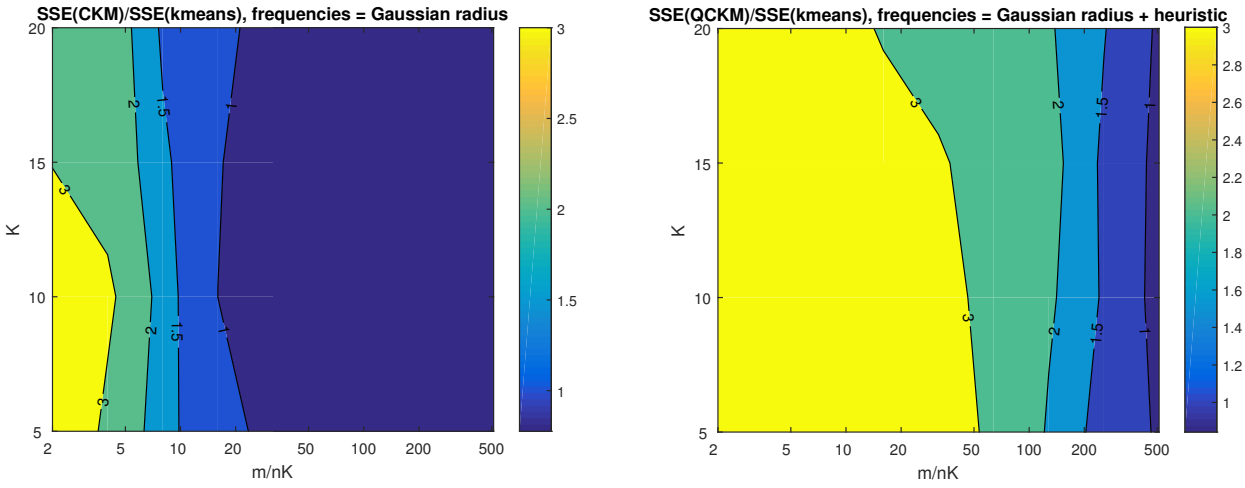
(b) Sampling : (Gr-h)

Figure 5.5: Phase transition diagram for the `QCKM` perfomances with the (Gr) frequency sampling (left) and (Gr-h) frequency sampling (right) when the dimension n varies. The phase transition happens both between $m = 10nK$ and $m = 20nK$ for (Gr) and (Gr-h), and depends linearly on n.

Experiment 3b: the performances as a function of m and K, the number of clusters

We are now going to vary the number of clusters K in fixed dimension $n = 5$. Here the learning examples are drawn from a mixture of K isotropic Gaussians with variance as described in experiment 3a (except we only have one value of n) and the K means of those Gaussian distributions are K random draws from the set $\{-1, +1\}^n$ (we place clusters in K of the $2^5 = 32$ hyperoctants of $\mathbb{R}^5$). We use K centroids to cluster this dataset. This time, we only use the adapted Gaussian

radius heuristic (Gr-h) for drawing the frequencies for QCKM since it was the best performing heuristic in the previous experiments; the results are shown Figure 5.6.



(a) CKM with (Gr)

(b) CKM with (Gr-h)

Figure 5.6: Phase transition diagram for the CKM perfomances with the (Gr) frequency sampling (left) and the QCKM performances with (Gr-h) frequency sampling (right) when the number of clusters and centroids K varies. For CKM, the phase transition happens for $m \simeq 5nK$, and for QCKM for $m \simeq 100nK$, and depends linearly on K.

From Figure 5.6 and experiment 3a we can conclude that CKM's empirical rule $m = \mathcal{O}(nK)$ still holds for QCKM. As in the previous experiment, there is a factor of at least 20 between the number of frequencies necessary for CKM and QCKM.

We now give some hypotheses for the increase of m in the context of QCKM. There are two factors that could play in this increase (the truth is most likely to be a combination of both of them). First, without considering the algorithm and the frequency distribution, the universal quantization might be less suited than the complex exponential to record information about the clusters in the form of a sketch. This would be due to an "inherent" difficulty of QCKM, intuitively justified by the fact that the sketch contains "less" information[1] than the complex exponential. Another way to see this is by remembering that the quantized sketch can be interpreted as a sampling of a "square wave" transform of the probability density function underlying the data whereas the usual sketch is its Fourier transform; this square wave transform is intuitively less suited to represent smooth probability density functions than the Fourier transform, i.e. it might require more samples for an accuracte estimate of the centroids. If the increase of m is solely due to this -potential- inherent inability of QCKM to perform clustering with as many frequencies as CKM, this would be bad news since there is nothing we can do about it; however we believe there is cause for the increase in m.

The second -and more motivating- reason for the increasing m could lie in the actual implementation of QCKM, more particularly in the way the sketch is designed (choice of Λ) and in the way the QCKM problem is solved. We have already seen that the choice of Λ is a crucial parameter regarding the sketch performances in terms of learning, and up to now we only have *heuristics*

[1]Since the quantized sketch can only take 2 values, it naturally contains less information than the complex exponential that can theoretically (i.e. not on an actual computer) take an infinite number of values. But as we discuss in Chapter 6, we must be careful when defining a "fair" comparison between CKM and QCKM, since the latter requires potentially less bits to store.

to select the frequency distributions. Empirical results -not reported here for conciseness- have shown that downscaling the frequencies Ω by, for example, a factor 10 yield much better clustering results, suggesting that there is room for improvement in the choice of Λ. On the other hand, for a given frequency distribution, we also do not know how close the local optimum discovered by `QCKM` lies from the true optimum. It might very well be the case that the optimization steps in `QCKM` are more easily stuck in (bad) local optima than `CKM`, calling for an increased number of frequencies to reduce the amount of local optima. As we suggest in the next experiment, the practical `QCKM` algorithm presented up to now might be improved by considering optimization schemes specialized for discontinuous, or very flat, optimization landscapes.

5.1.4 Experiment 4: the bottleneck of `QCKM`, aka execution time

The `QCKM` algorithm produces clustering results that compete favorably with `CKM` and `k-means++`, but the main practical drawback -right now- is its execution time. The source of this slower execution time is double: first, more frequencies m are needed for `QCKM` ($m = \mathcal{O}(nK)$ is still valid but with a higher *hidden constant* in the big-O notation, see experience 2 and 3), but as we will see now even for an equal number of frequencies m, the execution of `QCKM` is slower. If we look at the *complexity* of `QCKM`, we obtain the same complexity as `CKM`, that is, $\mathcal{O}(K^2mn)$ (this comes directly since the steps of both algorithms are the same, they are just executed differently). We show in this experiment that this rule indeed still holds (at least for the dependence in m) but -similarly to the rule for m- with a higher hidden constant. We have included the computation of the sketch in our measure of time even though it could be computed much faster using parallel computing, but we have empirically determined that the sketch computation time took up only a small fraction (less than 5%) of the total computation time for the modest chosen value of N, so the computation time computed can be interpreted as the time for running the algorithm and not for computing the sketch.

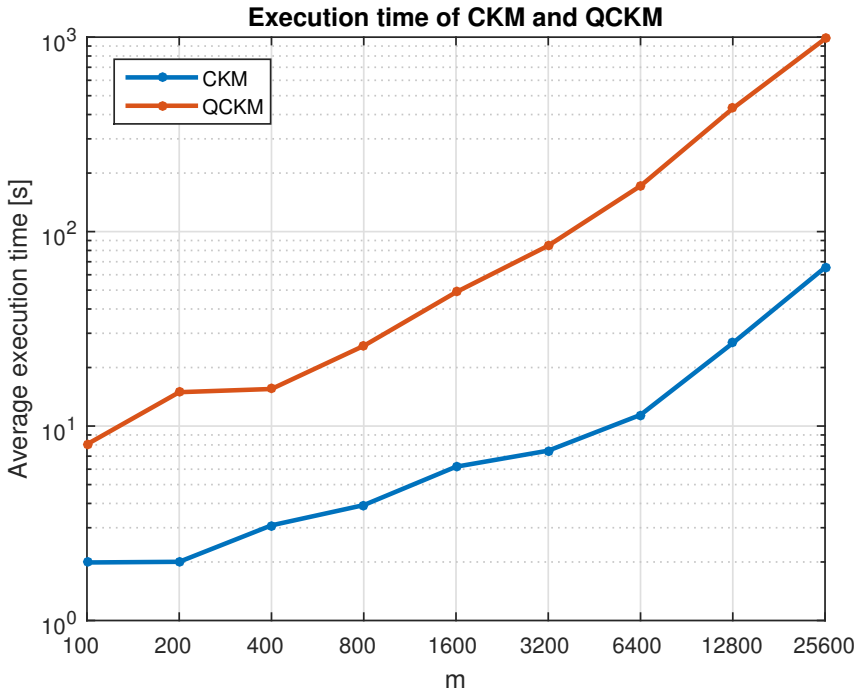


Figure 5.7: Average execution time of `CKM` (blue) and `QCKM` (orange) algorithms in seconds as a function of the number of frequencies m in a log-log scale for enhanced readability of the results.

In this experiment we consider the same dataset generation procedure as in experiment 2, and measure the average execution time of both algorithm over 10 realizations for different values of m. Figure 5.7 represents the evolution of the mean execution time as a function of m. The first observation is that empirically, the linear dependence on m of the time complexity of those algorithms ($\mathcal{O}(K^2mn)$, as reminder) is confirmed (the slope[2] of both curves is approximatively 1, this was also confirmed by a -less elegant- plot without logarithmic axes). However, there is roughly one order of magnitude between the execution time of `CKM` and `QCKM` (the latter being at least 10 times slower). Although it is true that the current implementation of `QCKM` could probably be further improved to run faster (the goal of our code is still to be a "proof of concept" rather than a competitive implementation), we strongly believe that the main reason for this slower execution resides in the solving time of the different optimization problems. More precisely, the optimization problems in `QCKM` are harder (hence more costly in time) to solve than for `CKM`. Indeed, the objective function of `QCKM`, even when smoothed, is very flat compared to the one of `CKM`. As a result, the gradients of `QCKM` will most of the time be close to zero, and the L-BFGS optimization scheme will be forced to make a lot more, smaller, iterations than when solving the problems for `CKM`. In future work, more specialized optimization methods could be considered to counter this drawback.

5.2 MNIST dataset

We now consider a "real-life" dataset to assert the performances of `QCKM` on real data. We thus consider the well-known MNIST dataset, a collection of $N = 70000$ 28×28 pixel images of hand-written digits. Our goal will be to detect, in an unsupervised manner (i.e., without using the labels of the images) the 10 clusters corresponding to the different numbers. Since the shape of the clusters in the original 28×28-dimensional space are possibly very complex and not representable by a single centroid, we will here use a spectral clustering algorithm, and use `QCKM` on the "new" dataset Y made of the eigenvectors, as illustrated Figure 5.8. To construct the adjacency matrix of the graph associated with the dataset[3], the SIFT descriptors of the different images are computed, then, the a nearest-neighbor search is performed on those features, and edges are added between training examples if they share similar descriptors. The graph Laplacian of this graph is then computed and the 10 eigenvectors associated with the samllest eigenvalues are extracted, each of those vectors defning a feature in the new dataset Y. Finally, we compare the different clustering algorithms on this new dataset Y. We would like to stress the fact that this experiment does not fit completely in a "compressive" framework since there is a heavy pre-processign step prior to the compressive part, but the goal is to evaluate the clustering performances on real data that are not necessarily Gaussian.

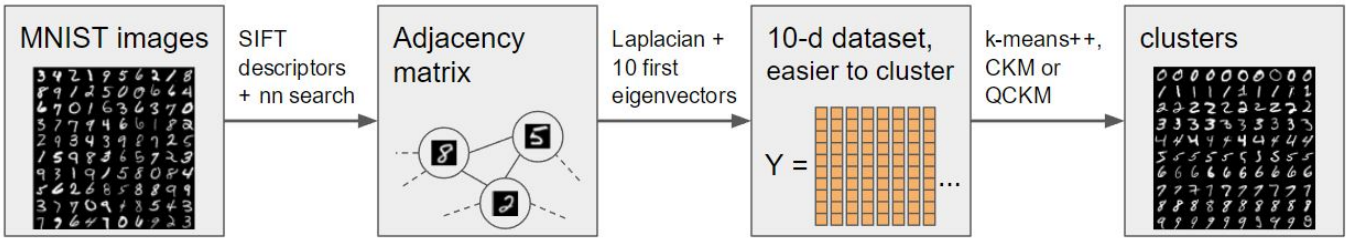


Figure 5.8: Experimental setup on the MNIST dataset : the first two setps are spectral clustering "pre-processing", and it is only in the last step that the clustering is performed.

[2]If T represents the time and $T = cm^p$ for a constant c, we must have $\log T = p \log m + \log c$, where in our case $p = 1$.

[3]Generously provided to us by the authors of [1].

We will use two metrics to evaluate the performances of the clustering algorithms. While the first one, the Sum of Squared Errors (or sum of distances to the closest centroids), is the most used metric for evaluating clustering performances, it is not necessarily the most meaningful for the application at hand. In our case for example, we would like to "discover" in an unsupervised manner (that is, without using the labels of the training examples) the clusters *corresponding to the actual numbers*: we somehow want to compare the clusters produced by our clustering algorithm with the ground truth labels of the images. We therefore also use the Adjusted Rand Index (ARI) that measures the similarity between partitions of datasets by comparing the number of pairs of examples that are put in the same cluster for two different clusterigns. Without entering into the details, the higher the ARI the better, with having $ARI = 1$ for two clusterings corresponds to having the exact same clusters, and $ARI = 0$ corresponds to assigning clusters at random. We compare the results obtained by `k-means` with `range` initialization[4], `CKM` with $m = 1000$ and (Gr) frequency sampling, and `QCKM` with $m = 20000$ and (Gr-h) frequency sampling (since we observed on synthetic data that there was a factor of at least 20 in the number of frequencies needed).

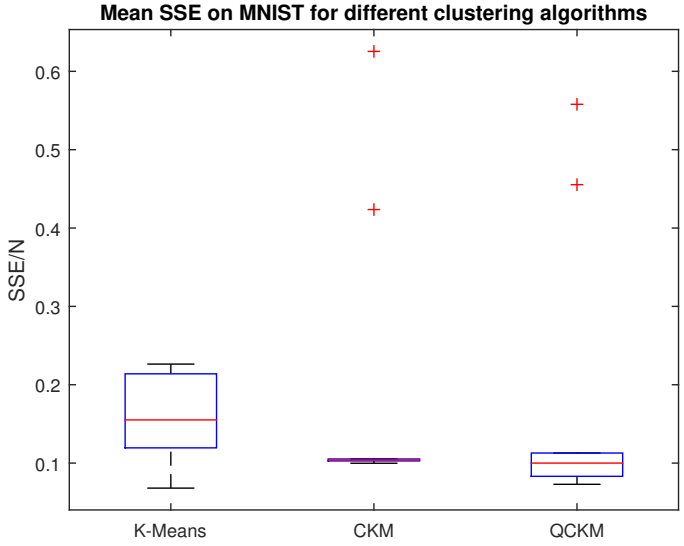


Figure 5.9: Boxplot of the mean SSE obtained on the spectral MNIST dataset for different clustering algorithms. The red line is the median of the performances, the blue box contains 50% of the performances closest to the median, and the box with the whiskers contain all the performances except eventual outliers represented by red crosses.

Figure 5.9 represents with a boxplot all the means SSE obtained by the three algorithms for 10 experiments. We can see that both `CKM` and `QCKM`, for the selected numbers of frequencies, globally outperform `k-means` in terms of SSE, except for some outliers. Those algorithms are also more stable to initialization as the variance of their performances is smaller, and compared to `CKM`, the `QCKM` algorithm presents more variance in the results obtained, but with the positive effect of obtaining globally a better SSE (keeping in mind however that `QCKM` has here access to a sketch with 20 times more frequencies).

We now compare the clusters obtained with the ground truth labels through the ARI, as shown in Figure 5.10 (again for 10 experiments). Both compressive clustering algorithms produced clusters that are closer to the true labels (higher ARI) compared to `k-means`. Once again, `CKM` is very stable except for some outliers. `QCKM` presents a higher variance than `CKM` but its median

[4]The initialization stage can thus be done in a compressive setting, where the dataset Y is forgotten but only the bounds are stored in memory.

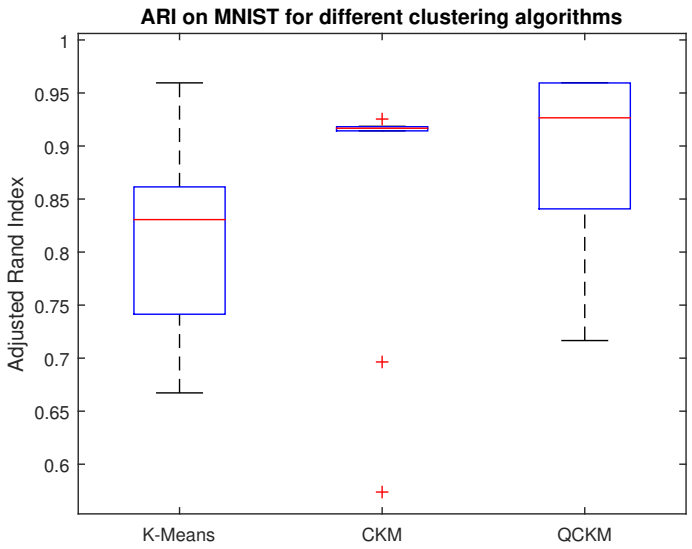


Figure 5.10: Boxplot of the Adjusted Rand Index obtained on the spectral MNIST dataset for different clustering algorithms (the closer to 1, the better). The red line is the median of the performances, the blue box contains 50% of the performances closest to the median, and the box with the whiskers contain all the performances except eventual outliers represented by red crosses.

ARI is better for the selected number of frequencies: on this example, `QCKM` produces the best digit classification results but with a high variance on the drawing on the frequencies.

Chapter 6

Conclusion

After reviewing the existing CKM compressive clustering algorithm in Chapter 2, we have proposed the new QCKM algorithm in Chapter 3 that should be easier to implement in hardware. However, this algorithm requires to solve optimization problems whose objective function is discontinuous in the decision variables. We thus have proposed two ways to smoothen this objective function in order to solve the original discontinuous problem approximatively. To better understand the implications of one of those approximations, i.e. approximating the discontinuous periodic signature function by a (truncated) Fourier series, Chapter 4 presents some theoretical results for general periodic signature functions and we particularized them to the universal quantization function. We obtained new interpretations of the modified objective function of QCKM and indications about how the frequency distributions Λ should be adapted to the universal quantized sketch. In Chapter 5 we have shown empirically that QCKM is feasible in practice at the cost of an increased number of frequencies m and slower execution time, although the adapted frequency distributions allow to reduce this increase in frequencies. The encouraging good news is that we have shown that, provided m is large enough, QCKM is competitive with the state of the art k-means++ and CKM methods, even on "real" data.

6.1 Drawbacks and discussion

The main bottleneck of the existing implementation of QCKM is its speed: since we observed empirically that QCKM requires at least 20 times more frequencies and is at least 10 times slower to converge for a constant amount of frequencies, for a particular instance QCKM is at least 200 times slower than CKM if we want to obtain similar clustering performances. Two ideas can be explored to alleviate this computational cost: 1) to reduce m, **refining the sketching operator** (and in particular, the frequency density Λ) in order to obtain a more "meaningful" sketch for clustering, and 2) to reduce the time spent optimizing, finding **specialized optimization algorithms** that are more suited to "stairsteps-like" objective functions.

The frequency selection heuristics, even when adapted to match the universal quantization signature function, are still originally selected to solve the Gaussian Mixture Estimation problem [2], and it has been shown *empirically* that they produce "good" results for CKM but are not necessarily adapted to a clustering algorithm [1]. We believe that there is room for improvement in the choice of Λ by developing specialized heuristics to the clustering problem. This claim is supported by the fact that *for GMM estimation*, the Adapted radius heuristic (Ar) outperforms the Gaussian radius heuristic (Gr) [2] while our empirical results from Chapter 5 suggest that *for clustering problems* the (Gr) heuristic yields better results. This suggests a fundamental difference in the optimal frequency distribution for those two problems. This fact is particularly noticeable in the way that the Adapted radius frequency heuristic is defined: the goal is to be

able to differentiate Gaussian probability distributions.

Another line of though to improve the choice of frequency distribution lies in the way we adapted our heuristics to the general signature function in Chapter 4: we argued that we wanted to sample $\overline{\psi}_{\mathcal{P}}$ "well enough" or based our choice on an intuitive gradient of this quantity. We propose another way to adapt the frequency distribution: since we observe that the frequency distribution Λ of `CKM` produces a better objective function (see Figure 3.1 in Chapter 3), why not *choose the new frequency sampling $\overline{\Lambda}$ such that the objective function defined by `QCKM` is the same as the one in `CKM`* when Λ is used? Building on results from Chapter 4, this is possible if we are able to solve the following equation for $\overline{\Lambda}$:

$$\sum_k |F_k|^2 D_{2\pi k/T} \overline{\Lambda} = \Lambda, \tag{6.1}$$

because we want to obtain the same kernel with `QCKM` and frequencies drawn from $\overline{\Lambda}$ as the kernel obtained with `CKM` and frequencies drawn from Λ. If the solution $\overline{\Lambda}$ exists and can be found, and assuming that the quality of the approximation of the γ-distance is the same for `CKM` and `QCKM` for a given m, it would be theoretically possible to have $m_{\texttt{QCKM}} \simeq m_{\texttt{CKM}}$. Solving (6.1) is however not a trivial task and remains an open question.

Towards a fair comparison of `CKM` and `QCKM`

Besides being closer to hardware-like operations, quantized sketching could provide advantages in the sketch storage requirements. The signature function of the quantized sketch maps to $\{-1, 1\}$ or equivalently $\{0, 1\}$ (represented by one single bit) and after pooling of N such signatures, we obtain an integer $\in [0, N]$ that can be stored on $\log_2 N+1$ bits for each frequency. Thus `QCKM`'s sketch needs $m(\log_2 N+1)$ bits. We are now going to see that this sketch size can be reduced even further

We assume that each contribution of an example i to the sketch entry j, $q_\Delta(\boldsymbol{\omega}_j^T \boldsymbol{x}_i + \xi_j)$, is an independent random variable Z_{ij} with $\mathbb{P}[Z_{ij} = 1] = \mathbb{P}[Z_{ij} = 0] = 1/2$. Therefore, the entries $z_j = \sum_i Z_{ij}$ of the universal quantization sketch can be modeled as Binomial distributed random variables. The sketch coefficients therefore concentrate around $N/2$ with very high probability, and it should be possible to avoid encoding the whole range $[0, N]$ with no perceptual loss (for example, this could allow to encode the sketch on only $0.5m \log_2 N$ bits).

To compare `CKM` and `QCKM` on a fair basis, we should compare their performances with an *equal bit budget*. The bit budget needed to encode `CKM`'s sketch is $2Bm$ where B is the amount of bits allocated to represent the real or the imaginary part of one sketch coefficient. In order to implement a comparison of the performances with equal bit budget, we can act on two degrees of freedom. First, `CKM`'s sketch can be quantized (we reduce B) introducing a quantization error. As another option, the number of frequencies can be reduced (we reduce m) and the sketch coefficients are encoded with full precision (where the precicison depends on the architecture running the algorithms, for example $B = 32$ bits). An in-depth comparison of `CKM` and `QCKM` at equal bit budget is left for future work. Our hope is that the conclusions of such an anlysis would balance the drawback of `QCKM`, that needs a higher amount of measurements m.

6.2 Additional directions for future work

As suggested at the end of Chapter 4, there is still work to be done concerning the bounds characterizing the quality of the solution achieved by the `QCKM` algorithm: so far, no guarantees

are known for the practical solution given by running `QCKM` relative to the actual optimal solution to the sketch matching optimization problem. In turn, the quality of the approximation of the pdf matching problem by the sketch matching problem as a function of m still needs to be quantified precisely. The theorems of Section 7 in [2] are a start but i) the authours acknowledge that they have to be improved, ii) they need to be particularized to the clustering problem and iii) adapted to the sketch by universal quantization.

In this work, we have focused on discovering the centroids in a dataset that is observed through a compressed version of it (a sketch). In practical applications, it would be interesting to actually use those centroids for classifying new learning examples (generalization). Future work should also focus on this "second phase" of compressive clustering, that is, classifying previously unseen examples that are compressed at acquisition.

Towards using `QCKM` for real applications

To motivate further study of compressive clustering by 1-bit sketching, two conditions still need to be fulfilled. First, the **practical feasability of hardware implementations of sensors** that would acquire the sketch by universal quantization is still an open question. When we proposed `QCKM`, we assumed that the universal quantization was easy to implement in hardware, but this assumption needs to be backed up by practical design of sensors relying on this operation to compute the sketch.

The second requirement to motivate further work on `QCKM` is to find good **applicative contexts** where clustering with `QCKM` provides substantial advantages over traditional clustering algorithms. For example, an imager that is able to detect certain events based on few compressive measurements could benefit from the datarate reduction achieved by `QCKM`. An other application could be automated signature detection in hyperspectral images, where each hyperspectral pixel can be seen as an "example" that can belong to different classes (spectral signatures). Acquiring the whole hyperspectral volume would then not be needed, but only a sketch of it should suffice. However, since the hyperspectral pixels can be modeled by a Dirichelet distribution [39] and not as a simple clusters, the compressive clustering ideas would need to be modified before being applied to this setting.

Going even further

So far, compressive learning strategies that use the sketch of a dataset have been restricted to GMM estimation and clustering. However, there are a lot of other machine learning tasks that could benefit from a reduced size of the dataset. In the future, the entire machine learning field could be revisited in the light of the possibility to sketch very large datasets without even aquiring the whole dataset, with the possibility to reduce the heavy computational load of existing algorithms. This work also raises an interesting question: besides the complex exponential and the universal quantization, are there other interesting signature functions that can be used to build a sketch? Maybe, there exists some "ideal" sketch signature function, that captures exactly the information needed by given machine learning algorithms with a very small amount of measurements m?

In short, we only scratched the surface of the possibilities that could be explored by the concept of the *sketch* of a dataset, both in the diversity of the applications that could benefit from its datarate reduction capabilities and in its power to succintly represent "just the information we need" of very large datasets.

Bibliography

[1] N. Keriven, N. Tremblay, Y. Traonmilin, and R. Gribonval, "Compressive K-means," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (New Orleans, United States), Mar. 2017.

[2] N. Keriven, A. Bourrier, R. Gribonval, and P. Pérez, "Sketching for Large-Scale Learning of Mixture Models," *CoRR*, vol. abs/1606.02838, 2016.

[3] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "NP-hardness of Euclidean sum-of-squares clustering," *Machine Learning*, vol. 75, no. 2, pp. 245–248, 2009.

[4] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[5] D. Arthur and S. Vassilvitskii, "k-means++: The Advantages of Careful Seeding," *ACM-SIAM symposium on Discrete algorithms*, pp. 1027–103, 2007.

[6] G. Hamerly and J. Drake, *Accelerating Lloyd's Algorithm for k-Means Clustering*, pp. 41–78. Cham: Springer International Publishing, 2015.

[7] D. Arthur and S. Vassilvitskii, "On the Worst-Case Complexity of the k-means Method," *Technical Report, Stanford 698*, 2005.

[8] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[9] D. Wagner and F. Wagner, "Between Min Cut and Graph Bisection," in *Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science*, MFCS '93, (London, UK, UK), pp. 744–750, Springer-Verlag, 1993.

[10] E. J. Candes and M. B. Wakin, "An Introduction To Compressive Sampling," *IEEE Signal Processing Magazine*, vol. 25, pp. 21–30, March 2008.

[11] E. J. Candès, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[12] H. R. Simon Foucart, *A Mathematical Introduction to Compressive Sensing*. Applied and Numerical Harmonic Analysis, Birkhäuser Basel, 1 ed., 2013.

[13] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, pp. 83–91, March 2008.

[14] L. Jacques, P. Vandergheynst, A. Bibet, V. Majidzadeh, A. Schmid, and Y. Leblebici, "CMOS compressed imaging by Random Convolution," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1113–1116, April 2009.

[15] B. K. Natarajan, "Sparse Approximate Solutions to Linear Systems," *SIAM Journal on Computing*, vol. 24, no. 2, pp. 227–234, 1995.

[16] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic Decomposition by Basis Pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[17] E. J. Candès, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, vol. 346, no. 9, pp. 589 – 592, 2008.

[18] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3397–3415, Dec 1993.

[19] Y. C. Pati, R. Rezaiifar, Y. C. P. R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition," in *Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems, and Computers*, pp. 40–44, 1993.

[20] P. Jain, A. Tewari, and I. S. Dhillon, "Orthogonal Matching Pursuit with Replacement," *CoRR*, vol. abs/1106.2774, 2011.

[21] P. T. Boufounos, S. Rane, and H. Mansour, "Representation and Coding of Signal Geometry," *CoRR*, vol. abs/1512.07636, 2015.

[22] L. Jacques, J. N. Laska, P. Boufounos, and R. G. Baraniuk, "Robust 1-Bit Compressive Sensing via Binary Stable Embeddings of Sparse Vectors," *CoRR*, vol. abs/1104.3160, 2011.

[23] P. T. Boufounos and S. Rane, "Efficient Coding of Signal Distances Using Universal Quantized Embeddings," in *2013 Data Compression Conference*, pp. 251–260, March 2013.

[24] M. A. Davenport, P. T. Boufounos, M. B. Wakin, and R. G. Baraniuk, "Signal Processing With Compressive Measurements," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, pp. 445–460, April 2010.

[25] M. F. Duarte, M. A. Davenport, M. B. Wakin, and R. G. Baraniuk, "Sparse Signal Detection from Incoherent Projections," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 3, pp. III–III, May 2006.

[26] A. R. Hall, *Generalized method of moments*. Oxford University Press, 2005.

[27] B. K. Sriperumbudur, "Mixture density estimation via Hilbert space embedding of measures," in *2011 IEEE International Symposium on Information Theory Proceedings*, pp. 1027–1030, July 2011.

[28] J. Haupt, R. M. Castro, and R. Nowak, "Distilled Sensing: Adaptive Sampling for Sparse Detection and Estimation," *IEEE Transactions on Information Theory*, vol. 57, pp. 6222–6235, Sept 2011.

[29] B. Schölkopf, "The Kernel Trick for Distances," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, (Cambridge, MA, USA), pp. 283–289, MIT Press, 2000.

[30] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the Amererican Mathematical Sociecty*, no. 68, pp. 337–404, 1950.

[31] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. R. Lanckriet, "Hilbert Space Embeddings and Metrics on Probability Measures," *J. Mach. Learn. Res.*, vol. 11, pp. 1517–1561, Aug. 2010.

[32] W. Rudin, *Fourier Analysis on Groups*. Interscience Publishers, 1962.

[33] A. Rahimi and B. Recht, "Random Features for Large-Scale Kernel Machines," in *Advances in Neural Information Processing Systems 20* (J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, eds.), pp. 1177–1184, Curran Associates, Inc., 2008.

[34] T. Yang, Y.-f. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou, "Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 476–484, Curran Associates, Inc., 2012.

[35] B. Razavi, "A tale of Two ADCs: Pipelined Versus SAR," *IEEE Solid-State Circuits Magazine*, vol. 7, pp. 38–46, Summer 2015.

[36] W. Kester, "ADC Architectures VI: Folding ADCs," *Analog Devices Inc., Rev. A*, vol. 10, no. 08.

[37] J. Rhee and Y. Joo, "Wide dynamic range CMOS image sensor with pixel level ADC," *Electronics Letters*, vol. 39, pp. 360–361, Feb 2003.

[38] A. Spivak, A. Belenky, A. Fish, and O. Yadid-Pecht, "Wide-Dynamic-Range CMOS Image Sensors - Comparative Performance Analysis," *IEEE Transactions on Electron Devices*, vol. 56, pp. 2446–2461, Nov 2009.

[39] S. Zou, H. Sun, and A. Zare, "Hyperspectral Unmixing with Endmember Variability using Semi-supervised Partial Membership Latent Dirichlet Allocation," Under Review.

Appendix A

Newton and quasi-Newton methods for nonlinear optimization

This section[1] introduces the L-BFGS method that is used in this work to solve the many non-convex optimization problems that arise in the `CKM` algorithm and its variants. We gonsider the general, *unconstrained* optimization problem with respect to decision variables $\boldsymbol{x}$:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \tag{A.1}$$

and we want to find a *local optimum* for $f(\boldsymbol{x})$.

A.1 Line search methods

Most optimization schemes for nonlinear optimization are line search methods. A line search method starts from an initial iterate $\boldsymbol{x}_0$ and updates the current iterate $\boldsymbol{x}_k$ by moving it in the direction $\boldsymbol{p}_k$ with a step size α_k:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k. \tag{A.2}$$

Any line search method proceeds in two steps. First, the search direction $\boldsymbol{p}_k$ is computed (the way this direction is chosen determines the particular optimization method). Then, the step size α_k is chosen such that it minimizes

$$\alpha_k \simeq \arg\min_{\alpha} f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k), \tag{A.3}$$

eventually approximatively, with the use of some conditions such as Wolfe's conditions. The most popular line search method is the gradient descent method, where $\boldsymbol{p}_k = -\boldsymbol{\nabla} f(\boldsymbol{x}_k)$. Although gradient descent has the advantage of converging globally (with assumptions on α_k) it is however quite slow (compared to Newton and quasi-Newton) because it has a linear convergence speed (intuitively, the gradient "zigzags" towards an optimum).

A.2 The Newton method

The gradient descent is a first-order method since it relies on the first-order Taylor expansion of $f(\boldsymbol{x})$ around $\boldsymbol{x}_k$. The Newton method is a second-order line search method since it relies on the second-order Taylor expansion approximation, that is,

[1] Inspired by the lectures of LINMA1702 - "Modèles et méthodes d'optimisation I" given at EPL by Prof. F. Glineur.

$$f(\boldsymbol{x}_k + \boldsymbol{p}) \simeq f(\boldsymbol{x}_k) + \boldsymbol{\nabla} f(\boldsymbol{x}_k)^T \boldsymbol{p} + \frac{1}{2} \boldsymbol{p}^T \nabla^2 f(\boldsymbol{x}_k) \boldsymbol{p} =: T_{f,\boldsymbol{x}_k}(\boldsymbol{p}), \tag{A.4}$$

and the Newton step $\boldsymbol{p}_k$ is then the direction $\boldsymbol{p}$ that minimises this second-order approximation

$$\boldsymbol{p}_k = \arg\min_{\boldsymbol{p}} T_{f,\boldsymbol{x}_k}(\boldsymbol{p}) = -\nabla^2 f(\boldsymbol{x}_k)^{-1} \boldsymbol{\nabla} f(\boldsymbol{x}_k). \tag{A.5}$$

The Newton steps have the advantage of having a quadratic convergence speed in the proximity of local optima. However, the Newton method requires to compute the (inverse of the) Hessian of f which is costly especially for high dimension n.

A.3 The quasi-Newton methods and (L-)BFGS

To combine the advantages of the Newton method (quadratic convergence speed) with those of gradient descent (only need to evaluate the gradient of f, and not the Hessian), quasi-Newton methods aim at approximating the Hessian at each iteration by $B_k \simeq \nabla^2 f(\boldsymbol{x}_k)$, and the quasi-Newton steps are then

$$\boldsymbol{p}_k = -B_k^{-1} \nabla f(\boldsymbol{x}_k) \tag{A.6}$$

where the particular choice of approximation B_k (usually updated from B_{k-1}) give rise to different methods. In particular, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) approximation rule for the Hessian approximates the Hessian by a matrix of rank 2 with the following update formula:

$$B_{k+1} = B_k + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k} - \frac{B_k \boldsymbol{s}_k \boldsymbol{s}_k^T B_k}{\boldsymbol{s}_k^T B_k \boldsymbol{s}_k} \quad \text{with} \quad \boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k, \quad \boldsymbol{y}_k = \boldsymbol{\nabla} f(\boldsymbol{x}_{k+1}) - \boldsymbol{\nabla} f(\boldsymbol{x}_k) \tag{A.7}$$

where we start from some initial approximation, for example $B_0 = I$.

Under some mild assumptions on f, the BFGS method converges superlinearly (faster than the gradient descent method) while only requiring access to the evaluation of the function f and its gradient $\boldsymbol{\nabla} f$. The L-BFGS (Limited memory BFGS) is a BFGS variant that limits the memory consumption of the algorithm. Finally, the solver also imposes linear constraints on the decision variables.

Appendix B

Computing the gradients in CKM and QCKM

This appendix gives more details about how `CKM` and `QCKM` can be implemented in practice by computing explicitly the cost functions and gradients that will have to be passed on to the LFGB-S optimization method.

B.1 Gradients for CKM

We detail here `CKM` (algorithm 5), step by step. As a preliminary result, note that the sketch for a single centroid $\boldsymbol{c}$ is obviously $\mathcal{A}\delta_{\boldsymbol{c}} = e^{-\mathrm{i}\Omega^T\boldsymbol{c}}$. As we will see, the gradients of step 1 and 5 require the Jacobian $\nabla_{\boldsymbol{c}}\mathcal{A}\boldsymbol{\delta_c}$, that will be ill-defined in the context of `QCKM`.

Step 1 : find a new centroid

Objective function

In this step we must find a new centroid $\boldsymbol{c}$ whose sketch matches the residual $\boldsymbol{r}$ the most (maximizes the real part of the correlation between them). More precisely, we must solve (since LFGB-S is a minimization algorithm, we turn the problem into a minimization by multiplying the objective by -1).

$$\arg\min_{\boldsymbol{c}} -\Re\left(\langle \frac{\mathcal{A}\delta_{\boldsymbol{c}}}{\|\mathcal{A}\delta_{\boldsymbol{c}}\|_2}, \boldsymbol{r}\rangle_2\right). \tag{B.1}$$

First, it is interesting to note that the norm of the sketch for one centroid is

$$\|\mathcal{A}\delta_{\boldsymbol{c}}\|_2 = \|e^{-\mathrm{i}\Omega^T\boldsymbol{c}}\|_2 = \sqrt{(e^{-\mathrm{i}\Omega^T\boldsymbol{c}})^T(e^{-\mathrm{i}\Omega^T\boldsymbol{c}})^*} = \sqrt{\sum_{j=1}^{m} e^{-\mathrm{i}\omega_j^T\boldsymbol{c}}e^{+\mathrm{i}\omega_j^T\boldsymbol{c}}} = \sqrt{m}$$

and thus doesn't depend on $\boldsymbol{c}$. We can simply drop it from the problem, and we re-write (B.1) as

$$\arg\min_{\boldsymbol{c}} f(\boldsymbol{c}) = \arg\min_{\boldsymbol{c}} -\Re\left(\langle \mathcal{A}\delta_{\boldsymbol{c}}, \boldsymbol{r}\rangle_2\right) = -(\Re\mathcal{A}\boldsymbol{\delta_c})^T\Re\boldsymbol{r} - (\Im\mathcal{A}\boldsymbol{\delta_c})^T\Im\boldsymbol{r} \tag{B.2}$$

where we used that for complex vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ we have :

$$\begin{aligned}
\Re\{\langle \boldsymbol{a}, \boldsymbol{b}\rangle_2\} &= \Re\{\langle (\Re\boldsymbol{a} + \mathrm{i}\Im\boldsymbol{a}), (\Re\boldsymbol{b} + \mathrm{i}\Im\boldsymbol{b})\rangle\} \\
&= \Re\{(\Re\boldsymbol{a} + \mathrm{i}\Im\boldsymbol{a})^T(\Re\boldsymbol{b} - \mathrm{i}\Im\boldsymbol{b})\} \\
&= \Re\{(\Re\boldsymbol{a})^T\Re\boldsymbol{b} + (\Im\boldsymbol{a})^T\Im\boldsymbol{b} + \mathrm{i}(\Im\boldsymbol{a})^T\Re\boldsymbol{b} - (\Re\boldsymbol{a})^T\Im\boldsymbol{b}\} \\
&= (\Re\boldsymbol{a})^T\Re\boldsymbol{b} + (\Im\boldsymbol{a})^T\Im\boldsymbol{b}.
\end{aligned} \tag{B.3}$$

Gradient

To optimize this function, we need to compute the gradient with respect to each coordinate of the centroid, noted $\nabla_{\boldsymbol{c}} f(\boldsymbol{c})$. Let us compute the l-th entry of this gradient : by linearity we have that

$$(\nabla_{\boldsymbol{c}} f(\boldsymbol{c}))_l := \frac{\partial f(\boldsymbol{c})}{\partial c_l} = -(\Re\frac{\partial \mathcal{A}\boldsymbol{\delta_c}}{\partial c_l})^T \Re\boldsymbol{r} - (\Im\frac{\partial \mathcal{A}\boldsymbol{\delta_c}}{\partial c_l})^T \Im\boldsymbol{r} \tag{B.4}$$

where $\frac{\partial \mathcal{A}\boldsymbol{\delta_c}}{\partial c_l}$ is the result of applying the derivative with respect to c_l component-wise to $\mathcal{A}\boldsymbol{\delta_c}$: we now need to explicit this term. We can write that for the k-th component we have

$$\left(\frac{\partial \mathcal{A}\boldsymbol{\delta_c}}{\partial c_l}\right)_k = \frac{\partial \left(\mathcal{A}\boldsymbol{\delta_c}\right)_k}{\partial c_l} = \frac{\partial}{\partial c_l} e^{-\mathrm{i}\omega_k^T \boldsymbol{c}} = \frac{\partial}{\partial c_l} e^{-\mathrm{i}\sum_j (\omega_k)_j c_j} = -\mathrm{i}(\omega_k)_l e^{-\mathrm{i}\omega_k^T \boldsymbol{c}}. \tag{B.5}$$

So putting everything together we write that ($\circ$ denotes the Hadamard (or "element-wise") product)

$$\frac{\partial \mathcal{A}\boldsymbol{\delta_c}}{\partial c_l} = -\mathrm{i}(\Omega_{l,:})^T \circ e^{-\mathrm{i}\Omega^T \boldsymbol{c}} = (\Omega_{l,:})^T \circ \sin(\Omega^T \boldsymbol{c}) - \mathrm{i}(\Omega_{l,:})^T \circ \cos(\Omega^T \boldsymbol{c}) \tag{B.6}$$

and this can be written in a compact form using the Jacobian matrix[1]:

$$\nabla_{\boldsymbol{c}} \mathcal{A}\boldsymbol{\delta_c} = -\mathrm{i}\ \mathrm{diag}\left(e^{-\mathrm{i}\Omega^T \boldsymbol{c}}\right)\Omega^T, \tag{B.7}$$

we can then feed this term in (B.4) to obtain

$$(\nabla_{\boldsymbol{c}} f(\boldsymbol{c}))_l = -((\Omega_{l,:})^T \circ \sin(\Omega^T \boldsymbol{c}))^T \Re\boldsymbol{r} - ((\Omega_{l,:})^T \circ \cos(\Omega^T \boldsymbol{c}))^T \Im\boldsymbol{r}. \tag{B.8}$$

Finally, this can be written for $l = 1...n$ in a compact way as

$$\begin{aligned} \nabla_{\boldsymbol{c}} f(\boldsymbol{c}) &= -(\Omega^T \circ \sin(\Omega^T \boldsymbol{c})\mathbf{1}_n^T)^T \Re\boldsymbol{r} - (\Omega^T \circ \cos(\Omega^T \boldsymbol{c})\mathbf{1}_n^T)^T \Im\boldsymbol{r} \\ &= -(\Omega \circ \mathbf{1}_n \sin(\boldsymbol{c}^T \Omega))\Re\boldsymbol{r} - (\Omega \circ \mathbf{1}_n \cos(\boldsymbol{c}^T \Omega))\Im\boldsymbol{r} \end{aligned} \tag{B.9}$$

giving us the gradient we were looking for. Or again alternatively using compact notations we find

$$\begin{aligned} \nabla_{\boldsymbol{c}} f(\boldsymbol{c}) &= -(\Re\nabla_{\boldsymbol{c}} \mathcal{A}\boldsymbol{\delta_c})^T \Re\boldsymbol{r} - (\Im\nabla_{\boldsymbol{c}} \mathcal{A}\boldsymbol{\delta_c})^T \Im\boldsymbol{r} \\ &= -\Omega\left(\mathrm{diag}\left(\sin(\Omega^T c)\right)\Re\boldsymbol{r} + \mathrm{diag}\left(\cos(\Omega^T c)\right)\Im\boldsymbol{r}\right). \end{aligned} \tag{B.10}$$

Step 3 and 4 : adjusting the weights

In this step we want to find the weights for the K centroids that minimize the l_2 norm of the residual[2]; we want to solve (we note $\mathcal{A}\boldsymbol{\delta}_C$ for $[\mathcal{A}\boldsymbol{\delta_{c_1}}...\mathcal{A}\boldsymbol{\delta_{c_K}}]$)

$$\begin{aligned} \boldsymbol{\alpha}_{opt} = \arg\min_{0\leq\boldsymbol{\alpha}\leq 1} g(\boldsymbol{\alpha}) &= \arg\min_{0\leq\boldsymbol{\alpha}\leq 1} \|\boldsymbol{z} - \sum_{k=1}^{K} \alpha_k \mathcal{A}\boldsymbol{\delta_{c_k}}\|_2^2 \\ &= \arg\min_{0\leq\boldsymbol{\alpha}\leq 1} \|\boldsymbol{z} - \mathcal{A}\boldsymbol{\delta}_C\boldsymbol{\alpha}\|_2^2 \\ &= \arg\min_{0\leq\boldsymbol{\alpha}\leq 1} \|\Re\{\boldsymbol{z} - \mathcal{A}\boldsymbol{\delta}_C\boldsymbol{\alpha}\}\|_2^2 + \|\Im\{\boldsymbol{z} - \mathcal{A}\boldsymbol{\delta}_C\boldsymbol{\alpha}\}\|_2^2 \end{aligned} \tag{B.11}$$

This is in fact a constrained least-squares problem. The gradient is

[1] Here one column of the Jacobian matrix corresponds to the derivative with respect to one coordinate of $\boldsymbol{c}$.

[2] Actually we minimize the square of this quantity because it is easier and gives the same solution.

$$\nabla_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}) = -2(\Re\{\mathcal{A}\boldsymbol{\delta}_C\})^T(\Re\{\boldsymbol{z} - \mathcal{A}\boldsymbol{\delta}_C\boldsymbol{\alpha}\}) - 2(\Im\{\mathcal{A}\boldsymbol{\delta}_C\})^T(\Im\{\boldsymbol{z} - \mathcal{A}\boldsymbol{\delta}_C\boldsymbol{\alpha}\}) \tag{B.12}$$

which enables the use of LFGBS optimization. Note that although this is what is implemented in the SketchMLbox toolbox, it is also possible to solve the unconstrained least-squares problem and then to project the solution on the set of allowed values $\{\boldsymbol{\alpha}|\alpha_i \geq 0, \sum_i \alpha_i = 1\}$.

Step 5 : additional reduction of the cost function

Now we do a global gradient descent on the same objective function, initialized with the current parameters. All the centroids and their weights are allowed to change. We want to solve

$$\min_{\boldsymbol{\alpha},\boldsymbol{c}_1,...,\boldsymbol{c}_K} \quad g(\boldsymbol{\alpha}, \boldsymbol{c}_1, ..., \boldsymbol{c}_K) = \min_{\boldsymbol{\alpha},\boldsymbol{c}_1,...,\boldsymbol{c}_K} \quad \|\boldsymbol{z} - \mathcal{A}\boldsymbol{\delta}_C\boldsymbol{\alpha}\|_2^2 \tag{B.13}$$

so we need the gradient with respect to all the variables : $\nabla_{\boldsymbol{\alpha}} g$, $\nabla_{\boldsymbol{c}_1} g$, ... , $\nabla_{\boldsymbol{c}_K} g$. We already computed the expression of $\nabla_{\boldsymbol{\alpha}} g$ above, so it remains to compute $\nabla_{\boldsymbol{c}_l} g$ for $l = 1...K$.

We compute (first a general expression where $\nabla_{\boldsymbol{c}}\mathcal{A}\boldsymbol{\delta}_{\boldsymbol{c}}$ appears, then a more detailed expansion):

$$\begin{aligned}
\nabla_{\boldsymbol{c}_l} g &= \nabla_{\boldsymbol{c}_l}\|\boldsymbol{z} - \mathcal{A}\delta_C\boldsymbol{\alpha}\|_2^2 = 2(\boldsymbol{z} - \mathcal{A}\delta_C\boldsymbol{\alpha}) \cdot \nabla_{\boldsymbol{c}_l}(\boldsymbol{z} - \mathcal{A}\delta_C\boldsymbol{\alpha}) = -2(\boldsymbol{z} - \mathcal{A}\delta_C\boldsymbol{\alpha}) \cdot \alpha_l \nabla_{\boldsymbol{c}_l}\mathcal{A}\delta_{\boldsymbol{c}_l} \\
&= \nabla_{\boldsymbol{c}_l}\|\Re\{\boldsymbol{z} - \mathcal{A}\boldsymbol{\delta}_C\boldsymbol{\alpha}\}\|^2 + \|\Im\{\boldsymbol{z} - \mathcal{A}\boldsymbol{\delta}_C\boldsymbol{\alpha}\}\|^2 \\
&= \nabla_{\boldsymbol{c}_l}\|\Re\{\boldsymbol{z} - \sum_{k=1}^{K}\alpha_k e^{-i\Omega^T\boldsymbol{c}_k}\}\|^2 + \|\Im\{\boldsymbol{z} - \sum_{k=1}^{K}\alpha_k e^{-i\Omega^T\boldsymbol{c}_k}\}\|^2 \\
&= \nabla_{\boldsymbol{c}_l} \quad \sum_{j=1}^{m}\left(\Re z_j - \sum_{k=1}^{K}\alpha_k \Re e^{-i\omega_j^T\boldsymbol{c}_k}\right)^2 + \sum_{j=1}^{m}\left(\Im z_j - \sum_{k=1}^{K}\alpha_k \Im e^{-i\omega_j^T\boldsymbol{c}_k}\right)^2 \\
&= 2\sum_{j=1}^{m}\left(\Re z_j - \sum_{k=1}^{K}\alpha_k \Re e^{-i\omega_j^T\boldsymbol{c}_k}\right)\left(+\alpha_l \Re\{e^{-i\omega_j^T\boldsymbol{c}_l} i\omega_j\}\right) + \text{same for } \Im
\end{aligned} \tag{B.14}$$

and finally, we have the global gradient by putting all the gradients together.

B.2 Gradients for the generalized sketch

Note that we now have $\overline{\mathcal{A}}\delta_{\boldsymbol{c}} = f(\Omega^T\boldsymbol{c} + \boldsymbol{\xi}) = \sum_k F_k e^{\mathrm{i}k\frac{2\pi}{T}(\Omega^T\boldsymbol{c}+\boldsymbol{\xi})}$ and we must compute the Jacobian $\boldsymbol{\nabla}_{\boldsymbol{c}}\overline{\mathcal{A}}\delta_{\boldsymbol{c}}$ (it is the only term that changes in the previous expressions). We first compute the term associated with the j-th frequency using the chain rule

$$\boldsymbol{\nabla}_{\boldsymbol{c}}\left(\overline{\mathcal{A}}\delta_{\boldsymbol{c}}\right)_j = \boldsymbol{\nabla}_{\boldsymbol{c}} f(\boldsymbol{\omega}_j^T\boldsymbol{c} + \xi_j) = \left.\frac{\partial f(t)}{\partial t}\right|_{t=\boldsymbol{\omega}_j^T\boldsymbol{c}+\xi_j} \cdot \boldsymbol{\nabla}_{\boldsymbol{c}}\left(\boldsymbol{\omega}_j^T\boldsymbol{c} + \xi_j\right) = \boldsymbol{\omega}_j \left.\frac{\partial f(t)}{\partial t}\right|_{t=\boldsymbol{\omega}_j^T\boldsymbol{c}+\xi_j} \tag{B.15}$$

where we can express the derivative of the signature function using its Fourier series expression

$$\frac{\partial f(t)}{\partial t} = \frac{\partial}{\partial t}\sum_k F_k e^{\mathrm{i}k\frac{2\pi}{T}t} = \sum_k F_k \frac{\partial}{\partial t} e^{\mathrm{i}k\frac{2\pi}{T}t} = \sum_k F_k \mathrm{i}k\frac{2\pi}{T} e^{\mathrm{i}k\frac{2\pi}{T}t}. \tag{B.16}$$

Putting the derivatives of all elements of the sketch together, the complete Jacobian can be expressed compactly as

$$\boldsymbol{\nabla}_{\boldsymbol{c}}\overline{\mathcal{A}}\delta_{\boldsymbol{c}} = \frac{\mathrm{i}2\pi}{T}\left(\sum_k \left(kF_k e^{\mathrm{i}k\frac{2\pi}{T}(\Omega^T\boldsymbol{c}+\boldsymbol{\xi})}\right)\mathbf{1}_n^T\right) \circ \Omega^T. \tag{B.17}$$

B.3 Gradients for `QCKM`

In the general case, we had

$$\boldsymbol{\nabla}_{\boldsymbol{c}}\overline{\mathcal{A}}\delta_{\boldsymbol{c}} = \frac{\mathrm{i}2\pi}{T}\left(\sum_k \left(kF_k e^{\mathrm{i}k\frac{2\pi}{T}(\Omega^T\boldsymbol{c}+\boldsymbol{\xi})}\right)\mathbf{1}_n^T\right)\circ\Omega^T \tag{B.18}$$

but interestingly, this expression can be simplified when $F_k = \frac{2\mathrm{i}}{k\pi}$ for k odd since the k factor in the sum will simply:

$$\boldsymbol{\nabla}_{\boldsymbol{c}}\overline{\mathcal{A}}\delta_{\boldsymbol{c}} = \frac{\mathrm{i}2\pi}{T}\left(\sum_{k\text{ odd}}\left(\frac{2\mathrm{i}}{\pi}e^{\mathrm{i}k\frac{2\pi}{T}(\Omega^T\boldsymbol{c}+\boldsymbol{\xi})}\right)\mathbf{1}_n^T\right)\circ\Omega^T = \frac{-4}{T}\left(\underbrace{\sum_{k\text{ odd}}\left(e^{\mathrm{i}\frac{2\pi}{T}(\Omega^T\boldsymbol{c}+\boldsymbol{\xi})}\right)^k}_{\boldsymbol{b}}\mathbf{1}_n^T\right)\circ\Omega^T \tag{B.19}$$

which can be simplified since we obtain a geometric series for each component of $\boldsymbol{b}$

$$b_j = \sum_{k\text{ odd}}\left(e^{\mathrm{i}\frac{2\pi}{T}(\boldsymbol{\omega}_j^T\boldsymbol{c}+\xi_j)}\right)^k. \tag{B.20}$$

If we suppose without loss of generality that we are truncating the Fourier series at an odd coefficient index $K_{coef} = 2K'_{coef} + 1$ we can re-write this sum by change of variables $k = 2k' + 1$ since we are only taking the sum on k odd

$$b_j = \left(e^{\mathrm{i}\frac{2\pi}{T}(\boldsymbol{\omega}_j^T\boldsymbol{c}+\xi_j)}\right)\sum_{k'=-K'_{coef}+1}^{K'_{coef}}\left(e^{\mathrm{i}\frac{4\pi}{T}(\boldsymbol{\omega}_j^T\boldsymbol{c}+\xi_j)}\right)^{k'}, \tag{B.21}$$

which can then easily be computed using the formula for a geometric series. In particular, the computation time to compute the gradient does not increase with K_{coef}.

Appendix C

Additionnal proofs

C.1 Proof of Proposition 3

We prove here Proposition 3, that is

$$\frac{1}{m}\|\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P} - \overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{Q}\|^2 \simeq \overline{\gamma_\Lambda}^2(\mathcal{P},\mathcal{Q}) \tag{C.1}$$

with $\overline{\gamma_\Lambda}(\mathcal{P},\mathcal{Q})$ that can be written, when squared, as (equivalence 1)

$$\overline{\gamma_\Lambda}^2(\mathcal{P},\mathcal{Q}) = \sum_{k=-\infty}^{+\infty} |F_k|^2 \gamma_\Lambda^2(D_{2\pi k/T}\mathcal{P}, D_{2\pi k/T}\mathcal{Q}) \tag{C.2}$$

where we define the n-dimensional dilation operator $D_\alpha g(\cdot) := \frac{1}{|\alpha|^n} g(\frac{\cdot}{\alpha})$. It can also be written as (equivalence 2)

$$\overline{\gamma_\Lambda}^2(\mathcal{P},\mathcal{Q}) = \gamma_{\Lambda_f}^2(\mathcal{P},\mathcal{Q}) \tag{C.3}$$

with $\Lambda_f = \sum_k |F_k|^2 D_{2\pi k/T}\Lambda$.

Proof. We first write, by assuming m is large and invoking the law of large numbers

$$\frac{1}{m}\|\overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{P} - \overline{\mathcal{A}}_{\Omega,\boldsymbol{\xi}}\mathcal{Q}\|^2 = \frac{1}{m}\sum_{j=1}^{m} |\overline{\psi}_{\mathcal{P}}(\boldsymbol{\omega}_j;\xi_j) - \overline{\psi}_{\mathcal{Q}}(\boldsymbol{\omega}_j;\xi_j)|^2 \simeq \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim\Lambda,\xi} |\overline{\psi}_{\mathcal{P}}(\boldsymbol{\omega};\xi) - \overline{\psi}_{\mathcal{Q}}(\boldsymbol{\omega};\xi)|^2 =: \overline{\gamma_\Lambda}^2(\mathcal{P},\mathcal{Q}) \tag{C.4}$$

which defines $\overline{\gamma_\Lambda}(\mathcal{P},\mathcal{Q})$.

Equivalence 1:
We develop the expectation (of course, ξ and $\boldsymbol{\omega}$ are independent)

$$\begin{aligned}
\overline{\gamma_\Lambda}^2(\mathcal{P},\mathcal{Q}) &= \frac{1}{T}\int_{\mathbb{R}^n}\int_0^T |\overline{\psi}_{\mathcal{P}}(\boldsymbol{\omega};\xi) - \overline{\psi}_{\mathcal{Q}}(\boldsymbol{\omega};\xi)|^2 \Lambda(\boldsymbol{\omega})\mathrm{d}\xi\mathrm{d}\boldsymbol{\omega} \\
&= \frac{1}{T}\int_{\mathbb{R}^n}\int_0^T \left|\sum_k F_k \psi_{\mathcal{P}}\left(\frac{2\pi}{T}k\boldsymbol{\omega}\right) e^{\mathrm{i}k\xi} - \sum_k F_k \psi_{\mathcal{Q}}\left(\frac{2\pi}{T}k\boldsymbol{\omega}\right) e^{\mathrm{i}k\xi}\right|^2 \Lambda(\boldsymbol{\omega})\mathrm{d}\xi\mathrm{d}\boldsymbol{\omega} \\
&= \int_{\mathbb{R}^n}\sum_k |F_k|^2 \left|\psi_{\mathcal{P}}\left(\frac{2\pi}{T}k\boldsymbol{\omega}\right) - \psi_{\mathcal{Q}}\left(\frac{2\pi}{T}k\boldsymbol{\omega}\right)\right|^2 \Lambda(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega} \\
&= \sum_k |F_k|^2 \int_{\mathbb{R}^n} \left|\psi_{\mathcal{P}}\left(\frac{2\pi}{T}k\boldsymbol{\omega}\right) - \psi_{\mathcal{Q}}\left(\frac{2\pi}{T}k\boldsymbol{\omega}\right)\right|^2 \Lambda(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega}
\end{aligned} \tag{C.5}$$

and note that by change of variables $\boldsymbol{x}' = \alpha\boldsymbol{x}$ we have

$$\psi_{\mathcal{P}}(\alpha\boldsymbol{\omega}) = \int_{\mathbb{R}^n} \mathcal{P}(\boldsymbol{x}) e^{-\mathrm{i}\alpha\boldsymbol{\omega}^T\boldsymbol{x}} \mathrm{d}\boldsymbol{x} = \int_{\mathbb{R}^n} \mathcal{P}\left(\frac{\boldsymbol{x}}{\alpha}\right) e^{-\mathrm{i}\boldsymbol{\omega}^T\boldsymbol{x}'} \frac{1}{|\alpha|^n} \mathrm{d}\boldsymbol{x}' = \psi_{D_\alpha \mathcal{P}}(\boldsymbol{\omega}) \tag{C.6}$$

thus we can finally write

$$\begin{aligned}
\overline{\gamma_\Lambda}^2(\mathcal{P}, \mathcal{Q}) &= \sum_k |F_k|^2 \int_{\mathbb{R}^n} \left| \psi_{\mathcal{P}}\left(\frac{2\pi}{T} k\boldsymbol{\omega}\right) - \psi_{\mathcal{Q}}\left(\frac{2\pi}{T} k\boldsymbol{\omega}\right) \right|^2 \Lambda(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \\
&= \sum_k |F_k|^2 \int_{\mathbb{R}^n} \left| \psi_{D_{2\pi k/T}\mathcal{P}}(\boldsymbol{\omega}) - \psi_{D_{2\pi k/T}\mathcal{Q}}(\boldsymbol{\omega}) \right|^2 \Lambda(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \\
&= \sum_k |F_k|^2 \gamma_\Lambda^2(D_{2\pi k/T}\mathcal{P}, D_{2\pi k/T}\mathcal{Q})
\end{aligned} \tag{C.7}$$

where the last equality comes from (2.43).

Equivalence 2:
By starting with the same steps as equivalence 1 and then by change of variables $\boldsymbol{\omega}' = \frac{2\pi}{T} k\boldsymbol{\omega}$ we obtain

$$\begin{aligned}
\overline{\gamma_\Lambda}^2(\mathcal{P}, \mathcal{Q}) &= \sum_k |F_k|^2 \int_{\mathbb{R}^n} \left| \psi_{\mathcal{P}}\left(\frac{2\pi}{T} k\boldsymbol{\omega}\right) - \psi_{\mathcal{Q}}\left(\frac{2\pi}{T} k\boldsymbol{\omega}\right) \right|^2 \Lambda(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \\
&= \int_{\mathbb{R}^n} \left| \psi_{\mathcal{P}}(\boldsymbol{\omega}') - \psi_{\mathcal{Q}}(\boldsymbol{\omega}') \right|^2 \sum_k |F_k|^2 \frac{1}{|2\pi k/T|^n} \Lambda\left(\frac{\boldsymbol{\omega}'}{2\pi k/T}\right) \mathrm{d}\boldsymbol{\omega}' \\
&= \int_{\mathbb{R}^n} \left| \psi_{\mathcal{P}}(\boldsymbol{\omega}') - \psi_{\mathcal{Q}}(\boldsymbol{\omega}') \right|^2 \Lambda_f(\boldsymbol{\omega}') \mathrm{d}\boldsymbol{\omega}' = \gamma_{\Lambda_f}^2(\mathcal{P}, \mathcal{Q}).
\end{aligned} \tag{C.8}$$

□

C.2 Proof of Proposition 4

We prove here Proposition 4, that is, the kernel as defined in (4.22) associated with a zero-mean function f that has F_k as Fourier series coefficents is translation-invariant,

$$\overline{\kappa}_\Lambda(\boldsymbol{x}_1, \boldsymbol{x}_2) = \overline{K}_\Lambda(\boldsymbol{x}_1 - \boldsymbol{x}_2) \tag{C.9}$$

where $\overline{K}_\Lambda$ can moreover be written as (equivalence 1)

$$\overline{K}_\Lambda(\boldsymbol{x}_1 - \boldsymbol{x}_2) = \sum_{k=-\infty}^{\infty} |F_k|^2 K_\Lambda\left(-k\frac{2\pi}{T}(\boldsymbol{x}_1 - \boldsymbol{x}_2)\right) \tag{C.10}$$

or can also equivalently be written as (equivalence 2)

$$\overline{K}_\Lambda(\boldsymbol{x}_1 - \boldsymbol{x}_2) = K_{\Lambda_f}(\boldsymbol{x}_1 - \boldsymbol{x}_2) \quad \text{with} \quad \Lambda_f(\boldsymbol{\omega}) = \sum_{k=-\infty}^{\infty} |F_k|^2 D_{2\pi k/T}\Lambda(\omega) \tag{C.11}$$

where we introduce the n-dimensional dilation operator $D_\alpha g(\cdot) := \frac{1}{|\alpha|^n} g(\frac{\cdot}{\alpha})$ that dilates a function g by a factor α while preserving its L^1-norm in $\mathbb{R}^n$.

Proof. Translation-invariance of $\overline{\kappa}_\Lambda$ and equivalence 1:
We write, by definition of $\overline{\kappa}_\Lambda$ and developping f as a Fourier series,

$$\begin{aligned}
\overline{\kappa}_\Lambda(\boldsymbol{x}_1, \boldsymbol{x}_2) &= \mathop{\mathbb{E}}_{\boldsymbol{\omega},\xi}\left[f(\boldsymbol{\omega}^T\boldsymbol{x}_1+\xi)f^*(\boldsymbol{\omega}^T\boldsymbol{x}_2+\xi)\right] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{\omega}}\mathop{\mathbb{E}}_{\xi}\sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} F_k F_l^* e^{\mathrm{i}k\frac{2\pi}{T}(\boldsymbol{\omega}^T\boldsymbol{x}_1+\xi)} e^{-\mathrm{i}l\frac{2\pi}{T}(\boldsymbol{\omega}^T\boldsymbol{x}_2+\xi)} \\
&= \mathop{\mathbb{E}}_{\boldsymbol{\omega}}\sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} F_k F_l^* e^{\mathrm{i}\frac{2\pi}{T}\boldsymbol{\omega}^T(k\boldsymbol{x}_1-l\boldsymbol{x}_2)} \underbrace{\mathop{\mathbb{E}}_{\xi} e^{\mathrm{i}\frac{2\pi}{T}(k-l)\xi}}_{\delta[k-l]} \\
&= \mathop{\mathbb{E}}_{\boldsymbol{\omega}}\sum_{k=-\infty}^{\infty}|F_k|^2 e^{\mathrm{i}k\frac{2\pi}{T}\boldsymbol{\omega}^T(\boldsymbol{x}_1-\boldsymbol{x}_2)} = \sum_{k=-\infty}^{\infty}|F_k|^2\,\mathbb{E}_{\boldsymbol{\omega}} e^{\mathrm{i}k\frac{2\pi}{T}\boldsymbol{\omega}^T(\boldsymbol{x}_1-\boldsymbol{x}_2)} \\
&= \sum_{k=-\infty}^{\infty}|F_k|^2 K_\Lambda\left(-k\frac{2\pi}{T}(\boldsymbol{x}_1-\boldsymbol{x}_2)\right).
\end{aligned} \tag{C.12}$$

where in the last step we used that $K_\Lambda(\alpha(\boldsymbol{x}_1-\boldsymbol{x}_2)) := \mathbb{E}_{\boldsymbol{\omega}\sim\Lambda}\left[e^{-\mathrm{i}\alpha\boldsymbol{\omega}^T(\boldsymbol{x}_1-\boldsymbol{x}_2)}\right]$ to conclude the proof of the first equivalence.

Equivalence 2:
Starting from the second to last line of the proof of equivalence 1 we develop the expectation, then make a change of variables $\boldsymbol{\omega}' = \frac{2\pi}{T}k\boldsymbol{\omega}$. We assumed that the mean of the signature function is thus zero (which does not remove any information from the sketch) thus $F_k = 0$ so $k = 0$ does not pose a problem in the change of variables, and we have to keep in mind that for the terms $k < 0$ the integration limits are switched (hence the absolute value). We obtain

$$\begin{aligned}
\overline{\kappa}_\Lambda(\boldsymbol{x}_1, \boldsymbol{x}_2) &= \sum_{k=-\infty}^{\infty}|F_k|^2 \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim\Lambda} e^{\mathrm{i}k\frac{2\pi}{T}\boldsymbol{\omega}^T(\boldsymbol{x}_1-\boldsymbol{x}_2)} \\
&= \sum_{k=-\infty}^{\infty}|F_k|^2 \int_{\mathbb{R}^n}\Lambda(\boldsymbol{\omega}) e^{\mathrm{i}k\frac{2\pi}{T}\boldsymbol{\omega}^T(\boldsymbol{x}_1-\boldsymbol{x}_2)}\mathrm{d}\boldsymbol{\omega} \\
&= \sum_{k=-\infty}^{\infty}|F_k|^2 \int_{\mathbb{R}^n}\Lambda\left(\frac{T}{2\pi k}\boldsymbol{\omega}'\right) e^{\mathrm{i}\boldsymbol{\omega}'^T(\boldsymbol{x}_1-\boldsymbol{x}_2)}\left|\frac{T}{2\pi k}\right|^n \mathrm{d}\boldsymbol{\omega}' \\
&= \int_{\mathbb{R}^n}\sum_{k=-\infty}^{\infty}|F_k|^2 \underbrace{\left|\frac{T}{2\pi k}\right|^n \Lambda\left(\frac{T}{2\pi k}\boldsymbol{\omega}'\right)}_{D_{2\pi k/T}\Lambda(\boldsymbol{\omega}')} e^{\mathrm{i}\boldsymbol{\omega}'^T(\boldsymbol{x}_1-\boldsymbol{x}_2)}\mathrm{d}\boldsymbol{\omega}' \\
&= \int_{\mathbb{R}^n}\Lambda_f(\boldsymbol{\omega}') e^{\mathrm{i}\boldsymbol{\omega}'^T(\boldsymbol{x}_1-\boldsymbol{x}_2)}\mathrm{d}\boldsymbol{\omega}' \\
&= \mathop{\mathbb{E}}_{\boldsymbol{\omega}'\sim\Lambda_f}\left[e^{-\mathrm{i}\boldsymbol{\omega}'^T(\boldsymbol{x}_1-\boldsymbol{x}_2)}\right] = K_{\Lambda_f}(\boldsymbol{x}_1-\boldsymbol{x}_2)
\end{aligned} \tag{C.13}$$

with Λ_f defined as in (C.11).

□

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**