UCLouvain

École polytechnique de Louvain

Towards an improved nonconvex optimization step in a compressive clustering algorithm

Author: **Mathys Gouverneur**
Supervisors: **Laurent Jacques, Geovani Nunes Grapiglia**
Reader: **Vincent Schellekens**
Academic year 2024–2025
Master [120] in Mathematical Engineering

Abstract

Machine learning, the process of learning from data, has revolutionized many fields but faces computational challenges as datasets expand exponentially. *Compressive learning* is an emerging approach to address this issue by first compressing datasets into low-dimensional *sketch* vectors, before performing learning tasks.
One key application of compressive learning is *compressive clustering*, which aims to perform a clustering task, i.e., group together 'similar' data samples, using only the sketch of a dataset. To tackle the compressive clustering problem, the standard approach is the heuristic algorithm CL-OMPR. However, CL-OMPR can be difficult to tune and may fail in certain cases. In this work, we carefully analyze CL-OMPR to overcome its limitations. We identify that these issues arise from optimization challenges related to the structure of a correlation function used in key steps of the algorithm. We propose methods to improve the nonconvex optimization of this function, enhancing the robustness of CL-OMPR and simplifying its tuning.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to professors Laurent Jacques and Geovani Nunes Grapiglia. This work was made possible thanks to the numerous meetings, the time they dedicated, and their invaluable guidance.

I sincerely thank Rémi Delogne for the insightful discussions and valuable advice he shared.

A special mention goes to Vincent Schellekens for his outstanding toolbox [1], without which this work would not have been possible.

Writing my master's thesis required more than academic assistance, and I wish to thank my parents and my brother for their support and encouragement throughout my study. To my mother, in particular, I am forever grateful—merci pour tout.

Contents

List of Notations

$\boldsymbol{x}, \boldsymbol{y}, \ldots$	Vector/generic signal
$\|\cdot\|_2$	Euclidean distance between two points
$\langle \cdot, \cdot \rangle$	Hermitian inner product
j	Imaginary unit, defined as $j = \sqrt{-1}$
$\mathbf{I}_d$	Identity matrix of size $d \times d$
$\lvert\mathcal{S}\rvert$	Cardinality of a set, representing the number of elements in the set, denoted as $\lvert\mathcal{S}\rvert$ for a set $\mathcal{S}$
$\sim \mathcal{D}$	Indicates that a random variable, vector, or function is distributed according to the distribution $\mathcal{D}$
i.i.d.	Independent and identically distributed, a key assumption in probability and statistics
$\mathcal{U}(\mathcal{A})$	Uniform distribution over the set $\mathcal{A}$
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate Gaussian distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$
$\delta_{\boldsymbol{c}}$	Dirac distribution at $\boldsymbol{c} \in \mathbb{R}^d$, assigning all mass to the point $\boldsymbol{c}$
$\Gamma(x)$	Gamma function, defined as $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt$ for $x > 0$
M_+^1	The set of all probability measures over a given space
$\mathbb{E}[\cdot]$	Expected value of a random variable

Introduction

Thanks to rapid technological advancements, we now generate and store vast amounts of data in countless forms. From global phenomena like climate metrics and financial markets to intricate records of our personal lives—our locations, online activities, or shopping preferences—this explosion of data opens the door to solving complex problems with impressive accuracy.
The growing size of datasets also brings important computational challenges. It is crucial to find ways to get useful information from large datasets while keeping computational resources under control.
To address these challenges, one approach is to create a compact representation that captures the essential structure of the dataset by summarizing its key information. Such representations make it possible to extract valuable insights efficiently, reducing both memory usage and the need for direct access to the full dataset. This is the idea behind *compressive learning*, where the dataset is first compressed into a *sketch* before learning.

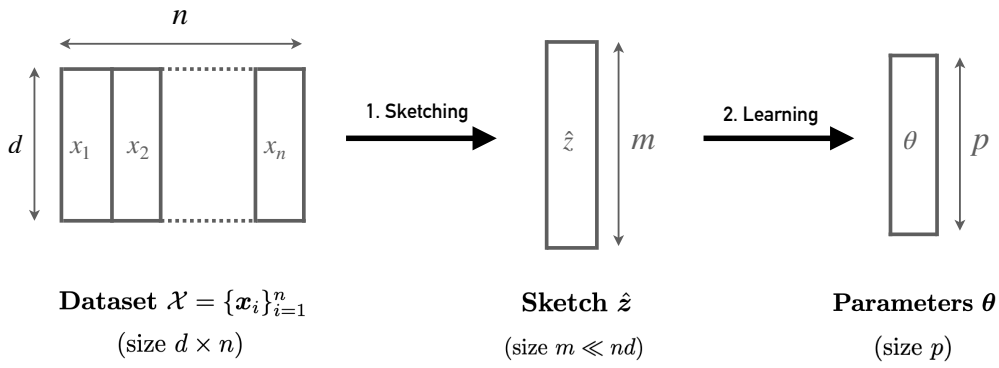


Figure 1: A potentially massive (large n) dataset is first compressed into a low dimensional vector called *sketch* before a learning task is applied.

A well-known learning task is clustering, which aims to group 'similar' data samples together. The goal is to organize the dataset into clusters, and the parameters to be determined are the centers of these clusters, i.e., the *centroids*, which represent the groups.
The compressive approach of this problem is called *compressive clustering*, and aims at retrieving those parameters based on a sketch of the data.

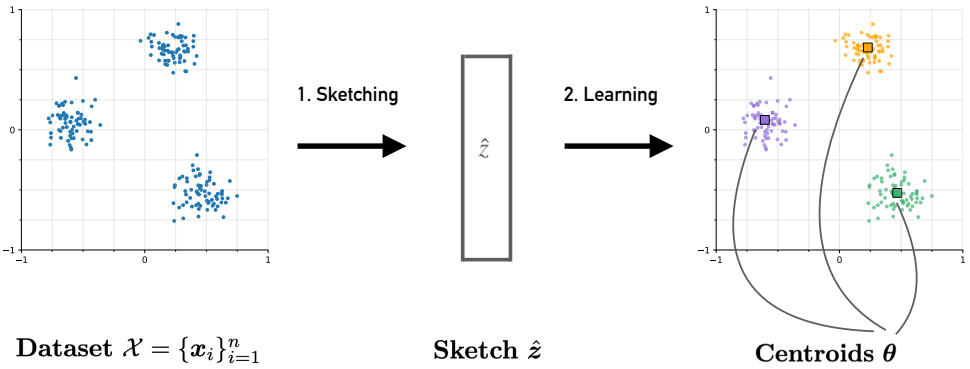


Figure 2: An overview of compressive clustering

The CL-OMPR algorithm [2] was proposed to solve the compressive clustering problem and recover the centroids from the sketch. However, this recovery can sometimes fail due to the presence of nonconvex optimization steps. In our context, these steps correspond to the maximization of a nonconvex function, which may lead to convergence towards non-global maximizers and result in suboptimal solutions. An example of such a nonconvex function is illustrated in Figure 3.
In this work, we will investigate the potential causes of these failures in CL-OMPR and propose an optimization method to address them.

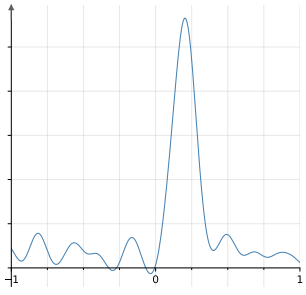


Figure 3: An example of a nonconvex function with many non-global maximizers

The CL-OMPR algorithm can be challenging to tune. This work, through an improvement of a nonconvex optimization step within the CL-OMPR algorithm, aims to demonstrate an expansion of its operational range.

This report is organized as follows: Chapter 1 introduces the compressive learning framework. Chapter 2 delves into the CL-OMPR algorithm, identifying its failures and potential areas for improvement. Building on this analysis, chapter 3 presents our proposed improvement to the optimization step of the algorithm and evaluates its performance. Finally, we conclude the work and discusses potential improvements and future perspectives.

Chapter 1

Foundations of Compressive Learning

This chapter aims to provide the reader with a general understanding of Compressive Learning. Before delving into Compressive Learning itself in Section 1.2, we first introduce key machine learning concepts in Section 1.1 that form the foundation for this framework.

1.1 Machine Learning

In machine learning, we are given some *data*. We want to turn this data into useful *mathematical models*. We will first define more precisely the terms *data*, *mathematical models*, as well as the *learning process*.

1.1.1 Data

Data refers to the collection of information, often organized as a dataset, that serves as the foundation for building machine learning models. A dataset typically consists of examples or instances, where each instance is described by a set of attributes (also called features).

Mathematically, a dataset $\mathcal{X}$ is a collection of n instances $\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}$, which belong to the same domain Σ (a common case is $\Sigma = \mathbb{R}^d$, with d representing the number of features), i.e.,

$$\mathcal{X} := \{\boldsymbol{x}_i \in \mathbb{R}^d \mid i = 1, \ldots, n\} \tag{1.1}$$

A key idea in machine learning is that the instances in our dataset are assumed to be drawn from an underlying probability distribution $\mathcal{D}$. This distribution

represents the statistical properties of the data generation process, and it provides a framework for understanding how the data is structured.

Mathematically, we assume each instance $\boldsymbol{x}_i$ in the dataset $\mathcal{X}$ to be an independent and identically distributed (i.i.d.) realization (observation) of the distribution $\mathcal{D}$, i.e.,

$$\boldsymbol{x_1}, \ldots, \boldsymbol{x}_n \sim_{\text{i.i.d}} \mathcal{D} \tag{1.2}$$

By understanding and capturing the statistical properties of $\mathcal{D}$, we aim to build mathematical models that generalize well to new, unseen data that is also drawn from the same distribution.

1.1.2 Mathematical Models

In machine learning, the objective is to identify a good model that can effectively solve a specific task. These tasks can vary widely, leading to different categories of machine learning problems.
For instance, in supervised learning, the model's purpose might be to predict an output label given an input (e.g., classifying emails as spam or not). In contrast, in unsupervised learning, the model might aim to uncover hidden structures within the data (e.g., clustering similar documents together).
Despite the variety of tasks, most machine learning models share a common feature: they are *parametrized*. This means that each model is defined by a set of parameters, represented by a vector $\boldsymbol{\theta}$, which belongs to a parameter space Θ. Often, $\Theta \subseteq \mathbb{R}^p$, meaning the model is described by p real-valued parameters.

The goal of machine learning is to find the parameter vector $\boldsymbol{\theta}$ that best fits the dataset $\mathcal{X}$. How these parameters are learned will be explained in the next subsection.

1.1.3 Learning

In machine learning, the process of learning revolves around finding the optimal model parameters that best fit the data. A common and unifying approach to this process is to define a loss function $l : \Sigma \times \Theta \to \mathbb{R}$, which quantifies how well a given model parameter vector $\boldsymbol{\theta}$ performs on any data point $\boldsymbol{x} \in \Sigma$. The loss function assigns a numerical value to the pair $(\boldsymbol{x}, \boldsymbol{\theta})$, with a lower loss indicating a better fit. The ultimate goal in this learning framework is to find the parameters $\boldsymbol{\theta}^* \in \Theta$ that minimize the expected loss, also known as the risk.

Mathematically, this involves minimizing the risk objective $R(\boldsymbol{\theta}; \mathcal{D}) := \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[l(\boldsymbol{x}, \boldsymbol{\theta})]$,

where the expectation is taken over the data distribution $\mathcal{D}$:

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}\in\Theta} R(\boldsymbol{\theta}; \mathcal{D}) = \arg\min_{\boldsymbol{\theta}\in\Theta} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[l(\boldsymbol{x}, \boldsymbol{\theta})]. \tag{1.3}$$

This equation represents the ideal objective, where we seek the parameters that would minimize the average loss over all possible data points from the true data distribution $\mathcal{D}$.

However, in practice, the true distribution $\mathcal{D}$ is unknown. Instead, we have access to a dataset $\mathcal{X} = \{\boldsymbol{x_i}\}_{i=1}^n$ consisting of n i.i.d. observations of $\mathcal{D}$. We thus approximate the true risk by the empirical risk, by replacing the true distribution with the empirical distribution $\hat{\mathcal{D}}_{\mathcal{X}}$ derived from the dataset. The empirical distribution is defined as:

$$\hat{\mathcal{D}}_{\mathcal{X}} := \frac{1}{n}\sum_{i=1}^{n} \delta_{\boldsymbol{x_i}}, \tag{1.4}$$

where $\delta_{\boldsymbol{x_i}}$ is the Dirac delta function centered at $\boldsymbol{x_i}$.

We now seek to find the best model parameter vector $\tilde{\boldsymbol{\theta}}$ that minimizes the empirical risk :

$$\tilde{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}\in\Theta} R(\boldsymbol{\theta}; \hat{\mathcal{D}}_{\mathcal{X}}) = \arg\min_{\boldsymbol{\theta}\in\Theta} \frac{1}{n}\sum_{\boldsymbol{x_i}\in\mathcal{X}} l(\boldsymbol{x_i}, \boldsymbol{\theta}), \tag{1.5}$$

which can be interpreted as finding $\tilde{\boldsymbol{\theta}}$ that minimizes the dataset's average loss function over $\mathcal{X}$.

This formulation allows us to practically determine the best-fitting parameters using the observed data, providing a foundation for most machine learning algorithms.

1.1.4 A concrete example : K-means clustering

We will now use the theoretical foundations we have just developed to define the specific machine learning problem that we will focus on exclusively in this report: K-means clustering.

Data

Recall that in our framework (1.1.1), data is represented by a dataset $\mathcal{X} = \{\boldsymbol{x_i}\}_{i=1}^n$, where each $\boldsymbol{x_i}$ is a d-dimensional vector in $\mathbb{R}^d$. The dataset $\mathcal{X}$ consists of n instances drawn from an underlying distribution $\mathcal{D}$.

Mathematical Model

The objective of clustering is to group together 'similar' data samples from $\mathcal{X}$. In the K-means approach to clustering, we aim to find a set of K centroids, $\boldsymbol{\theta} := \mathcal{C} = \{\boldsymbol{c_k} \in \mathbb{R}^d\}_{k=1}^K$, that are 'representative' of clusters in the data. To do this, each data sample $\boldsymbol{x_i}$ is associated to its closest (according to a metric) centroid. The metric in this problem is the squared Euclidean distance $||\cdot||_2^2$.

The parameters $\boldsymbol{\theta}$ for this task are thus the set of centroids $\mathcal{C} = \{\boldsymbol{c_k} \in \mathbb{R}^d\}_{k=1}^K$. The parameters are basically the positions of the K centroids.

Learning

Recall from 1.1.3, that, in order to find the best set of centroids, we need to define the loss function l that quantifies how well a given model parameters $\boldsymbol{\theta}$ perform on any data point $\boldsymbol{x} \in \Sigma$.

In our case, the loss function is the squared distance from $\boldsymbol{x}$ to its nearest centroid $\boldsymbol{c} \in \mathcal{C}$:

$$l(\boldsymbol{x}, \boldsymbol{\theta}) = \min_{1 \leq k \leq K} ||\boldsymbol{x} - \boldsymbol{c_k}||_2^2 \tag{1.6}$$

The best set of centroids $\tilde{\mathcal{C}}$ is the one that minimizes the empirical risk, i.e.,

$$\tilde{\mathcal{C}} \in \arg\min_{\mathcal{C}} \frac{1}{n} \sum_{\boldsymbol{x}_i \in \mathcal{X}} l(\boldsymbol{x}_i, \boldsymbol{\theta}) = \arg\min_{\mathcal{C}} \frac{1}{n} \sum_{\boldsymbol{x}_i \in \mathcal{X}} \min_{1 \leq k \leq K} ||\boldsymbol{x}_i - \boldsymbol{c}_k||_2^2.$$

This quantity we aim to minimize is called the Mean Squares Errors (MSE), and is thus defined by

$$\mathrm{MSE}(\mathcal{C}; \mathcal{X}) := \frac{1}{n} \sum_{\boldsymbol{x_i} \in \mathcal{X}} \min_{1 \leq k \leq K} ||\boldsymbol{x}_i - \boldsymbol{c}_k||_2^2 \tag{1.7}$$

Finding the global minimum of (1.7) is NP-hard [3], but there are heuristics to approximate a solution, notably the popular Lloyd's algorithm [4, 5]. However, when the number of data samples n is very large, Lloyd's algorithm becomes computationally expensive. This limitation motivates the use of compressive learning for the K-means clustering problem (see Section 1.2.4).

1.2 Introduction to Compressive Learning

Now that we have a better understanding of machine learning, we turn our attention to compressive learning. The objective remains the same: to find parameterized mathematical models, but this time based on a sketch of the data. In this section,

we will focus on the two main components of compressive learning: sketching, discussed in Section 1.2.1, and learning, covered in Section 1.2.3. Between these, key concepts from *compressive sensing* [6, 7], presented in Section 1.2.2, will be essential for a proper understanding.

1.2.1 Sketching

As mentioned in Section 1.1.1, we assume each instance $\boldsymbol{x}_i \in \mathcal{X}$ is drawn i.i.d. from a distribution $\mathcal{D}$. It follows that a mathematical model (see 1.1.2) would identify its optimal parameters with respect to this underlying distribution $\mathcal{D}$.
Ideally, we would like to capture the essence of this distribution $\mathcal{D}$ in a *sketch*—a vector of dimension m.
To achieve this, we introduce a **sketching operator**, a mathematical tool that projects any probability distribution $\mathcal{P}$ into a lower-dimensional space.
This sketching operator is formally defined as follows:

Definition 1.1 (Sketching operator)**.** Given a feature map $\Phi : \Sigma \to \mathbb{C}^m$, the associated sketching operator $\mathcal{A}_\Phi : M_+^1(\Sigma) \to \mathbb{C}^m$ is defined by the expectation of the feature map $\Phi(\boldsymbol{x})$ with respect to the probability distribution P:

$$\mathcal{A}_\Phi(P) := \mathbb{E}_{\boldsymbol{x}\sim P}[\Phi(\boldsymbol{x})] = \int_\Sigma \Phi(\boldsymbol{x})\, dP(\boldsymbol{x}) \tag{1.8}$$

However, in practice, the underlying distribution $\mathcal{D}$ is unknown, but we have access to a dataset $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ with $\boldsymbol{x}_i \sim \mathcal{D}$. Instead of the unknown true distribution $\mathcal{D}$, we can practically use the empirical distribution $\hat{\mathcal{D}_\mathcal{X}} = \frac{1}{n}\sum_{i=1}^n \delta_{\boldsymbol{x}_i}$. We can compute the empirical sketch of the dataset $\mathcal{X}$, associated with the feature map $\Phi : \Sigma \to \mathbb{C}^m$, denoted $\hat{z}_{\Phi,\mathcal{X}}$, by passing the empirical distribution $\hat{\mathcal{D}_\mathcal{X}}$ through the sketching operator defined above (1.1). We get :

$$\begin{aligned}
\hat{z}_{\Phi,\mathcal{X}} := \mathcal{A}_\Phi(\hat{\mathcal{D}_\mathcal{X}}) &= \int_\Sigma \Phi(\boldsymbol{x})\, d\hat{\mathcal{D}_\mathcal{X}}(x) \\
&= \int_\Sigma \Phi(\boldsymbol{x})\, d\Big(\frac{1}{n}\sum_{i=1}^n \delta_{\boldsymbol{x}_i}\Big)(\boldsymbol{x}) \\
&= \frac{1}{n}\sum_{i=1}^n \Phi(\boldsymbol{x}_i)
\end{aligned} \tag{1.9}$$

1.2.2 Overview of Compressive Sensing

In the field of *compressive sensing* [6, 7], a signal of interest, modeled as a vector $\boldsymbol{x} \in \mathbb{R}^d$, is measured through a linear operator $\boldsymbol{A} \in \mathbb{R}^{m\times d}$, i.e.,

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e} \tag{1.10}$$

where $\boldsymbol{e} \in \mathbb{R}^m$ is additive noise.
The goal is to recover an estimate $\hat{\boldsymbol{x}}$ of $\boldsymbol{x}$ from the noisy and dimension-reduced measurement $\boldsymbol{y}$.
To solve this *inverse problem*, one might consider the following method: :

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} ||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}||_2^2 \tag{1.11}$$

However, this problem is theoretically ill-posed, even in the absence of additive noise $\boldsymbol{e}$.

Sparsity

However, it is possible to regularize this problem by making a sparsity assumption on $\boldsymbol{x}$. If we assume that $\boldsymbol{x}$ is *sparse*, i.e., it has only a few non-zero entries, then we can regularize the problem (1.11) :

$$\hat{x} = \arg\min_{\boldsymbol{x}\in\mathcal{K}} ||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}||_2^2 \tag{1.12}$$

with $\mathcal{K}$ the set of K*-sparse* signals, i.e. signals with at most K non-zero entries. Solving (1.12) returns the K-sparse vector that minimizes the measurement error. Guarantees for robust recovery exist for (1.12) when the measurement operator $\boldsymbol{A}$ satisfies the restricted isometry property (RIP) [6, 7].
One way to ensure that $\boldsymbol{A}$ satisfies the RIP is to construct it randomly.

Solving the regularized problem

Solving (1.12) is NP-Complete [6] and therefore cannot be solved exactly in practice. Instead, two common approaches are typically used: *convex relaxation* or *greedy algorithms*.
Greedy algorithms are local search methods that iteratively improve a candidate solution without considering the global optimization landscape.
A well-known family of greedy algorithms for sparse recovery problems like (1.12) is Orthogonal Matching Pursuit (OMP) [8] .

1.2.3 Learning

We have just introduced the fundamental concepts of compressive sensing (CS).
To summarize: a **random** linear operator $\boldsymbol{A}$ is designed, and we obtain compressed and noisy measurements of a signal of interest $\boldsymbol{x}$, expressed as $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}$. An approximate solution $\hat{\boldsymbol{x}} \approx \boldsymbol{x}$ can be recovered by assuming **sparsity** on $\boldsymbol{x}$.

We will build on this framework to introduce and understand compressive learning.

Recall from Section 1.2.1, that the sketch $\hat{\boldsymbol{z}}$ of a dataset $\mathcal{X} = \{x_i\}_{i=1}^n$, with $x_i \in \mathbb{R}^d$ is :

$$\hat{\boldsymbol{z}} = \mathcal{A}(\hat{\mathcal{D}}) = \mathcal{A}(\mathcal{D}) + \boldsymbol{e} \tag{1.13}$$

where $\mathcal{D}$ is the distribution underlying the dataset, $\hat{\mathcal{D}}$ is its empirical distribution, and $\mathcal{A}$ is a sketching operator (defined in Section 1.1).

Similarly to CS, the goal of CL is to recover an estimate $\tilde{\mathcal{D}}$ of the true distribution $\mathcal{D}$ (or of some distributional parameters $\boldsymbol{\theta}$, depending on the specific task), from the noisy and dimension-reduced[1] measurement $\hat{\boldsymbol{z}}$.

Sparsity

As we have already seen in Section 1.2.2, the problem $\tilde{\mathcal{D}} = \arg\min_{\mathcal{D} \in \mathcal{M}(\mathbb{R}^d)} ||\hat{z} - \mathcal{A}(\mathcal{D})||_2^2$ is ill posed, and we need to make sparsity assumptions (modelling assumptions) to regularize it.
We thus assume that $\mathcal{D}$ can be modeled by a 'family' of simple distributions. More formally, we consider $\{\mathcal{P}_{\boldsymbol{\theta}} \mid \boldsymbol{\theta} \in \Theta\}$, a basic set of distributions parameterized by $\boldsymbol{\theta}$. Consequently, we aim to find the optimal parameters $\hat{\boldsymbol{\theta}}$ that minimize the measurement error. This leads to the following general regularized problem for compressive learning:

$$\hat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta} \in \Theta} \mathcal{C}(\boldsymbol{\theta}; \hat{\boldsymbol{z}}), \quad \text{where} \quad \mathcal{C}(\boldsymbol{\theta}; \hat{\boldsymbol{z}}) = ||\hat{z} - \mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}})||_2^2 \tag{1.14}$$

1.2.4 Compressive K-means

A common machine learning task is mixture model fitting problem. In this problem, data is assumed to be generated from a mix of different probability distributions. For example, in a Gaussian Mixture Model (GMM), each component is a Gaussian distribution. Each of these distributions is called a component. Each component has an associated weight, which represents the proportion of the entire data set that is generated by that component. Each component has parameters that define its shape. In the case of a GMM, these parameters would include the mean and variance of each Gaussian component. The main problem in mixture model fitting is to estimate the parameters of the mixture model, both the component parameters and the weights.

[1] In mathematical terms, probability distribution belong to the infinite dimensional vector space of so-called *finite* measures [9, 10].

Learning such models with the compressive learning approach suggests the following *sparsity assumption* :

$$\mathcal{P}_{\boldsymbol{\theta}} = \sum_{k=1}^{K} \alpha_k \pi_k, \tag{1.15}$$

with the parameter vector $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \{\pi_k\}_{k=1}^K)$.

This setting encompasses both *compressive Gaussian mixture modeling* [2], where $\pi_k = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, and *compressive K-means* [11], where $\pi_k = \delta_{\boldsymbol{c}_k}$, which can be viewed as a weighted mixture of Dirac deltas located at the centroids $\boldsymbol{c}_k$ positions. In this framework, the compressive K-means problem derived from (1.14) is:

$$\hat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}\in\Theta} ||\hat{z} - \mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}})||_2^2 = \arg\min_{\boldsymbol{\theta}\in\Theta} ||\hat{z} - \sum_{k=1}^{K} \alpha_k \mathcal{A}(\delta_{\boldsymbol{c}_k})||_2^2 \tag{1.16}$$

Similar to the (1.12) in compressive sensing, solving the problem (1.16) is challenging due to its nonconvex nature. In this work, we will explore a heuristic proposed in [2] for this problem: the CL-OMP algorithm, which will be discussed in detail in Chapter 2.

1.2.5 Practical design of the sketching operator

In Section 1.2.1, we introduced the sketch $\hat{\boldsymbol{z}}$ of a dataset $\mathcal{X}$, where samples reside in $\mathbb{R}^d$, as

$$\hat{z} = \mathcal{A}(\hat{\mathcal{D}_{\mathcal{X}}}) = \frac{1}{n}\sum_{i=1}^{n} \Phi(x_i) \tag{1.17}$$

with Φ, a feature map from $\mathbb{R}^d \to \mathbb{C}^m$.

A common choice, similar to the CS approach, is to design the measurement operator $\mathcal{A}$ using **randomness**. To do this, we need to choose an appropriate randomized feature map Φ. A common choice is to take one based on Random Fourier Features (RFF) [12]:

$$\Phi_{\text{RFF}}(\boldsymbol{x}) = [\exp(j\boldsymbol{w}_l^T\boldsymbol{x})]_{l=1}^m, \tag{1.18}$$

where $j = \sqrt{-1}$ and the frequencies $\boldsymbol{w}_1, ..., \boldsymbol{w}_m$ are random vectors in $\mathbb{R}^d$ drawn i.i.d from a probability distribution Λ, i.e., $\boldsymbol{w_l} \sim_{\text{i.i.d}} \Lambda$.
Note that the sketching operator $\mathcal{A}$ defined in (1.1), associated with the feature map Φ_{RFF}, now becomes :

$$\begin{aligned}\mathcal{A}(\mathcal{P}) &= \mathbb{E}_{\boldsymbol{x}\sim P}\Big[[\exp(j\boldsymbol{w}_l^T\boldsymbol{x})]_{l=1}^m\Big] \\ &= \Big[\mathbb{E}_{\boldsymbol{x}\sim P}[\exp(j\boldsymbol{w}_l^T\boldsymbol{x})]\Big]_{l=1}^m \\ &= [\varphi_{\mathcal{P}}(\boldsymbol{w_l})]_{l=1}^m,\end{aligned} \tag{1.19}$$

where $\varphi(\boldsymbol{w}) := \mathbb{E}_{\boldsymbol{x}\sim P}[\exp(j\boldsymbol{w}_l^T\boldsymbol{x})]$ is called the *characteristic function* of $\mathcal{P}$. From a statistical point of view, the sketching operator $\mathcal{A}$, equipped with Φ_{RFF} consists in taking m samples of the characteristic function of $\mathcal{P}$.

The only missing piece to fully define a practical sketching operator $\mathcal{A}$ is the distribution Λ. A common choice is to take $\Lambda = \mathcal{N}(0, \sigma^{-2}\mathbf{I}_d)$. In this case, σ, called the bandwidth, can be interpreted as the 'resolution' at which the distribution is captured (this will be discussed further in Chapter 2 and Chapter 3).
Other choices for the feature map Φ and the distribution Λ have been proposed [2]. However, in this work, we will focus exclusively on the choices presented above.

Chapter 2

A Greedy algorithm for CKM

Context. In the first chapter, we briefly introduced Compressive Learning. We defined the sketching operator $\mathcal{A}$, which led to the sketch of a dataset $\mathcal{X}$: $\hat{z}$ ($\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^d \mid i = 1, \dots, n\}$). We also defined a machine learning task: K-means clustering, and explored its compressive version: compressive K-means (CKM). Mathematically, CKM consists of solving the following optimization problem:

$$\hat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}} \mathcal{C}_{\text{CKM}}(\boldsymbol{\theta}; \hat{z}), \quad \text{where} \quad \mathcal{C}_{\text{CKM}}(\boldsymbol{\theta}; \hat{\boldsymbol{z}}) = ||\hat{z} - \sum_{k=1}^{K} \alpha_k \mathcal{A}(\delta_{\boldsymbol{c}_k})||_2^2 \tag{2.1}$$

The model parameters are $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \mathcal{C})$, where $\boldsymbol{\alpha} = \{\alpha_k\}_{k=1}^K$ and $\mathcal{C} = \{\boldsymbol{c}_k\}_{k=1}^K$. Therefore, we aim to determine the locations of the centroids $\mathcal{C}$ and their associated weights $\boldsymbol{\alpha}$.

In practice, a heuristic algorithm is used to solve it approximately, i.e., a decoder Δ that takes the sketch of the dataset and returns some parameters, i.e., $\Delta : \hat{z} \to \boldsymbol{\theta}_\Delta$. For the CKM problem, a heuristic algorithm has been proposed in [2] : the CL-OMP algorithm.

This chapter. In this chapter, we will explore and analyze the CL-OMP algorithm for the compressive K-means problem. The layout of this chapter is as follows:

- In Section 2.1, we will describe the method and its variants.
- In Section 2.2, we will analyze the different situations where CL-OMP may fail and identify directions for improvement.

2.1 Compressive Learning-OMP (CL-OMP)

The CL-OMP algorithm builds upon the Orthogonal Matching Pursuit (OMP) algorithm [8], a well-known greedy method for solving regularized problems in compressive sensing (see Section 1.2.2). The core idea is to begin with an empty support and iteratively refine the solution. At each iteration t, the algorithm greedily selects the most promising index and adds it to the support of the solution.

CL-OMP adapts this approach to the context of compressive K-means. Specifically, it starts with an empty set of centroids and iteratively adds new ones by approximately maximizing a nonconvex criterion. The method alternates between expanding the set of centroids and refining the solution through local minimization, initialized at the current set of centroids, to further reduce the cost function.

CL-OMP for CKM is described in Algorithm 1 :

Algorithm 1: CL-OMP for CKM

Data: Sketch $\hat{\boldsymbol{z}}$, sketching operator $\mathcal{A}$, number of centroids K, Ω
Result: Centroids C, weights $\boldsymbol{\alpha}$
Initialize residual $\boldsymbol{r} \leftarrow \hat{\boldsymbol{z}}$, centroids $\boldsymbol{C} \leftarrow \emptyset$;
for $t \leftarrow 1$ ***to*** K **do**
  Step 1: Find a new centroid;
    $\boldsymbol{c} \leftarrow \max_{\boldsymbol{c}\in\Omega} \left(\Re\left\langle \frac{\mathcal{A}(\delta_{\boldsymbol{c}})}{\|\mathcal{A}(\delta_{\boldsymbol{c}})\|}, \frac{\boldsymbol{r}}{\|\boldsymbol{r}\|}\right\rangle, \text{init} = \text{rand}\right)$;
  end
  Step 2: Expand support;
    $C \leftarrow C \cup \{\boldsymbol{c}\}$;
  end
  Step 3: Project to find weights $\boldsymbol{\alpha}$;
    $\boldsymbol{\alpha} \leftarrow \arg\min_{\boldsymbol{\alpha}\geq 0} \left\|\hat{\boldsymbol{z}} - \sum_{k=1}^{|\boldsymbol{C}|} \alpha_k \mathcal{A}(\delta_{\boldsymbol{c}_k})\right\|$;
  end
  Step 4: Global gradient descent;
    $C, \boldsymbol{\alpha} \leftarrow \min_{\boldsymbol{C},\boldsymbol{\alpha}} \left\|\hat{\boldsymbol{z}} - \sum_{k=1}^{|\boldsymbol{C}|} \alpha_k \mathcal{A}(\delta_{\boldsymbol{c}_k})\right\|$, $\boldsymbol{c}_k \in \Omega$, for all k's;
  end
  Update residual: $\boldsymbol{r} \leftarrow \hat{\boldsymbol{z}} - \sum_{k=1}^{|\boldsymbol{C}|} \alpha_k \mathcal{A}(\delta_{\boldsymbol{c}_k})$;
end

Let us go trough each step of the algorithm.
Step1. As mentioned, the algorithm, at each iteration, greedily selects a new centroid by maximizing a nonconvex criterion.
This criterion is the correlation between the residual $\boldsymbol{r}$ ($\boldsymbol{r} = \hat{\boldsymbol{z}}$ at the first iteration) and the sketch of a localized dirac $\mathcal{A}(\delta_c)$, i.e.,

$$\max_{\boldsymbol{c}\in\Omega} \left(\Re \left\langle \frac{\mathcal{A}(\delta_{\boldsymbol{c}})}{\|\mathcal{A}(\delta_{\boldsymbol{c}})\|}, \frac{\boldsymbol{r}}{||\boldsymbol{r}||} \right\rangle\right),$$

where Ω is the domain in which lies the data. More precisely, during the computation of the sketch $\hat{\boldsymbol{z}}$, bounds $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^d$ are computed during the sketching such that all data are comprised in these bounds : $\boldsymbol{l} \leq \boldsymbol{x}_i \leq \boldsymbol{u}, \quad \forall \boldsymbol{x}_i \in \mathcal{X}$, so $\Omega = \prod_{i=1}^{d}[l_i, u_i]$ (with $\boldsymbol{a} \leq \boldsymbol{b}$ the element-by-element comparison of vectors in $\mathbb{R}^d$).
The maximization is carried via an optimization scheme (e.g. quasi Newton optimization schemes), and the initialisation $\boldsymbol{c}_0$ is random, i.e., the sampling is performed uniformly, such that $\boldsymbol{c}_0 \sim \mathcal{U}(\Omega)$.
The criterion is highly dependent on the design of the sketching operator $\mathcal{A}$, more precisely on the number of considered measurements m and the bandwidth σ (recall Section 1.2.5). See Figure 2.1.

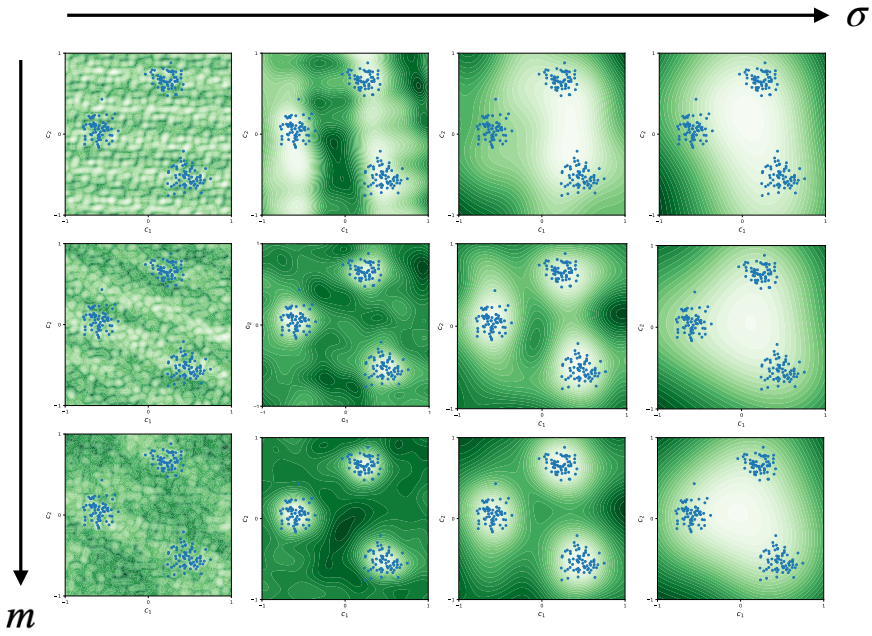


Figure 2.1: A given dataset of three clusters (points in blue), and some realizations of the criterion $\Re \left\langle \frac{\mathcal{A}(\delta_{\boldsymbol{c}})}{\|\mathcal{A}(\delta_{\boldsymbol{c}})\|}, \frac{\boldsymbol{r}}{||\boldsymbol{r}||} \right\rangle$ as a function of the centroid hypothesis location $\boldsymbol{c}$, for different values of the sketch size m and the bandwidth σ.

Let us delve into the different realizations of the criterion shown in Figure 2.1.
As mentioned in Section 1.2.5, the bandwidth σ can be interpreted as the 'resolution' at which the distribution is captured. This becomes clearer now: when σ is low, there is insufficient smoothing and the criterion exhibits numerous spurious local maxima. At mid-range values of σ, the smoothing becomes appropriate, allowing the criterion to preserve the dataset's geometry. In this case, the most significant local maxima correspond to the true cluster centers, although some spurious local maxima may still be present in certain instances. Conversely, when σ is high, excessive smoothing occurs, causing the criterion to lose the dataset's 'clustered' structure and exhibit only a single maximum.
The size of the sketch, m, also influences the quality of the criterion. For a value that is too low, the criterion lacks precision. A larger value of m appears to 'attenuate' the spurious local maxima and enhance precision.

Step2. We add the found centroid to the support.

Step3. We find the optimal weights $\boldsymbol{\alpha}$ for the current set of centroids. This is done through a convex optimization, i.e.,

$$\boldsymbol{\alpha} \leftarrow \arg\min_{\boldsymbol{\alpha} \geq 0} \left\| \hat{\boldsymbol{z}} - \sum_{k=1}^{|\boldsymbol{C}|} \alpha_k \mathcal{A}(\delta_{\boldsymbol{c}_k}) \right\|$$

Step4. Before updating the residual through $\boldsymbol{r} \leftarrow \hat{\boldsymbol{z}} - \sum_{k=1}^{|\boldsymbol{C}|} \alpha_k \mathcal{A}(\delta_{\boldsymbol{c}_k})$, another nonconvex optimization is performed to refine the current solution.

We repeat those steps K times, resulting in K found centroids, and their associated weights.

It is important to note that the CL-OMP decoder is not guaranteed to find the global minimizer $\hat{\boldsymbol{\theta}}$. This is because CL-OMP involves nonconvex optimization steps and relies on random initializations. As a result, general guarantees are typically not available for such methods. Indeed, we have observed that the criterion used to select a new centroid (**Step1**) can exhibit multiple spurious local maxima that do not correspond to the true cluster centers. If the optimization of this criterion converges to such a maximum, it is highly likely to result in a suboptimal solution.
Inspired from CS, the authors of [2] proposed an alternative to CL-OMP, CL-OMP with replacement (CL-OMPR), see Algorithm 2.
The adaptation of CL-OMPR is quite simple : first, CL-OMP is performed. Then, we continue to perform the same steps as CL-OMP for a number of additional iterations (a common choice is to perform K additional steps, so $2K$ iterations in total),

with the added *Hard Thresholding* step.This variant allows more exploration and the suppression of spurious found centroids. Although yielding better results in practice than CL-OMP, CL-OMPR is still not guaranteed to find the global minimizer $\hat{\boldsymbol{\theta}}$.

Algorithm 2: CL-OMPR for CKM

Data: Sketch $\hat{\boldsymbol{z}}$, sketching operator $\mathcal{A}$, sparsity parameter K, bounds $\boldsymbol{l}$, $\boldsymbol{u}$
Result: Centroids $\boldsymbol{C}$, weights $\boldsymbol{\alpha}$
Initialize residual $\hat{\boldsymbol{r}} \leftarrow \hat{\boldsymbol{z}}$, centroids $\boldsymbol{C} \leftarrow \emptyset$;
for $t \leftarrow 1$ ***to*** $2K$ **do**
  Perform **Step 1** and **Step2** of Algorithm 1.
  Additional Step : Enforce sparsity by Hard Thresholding if $t > K$;
  if $|\boldsymbol{C}| > K$ **then**
    $\beta \leftarrow \arg\min_{\beta \geq 0} \left\| \hat{\boldsymbol{z}} - \sum_{k=1}^{|\boldsymbol{C}|} \beta_k \frac{\mathcal{A}(\delta_{c_k})}{\|\mathcal{A}(\delta_{c_k})\|} \right\|$;
    Select K largest entries $\beta_{i1}, \ldots, \beta_{iK}$;
    Reduce the support: $\boldsymbol{C} \leftarrow \{\boldsymbol{c}_{i1}, \ldots, \boldsymbol{c}_{iK}\}$;
  end
  end
  Perform **Step 3** and **Step 4** from Algorithm 1.
  Update residual: $\hat{\boldsymbol{r}} \leftarrow \hat{\boldsymbol{z}} - \sum_{k=1}^{|\boldsymbol{C}|} \alpha_k \mathcal{A}(\delta_{c_k})$;
end

2.2 Analyzing the failures of CL-OMP(R)

In the first section (2.1), we introduced the CL-OMP algorithm and its variant CL-OMPR. We briefly mentioned that these algorithms might fail to recover the optimal parameters $\hat{\boldsymbol{\theta}}$ from (2.1), i.e., $\boldsymbol{\theta}_\Delta \neq \hat{\boldsymbol{\theta}}$, where the decoder Δ is either CL-OMP or CL-OMPR. Such failures could occur due to **Step 1** in Algorithm 1, which seeks to find the most correlated centroid by maximizing a nonconvex function.

We will first identify the different failure scenarii of CL-OMP(R) in Section 2.2.1. Based on this analysis, we will highlight potential improvements for CL-OMP(R) and present them in Section 2.2.2.

2.2.1 The different scenarii

The problem we aim to solve is the K-means clustering problem. K-means clustering is a machine learning problem, and as defined in Section 1.1.3, such a problem has optimal parameters denoted by $\boldsymbol{\theta}^*$.
We have seen that Compressive K-means involves solving the problem (2.1), i.e., minimizing a cost function $\mathcal{C}(\boldsymbol{\theta}, \hat{z})$, where the optimal parameters $\hat{\boldsymbol{\theta}}$ are expected to be close to $\boldsymbol{\theta}^*$, i.e., $\hat{\boldsymbol{\theta}} \approx \boldsymbol{\theta}^*$. However, we will see that, under certain conditions, this is not the case.
We then introduced the heuristics CL-OMP(R), which aim to approximate the solution of problem (2.1), i.e., CL-OMP(R) acts as a decoder $\Delta : \hat{z} \to \boldsymbol{\theta}_\Delta$. Ideally, we would like $\boldsymbol{\theta}_\Delta \approx \hat{\boldsymbol{\theta}}$. However, we will see that this is not always true.

We thus understand that, when running the CL-OMP or CL-OMPR decoder Δ on a given sketch $\hat{z}$ of a dataset, different outcomes are possible.
To better understand these outcomes, we refer to an illustration from [13], see Figure 2.2.

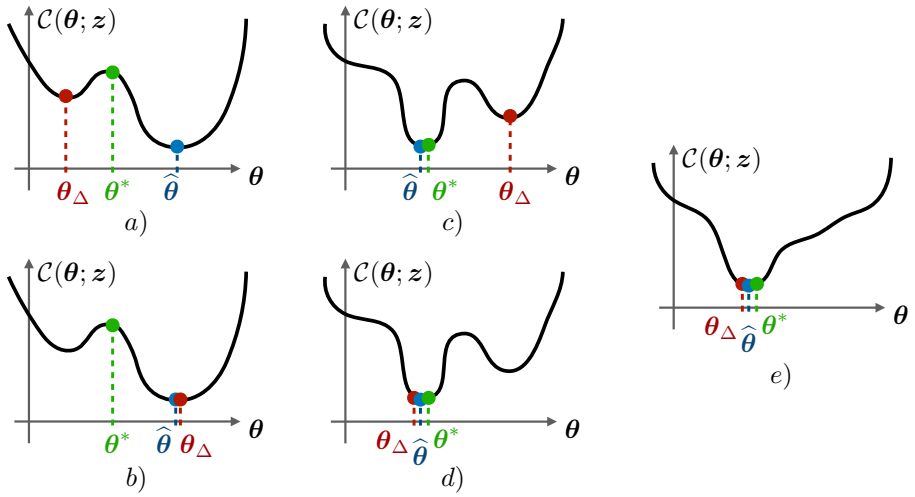


Figure 2.2: The different scenarii of a run of the decoder

We can summarize the different outcomes as follows:

- The first outcome is the ideal case where $\boldsymbol{\theta}_\Delta \approx \hat{\boldsymbol{\theta}} \approx \boldsymbol{\theta}^*$ (scenarii (d) and (e)). In this case, the ideal parameters are successfully recovered.
- The second outcome is characterized by a failure of the decoder, despite $\hat{\boldsymbol{\theta}} \approx \boldsymbol{\theta}^*$ (scenario (c)).

- The third outcome occurs when the sketch is poorly designed, leading to a global minimum of $\mathcal{C}(\boldsymbol{\theta}, \hat{z})$ where $\hat{\boldsymbol{\theta}}$ does not correspond to the desired parameters $\boldsymbol{\theta}^*$, i.e., $\hat{\boldsymbol{\theta}} \neq \boldsymbol{\theta}^*$. In this case, failure occurs regardless of whether the decoder Δ succeeds (scenario (b)) or fails (scenario (a)).

We would like to identify in which scenario we are more likely to fall, depending on the design of the sketch. This is precisely what the authors of [13] accomplished. They conducted numerical experiments to examine the influence of the sketching operator design, specifically the sketch size m and the bandwidth σ. We will briefly explain their findings. These results are supported by two experiments depicted in Figure 2.3 and Figure 2.4.

The influence of the sketch scale.
We consider a large sketch size m, to reduce its influence on potential fails.
For low values of σ, failures of the decoder Δ were detected, meaning that scenario (c) is possible.
High values of σ are more likely to result in a poor sketch design, i.e., scenarii (a)-(b).
This relates to our analysis of the criterion in **Step 1** of CL-OMPR (Section 2.1). Indeed, a low value of σ leads to spurious local maxima and may cause the decoder to fail. Conversely, a high value of σ results in an over-smoothed criterion, which fails to retain the dataset's structure.
As illustrated in Figure 2.3, there is a 'window' of σ values where CL-OMPR performs well.

The influence of the sketch size.
We consider a value of the bandwidth σ within the 'window' where CL-OMPR performs well to minimize its influence on potential failures.
A large sketch size $m \geq 10Kd$ will most likely lead to scenario (e).
A small sketch size $m \leq Kd$ will most likely lead to scenarii (a)-(b).
For intermediate values, $1 \leq m/Kd \leq 10$, the situation is less clear: as the number of measurements increases, we transition from scenarii (a)-(b) to (c) with a gradual increase in (d), and eventually to (e). However, it remains unclear at what point the transition from (a)-(b) to (c) occurs.
As illustrated in Figure 2.4, an increase in the number of measurements improves the performance of CL-OMPR.

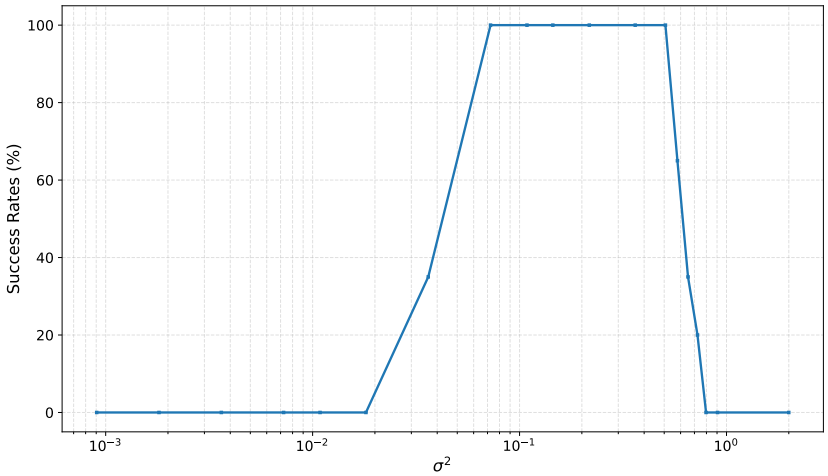


Figure 2.3: Empirical success rate of CL-OMPR (defining success as $\text{MSE}(\boldsymbol{\theta}_\Delta; \mathcal{X}) \leq 1.2 \times \text{MSE}(\tilde{\boldsymbol{\theta}}; \mathcal{X})$, with $\boldsymbol{\theta}_\Delta$the centroids found with CL-OMPR and $\tilde{\boldsymbol{\theta}}$ the centroids found with the Lloyd algorithm), over 20 random realizations of the sketching operation. The sketch size is fixed ($m = 20Kd$) and the bandwidth σ is varying. The data $\mathcal{X}$ is drawn from a mixture of $K = 7$ Gaussians and consists of $n = 10^5$ samples in $d = 3$.

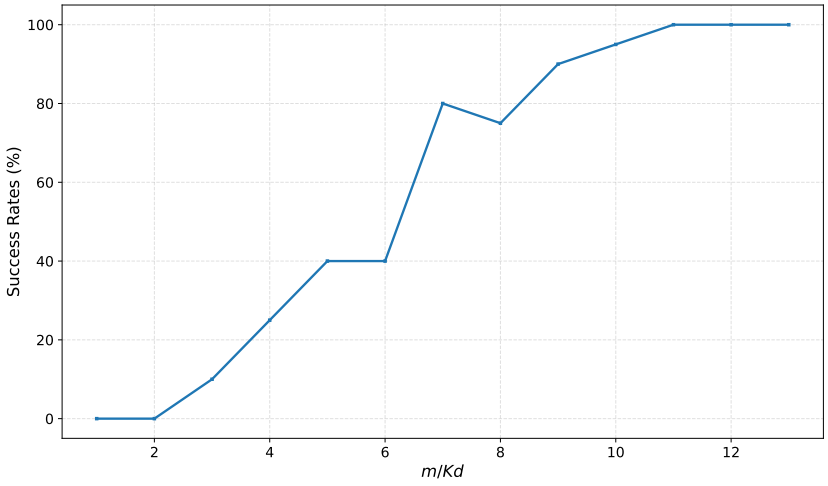


Figure 2.4: Empirical success rate of CL-OMPR (defining success as $\text{MSE}(\boldsymbol{\theta}_\Delta; \mathcal{X}) \leq 1.2 \times \text{MSE}(\tilde{\boldsymbol{\theta}}; \mathcal{X})$, with $\boldsymbol{\theta}_\Delta$the centroids found with CL-OMPR and $\tilde{\boldsymbol{\theta}}$ the centroids found with the Lloyd algorithm), over 20 random realizations of the sketching operation. The bandwidth σ is fixed and the sketch size is varying from $m = 1Kd$ to $m = 13Kd$. The data $\mathcal{X}$ is drawn from a mixture of $K = 7$ Gaussians and consists of $n = 10^5$ samples in $d = 3$.

2.2.2 Possible improvements

We presented CL-OMP(R) and analyzed its various possible failures. We identified two types of failures.
The first arises from the design of the sketch, such as an insufficient number of measurements m or a bandwidth that is too large.
The second type of failure is related to the decoder. Such failures were observed for low values of σ, when the sketch size is sufficiently large for the problem to be well-posed, i.e., $\hat{\boldsymbol{\theta}} \approx \boldsymbol{\theta}^*$.
In this work, we will focus on these specific situations. We will investigate wether an improvement to CL-OMPR is possible in such cases. Such an improvement would broaden the effective operating range of CL-OMPR.

Tuning of the sketching operator

One of the difficulties faced by practitioners of compressive clustering resides in the selection of the bandwidth parameter σ. A heuristic has been proposed in [2] for the choice of σ. Given a dataset $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^d \mid i = 1, \dots, n\}$, where samples are assumed to be drawn from a mixture of K-Gaussians ($\sum_{k=1}^{K} \alpha_k \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k)$), the idea is to estimate the average variance $\bar{\sigma}^2 = \frac{1}{Kd} \sum_{k=1}^{K} \sum_{l=1}^{d} \sigma_{k,l}^2$, where $\sigma_{k,1}^2, \dots, \sigma_{k,d}^2$ are the eigenvalues of Σ_k.
Then, the bandwidth can be set as $\sigma^2 = \bar{\sigma}^2$. The authors of [2] reported that this choice did not seem to yield very good results on real data. The authors left the search for a better choice of σ as an open question for future works.

The authors of [14] describe the choice of the bandwidth σ is currently something of an art. This highlights the necessity of broadening the operating range of CL-OMPR for different values of σ. Doing so would make the algorithm easier to parameterize.
We investigate such an improvement to CL-OMPR in the next chapter.

Chapter 3

Towards a robust decoder

Context. In Chapter 1, we briefly introduced the framework of compressive learning and described a practical problem: k-means clustering. In Chapter 2, we explored heuristics to approximately solve this problem: CL-OMP and its variant, CL-OMPR. We also analyzed the performance of these heuristics and identified areas for improvement. Notably, we highlighted the potential for expanding the operational range of a key parameter, the bandwidth σ. As discussed at the end of Chapter 2, such an expansion is desirable because it would make CL-OMP(R) easier to tune.
More specifically, we understood that the failures of the decoder in the situations we aim to improve are likely linked to the maximization of the nonconvex criterion in **Step 1** of CL-OMP(R), during the search for a new promising centroid.

This chapter. In this chapter, we will propose a method to address the deficiencies of CL-OMP(R) in these situations by improving the optimization of the nonconvex criterion. We will tailor our method by assuming that the data is drawn from an isotropic mixture of Gaussians.

The layout of this chapter is the following :

- In Section 3.1, we will begin with some theoretical background to ensure the reader has a solid understanding of what follows.
- In Section 3.2, we will analyze the correlation function to be optimized.
- Following this analysis, we will propose strategies to improve the optimization of the correlation function in Section 3.3
- Finally, we will perform numerical simulations to assess the performance of the proposed method and discuss these results in Section 3.4.

3.1 Theoretical Background

3.1.1 Optimization methods

Iterative schemes

Let us consider the generic optimization problem

$$\max_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \tag{3.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, and possibly nonconvex.
In practice, such a problem is approximated using iterative schemes. We start from an initial guess $\boldsymbol{x}_0$. At each iteration k, the next iterate $\boldsymbol{x}_{k+1}$ is computed from $\boldsymbol{x}_k$ using a **direction** $\boldsymbol{d}_k$ and a **step size** s_k:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + s_k\boldsymbol{d}_k \tag{3.2}$$

The choices of the directions and step sizes influence the method's behavior, particularly regarding the convergence of the method.

Gradient ascent method

Let us revisit the problem (3.1) introduced earlier in this section, where the function is assumed to be L-smooth.
A function $f(\cdot)$ is L-smooth when it is differentiable, and for any $\boldsymbol{x} \in \mathbb{R}^d$, we have:

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \Delta f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle - \frac{L}{2}||\boldsymbol{y} - \boldsymbol{x}||^2, \quad \forall \boldsymbol{y} \in \mathbb{R}^d \tag{3.3}$$

Given $\boldsymbol{x} \in \mathbb{R}^d$, we define $m_{\boldsymbol{x}} : \mathbb{R}^d \to \mathbb{R}$:

$$m_{\boldsymbol{x}}(\boldsymbol{y}) = f(\boldsymbol{x}) + \langle \Delta f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle - \frac{L}{2}||\boldsymbol{y} - \boldsymbol{x}||^2 \tag{3.4}$$

Combining (3.3) and (3.4), we can write $f(\boldsymbol{y}) \geq m_{\boldsymbol{x}}(\boldsymbol{y})$, thus suggesting the following iterative scheme to maximize $f(\cdot)$:

$$\boldsymbol{x}_{k+1} = \arg\max_{\boldsymbol{y}\in\mathbb{R}^d} m_{x_k}(\boldsymbol{y}) \tag{3.5}$$

Since $m_{\boldsymbol{x}}(\cdot)$ is a concave function, we have:

$$\begin{aligned} \boldsymbol{x}_{k+1} = \arg\max_{\boldsymbol{y}\in\mathbb{R}^d} m_{x_k}(\boldsymbol{y}) &\iff \Delta m_{x_k}(\boldsymbol{x}_{k+1}) = 0 \\ &\iff \Delta f(\boldsymbol{x}_k) - L(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) = 0 \\ &\iff \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \frac{1}{L}\Delta f(\boldsymbol{x}_k) \end{aligned} \tag{3.6}$$

This method, defined by (3.6), is called the **gradient ascent method**. It fits into the framework of iterative schemes (3.2), by taking the direction of the steepest slope, i.e., $d_k = \Delta f(\boldsymbol{x}_k)$, and the step size as $s_k = \frac{1}{L}$, with L being the Lipschitz constant of $\Delta f(\cdot)$.
To assess the efficiency of the gradient method, it is crucial to understand its global convergence properties. The following theorem provides a global convergence guarantee, specifying the number of iterations required to reduce the gradient norm below a given threshold ϵ.

Theorem 3.1. Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is an L-smooth function, bounded from above by f_{high}, and let $\{\boldsymbol{x}_k\}$ be the sequence generated by the gradient method applied to $f(\cdot)$. Given $\epsilon > 0$, if T is the first iteration such that $\|\nabla f(\boldsymbol{x}_T)\| \leq \epsilon$, then:

$$T \leq \frac{2L(f_{\text{high}} - f(\boldsymbol{x}_0))}{\epsilon^2}$$

In other words, the gradient method applied to a nonconvex function $f(\cdot)$, takes at most $\mathcal{O}(\epsilon^{-2})$ iterations to generate $\boldsymbol{x}_k$ such that $||\nabla f(\boldsymbol{x}_k)|| \leq \epsilon$.

Projected gradient

We can generalize the gradient method to the constrained optimization problem

$$\begin{array}{l} \max_{\boldsymbol{x}} f(\boldsymbol{x}) \\ \text{s.t. } \boldsymbol{x} \in \Theta \end{array} \tag{3.7}$$

where $\Theta \subset \mathbb{R}^n$ is a nonempty, closed, and convex set, and $f : \mathbb{R}^n \to \mathbb{R}$ is an L-smooth function on Θ, i.e.,

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \Delta f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle - \frac{L}{2}||\boldsymbol{y} - \boldsymbol{x}||^2, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \Theta \tag{3.8}$$

Given an approximation $\boldsymbol{x}_k \in \Theta$ for a solution of (3.7), a natural generalization of the gradient method is to define

$$\boldsymbol{x}_{k+1} = P_\Theta\Big(\boldsymbol{x}_k + \frac{1}{L}\nabla f(\boldsymbol{x}_k)\Big), \tag{3.9}$$

where $P_\Theta(\boldsymbol{z})$ is the projection of $\boldsymbol{z}$ onto Θ.
Let us define the vector $G_L(\boldsymbol{x}_k) = L(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)$, called the reduced gradient. $||G_L(\boldsymbol{x}_k)||$ can be used as a stationarity measure for (3.7).
Similar to the gradient method, we can show that the projected gradient method (3.9) takes at most $\mathcal{O}(\epsilon^{-2})$ iterations to generate $\boldsymbol{x}_k \in \Theta$ such that $||G_L(\boldsymbol{x}_k)|| \leq \epsilon$.

The projected gradient method alternates between a gradient step to find the next iterate $\boldsymbol{x}_{k+1}$ as if there were no constraints and a projection step to ensure that the updated $\boldsymbol{x}_{k+1}$ lies within the feasible region Θ.

Armijo line search

The gradient method requires knowledge of the Lipschitz constant L of the gradient of $f(\cdot)$, which might be difficult to compute in practice. We will examine a method that does not require the Lipschitz constant L but shares the same complexity guarantees as the gradient method (3.6).

The Armijo rule is used in the context of backtracking line search and was introduced by Armijo in [15]. The idea is to test smaller and smaller step sizes until an increase in the objective function is sufficient. The Armijo rule can be used as a stopping criterion for this method and is given by the following condition:

$$f(\boldsymbol{x}_k + s_k \boldsymbol{d}_k) \geq f(\boldsymbol{x}_k) + c s_k \langle \boldsymbol{d}_k, \nabla f(\boldsymbol{x}_k) \rangle \tag{3.10}$$

In this equation, $\boldsymbol{d}_k$ is the direction, s_k is the step size, and $c \in (0, 1)$. If the condition is not satisfied, we set $s_k = \rho s_k$ with $\rho \in (0, 1)$ and check the condition until it is satisfied. Fortunately, this condition will always be satisfied in a finite time, as the following theorem states.

Theorem 3.2. Let f be differentiable at $\boldsymbol{x} \in \mathbb{R}^n$. If $\langle \nabla f(\boldsymbol{x}), \boldsymbol{d} \rangle > 0$ and $\upsilon \in (0, 1)$, then $\exists \delta > 0$ such that

$$f(\boldsymbol{x} + s\boldsymbol{d}) \geq f(\boldsymbol{x}) + \upsilon s \langle \nabla f(\boldsymbol{x}), \boldsymbol{d} \rangle, \quad \forall s \in [0, \delta).$$

The gradient ascent method with Armijo line search is given in Algorithm 3.

Algorithm 3: Gradient ascent method with Armijo line search

Step 0: Given $\boldsymbol{x}_0 \in \mathbb{R}^n$, step size $s_0 > 0$, and parameters $\beta, \rho \in (0, 1)$, set $k := 0$;
Step 1: Set $l := 0$;
while $f(\boldsymbol{x}_k + s_k \beta^l \nabla f(\boldsymbol{x}_k)) < f(\boldsymbol{x}_k) + \rho s_k \beta^l \|\nabla f(\boldsymbol{x}_k)\|^2$ **do**
    Set $l := l + 1$;
end
Step 2: Update: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + s_k \beta^l \nabla f(\boldsymbol{x}_k)$, $s_{k+1} = s_k \beta^l$, set $k := k + 1$ and go back to **Step 1**.

Furthermore, the backtracking line search with the stopping criterion of the Armijo rule has the same global convergence properties as the gradient method.
We can also generalize the method for constrained optimization problem (3.7), choosing the next iterate $\boldsymbol{x}_{k+1}$ following the algorithm, and then projecting it onto the feasible region Θ, i.e., $P_\Theta(\boldsymbol{x}_{k+1})$, with $P_\Theta(\boldsymbol{z})$ the projection of $\boldsymbol{z}$ onto Θ.

3.1.2 Kernels and Random Fourier Features

Kernels are widely used in machine learning for capturing non-linear relationships within data. Formally, a kernel is a function $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that satisfies the following conditions:

- **Symmetry:** $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \kappa(\boldsymbol{y}, \boldsymbol{x})$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,
- **Positive Semi-Definiteness:** For any finite set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$, the Gram matrix $[K]_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is positive semi-definite, i.e., $\mathbf{v}^T K \mathbf{v} \geq 0$ for all $\mathbf{v} \in \mathbb{R}^n$.

One of the most widely used kernels is the Gaussian kernel, defined as:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma^2}\right),$$

where $\sigma > 0$ is the bandwidth parameter. The Gaussian kernel is shift-invariant, meaning it can be written as:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \kappa^{\Delta}(\boldsymbol{x} - \boldsymbol{y}),$$

where $\kappa^{\Delta} : \mathbb{R}^d \to \mathbb{R}$.

Bochner's Theorem

The approximation of shift-invariant kernels using Random Fourier Features is based on Bochner's theorem, which states:

Theorem 3.3 (Bochner's Theorem)**.** A continuous, shift-invariant kernel $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \kappa^{\Delta}(\boldsymbol{x} - \boldsymbol{y})$ on $\mathbb{R}^d$ is positive semi-definite if and only if κ^{Δ} is the Fourier transform of a non-negative measure μ:

$$\kappa^{\Delta}(\boldsymbol{\tau}) = \int_{\mathbb{R}^d} e^{j\langle \boldsymbol{\omega}, \boldsymbol{\tau} \rangle} \, d\Lambda(\boldsymbol{\omega}),$$

where $\boldsymbol{\tau} = \boldsymbol{x} - \boldsymbol{y}$.

In the case of the Gaussian kernel, where $\kappa^{\Delta}(\boldsymbol{\tau}) = e^{-\frac{\|\boldsymbol{\tau}\|^2}{2\sigma^2}}$, the corresponding measure Λ is a multivariate Gaussian distribution with mean zero and covariance $\sigma^{-2}\mathbf{I}_d$.

Random Fourier Features

Rahimi and Recht [12] proposed an efficient method to approximate kernels by sampling from the measure Λ. The key idea is to rewrite the kernel function as:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \Re\Big(\mathbb{E}_{\boldsymbol{\omega}\sim\Lambda}\left[e^{\boldsymbol{j}\langle\boldsymbol{\omega}, \boldsymbol{x}-\boldsymbol{y}\rangle}\right]\Big).$$

To approximate this expectation, they use Monte Carlo sampling. Let $\{\boldsymbol{\omega}_l\}_{l=1}^m$ be i.i.d. samples from Λ. The kernel can then be approximated as:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) \approx \Re\left\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y})\right\rangle,$$

where $\Phi(\boldsymbol{x}) \in \mathbb{R}^m$ is the feature map defined as:

$$\Phi(\boldsymbol{x}) = [\phi_{w_l}(\boldsymbol{x})]_{l=1}^m,$$

with

$$\phi_{\boldsymbol{\omega}}(\boldsymbol{x}) = \frac{1}{\sqrt{m}} e^{\boldsymbol{j}\langle\boldsymbol{\omega}, \boldsymbol{x}\rangle}.$$

3.1.3 A covering problem

In this section, we introduce the problem of covering a rectangular box domain Ω with spheres of radius r.
The objective is to determine the minimal number of centers $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \Omega$ such that every point in Ω lies within a distance r of at least one center. Mathematically, this can be formulated as follows:

Objective: Minimize the number of centers n:

$$\min n$$

Constraints:

- **Coverage constraint:** Every point in Ω must lie within a distance r of at least one selected center:

$$\forall \boldsymbol{\mu} \in \Omega, \exists i \in \{1, \ldots, n\} \text{ such that } \|\boldsymbol{\mu} - \boldsymbol{x}_i\|_2 \leq r.$$

- **Domain constraint:** The selected centers must lie within the domain Ω:

$$\boldsymbol{x}_i \in \Omega, \quad \forall i \in \{1, \ldots, n\}.$$

The exact solution to this problem is challenging due to the continuous nature of the domain Ω and the high-dimensional geometry involved. Computing the minimal number of spheres requires optimizing their placement to ensure full coverage, which is computationally infeasible in high dimensions. To address this, we use bounds to study the scaling of n with respect to the problem parameters.

Lower bound : A lower bound on n can be derived by comparing the total volume of the domain Ω and the volume of a single sphere. For a general hypercube $\Omega = [-1, 1]^d$, the domain volume is:

$$V_{\text{domain}} = 2^d.$$

The volume of a sphere of radius r in d-dimensions is:

$$V_{\text{sphere}} = \frac{\pi^{d/2}}{\Gamma(d/2+1)} r^d.$$

Since the spheres must cover the entire domain, we have:

$$n \cdot V_{\text{sphere}} \geq V_{\text{domain}}.$$

This gives the lower bound:

$$n \geq \frac{2^d \cdot \Gamma(d/2+1)}{\pi^{d/2} r^d} \geq \left(\frac{C_{\text{lower}}}{r}\right)^d,$$

where C_{lower} is a constant that depends on the geometry of the domain Ω.

Upper bound : An upper bound on n can be derived by using a grid-based approximation. For a general hypercube $\Omega = [-1, 1]^d$, the domain has side length 2 in each dimension. To ensure full coverage, we place grid points spaced at most r apart.
The number of grid points required along each dimension is:

$$n_{\text{1D}} = \left\lceil \frac{2}{r} \right\rceil.$$

The total number of points is therefore:

$$n \leq \left\lceil \frac{2}{r} \right\rceil^d.$$

Ignoring the ceiling for simplicity, we have:

$$n \leq \left(\frac{2}{r}\right)^d \leq \left(\frac{C_{\text{upper}}}{r}\right)^d,$$

where C_{upper} is a constant that depends on the geometry of the domain Ω.

Scaling of n : Combining the lower and upper bounds derived above, we can establish the scaling behavior of n as a function of the sphere radius r, the dimension d, and the geometry of the domain Ω.
Thus, the number of points n required to cover the domain satisfies:

$$\left(\frac{C_{\text{lower}}}{r}\right)^d \leq n \leq \left(\frac{C_{\text{upper}}}{r}\right)^d,$$

where C_{lower} and C_{upper} are constants that depend on the geometry of the domain Ω.
This result shows that n scales as:

$$n \sim \left(\frac{C}{r}\right)^d,$$

where C is a constant that depends on the geometry of the domain Ω.

3.2 Analysis of the Correlation Function

In this section, we will analyze the correlation function to be optimized. This optimization takes place during **Step 1** of CL-OMPR. The analysis focuses on the first iteration of CL-OMPR.

An approximate of the kernel density estimator

Given a sketching operator $\mathcal{A}$, associated with RFF and frequencies $W = \{\boldsymbol{w}_l\}_{l=1}^m$, with $\boldsymbol{w}_l \sim_{\text{i.i.d}} \mathcal{N}(0, \sigma^{-2}\mathbf{I}_d)$, and given $\boldsymbol{z}_{\mathcal{X}} \in \mathbf{C}^m$, the sketch of some dataset $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^d \mid i = 1, \ldots, n\}$, the correlation function corresponding to the criterion to maximize in **Step1** of CL-OMP(R), $f_W(\boldsymbol{c}) : \mathbb{R}^d \to \mathbb{R}$, associated with frequencies W, is

$$f_W(\boldsymbol{c}) := \Re \left\langle \mathcal{A}(\delta_{\boldsymbol{c}}), \boldsymbol{z}_{\mathcal{X}} \right\rangle.$$

Developing this function further, we use the fact that $\boldsymbol{z}_{\mathcal{X}} = \mathcal{A}(\hat{D}_{\mathcal{X}}) = \frac{1}{n}\sum_{i=1}^n \Phi(\boldsymbol{x}_i)$ and $\mathcal{A}(\delta_{\boldsymbol{c}}) = \Phi(\boldsymbol{c})$, where:

$$\Phi(\boldsymbol{x}) = [\phi_{\boldsymbol{w}_l}(\boldsymbol{x})]_{l=1}^m, \quad \phi_{\boldsymbol{w}}(\boldsymbol{x}) = \frac{1}{\sqrt{m}} e^{\boldsymbol{j}\langle \boldsymbol{w}, \boldsymbol{x}\rangle}.$$

Substituting these, we obtain:

$$f_W(\boldsymbol{c}) = \Re \left\langle \Phi(\boldsymbol{c}), \frac{1}{n}\sum_{i=1}^n \Phi(\boldsymbol{x}_i) \right\rangle = \frac{1}{n}\sum_{i=1}^n \Re \left\langle \Phi(\boldsymbol{c}), \Phi(\boldsymbol{x}_i) \right\rangle.$$

Using the Random Fourier Features framework [12] (Section 3.1.2), the inner product $\Re\langle\Phi(\boldsymbol{c}), \Phi(\boldsymbol{x}_i)\rangle$ approximates the Gaussian kernel. This is due to Bochner's theorem, which states that the Gaussian kernel:

$$\kappa(\boldsymbol{c}, \boldsymbol{x}_i) = \exp\left(-\frac{\|\boldsymbol{c} - \boldsymbol{x}_i\|^2}{2\sigma^2}\right),$$

can be expressed as:

$$\kappa(\boldsymbol{c}, \boldsymbol{x}_i) = \Re(\mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(0,\sigma^{-2}\mathbf{I}_d)}\left[e^{\boldsymbol{j}\langle\boldsymbol{w}, \boldsymbol{c}-\boldsymbol{x}_i\rangle}\right]).$$

By sampling m frequencies $\boldsymbol{w}_l \sim \mathcal{N}(0, \sigma^{-2}\mathbf{I}_d)$ and applying the Monte Carlo approximation, we obtain:

$$\Re\langle\Phi(\boldsymbol{c}), \Phi(\boldsymbol{x}_i)\rangle \approx \kappa(\boldsymbol{c}, \boldsymbol{x}_i).$$

Thus, $f_W(\boldsymbol{c})$ becomes:

$$f_W(\boldsymbol{c}) \approx \frac{1}{n}\sum_{i=1}^{n}\kappa(\boldsymbol{c}, \boldsymbol{x}_i),$$

which corresponds to the kernel density estimator:

$$f_{\text{KDE}}(\boldsymbol{c}) = \frac{1}{n}\sum_{i=1}^{n}\kappa(\boldsymbol{c}, \boldsymbol{x}_i).$$

The function $f_W(\boldsymbol{c})$ can be interpreted as a computationally efficient approximation of the Kernel Density Estimator (KDE), see Figure 3.1. By summing the contributions of an approximate Gaussian kernel $\kappa(c, \boldsymbol{x}_i)$ centered at each data point $\boldsymbol{x}_i \in \mathcal{X}$, $f_W(\boldsymbol{c})$ provides a measure of the density of data points in the vicinity of c. This means that higher values of $f_W(\boldsymbol{c})$ correspond to regions in the space where data points are more densely clustered, while lower values indicate sparser regions. As such, $f_W(\boldsymbol{c})$ serves as an effective proxy for identifying promising centroids during **Step 1** of CL-OMPR.

We understand that, in the situations of interest, for a low σ value, although the most significant maxima correspond to the true cluster centers, some spurious local maxima appear (Figure 3.1). As explained in Chapter 2, we believe that the decoder's failures in these situations are due to the optimization process, which may have converged to one of these spurious local maxima.
In the following section, we address an optimization method to avoid such behavior.

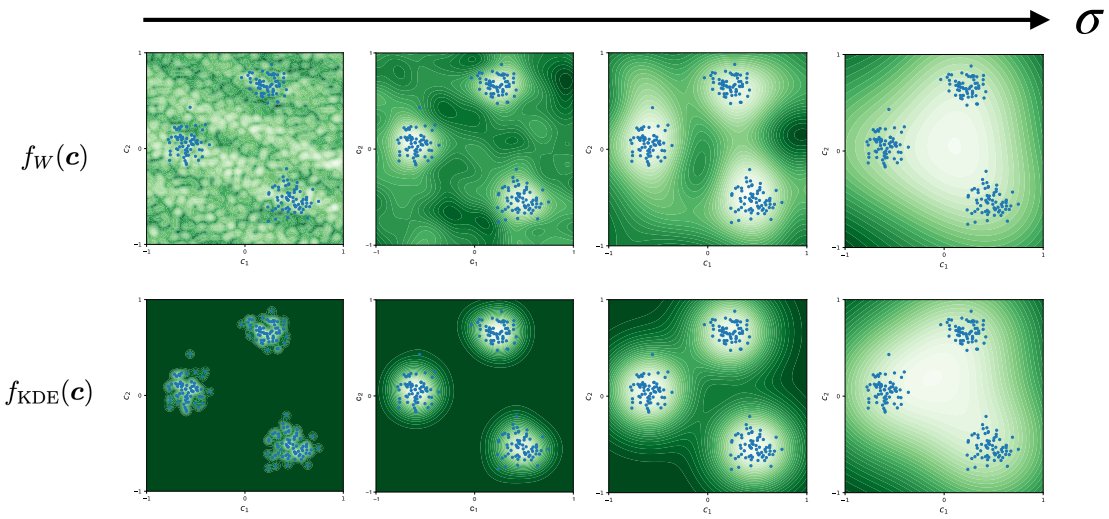


Figure 3.1: We consider a dataset of three clusters (blue points) in two dimensions. **Top row**: realizations of the correlation function $f_W(\boldsymbol{c})$ are shown. The sketch size m is fixed, while the bandwidth σ varies. **Bottom row**: the associated Kernel Density Estimator, representing the expected value of $f_W(\boldsymbol{c})$ as $m \to \infty$.

3.3 The optimization of the correlation function

The optimization of the correlation function constitutes **Step 1** of CL-OMPR (Algorithm 2). This step is therefore encountered at every iteration of CL-OMPR. The correlation function changes at each iteration because the residual is updated in **Step 5** by removing from $\boldsymbol{z}_{\mathcal{X}}$ the sketch of an estimated $|C|$-mixture of Diracs, corresponding to the term $\sum_{k=1}^{|C|} \alpha_k \mathcal{A}(\delta_{\boldsymbol{c}_k})$ (see Chapter 2).
In the most favorable case, where each cluster is well-localized by the correlation function, this process is likely to produce a new correlation function with no local maximum around any of the already identified centroids, particularly when these clusters are isotropic [14].
For the remainder of this chapter, we will assume this to be true, and that when a cluster is identified, its 'removal' during **Step 5** of CL-OMPR is perfect, leaving no residual local maximum in its vicinity.

The constrained optimization problem we face is formulated as follows:

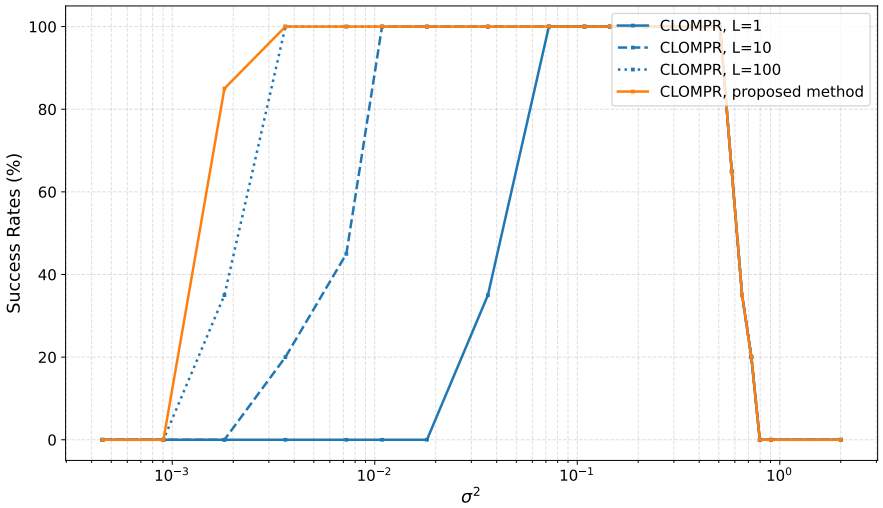
$$\max_{\boldsymbol{c} \in \Omega} f_W(\boldsymbol{c}),$$

where Ω is the domain (rectangular box) in which lies the data. More precisely, it is defined as $\Omega = \prod_{i=1}^{d} [l_i, u_i]$, with the bounds $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^d$, computed during the

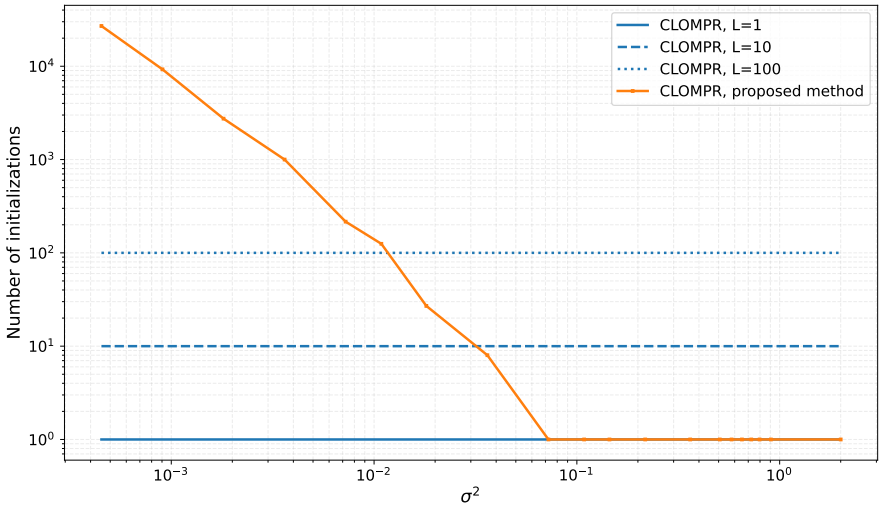
sketching such that all data are comprised in these bounds : $\boldsymbol{l} \leq \boldsymbol{x}_i \leq \boldsymbol{u}, \quad \forall \boldsymbol{x}_i \in \mathcal{X}$ (with $\boldsymbol{a} \leq \boldsymbol{b}$ the element-by-element comparison of vectors in $\mathbb{R}^d$).

Overcoming the limitations of CL-OMPR, particularly addressing the nonconvex optimization of the correlation function, has already been studied. One of the most recent contributions to this topic is the work of Belhadji and Gribonval [14]. The authors found out that, by taking a number L of random initializations (uniformly sampled within the domain), performing a kind of projected gradient ascent from each, and selecting the best output, the range of bandwidth σ values for which the algorithm performs well (matching Lloyd's algorithm) expands, provided that the number of frequencies m is sufficiently large (see Figure 3.2a).
More precisely, they observed that the enlargement only occured for lower values of σ, and a greater range was observed as the number of initialisations L got bigger. They also noticed that, for large values of σ, the perfomances of the algorithm did not depend on the value L.
They also made numerical experiments to understand the relation between the number of initializations L needed to achieve good performance related to the dimension d. They observed empirically that L should grow with the dimension.

Based on these observations, we understand that it is indeed possible to improve the decoder (and extend its operational range) by improving the optimization step of the correlation function.
The intuitions behind the promising results of the method proposed by Belhadji and Gribonval are as follows. As previously mentioned, for very large values of the bandwidth σ, oversmoothing occurs: clusters merge, and the KDE (approximated by the correlation function) exhibits only a single maximum. This explains why the number of initializations L does not impact the performance for large σ, as all initializations converge to the same maximum and are therefore useless.
Then, as the bandwidth σ decreases, the number of initializations L must increase to ensure good performance. This observed effect is due to the fact that, for well-clustered data, the size of the clusters as perceived by the KDE decreases with σ. Moreover, $f_W(\boldsymbol{c})$ will exhibit spurious local maxima. Therefore, the search space needs to be explored more extensively (by increasing the number of initializations L) to increase the chances of finding a local maximum aligned with a cluster location as σ becomes smaller.



(a)



(b)

Figure 3.2: **Top**: Comparison of the empirical success rates between CL-OMPR with L initializations ($L \in \{1, 10, 100\}$) and the proposed method. The results represent the mean over 20 random realizations of the sketching operation. The sketch size is fixed ($m = 20Kd$) while the bandwidth σ varies. A success is defined as $\text{MSE}(\boldsymbol{\theta}_\Delta; \mathcal{X}) \leq 1.2 \times \text{MSE}(\tilde{\boldsymbol{\theta}}; \mathcal{X})$, with $\boldsymbol{\theta}_\Delta$ the centroids found with the considered method and $\tilde{\boldsymbol{\theta}}$ the centroids found with the Lloyd algorithm. **Bottom**: comparison of the number of initializations for different values of σ. The data $\mathcal{X}$ is drawn from a mixture of $K = 7$ Gaussians and consists of $n = 10^5$ samples in $d = 3$.

In this section, we will provide a theoretical framework to support these empirical observations. We will establish a reasoning to estimate the minimum number of required initializations based on the choice of the bandwidth σ, as well as a way to select these initializations within the domain Ω, to maintain the optimization performance necessary for the proper functioning of CL-OMPR. In other words, this amounts to finding a set of initializations in the domain Ω.
From this reasoning, we will propose a method. As shown in Figure 3.2, the proposed method provides a wider operational range for CL-OMPR while limiting the required number of initializations, depending on the situation.

Before getting to the actual adaptative set of initialisations in Section 3.3.2, let use dive more into how the spurious local maxima appear in the correlation function in Section 3.3.1. We will then investigate strategies to detect those spurious local maxima in Section 3.3.3, to speed-up the optimization process.

3.3.1 The spurious local maxima

Let us recall the form of the correlation function for a dataset $\mathcal{X}$, which we derived in the previous section as an approximation of the Kernel Density Estimator:

$$f_W(\boldsymbol{c}) \approx \frac{1}{n}\sum_{i=1}^{n} \kappa(\boldsymbol{c}, \boldsymbol{x}_i) = f_{\text{KDE}},$$

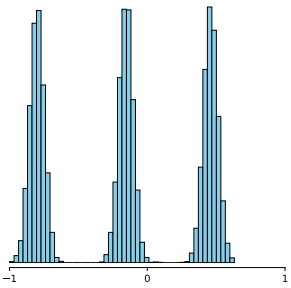
with $\kappa(\boldsymbol{c}, \boldsymbol{x}_i) = \exp\left(-\frac{\|\boldsymbol{c}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right)$, σ being the associated bandwidth.

When the data is well clustered (e.g. when it is drawn from a well-separated mixture of Gaussians), the associated KDE will exhibit local maxima in regions of high density, and a flatter behaviour in regions of low density (see Fig 3.1). When assuming a low enough bandwidth σ to avoid merging between clusters, then each local maximum will most likely correspond to the center of each cluster in the data. In the rest of the section, we will assume such a σ (note that improving the optimization of the correlation for low σ is the primary concern), but we will generalize for larger σ later at the end of Section 3.3.2.

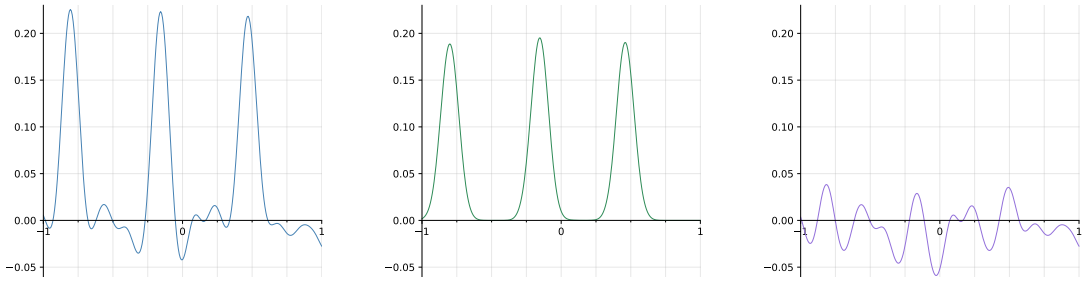
The spurious local maxima are due to 'added noise', since we only take a finite number of frequencies m to approximate the true kernel. Mathematically, we have

$$\begin{aligned} f_W(\boldsymbol{c}) &= f_{\text{KDE}}(\boldsymbol{c}) + \Big(f_W(\boldsymbol{c}) - f_{\text{KDE}}(\boldsymbol{c})\Big) \\ &= f_{\text{KDE}}(\boldsymbol{c}) + \Delta(\boldsymbol{c}), \end{aligned}$$

with $\Delta(\boldsymbol{c}) = f_W(\boldsymbol{c}) - f_{\text{KDE}}(\boldsymbol{c})$, the 'noise' responsible for the spurious local maxima.



(a) The dataset



(b) $f_W(\boldsymbol{c})$ (c) $f_{\mathrm{KDE}}(\boldsymbol{c})$ (d) $\Delta(\boldsymbol{c})$

Figure 3.3: **Top**: we consider a dataset of three clusters in 1 dimension. **Bottom**: a realization of the correlation function $f_W(\boldsymbol{c})$, the associated Kernel Density Estimator $f_{\mathrm{KDE}}(\boldsymbol{c})$ and their difference $\Delta(\boldsymbol{c})$.

We will differentiate between two distinct regions:
Regions of high density: In regions of high density, i.e., near the center of a cluster, the influence of $f_{\mathrm{KDE}}(\boldsymbol{c})$ is dominant. In such areas, the appearance of spurious local maxima is less likely. This is because the noise term $\Delta(\boldsymbol{c})$ would need to counteract the gradient of $f_{\mathrm{KDE}}(\boldsymbol{c})$ in order to satisfy the condition $\nabla f_{\mathrm{KDE}}(\boldsymbol{c}) + \nabla\Delta(\boldsymbol{c}) = 0$, and also perturb the Hessian $\nabla^2 f_{\mathrm{KDE}}(\boldsymbol{c})$ enough to create a new local maximum. Such a perturbation requires a significant magnitude of $\Delta(\boldsymbol{c})$, as the gradient and the curvature of $f_{\mathrm{KDE}}(\boldsymbol{c})$ are strong near the center.

Regions of low density: In regions far from any center of the clusters, $f_{\mathrm{KDE}}(\boldsymbol{c})$ is weak, and the appearance of spurious local maxima is more likely. This happens because the gradient of $f_{\mathrm{KDE}}(\boldsymbol{c})$ is close to zero, and the Hessian is nearly flat (with little curvature). The noise $\Delta(\boldsymbol{c})$ can easily disturb the landscape in these flat areas, leading to the formation of spurious local maxima.

This is illustrated by Figure 3.3.

3.3.2 An adaptative set of initializations

We consider that our data is drawn from a mixture of well-separated isotropic gaussians, i.e. $\mathcal{X} = \{\boldsymbol{x}\}_{i=1}^{n}$, with $\boldsymbol{x}_1, ..., \boldsymbol{x}_n \sim_{\text{i.i.d}} \sum_{k=1}^{K} \alpha_k \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k)$, with $\Sigma_k = \sigma_k \mathbf{I}_d$.
As the number of samples n goes to ∞, f_{KDE} with bandwidth σ converges to the kernel mean embedding f_{KME}, which is given by (see Appendix B for the mathematical development) :

$$f_{\text{KME}}(\boldsymbol{c}) := \sum_{k=1}^{K} \beta_k \exp\Big(-\frac{1}{2}(\boldsymbol{c} - \boldsymbol{\mu}_k)^T (\Sigma_k + \sigma^2 \mathbf{I}_d)^{-1} (\boldsymbol{c} - \boldsymbol{\mu}_k)\Big),$$

with $\beta_k = (\frac{\sigma^2}{\sigma^2 + \sigma_k^2})^{\frac{d}{2}}$.
This form corresponds to a sum of Gaussians, each associated with a weight β_k and a spread $\Sigma_k + \sigma^2 \mathbf{I}_d$. Here, we assume a sufficiently small σ, which ensures not only that each maximum of f_{KME} corresponds to the center of a Gaussian in the mixture from which the data is drawn, meaning that the influence of each Gaussian in f_{KME} on the others becomes negligible. This is reasonable given that the data is assumed to be drawn from well-separated Gaussians. We will revisit the case of a larger σ later.
If we were to optimize f_{KME}, starting from any $\boldsymbol{c}_0 \in \Omega$ (note that $\Omega = \mathbb{R}^d$ when $n \to \infty$) and performing a gradient ascent (like Algorithm 3), we would end up finding a local maximum, thus finding the center of a cluster.
When dealing with its approximation $f_W(\boldsymbol{c})$, this is different, since spurious local maxima tend to appear, making the optimization more difficult, as it is possible to find one of those spurious maxima instead of one of the good local maxima, corresponding to one of the K centers $\boldsymbol{\mu}_k$.
However, as we have seen in Section 3.3.1, these spurious local maxima are more likely to appear in regions of low density, i.e., far from any center of the Gaussians. In regions of high density, i.e., close to one center of the Gaussians, spurious local maxima are less likely to appear, leaving the high-density regions as basins of attraction for the good local maxima.
Moreover, those regions of high density are related to the spread of the gaussians in $f_{\text{KME}}(\boldsymbol{c})$; $f_W(\boldsymbol{c})$ being its approximate. As we have seen, the spread is given by $\Sigma_k + \sigma^2 \mathbf{I}_d$, for each k-gaussian in the mixture. Without taking any assumptions on the covariance matrices Σ_k, the spread of each k-Gaussian is at least equal to $\sigma^2 \mathbf{I}_d$. Indeed, since Σ_k is a symmetric positive semi-definite matrix by definition (covariance), we have $\Sigma_k + \sigma^2 \mathbf{I}_d \succcurlyeq \sigma^2 \mathbf{I}_d$.
The size of the basins of attraction is therefore linked to σ. We consider that, at a distance $\eta\sigma$ from the center of a Gaussian, its influence remains very strong, making it highly likely to belong to its basin of attraction. Typical choices for η

that guarantee a strong influence are $\eta \in [0, 3]$. The parameter η can be adjusted based on the quality of the approximation of f_{KDE} by f_W, which depends on the number of considered measurements m (with a lower value of η providing a more conservative estimate). Also, note that if $\Sigma_k \neq 0$, the spread of each Gaussian is larger than $\sigma^2 \mathbf{I}_d$, meaning the regions of strong influence may extend beyond $\eta\sigma$, ensuring to cover the worst case scenario.

Let us now consider an optimization scheme that converges to the local maximum within the basin of attraction where it is initialized. The specific construction of this scheme will be detailed later in this section.
Then, to converge to a good local maximum corresponding to the center of a Gaussian, we need at least one starting point $\boldsymbol{c}_0$ within the basin of attraction of such a local maximum, i.e., at most $\eta\sigma$ away from it.

This challenge can be reformulated as the problem of placing a set of points within the domain to ensure that at least one of them lies within the basin of attraction of each Gaussian center. Determining the minimum number of such points can then be expressed as the following optimization problem:

A Covering Problem

We aim to find the minimal number of points $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_L \in \Omega$, such that at least one point lies inside a sphere of radius $\eta\sigma$ centered at any $\boldsymbol{\mu} \in \Omega$. This ensures coverage regardless of the center $\boldsymbol{\mu}$.
This leads to the following formulation:

Objective:
Minimize the number of points n:

$$\min n$$

Constraints:

- **Coverage constraint:** Every point in Ω must lie within a distance $\eta\sigma$ of at least one selected point:

$$\forall \boldsymbol{\mu} \in \Omega, \exists i \in \{1, \ldots, L\} \text{ such that } \|\boldsymbol{\mu} - \boldsymbol{c}_i\|_2 \leq \eta\sigma.$$

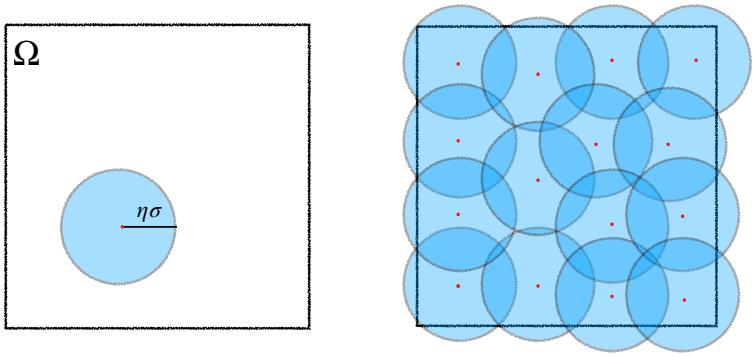
- **Domain constraint:** The selected points must lie within the domain:

$$\boldsymbol{c}_i \in \Omega, \quad \forall i \in \{1, \ldots, L\}.$$

This problem can be directly interpreted as a covering problem with spheres in a continuous domain. An equivalent formulation consists of determining the minimal number of centers $\boldsymbol{c}_i \in \Omega$ such that spheres of radius $\eta\sigma$, centered at $\boldsymbol{c}_i$, fully cover the domain Ω (see Figure 3.4).
This covering problem and the scaling of n are detailed in Section 3.1.3. The scaling of L can be expressed as:

$$L \sim \left(\frac{C}{\eta\sigma}\right)^d,$$

where C is a constant that depends on the geometry of the domain Ω.



(a) (b)

Figure 3.4: Illustrative example in 2D of the covering problem. Given a domain Ω and a sphere of radius $\eta\sigma$ (a), the objective is to cover the entire domain by centering the spheres at some points (in red) of the domain Ω (b).

The optimization scheme

We will again use the fact that $f_W(\boldsymbol{c})$ converges to f_{KME}, when the number of measurements $m \to \infty$ and the number of samples $n \to \infty$.
Assuming negligible influence of the Gaussians on one another, we focus on a region near a center $\boldsymbol{\mu}_k$, for $k = 1, \ldots, K$. Mathematically, this region is defined as:

$$\boldsymbol{c} \in \{\boldsymbol{c} \in \Omega : \|\boldsymbol{c} - \boldsymbol{\mu}_k\| \leq \eta\sigma\},$$

which is a ball of radius $\eta\sigma$ centered at $\boldsymbol{\mu}_k$. In this region, the Kernel Mean Embedding function $f_{\text{KME}}(\boldsymbol{c})$ is given by:

$$f_{\text{KME}}(\boldsymbol{c}) = \beta_k \exp\left(-\frac{1}{2}(\boldsymbol{c} - \boldsymbol{\mu}_k)^T\Big((\sigma_k^2 + \sigma^2)\mathbf{I}_d\Big)^{-1}(\boldsymbol{c} - \boldsymbol{\mu}_k)\right),$$

The gradient of $f_{\mathrm{KME}}(\boldsymbol{c})$ is:

$$\nabla f_{\mathrm{KME}}(\boldsymbol{c}) = f_{\mathrm{KME}}(\boldsymbol{c}) \cdot \left(- \left((\sigma_k^2 + \sigma^2)\mathbf{I}_d \right)^{-1} (\boldsymbol{c} - \boldsymbol{\mu}_k) \right).$$

The vector $(\boldsymbol{c} - \boldsymbol{\mu}_k)$ represents the direction from the center $\boldsymbol{\mu}_k$ to the point $\boldsymbol{c}$. Since $-(\boldsymbol{c} - \boldsymbol{\mu}_k)$ points towards $\boldsymbol{\mu}_k$, the gradient $\nabla f_{\mathrm{KME}}(\boldsymbol{c})$ always points directly toward the center $\boldsymbol{\mu}_k$ within this region.

Consider the following iterative scheme (see Section 3.1.1):

$$\boldsymbol{c}_{k+1} = \boldsymbol{c}_k + s_k \nabla f(\boldsymbol{c}_k).$$

Starting from $\boldsymbol{c}_0$ within the defined region (a ball of radius $\eta\sigma$ centered at $\boldsymbol{\mu}_k$), we aim to ensure that the iterates remain inside this ball and converge to the associated center $\boldsymbol{\mu}_k$.
Since the gradient $\nabla f_{\mathrm{KME}}(\boldsymbol{c})$ points directly towards the center $\boldsymbol{\mu}_k$ within this region, a sufficient condition for staying inside the ball is to limit the distance traveled at each iteration. This can be achieved by ensuring that successive steps do not exceed a maximum distance of $\eta\sigma$ (See Figure 3.5).

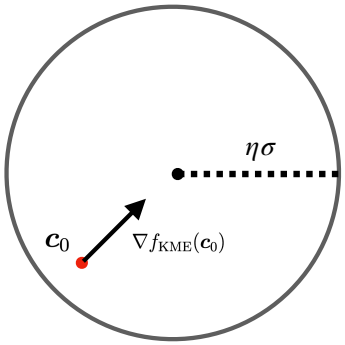


Figure 3.5: To ensure that $\boldsymbol{c}_0$ remains within the ball, it is sufficient to constrain the step size at each iteration to not exceed the radius $\eta\sigma$, as the gradient $\nabla f_{\mathrm{KME}}(\boldsymbol{c})$ directs the updates toward the center within the ball.

This suggests the following modification of the gradient ascent with Armijo line search (Algorithm 3) :

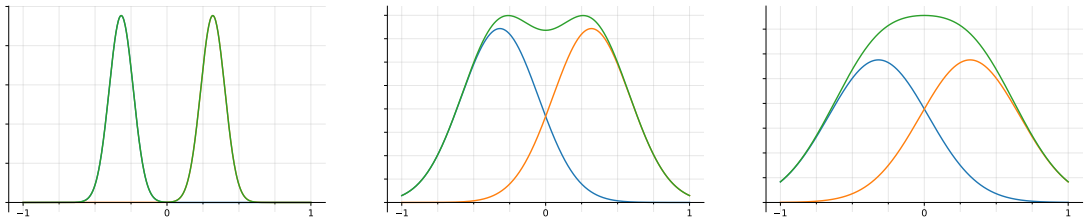
Algorithm 4: Gradient ascent method with Armijo line search and distance constraint

Step 0: Given $\boldsymbol{x}_0 \in \mathbb{R}^n$, step size $s_0 > 0$, and parameters $\beta, \rho \in (0,1)$, $\eta\sigma > 0$, set $k := 0$;
Step 1: Set $l := 0$;
while
$f(\boldsymbol{x}_k + s_k\beta^l \nabla f(\boldsymbol{x}_k)) < f(\boldsymbol{x}_k) + \rho s_k \beta^l \|\nabla f(\boldsymbol{x}_k)\|^2$ ***or*** $\|s_k\beta^l \nabla f(\boldsymbol{x}_k)\| > \eta\sigma$
do
    Set $l := l + 1$;
end
Step 2: Update: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + s_k\beta^l \nabla f(\boldsymbol{x}_k)$, $s_{k+1} = s_k\beta^l$, set $k := k + 1$ and go back to **Step 1**.

Considerations and generalizations

The method we propose involves modifying the optimization step (**Step 1**) of CL-OMPR by performing Algorithm 4 for each starting point in the set of initializations described in Section 3.3.2. The optimization algorithm and the set of initializations have been designed under the assumption that the data $\mathcal{X}$ is drawn from a mixture of well-separated isotropic Gaussians, which aligns with the sparsity assumption underlying the inverse problem that CL-OMPR aims to solve (see Section 1.2.4).
We also used the fact that the correlation function f_W converges to the Kernel Density Estimator f_{KDE} as the number of Random Fourier Features $m \to \infty$, and that f_{KDE} converges to the Kernel Mean Embedding as the number of samples $n \to \infty$. Thus, the effectiveness of the proposed method depends critically on both the number of random frequencies m and the number of samples n.

The issue observed empirically by the authors of [14] demonstrated that CL-OMPR fails for low values of σ, and that improving the optimization step enhances its success for such values of σ. Consequently, we focused our approach on addressing the low-σ case. However, while our method is specifically designed for low values of σ, where the Gaussians in f_{KME} have negligible influence on each other (e.g. Figure 3.6a), we can explore its behavior in scenarii with larger σ.
For higher values of σ, the Gaussians in f_{KME} begin to influence each other (see Figures 3.6b-3.6c), causing the maxima to slightly shift away from the true centers of the associated Gaussians (3.6b) or merge together, resulting in a single maximum (3.6c).



(a) low σ (b) mid σ (c) high σ

Figure 3.6: We consider two Gaussian functions (in blue and orange), and their sum (in green). The Gaussian centers are fixed, but their (shared) spread is increasing from left to right.

In such scenarii, the concept of good local maxima corresponding to the center of a Gaussian no longer holds. The behavior of CL-OMPR in these cases is not fully understood; however, it can still succeed by greedily identifying regions of high density. Let us now analyze how our approach works under these conditions.

Let us extend the concept of good local maxima to include any maximum resulting from the influence of one or more Gaussians. As we can observe, merging in case 3.6c will only enlarge the regions of high influence, thereby expanding the basin of attraction. In case 3.6b, although the basin of attraction for a single Gaussian is reduced, since any maximum can be considered a good one, we can interpret this as an effective enlargement of the overall basin.

In other words, while our method is designed for situations where the Gaussians of f_{KME} have negligible influence on each other, it remains valid for values of σ that result in non-negligible influence. Indeed, the regions of influence of merged Gaussians can only expand.

Interpretations

The goal of this section was to establish a theoretical framework to support the empirical observations presented in Section 3.3 . We have reached the conclusion that the number of initializations required to guarantee good performance is

$$L \sim \left(\frac{C}{\eta\sigma}\right)^d$$

where C is a constant that depends on the geometry of the domain Ω.
This explains the various observations discussed, notably that the number of

initializations must increase as σ decreases, that this number does not affect performance for large values of σ, and that it suffers from the curse of dimensionality. Moreover, the proposed method allows us to limit the number of initializations depending on the situation (see Figure 3.2) and to explore the space in a structured manner rather than randomly. Note that the results presented in Figure 3.2 use a naïve solution for the covering problem.
The interpretation is that we explore the space based on the size of the objects we are trying to locate.

3.3.3 Detecting the spurious local maxima

In the last subsection, we developed an adaptative set of initializations to explore the domain, and hopefully visit all the good maxima.

It would be convenient to be able to detect wether a local maximum is a spurious one or a good one. In the latter case, we would stop the search, speeding-up the process. In this subsection, we explore how to achieve this.

We assume that our data is drawn from a well-separated mixture of isotropic Gaussians, i.e. $\mathcal{X} = \{\boldsymbol{x}\}_{i=1}^{n}$, with $\boldsymbol{x}_1, ..., \boldsymbol{x}_n \sim_{\text{i.i.d}} \sum_{k=1}^{K} \alpha_k \mathcal{N}(\boldsymbol{\mu}_k, \sigma_k^2 \mathbf{I}_d)$, and assuming a very large number of samples n such that f_{KDE} converges to f_{KME}, with

$$f_{\text{KME}}(\boldsymbol{c}) = \sum_{k=1}^{K} \alpha_k \Big(\frac{\sigma^2}{\sigma^2 + \sigma_k^2} \Big)^{\frac{d}{2}} \cdot \exp\Big(-\frac{1}{2} \frac{||\boldsymbol{\mu}_k - \boldsymbol{c}||^2}{\sigma^2 + \sigma_k^2} \Big)$$

We still consider a low enough σ such that each local maximum of f_{KME} corresponds to a center $\boldsymbol{\mu}_k$ and that the influence of the Gaussians on one another remains negligible.
We have seen that, to find a centroid, we use f_W, the approximation of f_{KDE}, and maximize it. If the problem is well-posed, then every good local maximum (i.e., close to a center $\boldsymbol{\mu}_k$) should have a higher function value than any spurious local maximum.
Mathematically,let $\boldsymbol{\mu}_p \in \{\boldsymbol{\mu}_k : k = 1, \ldots, K\}$ denote the good local maximum with the lowest function value, i.e., $\boldsymbol{\mu}_p = \arg\min_{\boldsymbol{\mu}_k \in \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K\}} f_W(\boldsymbol{\mu}_k)$. And $\boldsymbol{x}_s^*$ denote the spurious local maximum with the highest function value. The problem is considered well-posed if

$$f_W(\boldsymbol{x}_s^*) < f_W(\boldsymbol{\mu}_p)$$

The idea would be to find a threshold value T to distinguish between good and spurious local maxima, such that :

$$f_W(\boldsymbol{x}_s^*) < T < f_W(\boldsymbol{\mu}_p)$$

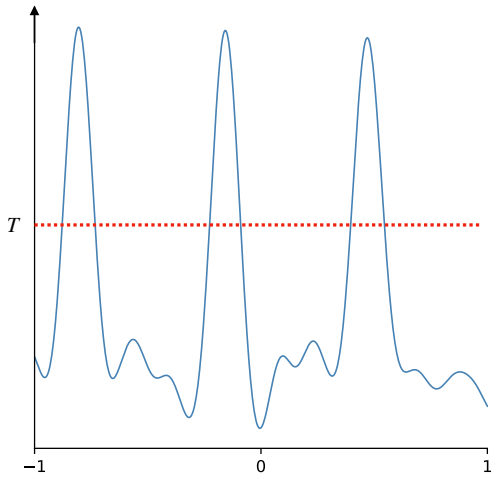


Figure 3.7: Illustrative example of a threshold T to distinguish between good and spurious local maxima

This way, when performing the optimization with multiple initializations, we can evaluate each critical point $\boldsymbol{c}^*$ encountered. If $f_W(\boldsymbol{c}^*) > T$, the search is stopped, as the condition indicates that a good local maximum has been found.
Recall from Section 3.3.1 that $f_W(\boldsymbol{c}) = f_{\text{KDE}}(\boldsymbol{c}) + \Delta(\boldsymbol{c})$, which implies:

$f_W(\boldsymbol{\mu}_p) = f_{\text{KME}}(\boldsymbol{\mu}_p) + \Delta(\boldsymbol{\mu}_p)$, and $f_W(\boldsymbol{x}_s^*) = f_{\text{KME}}(\boldsymbol{x}_s^*) + \Delta(\boldsymbol{x}_s^*)$,

using the fact that f_{KDE} converges to f_{KME}.
Given the unpredictability of $\Delta(\boldsymbol{c})$, the most rational choice for T is to maximize the reliance on the predictable component $f_{\text{KME}}(\boldsymbol{c})$. A natural choice is the midpoint:

$$T = \frac{f_{\text{KME}}(\boldsymbol{\mu}_p) + f_{\text{KME}}(\boldsymbol{x}_s^*)}{2}$$

This choice ensures a symmetric and unbiased threshold, separating the good and spurious local maxima by focusing on the stable part of $f_W(\boldsymbol{c})$. However, such a T requires the value $f_{\text{KME}}(\boldsymbol{\mu}_p)$ which is unknown, since the weights α_k and the covariances $\sigma_k^2\mathbf{I}_d$ are unknown a priori. It also requires the value of $f_{\text{KME}}(\boldsymbol{x}_s^*) > 0$, which represents the positive influence of the Kernel Density Estimator over which the spurious local maximum $f_W(\boldsymbol{x}_s^*)$ appeared. Recall from Section 3.3.1 that spurious local maxima typically arise in regions of low density, i.e., far from any Gaussian center. Consequently, $f_{\text{KME}}(\boldsymbol{x}_s^*)$ is likely to be much smaller than $f_{\text{KME}}(\boldsymbol{\mu}_p)$.

Controlling the spurious maxima

Another idea for the design of T is to rely only on $f_W(\boldsymbol{x}_s^*)$ and set $T = f_W(\boldsymbol{x}_s^*)$. $f_W(\boldsymbol{x}_s^*)$ is unknown, but depends entirely on $\Delta(\boldsymbol{c})$, since it is a spurious local maximum. Thus, we would like to somehow control $f_W(\boldsymbol{x}_s^*)$.
The idea is to establish a relationship between the number of random frequencies m and the probability that $f_W(\boldsymbol{x}_s^*)$ exceeds a certain threshold.
Formally, we are interested in bounding the probability:

$$P\left(f_W(\boldsymbol{x}_s^*) > \epsilon\right),$$

where $f_W(\boldsymbol{x}_s^*)$ depends on $\Delta(\boldsymbol{c})$, which itself diminishes as $m \to \infty$. The goal is to determine how m influences this probability, ensuring that for sufficiently large m, the spurious local maxima values $f_W(\boldsymbol{x}_s^*)$ remain below the desired threshold ϵ with high probability.
This could involve analyzing the statistical concentration of $f_W(\boldsymbol{x}_s^*)$ around its expected value as m increases, potentially using concentration inequalities to derive bounds on:

$$P\left(f_W(\boldsymbol{x}_s^*) > \epsilon\right) \leq g(m, \epsilon),$$

where $g(m, \epsilon)$ is a function that decreases with m, capturing how the likelihood of exceeding ϵ diminishes as more random frequencies are used.
The derivation of such a bound is beyond the scope of this work and is left for future research.

A Special Case

We derived above a method to design T if we had access to $f_{\text{KME}}(\boldsymbol{\mu}_p)$, the lowest function value of f_{KME} at a center $\boldsymbol{\mu}_p \in \{\boldsymbol{\mu}_k : k = 1, \ldots, K\}$, given by:

$$f_{\text{KME}}(\boldsymbol{\mu}_p) = \alpha_p \Big(\frac{\sigma^2}{\sigma^2 + \sigma_p^2}\Big)^{\frac{d}{2}} \cdot \exp\Big(-\frac{1}{2}\frac{\|\boldsymbol{\mu}_p - \boldsymbol{\mu}_p\|^2}{\sigma^2 + \sigma_p^2}\Big) \tag{3.11}$$

$$= \alpha_p \Big(\frac{\sigma^2}{\sigma^2 + \sigma_p^2}\Big)^{\frac{d}{2}}. \tag{3.12}$$

However, this value is unknown a priori since we do not have access to the weight α_p or the variance σ_p.
If we consider a balanced dataset such that $\alpha_k = \frac{1}{K}, \quad \forall k \in \{1, \ldots, K\}$, with similar cluster sizes, i.e., $\sigma_k = \sigma_{\mathcal{X}}, \quad \forall k \in \{1, \ldots, K\}$, then the weights are known,

and $\sigma_{\mathcal{X}}$ can be approximated using the heuristic from [2] (see Section 2.2.2), such that $\sigma_{\mathcal{X}} \approx \bar{\sigma_{\mathcal{X}}}$. Thus, we have:

$$f_{\text{KME}}(\boldsymbol{\mu}_p) = \alpha_p \left(\frac{\sigma^2}{\sigma^2 + \sigma_p^2} \right)^{\frac{d}{2}} \approx \frac{1}{K} \left(\frac{\sigma^2}{\sigma^2 + \bar{\sigma_{\mathcal{X}}}^2} \right)^{\frac{d}{2}}.$$

Following the design of T derived earlier, we have:

$$T = \frac{1}{2} \Big(f_{\text{KME}}(\boldsymbol{\mu}_p) + f_{\text{KME}}(\boldsymbol{x}_s^*) \Big).$$

Since $f_{\text{KME}}(\boldsymbol{x}_s^*)$ is likely to be much smaller than $f_{\text{KME}}(\boldsymbol{\mu}_p)$, we set:

$$f_{\text{KME}}(\boldsymbol{x}_s^*) = \epsilon \cdot f_{\text{KME}}(\boldsymbol{\mu}_p), \quad \text{with } 0 \leq \epsilon \ll 1.$$

Substituting this expression into T, we get:

$$T = \frac{1+\epsilon}{2} \cdot f_{\text{KME}}(\boldsymbol{\mu}_p)$$

Replacing $f_{\text{KME}}(\boldsymbol{\mu}_p)$ with its explicit form, we obtain:

$$T \approx \frac{1+\epsilon}{2} \cdot \frac{1}{K} \left(\frac{\sigma^2}{\sigma^2 + \bar{\sigma}_{\mathcal{X}}^2} \right)^{\frac{d}{2}}$$

This proposed threshold leverages the fact that the correlation function f_W converges to the Kernel Mean Embedding f_{KME} as the number of Random Fourier Features $m \to \infty$ and the number of data samples $n \to \infty$. Consequently, the quality of the threshold T critically depends on both the number of random frequencies m and the sample size n.

Moreover, the threshold is designed for a sufficiently small σ such that the influence of the Gaussians on one another remains negligible. When the Gaussians do interact, their combined influence is always positive. Consequently, the values of maxima formed due to interaction or merging (see Figure 3.6) can only be higher than those observed in the case of minimal Gaussian overlap (i.e., low σ), making the threshold T valid.
However, this reasoning holds primarily for the first iteration. For subsequent iterations of CL-OMPR, the threshold value T is not guaranteed to remain effective in cases of high σ, where interactions between Gaussians become more significant.

3.4 Numerical simulations

Let us summarize the method proposed in the previous sections with the following algorihtm :

Algorithm 5: Summary of the proposed method

Step 0: Given f_W, a set of initializations $\mathcal{S}$, a threshold T, and a radius $\eta\sigma$;
Step 0.1: Pick $\boldsymbol{c}_0 \in \mathcal{S}$ uniformly at random and set $\boldsymbol{c}_{\text{best}} \leftarrow \boldsymbol{c}_0$;
while *not stopped* **do**
  Step 1: Perform **Algorithm 4** with radius $\eta\sigma$ starting from $\boldsymbol{c}_0$ until a critical point $\boldsymbol{c}^*$ is reached;
  Step 2: if $f_W(\boldsymbol{c}^*) > T$ **then**
    Stop and return $\boldsymbol{c}^*$;
  else
    if $f_W(\boldsymbol{c}^*) > f_W(\boldsymbol{c}_{best})$ **then**
      Update $\boldsymbol{c}_{\text{best}} \leftarrow \boldsymbol{c}^*$;
    end
    Update $\mathcal{S}$ as $\mathcal{S} \setminus \{\boldsymbol{c}_0\}$;
    if $\mathcal{S}$ *is empty* **then**
      Stop and return $\boldsymbol{c}_{\text{best}}$;
    end
    Pick a new $\boldsymbol{c}_0 \in \mathcal{S}$ uniformly at random and go to **Step 1**;
  end
end

We define two methods:

1. **CL-OMPR+**: This method corresponds to CL-OMPR where (**Step 1**) (maximization of the correlation function) is handled using Algorithm 5. The set of initializations $\mathcal{S}$ is chosen as described in Section 3.3.2, and the threshold value is set to $T = \infty$. In other words, this method does not include the detection of spurious local maxima.

2. **CL-OMPR++**: This method builds on CL-OMPR+ by introducing a threshold T, as detailed in Section 3.3.3. It is specifically designed for cases where the data is well-clustered, the number of samples per cluster is balanced, and the clusters have similar sizes.
Note that, Algorithm 5, which incorporates the detection of spurious local maxima, can indicate whether a good local maximum has been reached. This information allows us to limit the number of iterations of CL-OMPR. Specifically, the algorithm terminates once K good local maxima have been identified, speeding up the process.

We will now perform and analyze the results of numerical simulations. The analysis will be conducted on synthetic data and on real data.
The set of initializations $\mathcal{S}$ is a solution to the covering problem, as described in Section 3.3.2. We will use a naive solution to this problem, i.e., the upper bound presented in Section 3.1.3. The threshold T for CL-OMPR++ is set to $T = \frac{1+\epsilon}{2} \cdot \frac{1}{K} \left(\frac{\sigma^2}{\sigma^2 + \bar{\sigma}^2_{\mathcal{X}}} \right)^{\frac{d}{2}}$, with $\epsilon = 0.3$, and $\bar{\sigma}^2_{\mathcal{X}}$ is given by the heuristic in [2] (see Appendix A for the details of the method).

For every simulation, a success is defined as $\text{MSE}(\boldsymbol{\theta}_\Delta; \mathcal{X}) \leq 1.2 \times \text{MSE}(\tilde{\boldsymbol{\theta}}; \mathcal{X})$, with $\boldsymbol{\theta}_\Delta$ the centroids found with the considered method and $\tilde{\boldsymbol{\theta}}$ the centroids found with the Lloyd algorithm.

Synthetic data.
We place ourselves in ideal conditions for CL-OMPR++. Specifically, we consider a dataset $\mathcal{X} := \{\boldsymbol{x}_i \in \mathbb{R}^d \mid i = 1, \ldots, n\}$ with $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \sim_{\text{i.i.d.}} \sum_{k=1}^{K} \frac{1}{K} \mathcal{N}(\boldsymbol{\mu}_k, \sigma^2_{\mathcal{X}} \mathbf{I}_d)$. We consider $K = 5$ Gaussians in dimension $d = 5$, with $n = 10^5$ samples. Results are shown in Figure 3.8.

Real data.
The dataset used in this analysis is the CelebA dataset [16], a large-scale dataset of celebrity images commonly used in facial attribute recognition and feature extraction tasks. The facial embeddings were extracted using pre-trained InsightFace models [17, 18]. Specifically, we randomly selected 5 celebrities from the dataset. To further reduce the complexity of the data, we performed a Principal Component Analysis (PCA) on the selected embeddings, reducing their dimensionality to $d = 4$. Results are shown in Figure 3.9.

Interpretations

The results are conclusive, showing a clear extension of the effective operating range of the CL-OMPR algorithm with respect to the choice of the bandwidth σ when the algorithm is paired with our proposed methods, i.e., CL-OMPR+ and CL-OMPR++. These improvements are observed for both synthetic and real data.

We observe that the number of initializations required by the proposed method tends to significantly overestimate the number needed for low values of σ. Indeed, this can be seen by comparing CL-OMPR+ and CL-OMPR with L initializations. The number of initializations for CL-OMPR+ is considerably higher than that of CL-OMPR with L initializations within their respective operating ranges. This

effect is due to several factors. In constructing the set of initializations $\mathcal{S}$, we considered the worst-case scenario where the covariance matrices of the data clusters are 0, which is not the case in practice. Moreover, the minimal number of initializations in the set $\mathcal{S}$ is the solution to a covering problem. We considered a naive solution to this problem, resulting in a larger number of initializations. This solution can be further refined.
However, we can observe that the increase in the number of initializations compared to CL-OMPR with a single initialization ($L = 1$) aligns well with the decline in the performance of the latter. The proposed method effectively captures situations where a broader exploration of the domain is necessary.

CL-OMPR++ significantly reduces the average number of initializations compared to CL-OMPR+ while improving the operating range. This is due to the detection of good local maxima achieved using the threshold T (Section 3.3.3).
Moreover, CL-OMPR++ requires fewer initializations than any other method with equivalent performance and provides the widest operating range, along with CL-OMPR+.
We observe that, unlike CL-OMPR+, where the number of initializations consistently increases as σ decreases, CL-OMPR++ exhibits oscillations. Specifically, for very low values of σ, when even CL-OMPR+ fails to perform well, we observe a low number of initializations that increases until both CL-OMPR++ (and CL-OMPR+) reach effective performance. This behavior is due to the fact that, for very small σ, the spurious local maxima become indistinguishable from the true ones (see Figure 2.1 for a better visual interpretation).
Then, the number of initializations decreases until a certain value of σ, where it starts to rise again (Figure 3.8b). This behavior is due to the fact that the method is tailored for sufficiently low values of σ. In these situations, the number of initializations required by CL-OMPR++ is similar to that of CL-OMPR+.

These compelling observations, particularly for CL-OMPR++, should nevertheless be interpreted with caution. Indeed, these experiments were conducted under optimal conditions for CL-OMPR++ with respect to the synthetic data. The real data is also well-clustered, with balanced and relatively similar-sized clusters, creating an ideal scenario for CL-OMPR++.
However, they offer a perspective for designing a decoder that is both computationally efficient and easy to tune.

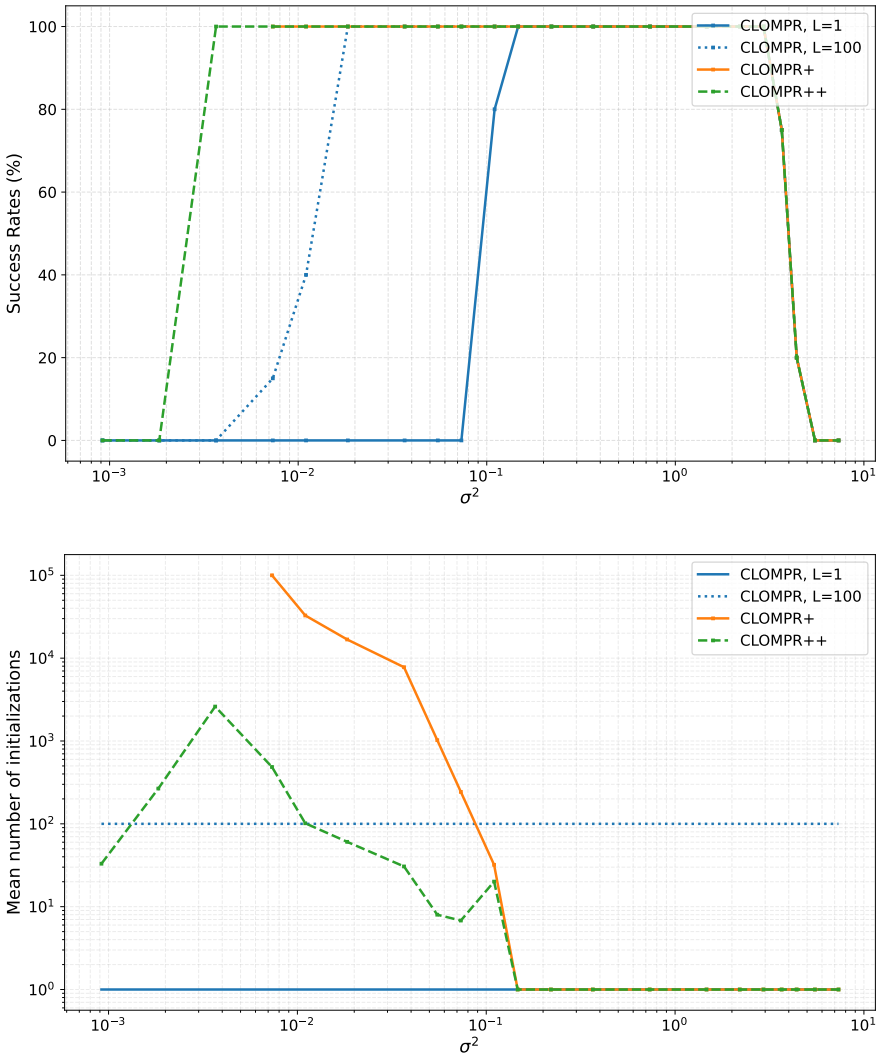


Figure 3.8: **Top**: Comparison of the empirical success rates between CL-OMPR with L initializations ($L \in \{1, 10, 100\}$), CL-OMPR+, and CL-OMPR++. The results represent the mean over 20 random realizations of the sketching operation. The sketch size is fixed ($m = 20Kd$) while the bandwidth σ varies. **Bottom**: Comparison of the mean number of initializations for each method. Note that only CL-OMPR++ has a varying number of initializations for a given σ. The data is drawn from a mixture of $K = 5$ Gaussians and consists of $n = 10^5$ samples in $d = 5$.

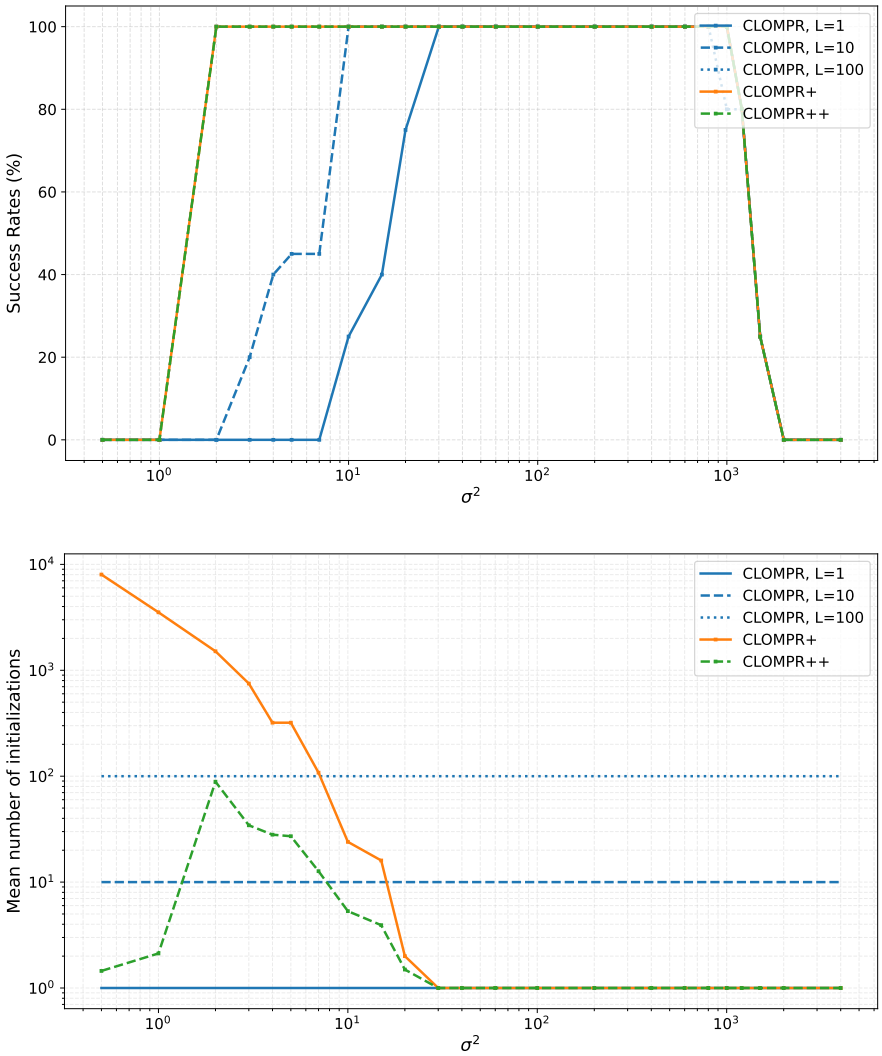


Figure 3.9: **Top**: Comparison of the empirical success rates between CL-OMPR with L initializations ($L \in \{1, 10, 100\}$), CL-OMPR+, and CL-OMPR++. The results represent the mean over 20 random realizations of the sketching operation. The sketch size is fixed ($m = 20Kd$) while the bandwidth σ varies. **Bottom**: Comparison of the mean number of initializations for each method. Note that only CL-OMPR++ has a varying number of initializations for a given σ. The data is derived from the CelebA dataset [16].

Conclusion

CL-OMPR is a greedy algorithm introduced by the authors of [2] to address the nonconvex compressive clustering problem. This approach seeks to solve the K-means clustering problem using only a sketch of the data—a low-dimensional vector that captures its essential information. Such a method can be particularly useful in various tasks where data efficiency is crucial, such as memory-constrained scenarii, distributed implementations, streaming contexts, or settings with strict privacy constraints.

By investigating CL-OMPR, we observed that the algorithm could fail to recover the optimal clustering parameters in certain scenarii. In this work, we aimed to prevent such failures by improving a nonconvex optimization step within CL-OMPR.

The proposed methods, CL-OMPR+ and CL-OMPR++, demonstrate promising results and notably extend the operating range of CL-OMPR for a parameter that is recognized as challenging to tune by the community [2, 14].
The results of CL-OMPR++, which is capable of distinguishing between good and bad maxima of the nonconvex criterion to be maximized, offer a perspective for designing a decoder that is both easy to tune and computationally efficient.

However, this method is designed to operate on specific datasets, with required characteristics that are relatively uncommon in practical applications. Adapting CL-OMPR++ to more general situations could be a subject of future work. We have already proposed some potential directions in Section 3.3.3.

Bibliography

[1] V. Schellekens, "Pycle: a python compressive learning toolbox (v1.0.0)," 2020. Available at: `https://doi.org/10.5281/zenodo.3855114`.

[2] N. Keriven, A. Bourrier, R. Gribonval, and P. Pèrez, "Sketching for large-scale learning of mixture models," *Information and Inference: A Journal of the IMA*, vol. 7, no. 3, pp. 447–508, 2018.

[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," in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035, SIAM, 2007.

[6] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. Springer, May 2012.

[7] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[8] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proceedings of the 27th Asilomar Conference on Signals, Systems and Computers*, pp. 40–44, IEEE, 1993.

[9] E. J. Candès and C. Fernandez-Granda, "Towards a mathematical theory of super-resolution," *Communications on Pure and Applied Mathematics*, vol. 67, no. 6, pp. 906–956, 2014.

[10] K. Bredies and H. K. Pikkarainen, "Inverse problems in spaces of measures," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 19, no. 1, pp. 190–218, 2013.

[11] N. Keriven, N. Tremblay, Y. Traonmilin, and R. Gribonval, "Compressive K-means," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6369–6373, IEEE, 2017.

[12] A. Rahimi and B. Recht, "Random features for large scale kernel machines," in *Proceedings of the Neural Information Processing Systems Conference (NeurIPS)*, 2007.

[13] V. Schellekens, *Extending the Compressive Statistical Learning Framework: Quantization, Privacy, and Beyond*. PhD thesis, 06 2021.

[14] A. Belhadji and R. Gribonval, "Sketch and shift: a robust decoder for compressive clustering," 2024.

[15] L. Armijo, "Minimization of functions having lipschitz continuous first partial derivatives," *Pacific Journal of Mathematics*, vol. 16, no. 1, pp. 1–3, 1966.

[16] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[17] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Insightface: A series of 2d and 3d face analysis projects.," 2018.

[18] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4690–4699, 2019.

Appendices

Appendix A

Additional algorithms

Here, we present the heuristic proposed in [2] for estimating the average variance of a dataset.

Consider a database $\mathcal{Z} = \{\boldsymbol{z}_i \in \mathbb{R}^d \mid i = 1, \ldots, n\}$ that we aim to sketch. Instead of using the entire database, a small subset $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_0}\}$ with $n_0 \ll n$ is selected to determine a suitable frequency distribution through a lightweight procedure, which precedes the actual sketch computation.
The goal is to estimate the average variance

$$\bar{\sigma}^2 = \frac{1}{kd} \sum_{l=1}^{k} \sum_{i=1}^{d} \sigma_{l,i}^2,$$

where $\sigma_{l,1}^2, \ldots, \sigma_{l,d}^2$ are the eigenvalues of Σ_l. This parameter, $\bar{\sigma}^2$, may differ significantly from the global variance of the dataset, particularly in cases where components are well-separated with small individual variances. To estimate $\bar{\sigma}^2$, a lightweight sketch $\hat{\boldsymbol{y}}_0$ with m_0 frequencies is computed using n_0 data points, and a frequency distribution corresponding to a single isotropic Gaussian $\Lambda(\cdot)$ is then employed.
If the variances $\sigma_{l,1}^2$ are relatively similar, the amplitude of the empirical characteristic function

$$\left| \frac{1}{n_0} \sum_{i=1}^{n_0} e^{j\boldsymbol{w}^T \boldsymbol{z}_i} \right|$$

approximately follows $e^{-\frac{1}{2}\|\boldsymbol{w}\|_2^2 \bar{\sigma}^2}$, exhibiting high oscillations. This enables a simple amplitude estimation: by ordering the m_0 frequencies used in $\hat{\boldsymbol{y}}_0$ by increasing norm, dividing the sketch into blocks, and identifying the modulus peaks within each block, a curve is formed that closely follows $e^{-\frac{1}{2}R^2\bar{\sigma}^2}$. A basic regression is then applied to estimate $\bar{\sigma}^2$.
Since the range of frequencies required to compute $\hat{\boldsymbol{y}}_0$ is not known beforehand,

the procedure starts with an initialization $\bar{\sigma}^2 = 1$. It is iteratively refined by drawing m_0 frequencies adapted to the current estimate of $\bar{\sigma}^2$ based on an updated frequency distribution $\Lambda(\cdot)$. After a few iterations, the process converges.
A detailed summary of this procedure is provided in Algorithm 6.

Algorithm 6: EstimMeanSigma($\mathcal{Z}, m_0, c, T$): Estimation of the mean variance $\bar{\sigma}^2$

Input: Small training dataset $\mathcal{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_0}\}$, small number of frequencies m_0, number of blocks $c \in \mathbb{N}_+^*$, number of iterations T
Output: Estimated mean variance $\bar{\sigma}^2$
Initialize: $\bar{\sigma}^2 \leftarrow 1$;
for $t \leftarrow 1$ **to** T **do**
  Draw frequencies: $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{m_0} \sim_{\text{i.i.d.}} \mathcal{N}(0, (\bar{\sigma}^2 \mathbf{I}_d)^{-1})$;
  Sort frequencies: $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{m_0}\}$ by increasing radius $\|\boldsymbol{w}_j\|_2$;
  Compute empirical sketch: $\hat{\boldsymbol{y}}_0 \leftarrow \left[\frac{1}{n_0} \sum_{i=1}^{n_0} e^{j \boldsymbol{w}_j^T \boldsymbol{z}_i}\right]_{j=1}^{m_0}$;
  Divide sketch into blocks and find peaks:;
  $s \leftarrow \lfloor m_0 / c \rfloor$;
  for $q \leftarrow 1$ **to** c **do**
    $j_q \leftarrow \arg\max_{j \in [(q-1)s+1, qs]} |\hat{\boldsymbol{y}}_{0,j}|$;
  end
  Update:;
  $\hat{\boldsymbol{e}} \leftarrow [\hat{\boldsymbol{y}}_{0,j_q}]_{q=1}^{c}$;
  $\bar{\sigma}^2 \leftarrow \arg\min_{\sigma^2 > 0} \|\hat{\boldsymbol{e}} - [e^{-\frac{1}{2} R_{j_q} \sigma^2}]_{q=1}^{c}\|_2$;
end

Appendix B

Mathematical Derivations

We derive here the Kernel Mean Embedding of a mixture of isotropic Gaussians : We consider that the data $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^d | i = 1, ..., n\}$ is drawn i.i.d. from a mixture of Gaussians:

$$\boldsymbol{x}_i \sim \sum_{k=1}^{K} \alpha_k \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k), \quad \text{with } \Sigma_k = \sigma_k^2 \mathbf{I}_d$$

The correlation function is

$$f_W(\boldsymbol{c}) = \Re \langle \mathcal{A}(\delta_c), \boldsymbol{z}_\mathcal{X} \rangle \tag{B.1}$$

And approximates the Kernel Mean Embedding f_{KME} when $m \to \infty$ and $n \to \infty$.

Recall that the sketch $\boldsymbol{z}_\mathcal{X}$ is defined as:

$$\boldsymbol{z}_\mathcal{X} = \mathcal{A}(\hat{p}_\mathcal{X}) = \left[\frac{1}{n} \sum_{i=1}^{n} e^{j \boldsymbol{w}_l^T \boldsymbol{x}_i} \right]_{l=1}^{m},$$

where $\boldsymbol{x}_i \sim p_\mathcal{X} = \sum_{k=1}^{K} \alpha_k \mathcal{N}(\boldsymbol{\mu}_k, \sigma_k^2 \mathbf{I}_d)$ and $\boldsymbol{w}_l \sim \mathcal{N}(0, \sigma^{-2} \mathbf{I}_d)$.
Since we consider $n \to \infty$ and the operator $\mathcal{A}$ is linear with respect to probability distributions, we can write:

$$\boldsymbol{z}_\mathcal{X} = \mathcal{A}(p_\mathcal{X}) = \left[\sum_{k=1}^{K} \alpha_k \cdot \mathbb{E}_{\boldsymbol{x}_i^{(k)} \sim p_\mathcal{X}^{(k)}} \left[e^{j \boldsymbol{w}_l^T \boldsymbol{x}_i^{(k)}} \right] \right]_{l=1}^{m}$$

The characteristic function of a random variable Y is defined as $\varphi_Y(t) = \mathbb{E}\left[e^{jtY}\right]$.
For a multivariate normal distribution, we have:

$$\varphi_Y(\boldsymbol{t}) = e^{j \boldsymbol{t}^T \boldsymbol{\mu} - \frac{1}{2} \boldsymbol{t}^T \Sigma \boldsymbol{t}}$$

Using this property, we can write:

$$\boldsymbol{z}_{\mathcal{X}} = \left[\sum_{k=1}^{K} \alpha_k \cdot e^{j\boldsymbol{w}_l^T \boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{w}_l^T \sigma_k^2 \mathbf{I}_d \boldsymbol{w}_l}\right]_{l=1}^{m} = \left[\sum_{k=1}^{K} \alpha_k \cdot e^{j\boldsymbol{w}_l^T \boldsymbol{\mu}_k - \frac{1}{2}\sigma_k^2 \|\boldsymbol{w}_l\|^2}\right]_{l=1}^{m}$$

Substituting in B.1, we get

$$f_W(\boldsymbol{c}) = \Re\left(\sum_{k=1}^{K} \alpha_k \cdot \frac{1}{m} \sum_{l=1}^{m} e^{j\boldsymbol{w}_l^T(\boldsymbol{\mu}_k - \boldsymbol{c}) - \frac{1}{2}\sigma_k^2 \|\boldsymbol{w}_l\|^2}\right)$$

As $m \to \infty$, we can write:

$$f_{\text{KME}}(\boldsymbol{c}) = \sum_{k=1}^{K} \alpha_k \cdot \Re\left(\mathbb{E}_{\boldsymbol{w}_l \sim \mathcal{N}(0, \sigma^{-2}\mathbf{I}_d)}\left[e^{j\boldsymbol{w}_l^T(\boldsymbol{\mu}_k - \boldsymbol{c}) - \frac{1}{2}\sigma_k^2 \|\boldsymbol{w}_l\|^2}\right]\right)$$

To develop the term $\mathbb{E}_{\boldsymbol{w}_l \sim \mathcal{N}(0, \sigma^{-2}\mathbf{I}_d)}\left[e^{j\boldsymbol{w}_l^T(\boldsymbol{\mu}_k - \boldsymbol{c}) - \frac{1}{2}\sigma_k^2 \|\boldsymbol{w}_l\|^2}\right]$, we proceed as follows:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{w}_l \sim \mathcal{N}(0, \sigma^{-2}\mathbf{I}_d)}\left[e^{j\boldsymbol{w}_l^T(\boldsymbol{\mu}_k - \boldsymbol{c}) - \frac{1}{2}\sigma_k^2 \|\boldsymbol{w}_l\|^2}\right] &= \int_{\mathbb{R}^d} e^{j\boldsymbol{w}_l^T(\boldsymbol{\mu}_k - \boldsymbol{c}) - \frac{1}{2}\sigma_k^2 \|\boldsymbol{w}_l\|^2} p(\boldsymbol{w}_l)\, d\boldsymbol{w}_l \\
&= \int_{\mathbb{R}^d} e^{j\boldsymbol{w}_l^T(\boldsymbol{\mu}_k - \boldsymbol{c}) - \frac{1}{2}\sigma_k^2 \|\boldsymbol{w}_l\|^2} \cdot \left(\frac{\sigma^2}{2\pi}\right)^{d/2} e^{-\frac{\sigma^2 \|\boldsymbol{w}_l\|^2}{2}} d\boldsymbol{w}_l \\
&= \left(\frac{\sigma^2}{\sigma^2 + \sigma_k^2}\right)^{d/2} \cdot e^{-\frac{1}{2}\frac{\|\boldsymbol{\mu}_k - \boldsymbol{c}\|^2}{\sigma_k^2 + \sigma^2}}.
\end{aligned}$$

Reinserting the result into $f_{\text{KME}}(\boldsymbol{c})$, we obtain:

$$f_{\text{KME}}(\boldsymbol{c}) = \sum_{k=1}^{K} \alpha_k \cdot \left(\frac{\sigma^2}{\sigma^2 + \sigma_k^2}\right)^{d/2} \cdot e^{-\frac{1}{2}\frac{\|\boldsymbol{\mu}_k - \boldsymbol{c}\|^2}{\sigma^2 + \sigma_k^2}}$$

This concludes the derivation.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**