UCLouvain
Institute for Information
and Communication Technologies,
Electronics and Applied Mathematics

Extending the Compressive Statistical Learning Framework: Quantization, Privacy, and Beyond

Vincent Schellekens

Thesis submitted in partial fulfillment
of the requirements for the degree of
Ph.D. in Engineering Sciences

Dissertation committee:

Prof. Laurent Jacques (UCLouvain, advisor)
Prof. Christophe de Vleeshouwer (UCLouvain)
Prof. David Bol (UCLouvain, chair)
Dr. Rémi Gribonval (ENS de Lyon, France)
Prof. Mike Davies (University of Edinburgh, United Kingdom)

Version of 08/06/2021.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor, Laurent Jacques. The best advice a PhD-student-to-be can get is probably "don't chose the topic, chose the advisor", and personally, I could not have made a better choice. Laurent is not only a model of scientific rigor and integrity; a creative mind with a deep knowledge on an incredibly vast amount of topics (with an interesting question ready at the end of any talk, no matter the topic); but most importantly, an understanding, generous, and supportive mentor.

I am also very grateful to Rémi Gribonval: for accepting to be on my thesis committee, for our fruitful and pleasant interactions, for kindly hosting me for a week in Rennes, and inviting me for a longer stay in Lyon (I still deeply regret not being able to make it due to the pandemic).

I would also like to thank the other members of my very supportive thesis committee, Christophe de Vleeshouwer and David Bol. I specifically thank Christophe for inviting me to his working lunches so that I could learn more about deep learning and have interesting discussions. For his dedication to the greater good in the rarely humane world of technology and academia, David is an inspiring role model to me.

I also thank Mike Davies for accepting to be a part of the jury of my thesis, and for giving me the opportunity to present results to his research group.

I thank all the people I had the chance to collaborate with, and in particular Antoine Chatalic (for our numerous exchanges on compressive learning and for making sure I felt at home while I was in Rennes) and Florimond Houssiau (for introducing me to the world of privacy).

This thesis was funded by the F.R.S.-FNRS, which has my gratitude for giving me "la liberté de chercher". It would also not have been possible

without the the administrative and technical staff at UCLouvain and the ICTEAM institute in particular.

I would like to express my appropriately formal gratitude to my many brolleagues, that made the workplace so welcoming I was eager to go take the bus every morning, *i.e.*, my office mates, Pierre-Yves and François (whom I learned a lot from in my first days as a researcher), Beni (whom I learned a lot from too, although on maybe on less work-related matters; I will mess benistractions!), Benoît (enjoy your nice office on the north side!), and Olivier (which I regret not seeing more); the other members of Laurent's team, Valerio (gone by the time I arrived but whose help was precious when writing the grant application for this thesis), Kévin, Chunlei, Amir (except for that time when he humiliated me at the Citadelles board game), Stéphanie (whom I am particularly grateful to for pointing me towards Laurent's door when I was looking for an advisor), Thomas (a special thank for all the gossiping), Gilles, Alexander, Rémi; my teaching duties associates, Simon, Maxime, Gabriel (for Image Processing) as well as Cécile, Florian, Lucas (for Signals and Systems); the remaining dolphins, Anne-Sophie, Antoine, Alice, Victor, Niels, and more generally everyone that I had the chance to meet at the ICTEAM institute.

I have the incredible chance to have a lot of good friends that supported me during this thesis, and I would also like to thank them all for that. They will recognize themselves so I will not name them all here, and rather buy them a beer (or more) in good time, but let me just mention a special thanks to Oliver, my unofficial office mate during confinement times, and Antoine, Louis, Florimond and Andine who took the time to proofread some parts of this thesis.

I also thank my parents, brothers and sisters, and in-laws for their unconditional support. And of course, my deepest thanks are for my wife Anne-Sophie, who was with me at every step of this journey.

Abstract

Over the last few years, machine learning—the discipline of automatically fitting mathematical models or rules from data—revolutionized science, engineering, and our society. This revolution is powered by the ever-increasing amounts of digitally recorded data, which are growing at an exponential rate. However, these advances do not come for free, as they incur important computational costs, such as memory requirements, execution time, or energy consumption. To reconcile learning from large-scale data with a reasoned use of computational resources, it seems crucial to research new learning paradigms.

A particularly promising candidate is the compressive statistical learning framework. In a nutshell, the idea of this method is to first compress the learning data, in an efficient manner, as lightweight sketch vector (given by random feature moments of the data). The desired learning methods are then carried out using only this sketch, instead of the full dataset, which can sometimes save orders of magnitude of computational resources.

This thesis broadens the scope of the compressive learning framework by exploring three extensions of it. First, the quantization of sketch contributions is studied, which allows to further reduce the computational burden associated with computing the sketch. Second, the addition of a privacy-protecting layer on top of the sketch is considered, which allows to learn from the sketch while ensuring the privacy of the data contributors. Finally, generalizations of the framework to novel tasks are discussed.

Contents

1

Introduction

"WE ARE DROWNING IN INFORMATION, while starving for wisdom." When E. O. Wilson, myrmecologist[1] and popular scientific writer, wrote those words back in 1998 [Wil98], I doubt that he anticipated just how much this quote (and its many variants) would grow in popularity during the following two decades; contributing, in an ironic twist, to the flood it denounces by prompting new blog posts and social media shares every day. It is however not surprising that this feeling of "drowning in information" resonates so strongly with all of us today. Indeed, thanks to numerous technological advancements, we are able to record an incredibly wide variety of "raw information" in the form of *data*. From classic human-readable signals (images, sound, video, multispectral volumes, etc.) to highly structured ones (textual corpora, DNA sequences, molecular arrangements, social networks, etc.), from macroscopic quantities monitored across the globe (cosmological observations, financial markets, meterological recordings such as temperature, pressure, wind, etc.) to the most detailed accounts of our individual lives (geolocalization, movie preferences, buying habits, online behavior, etc.), the list could go on forever. Over the last few years, *the volume of data* that humankind produces on a daily basis *has drastically increased*, and the trend does not look like slowing down any time soon.

[1] Myrmecology, subfield of biology, is the study of ants; it is not related to this thesis at all.

This phenomenon, also known by the buzzword "*big data*", can be explained by a multitude of factors. The new availability of data-recording devices that are affordable to the general public immediately comes to mind, the main ambassador of this trend being the smartphone, cramped with diverse sensors and cameras [DKBM21]. Another part of the answer is certainly the rise of the internet (and social media in particular), which made a perpetual consumption—and generation—of data a crucial part of our lives [VD20]. Beside these and other "extrinsic" explanations (which explain the availability of new data acquisition modalities), I believe there is an even more important driving force behind our newfound obsession with data. Indeed, the fact that *the intrinsic value of data has drastically increased* creates strong incentives (*e.g.*, for commercial, military, or scientific reasons) for companies and institutions to invest in massive data collection.

This increased valuation of data is a consequence of important advances in the field of *machine learning*, which aims at transforming raw data into useful insights and decisions. Machine learning (ML) itself is not a new phenomenon; it has been an active research topic since the fifties (Alan Turing described "learning machines" in his seminal paper[2] as early as 1950 [Tur50]). However, the smashing success of *deep learning* around 2010 sparked massive interest in the field, attracting ever-increasing amounts of interest, funding, and researchers. As a result, the advances in ML have accelerated considerably over the last few years. Machine learning agents have become very good at extracting valuable knowledge from data, often surpassing human performance.

Large-scale machine learning (*i.e.*, the synergy between advances in big data and machine learning) led to scientific and engineering breakthroughs that were previously undreamed of—but these advances do not come for free, as they incur important *computational costs*. This claim is developed in **Section 1.1**, which paints a broad view of the large-scale ML landscape.

In order to realign large-scale machine learning with a more reasoned use of computational resources, it seems crucial to research new learning paradigms. Among others, the recent *compressive statistical learning* framework [GBKT17,GCK+20], also called Compressive Learning (CL) for short, is a promising candidate. In a nutshell, the idea is to "compress" the learning data, in an efficient manner, as a single *sketch* vector of moderate size.

[2]Many of Turing's ideas from that paper, such as the famous "Turing test", stood the test of time remarkably well. I cannot resist including a quote from it that should resonate with most of machine learning researchers today: "*An important feature of a learning machine is that its teacher will often be very largely ignorant of quite what is going on inside, (...)*".

The desired learning would then be carried out using only this sketch instead of the full dataset, which has the potential to save massive amounts of computational resources. **Section 1.2** further explains this framework[3] and why it is promising for tackling some of the challenges posed by large-scale machine learning.

This thesis is inscribed in the compressive learning framework. In particular, it explores several extensions of it: quantization of the sketch contributions, adding a privacy-protecting layer on top of the sketching mechanism, and a tentative generalization to novel tasks and sketching techniques. **Section 1.3** briefly describes those contributions, and explains how they are organized into the next chapters.

1.1 The big picture of "big data"

Over the last few years, machine learning—the discipline of automatically fitting mathematical models or rules from data—revolutionized science, engineering, and our society. This recency could maybe seem surprising. Indeed, storing data is not a new idea; in fact, humanity started to record numeric quantities more than five thousand years ago[4]. Taking algorithmic decisions from data is not a new idea either (although significantly more recent); for example, Moritz Hardt and Benjamin Recht begin their historic overview of machine learning around 1690 with the story of Halley's life table [HR21]. The earliest "machine learning" technique that is still applied today, the least-squares method, can be traced back to Legendre [Leg06] in the early 19th century. And as argued above, with its birth in the 1950s, the specific ideas of letting *machines* learn from data is almost as old as general-purpose computers. Contrasting this decades-old evolution with the new-fashioned explosion of ML in practical applications begs the question: *what changed*?

1.1.1 Large-scale machine learning: ingredients for modern success

Ingredient 1: more hardware An obvious but crucial difference between data-driven algorithms from the fifties and from today is the *exponential increase in raw computation power* (*e.g.*, following Moore's "law" [Mac11], but also the development of "dedicated" hardware such as GPUs [SBS05]),

[3] For now, we settle for a high-level description, postponing technical details to Chapter 2.

[4] See the fascinating Wikipedia entry on the subject: `https://en.wikipedia.org/wiki/History_of_writing`.

which allows machines to perform an ever-increasing amount of operations for a given "budget" (*i.e.*, for a given computation time or monetary amount). As we will see in the other two ingredients, the ability to perform enormous amounts of computations is a *sine qua non* prerequisite for the success of modern large-scale machine learning.

Ingredient 2: more layers The most successful machine learning models today, such as deep neural networks, are characteristic in that they involve *several thousands (sometimes up to billions) of tunable parameters*[5].These last few years, the general rule of thumb that dominates machine learning practice is that the more parameters you can train, the better your model will be. One of the hottest topics in the field is the "double descent" phenomenon (see *e.g.*, [NKB+19]), which shows that, contrary to previous knowledge, the test error of ML models (sometimes) decreases in over-parameterized regime (*i.e.*, when the number of parameters is larger than a critical "interpolation threshold").

Ingredient 3: more data This thesis opened with the claim that the increase of data collection was motivated by the rise of machine learning. But in turn, one could argue that modern machine learning models—deep neural networks in particular—also *require* massive amounts of data to be trained properly in the first place. To provide an engineering metaphor, this symbiotic relationship between data and models is similar to the one between fuel and combustion engines: on one hand, the continuous invention of more sophisticated and ambivalent engines requires to acquire ever-increasing amounts of fuel; on the other hand, the availability of fuel in larger quantities facilitates the invention of more complex (and power-hungry) engines. Both trends reinforce each other.

As it happens, the amount of digitally recorded data—*i.e.*, machine learning fuel—has *increased exponentially* over the last few years. A popular statistic online, often attributed to IBM, is that 90% of today's data was produced in the last two years. Other numbers that are often shared online are that in 2020, every second 1.7MB of data is produced *per person*[6], collectively adding up to a total of 44ZB of digital data stored in total[7] (a zettabyte (ZB) is 10^{21} bytes). While those values and other wild guesses are to be taken with a grain of salt (they should give, at best, a rough estimate

[5] https://www.speechmatics.com/blog/machine-learning-is-getting-big-part-i/
[6] https://www.domo.com/learn/data-never-sleeps-6.
[7] https://www.raconteur.net/infographics/a-day-in-data/.

of the orders of magnitude), there is no doubt that we are deep into the so-called Zettabyte Era[8] [Xu14]. Obviously, not all of this recorded data is suitable for learning, but it seems reasonable to postulate that the amount of usable data has increased at least proportionally.

1.1.2 The computational pricetag

As a result, large-scale machine learning models, which rely on massive amounts of data and computation power to train gigantic models, now perform extremely well (often exceeding human accuracy), and are already a crucial part of our lives [HEM19]. Given those impressive feats, large-scale machine learning seems to be a marvel of human engineering, and we should eagerly await the new wonders it will bring us as the trend continues. However, those wonders will not come for free, and the pricetag might somewhat dissipate our eagerness.

Indeed, training from ever larger quantities of data requires ever more *computational resources*: see, for example, one of the many studies on the subject [AJYM+15,QWD+16,LGEC17,SGM19,TGLM20,BGMMS21]. Alarmingly, as detailed in [TGLM20], although modern ML models keep improving in "performance" (according to a given metric such as classification accuracy or F1 score), *the computational cost*[9] *grows superlinearly with this performance metric* (*i.e.*, a polynomial of high degree, or even exponentially). This means each time the performance of ML on some benchmark improve by a fixed amount, the cost to reach the next improvement (by the same amount) drastically increases.

Why computational costs matter In most research areas, we measure the "computational cost" of an algorithm by the *time and space complexities* of an algorithm, *i.e.*, how the number of operations and memory elements increase with the size of the problem. These metrics are convenient because they are relatively straightforward to analyze, and are (by definition) relatively universal (*i.e.*, they do not depend on the implementation, language, machine they run on, etc.).

However, it is important to keep in mind what computational costs imply in practice. For example, a large computational cost also obviously implies a large *monetary price* (amount of money that training the model will cost). To give an extreme example, it is estimated [SGM19] that train-

[8] `https://en.wikipedia.org/wiki/Zettabyte_Era`.
[9] Here evaluated in GigaFLOPs.

ing a modern state-of-the-art large-scale model can cost up to hundreds of thousands of dollars. Such high costs can create a "gatekeeping" effect, which prevents smaller institutions to keep up with the evolution of machine learning, and conversely gives a lot of influence to large-scale institutions (such as Google) that are able to cover such expenses [AW20].

Another "bottom line" cost of crucial interest is the *energy consumption* of training models, which can typically be decomposed as the product of the training time and the power consumption of the required hardware. Of course, this energy is billed in the form of money to the company training the model (as explained above), but is also billed in the form of carbon emissions to the whole planet. To give another example from [SGM19] the authors estimate that the training of one modern model can be assimilated to the CO_2 emissions of five cars over their whole lifetime.

Although the exact computational costs are arguably difficult to estimate accurately (the numbers provided here are only rough estimates), the main takeaway seems clear: there is a strong interest in researching lightweight alternatives to large-scale machine learning, maybe trading off some accuracy for reduced amounts of computations. This is the main motivation of compressive learning, as described in the next section.

Remark 1.1 (Other impacts of large-scale ML). We focused on the *computational* cost because it is the most relevant for this thesis, but this is not the only "cost" that is being paid to enable modern large-scale machine learning. One can cite, for example *privacy infringements* (discussed in Chapter 5), *bias and fairness* issues [BG18, BGMMS21], and *ethical dilemmas* [HEM19], which all impact our society as a whole. Although a complete discussion of these issues exceeds by far the scope of this thesis, the main takeaway is that modern ML practices created a crowd of unique and diverse, yet pressing and daunting, challenges.

1.2 Compressive Learning, a possible solution?

The previous section makes the case that there is a need for new, computationally efficient, learning paradigms. One particularly promising candidate, which is explored in this thesis, is the *compressive statistical learning* (CL) framework [GBKT17]. In a nutshell, the main specificity of CL is that it does not learn from the large-scale dataset directly but rather from a lightweight summary—called *sketch*—constructed by aggregating *random feature moments* of the data.

To be more specific, the CL workflow breaks down the expensive learning operation into two efficient sub-steps. In a first *sketching phase*, the dataset is compressed into a small sketch vector, defined as the average of nonlinear random features of the dataset. Crucially, this compression requires only a single pass over the dataset, which ensures that its computation scales favorably with large-scale datasets.

Then, during the *learning phase*, the desired machine learning model is extracted from the sketch alone. Since the sketch is typically much smaller than the full dataset, this step is potentially orders of magnitude cheaper (*e.g.*, in terms of required memory and computation time) than learning from the dataset directly.

Compressive learning is a promising framework but is a relatively recent topic: it has not yet been researched intensively, and has seen limited applications in practice. The aim of this thesis is to contribute to the development of this field by several contributions, each broadening the scope of CL in unique ways.

1.3 Outline and contributions

This thesis extends the compressive learning framework along three different axes, namely, in rough order of importance: quantization of the sketch contributions, adding a privacy-preserving layer to the sketch, and tentative generalizations of the CL framework to other tasks. The organization of those contributions into the present manuscript is explained below.

Part 0: Solid foundations. This preliminary part comprises two chapters. **Chapter 1**, which the reader is currently looking at, introduced the broader context of compressive learning. An in-depth description of compressive learning is then given in **Chapter 2**. To do so, that chapter first provides a tutorial-style overview of the many disciplines supporting compressive learning, which serves as justification to gently introduce many of the tools and notations that will be needed afterwards, *e.g.*, random features, compressive sensing, general sketching methods, and metrics on the space of probability measures.

Part I: Quantized Sketches. This part, which could be seen as the main contribution of this thesis, tackles the question of *quantizing the sketch contributions*. In fact, as will become clearer, the obtained results apply to a broader framework, coined "asymmetric" random features and CL.

A first key result is presented in **Chapter 3**. This chapter is a bit apart from the others in that it does not consider sketching (*i.e.*, averaged random features) yet, but the individual features (without averaging) instead. More particularly, this chapter shows that, when two signals are to be compared through their random periodic features (a generalization of random Fourier features [RR08]), it is sometimes beneficial to use *different feature maps* for the two signals of interest. In particular, it is possible to *quantize* one of the two random feature maps while still recovering the same kernel that one would obtain without this quantization. In this chapter, a particular effort is devoted to proving guarantees holding for an infinite signal set while at the same time dealing with the possibly discontinous nature of features (as in quantized features). Those guarantees serve as one of the main building blocks of the next chapter.

Then, in the following **Chapter 4**, we study the so-called quantized sketching scheme, where the sketch contributions of the individual data vectors are binarized. Among others, the interest of this approach lies in the potential to implement dedicated hardware sketch sensors, which would drastically reduce the computational complexity of computing the sketch. Practically, we present a general strategy, asymmetric compressive learning (ACL), to learn from generic "distorted" sketch contributions. We then address the important question of *learning guarantees* obtained by learning from such a distorted (*e.g.*, quantized) sketch—this is where the guarantees of the previous chapter are leveraged. The approach is further validated by extensive numerical simulations.

Part II: Private Sketches. This second axis, consisting of a single **Chapter 5**, studies a modification to the sketch to ensure the privacy (and more specifically, the *differential privacy*) of the contributors to the dataset. Intuitively, the idea is that compressive learning is a good match for privacy, as both approaches try to extract as most as possible information from the dataset while forgetting as much as possible about the individual data vectors. As we will see, this intuitive hope is justified, as this approach shows encouraging empirical success when compared to other existing privacy-preserving mechanisms. This work is a joint work with Antoine Chatalic (IRISA, Université de Rennes) and Florimond Houssiau (Imperial College London), and our respective supervisors: Laurent Jacques, Rémi Gribonval and Yves-Alexandre de Montjoye.

Part III: Towards General Compressive Learning. One of the drawbacks of the compressive learning framework is that it is quite limited in the range of machine learning tasks that it can solve (*e.g.*, in the previous Parts we consider mainly k-means and Gaussian mixture modeling). In **Chapter 6**, we gather some tentative extensions of CL to other tasks that were initiated during this thesis (such as classification and generative network training). Note that those extensions are only proof-of-concepts and would deserve to be more thoroughly studied in the future. We also present a generic technique to investigate the result of pragmatic compressive learning algorithms, which is helpful in the context of designing novel CL cost functions.

The main takeaway of that second-to-last chapter seems to be that it is quite difficult to extend compressive learning to other tasks. In **Chapter 7**, after summarizing the main contributions of this thesis and providing a few perspectives on each chapter, we give a few last thoughts on these difficulties.

1.3.1 List of publications

Hereafter is the list of scientific publications submitted in the context of this thesis, either peer-reviewed or under review. Several of them inspired significant parts of this manuscript, which will be explicitly mentioned at the beginning of the relevant chapters. The publications are grouped according to the part they are related to and, within each part, listed by order of submission. Moreover, we mention the type of publication: journal paper **(J)**, conference paper **(C)**, or extended abstract **(A)**, *i.e.*, a short version of a conference paper (usually 2 pages).

Related to Part I (quantization of sketch contributions):

- **(J)** Vincent Schellekens and Laurent Jacques, "*Quantized Compressive K-Means*", in IEEE Signal Processing Letters (vol. 25, no. 8), August 2018, cited hereafter as [SJ18b].

- **(J)** Vincent Schellekens and Laurent Jacques, "*Breaking the waves: asymmetric random periodic features for low-bitrate kernel machines*", accepted for publication and **to appear** in Information and Inference: a journal of the IMA, 2021, cited hereafter as [SJ20a].

- **(J)** Vincent Schellekens and Laurent Jacques, "*Asymmetric compressive learning guarantees with applications to quantized sketches*", **under review** at

IEEE Transactions on Signal Processing, 2021, cited hereafter as [SJ21].

Related to Part II (private sketching):

(C) Vincent Schellekens, Antoine Chatalic, Florimond Houssiau, Yves-Alexandre de Montjoye, Laurent Jacques and Rémi Gribonval, "*Differentially Private Compressive k-Means*", IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, cited hereafter as [SCH+19a].

(A) Vincent Schellekens, Antoine Chatalic, Florimond Houssiau, Yves-Alexandre de Montjoye, Laurent Jacques and Rémi Gribonval, "*Compressive k-means with differential privacy*", Signal Processing with Adaptive Sparse Structured Representations workshop (SPARS), 2019, cited hereafter as [SCH+19b].

(J) Antoine Chatalic, Vincent Schellekens, Florimond Houssiau, Yves-Alexandre de Montjoye, Laurent Jacques and Rémi Gribonval, "*Compressive Learning with Privacy Guarantees*", accepted for publication and **to appear** in Information and Inference: a journal of the IMA, 2021, cited hereafter as [CSH+21].

Related to Part III (tentative extensions of CL):

(A) Vincent Schellekens and Laurent Jacques, "*Compressive Classification (Machine Learning without learning)*", international Traveling Workshop on Interactions between low-complexity data models and Sensing Techniques (iTWIST), 2018, cited hereafter as [SJ18a].

(C) Vincent Schellekens and Laurent Jacques, "*Compressive Learning of Generative Networks*", European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), 2020, cited hereafter as [SJ20b].

(A) Vincent Schellekens and Laurent Jacques, "*When compressive learning fails: blame the decoder or the sketch?*", international Traveling Workshop on Interactions between low-complexity data models and Sensing Techniques (iTWIST), 2020, cited hereafter as [SJ20c].

During this thesis, I also contributed to the following tutorial/review article on the field on compressive learning and connected topics:

(J) Rémi Gribonval, Antoine Chatalic, Nicolas Keriven, Vincent Schellekens, Laurent Jacques, Philip Schniter, "*Sketching Datasets for Large-Scale Learning*", **under review** at IEEE Signal Processing Magazine, 2021, cited hereafter as [GCK+20]

Finally, as a by-product of this thesis, the `pycle` Python toolbox for compressive learning was developed, which is described, for the interested reader, in Appendix B.

1.4 Notations and conventions

Throughout this thesis, vectors and matrices are denoted by bold symbols.

The unit imaginary number is noted $\mathrm{i} = \sqrt{-1}$. The real part, the imaginary part, and the complex conjugation of $a \in \mathbb{C}$ read $\Re(a)$, $\Im(a)$, and a^*, respectively.

The ℓ_p-norm of a vector $\boldsymbol{u} \in \mathbb{R}^d$ reads $\|\boldsymbol{u}\|_p = (\sum_i |u_i|^p)^{1/p}$ for $p \geq 1$, with $\|\boldsymbol{u}\|_\infty = \max_i |u_i|$, and $\|\boldsymbol{u}\|_0 = |\operatorname{supp} \boldsymbol{u}| = |\{i : u_i \neq 0\}|$.

The unit ℓ_p-ball ($p \geq 1$) in dimension d is noted $\mathbb{B}_p^d := \{\boldsymbol{u} \in \mathbb{R}^d \,|\, \|\boldsymbol{u}\|_p \leq 1\}$, with the shorthand $\mathbb{B}^d = \mathbb{B}_2^d$.

The cardinality of a finite set $\mathcal{S}$ is $|\mathcal{S}|$, the Minkowski sum of two sets $\mathcal{A}$ and $\mathcal{B}$ is $\mathcal{A} + \mathcal{B} = \{a + b : a \in \mathcal{A}, b \in \mathcal{B}\}$, the index set in $\mathbb{R}^d$ is $[d] := \{1, \cdots, d\}$ for $d \in \mathbb{N}$.

The identity matrix in $\mathbb{R}^d$ is $\boldsymbol{I}_d \in \mathbb{R}^{d \times d}$, and the Kronecker delta $\delta_{k,k'}$ is defined as $\delta_{k,k'} = 1$ if $k = k'$ and $\delta_{k,k'} = 0$ otherwise.

By abuse of notation, evaluating a scalar function $f : \mathbb{R} \to \mathbb{C}$ on a vector $\boldsymbol{u} \in \mathbb{R}^m$ means applying this function componentwise, *i.e.*, $f(\boldsymbol{u}) \in \mathbb{C}^m$ with $(f(\boldsymbol{u}))_j = f(u_j)$. Similarly, "inequalities" between vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^m$ are to be interpreted component-wise, *e.g.*, $\boldsymbol{u} \leq \boldsymbol{v}$ means $u_j \leq v_j$ for all $1 \leq j \leq m$.

The notation $\sim \mathcal{P}$ denotes that a random variable, vector, or function is distributed according to the distribution $\mathcal{P}$. The uniform distribution on a set $\mathcal{A}$ is noted $\mathcal{U}(\mathcal{A})$, and "i.i.d." means "identically and independently" distributed.

2

Preliminaries: Flavors of Compressive Learning

COMPRESSIVE LEARNING is the trunk from which the diverse contributions of this thesis branch off. In turn, this core field takes its roots from a diverse cast of rich research areas, that can be (somewhat arbitrarily) categorized in four major disciplines. This chapter provides a self-contained introduction to each of those disciplines: *machine learning* in Section 2.1, *signal processing* in Section 2.2, *data synopses* in Section 2.3, and finally the *geometry of probability distributions* in Section 2.4. Given the vastness of these topics, this presentation will be far from exhaustive; its main goal is rather to introduce, in a pedagogical manner (assuming little or no prior knowledge), the important concepts and notations that will be relevant throughout this thesis.

Compressive learning (CL) itself is then introduced in Section 2.5. This explanation will be supported by the four root disciplines, each providing different and complementary "lens" through which CL can be interpreted. To give more concrete idea of the understanding we will be building towards throughout the entirety of this chapter, Figure 2.1 provides a glimpse of the multiple intertwined interpretations that the root disciplines provide.

In this chapter, several explanations are inspired by, and some figures are even copied from, the accessible introductory paper on compressive

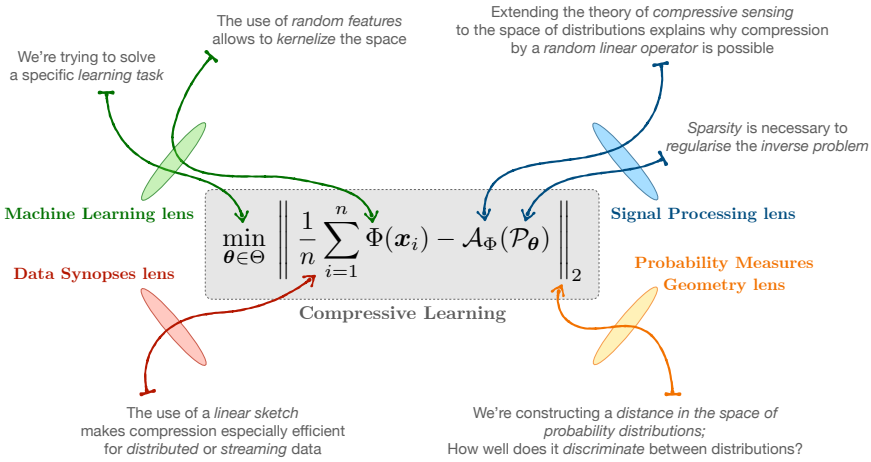


Fig. 2.1 The goal of this chapter is to progressively understand this figure. The grey box in the center contains the cornerstone formulation of compressive learning (explained in Sec. 2.5). The four corners provide examples of insights that come from looking at this formulation through the lens of the four root disciplines (introduced in Sec. 2.1 to 2.4).

learning [GCK+20] we collaborated on. A few other parts are also inspired by some publications related to this thesis; however the bulk of this chapter (in terms of content and structure) is original.

2.1 Machine Learning

The ambition of compressive learning (and thus, of this thesis) is to solve large-scale *Machine Learning* (ML) problems in a resource-efficient manner. In Chapter 1, we loosely defined machine learning as the discipline that turns some form of *data* into useful *mathematical models* by an *automatic extraction* procedure. To make this definition more meaningful, let us precise those three ingredients. This will lead us to one particular[1] theory of ML, called *statistical learning* (SL); see, for example, the textbooks [FHT01, SSBD14] or the shorter (but denser) introductory paper [Vap99]. This general framework will not only be our guide in our subsequent exploration of the machine learning landscape, but also serves as theoretical foundation

[1] While statistical learning is probably the most popular one, many other theoretical frameworks for ML exist. There is currently a strong research interest for such alternatives, in the hope of finally understanding the unreasonable efficiency of deep learning [ZBH+17].

for guarantees in CL (cfr. Section 2.5, Chapter 4).

Data (*e.g.*, a data*set*) The experience from which the machine learns takes, most often[2], the form of a *dataset*: a collection of multiple data points (that we will also call "signals"), assumed to be representative of the data distribution of interest. Mathematically, a dataset $\mathcal{X}$ is a multiset[3] of n learning examples $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n$, which belong to some *domain* or *signal space* Σ, *i.e.*,

$$\mathcal{X} := \{\boldsymbol{x}_i \in \Sigma \,|\, i = 1, ..., n\}. \tag{2.1}$$

The signal space depends on the nature of the learning examples (*e.g.*, Σ could be a set of time series, of images, of text documents...). But very often, the samples $\boldsymbol{x}_i \in \mathbb{R}^d$ are encoded by d-dimensional Euclidean vectors. For the moment, we thus consider that $\Sigma = \mathbb{R}^d$; we will come back to this notion later, when discussing the notion of *features*.

Above, we stated that $\mathcal{X}$ shoud be "representative" of the "data of interest". The implicit idea here is that there are other examples, not accessible during training, for which our model should to be applicable: the goal is to *generalize* on those unseen samples. To formalize this idea, statistical learning theory assumes that there exists a *true data distribution*[4] $\mathcal{P}_0 \in \mathcal{M}_+^1(\Sigma)$, which is such that: *(i)* each element of the dataset is an independent and identically distributed (i.i.d.) observation from that distribution, *i.e.*,

$$\boldsymbol{x}_1, ..., \boldsymbol{x}_n \sim_{\text{i.i.d.}} \mathcal{P}_0; \tag{2.2}$$

and *(ii)* the unknown samples we would like to generalize on also follow this distribution[5]. As we will see, the formal goal of machine learning will thus be to perform well on the true data distribution $\mathcal{P}_0$.

Mathematical models (*e.g.*, a parameter vector) In ML, the goal is to find a good *model* (or *hypothesis*). Models can serve a pleathora of purposes, each one leading to different task families. For example, in the family of *supervised learning* tasks, the model is a function $f_{\boldsymbol{\theta}} : \Sigma_{\overline{\boldsymbol{x}}} \to \Sigma_y : \overline{\boldsymbol{x}} \mapsto f_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}})$

[2]Some branches of ML consider other types of experience than the "fixed dataset" presented here: *e.g.*, in *active learning* [Set09], the algorithm starts with an empty or small dataset which is progressively enriched by actively querying new samples; or in *reinforcement learning* [SB18], it receives informative feedback ("rewards") after performing sequences of actions.

[3]A multiset is a set that can contain several instances (*i.e.*, copies) of its elements.

[4]As will become clearer in Section 2.4, $\mathcal{M}_+^1(\Sigma)$ is the set of probability distributions on Σ.

[5]If not, there is a *distribution shift*, which is a major issue in real-life machine learning.

whose goal is to predict some target features of the data $y_i \in \Sigma_y$ from the others $\overline{\boldsymbol{x}}_i \in \Sigma_{\overline{\boldsymbol{x}}}$ (we subdivide the data vectors as $\boldsymbol{x} = (\overline{\boldsymbol{x}}, y) \in \Sigma = \Sigma_{\overline{\boldsymbol{x}}} \times \Sigma_y$ with $\times$ the Cartesian product)[6]. This family can be further divided into *classification problems* where the set of possible predictions is finite (*e.g.*, $f_{\boldsymbol{\theta}} : \Sigma_{\overline{\boldsymbol{x}}} \to \{1, ..., K\}$ where the K classes are labeled by indices $1, ..., K$); or *regression problems* where the output is continuous (*e.g.*, $f_{\boldsymbol{\theta}} : \Sigma_{\overline{\boldsymbol{x}}} \to \mathbb{R}$). Another important family are *unsupervised learning* tasks, which seek to better understand and/or represent the data. Existing compressive learning approaches, and this thesis, mostly focus on unsupervised tasks.

Despite those different purposes, almost all ML models are "parameterized", *i.e.*, they are fully described by some *parameter vector* $\boldsymbol{\theta}$, which lives in a *parameter space* Θ. A common setting is to have $\Theta \subset \mathbb{R}^p$, which means that we have p real parameters to learn. The goal of ML is then to find the model[7] $\boldsymbol{\theta}$ which best fits the data $\mathcal{X}$; how precisely this happens is what we explain next.

Automatic extraction (*e.g.*, empirical risk minimization) In practice, the optimization procedure that dictates which model best "fits" the data is often crafted *ad hoc* for each specific task. However, the SL framework proposes a unifying principle covering most practical cases. One first crafts a *loss function* $\ell : \Sigma \times \Theta \to \mathbb{R} : (\boldsymbol{x}, \boldsymbol{\theta}) \mapsto \ell(\boldsymbol{x}, \boldsymbol{\theta})$, which describes how good the model $\boldsymbol{\theta}$ is for any possible data point $\boldsymbol{x} \in \Sigma$ (a smaller loss indicates a better fit). The ultimate goal in SL is then to find the model parameters $\boldsymbol{\theta}^* \in \Theta$ that minimize the *risk* objective $\mathcal{R}(\boldsymbol{\theta}; \mathcal{P}_0) := \mathbb{E}_{\boldsymbol{x} \sim \mathcal{P}_0} \ell(\boldsymbol{x}, \boldsymbol{\theta})$, *i.e.*, the expectation of the loss with respect to the data distribution $\mathcal{P}_0 \in \mathcal{M}_+^1(\Sigma)$:

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta} \in \Theta} \mathcal{R}(\boldsymbol{\theta}; \mathcal{P}_0) = \arg\min_{\boldsymbol{\theta} \in \Theta} \mathbb{E}_{\boldsymbol{x} \sim \mathcal{P}_0} \ell(\boldsymbol{x}, \boldsymbol{\theta}). \tag{2.3}$$

This describes the ideal solution we would like to find. In practice, $\mathcal{P}_0$ is unknown, but the dataset $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ provides n samples generated from this distribution $\boldsymbol{x}_i \sim_{\text{i.i.d.}} \mathcal{P}_0$. The "ideal" risk minimization (2.3) is thus replaced by *empirical risk minimization* (ERM), which uses the *empirical*

[6] Supervised (resp. unsupervised) learning is often defined by the presence (resp. absence) of a separate *label* or "known output". Here we rather focus on the *intended goal* of the model (predictive versus explanatory), and abstract possible labels inside the data vectors.

[7] Mathematically, the model/hypothesis h is distinct from the parameter vector $\boldsymbol{\theta}$ that describes it, *e.g.*, many different paramters $\boldsymbol{\theta}$ often equivalently describe the same model h. However, in this presentation we found it convenient to abuse notations and keep this distinction implicit; we'll often refer to a "model" as $\boldsymbol{\theta}$, the parameter vector that describe it. If the relationship between $\boldsymbol{\theta}$ and h is many-to-one, we simply pick one solution $\boldsymbol{\theta}$ arbitrarily.

distribution $\widehat{\mathcal{P}}_{\mathcal{X}}$ instead: with $\delta_{\boldsymbol{c}}$ the Dirac delta at $\boldsymbol{c}$, it is defined as[8]

$$\widehat{\mathcal{P}}_{\mathcal{X}} := \frac{1}{n}\sum_{i=1}^{n} \delta_{\boldsymbol{x}_i}.$$

The ERM problem thus consists in finding the model $\widetilde{\boldsymbol{\theta}}$ that minimizes the average (over the dataset) of the loss function:

$$\widetilde{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}\in\Theta} \mathcal{R}(\boldsymbol{\theta}; \widehat{\mathcal{P}}_{\mathcal{X}}) = \arg\min_{\boldsymbol{\theta}\in\Theta} \frac{1}{n}\sum_{\boldsymbol{x}_i\in\mathcal{X}} \ell(\boldsymbol{x}_i, \boldsymbol{\theta}). \tag{2.4}$$

Learning guarantees Equipped with this general learning principle, we would like to know whether or not, and under which conditions, it "succeeds": is the found solution $\widetilde{\boldsymbol{\theta}}$ "good" enough on the true data distribution $\mathcal{P}_0$? In other words, we would like to characterize its "true risk" $\mathcal{R}(\widetilde{\boldsymbol{\theta}}; \mathcal{P}_0)$, or *generalization error*, which can be interpreted as the sum of two terms

$$\mathcal{R}(\widetilde{\boldsymbol{\theta}}; \mathcal{P}_0) = \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0) \; + \; [\mathcal{R}(\widetilde{\boldsymbol{\theta}}; \mathcal{P}_0) - \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0)]. \tag{2.5}$$

The first term $\mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0)$, sometimes called the *bias* or *approximation error*, is the ideal, lowest possible risk that is achievable given our modeling choices (by definition (2.3)). The second term $\mathcal{R}(\widetilde{\boldsymbol{\theta}}; \mathcal{P}_0) - \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0)$, often called *variance*, *estimation error*, or *excess risk*, captures how much the pragmatic ERM solution $\widetilde{\boldsymbol{\theta}}$ is worse than the ideal one $\boldsymbol{\theta}^*$. Once the machine learning task and parameter space Θ have been chosen (see later), it is the only term that can vary.

The goal of the game is thus to bound, or "control", this excess risk. Note that, since the generation of the dataset is considered random (2.2), the ERM solution $\widetilde{\boldsymbol{\theta}}$ is also a random quantity[9]. The excess risk control is thus necessarily probabilistic, and we seek *statistical guarantees* of the form

$$\mathbb{P}[\mathcal{R}(\widetilde{\boldsymbol{\theta}}; \mathcal{P}_0) - \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0) \leq \eta] \geq 1 - \delta. \tag{2.6}$$

Probabilistic bounds of the type (2.6) guarantee *Probably Approximately Correct* (PAC) learning[10]: the learning procedure is allowed to commit a

[8] Intuitively, the empirical distribution $\widehat{\mathcal{P}}_{\mathcal{X}}$ is the probability distribution associated with the selection of one element from the dataset at random (more on this later).

[9] We can thus be unlucky and get a bad "draw": a dataset $\mathcal{X}$ that isn't representative of $\mathcal{P}_0$.

[10] Statistical learning (as presented here) and (agnostic) PAC learning (omnipresent in ML literature), are practically the same framework. The subtle distinction is that the usual PAC learning framework [Val84] moreover explicitly considers the *practical feasibility* of learning, in particular with respect to two aspects: the existence of a sufficiently small sample size n, and the existence of a sufficiently efficient practical algorithm, such that (2.6) is satisfied [SSBD14].

small error $\eta > 0$, as well as to fail with a probability $\delta > 0$ (the probability is here over the draw of the dataset). Of course, the idea is then to show that a given procedure is able to simultaneously achieve a small error η while having a small probability of failure δ.

Bias-complexity tradeoff: under- and overfitting Formal learning guarantees can provide useful insights on the practical design of machine learning systems. A typical question is how much *prior knowledge* to incorporate into the model (here implicitly represented by the parameter space Θ). The answer to this question is given by the so-called *bias-complexity* (or bias-variance) *trade-off*, which can be understood by inspecting the two error contributions in (2.5). On one hand, to have a small bias error term $\mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0)$, we should give the model as much freedom as possible, or put differently, we should impose little to no prior knowledge (Θ should be "large"). Indeed, if the considered model class is not rich enough, even the best possible model $\boldsymbol{\theta}^*$ may not be able to fit the data well enough (we have *underfitting*). On the other hand, to have a small variance (excess risk) term $\mathcal{R}(\widetilde{\boldsymbol{\theta}}; \mathcal{P}_0) - \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0)$, we should impose as much prior knowledge as possible. Indeed, for any finite sample size n and failure probability δ, statistical learning guarantees (2.6) typically predict that the excess risk bound η increases with the "size" of the model set[11] (we have *overfitting*). We should thus impose some prior knowledge, but not too much.

One particularly convenient way to introduce prior knowledge is to add a *regularization* term to the ERM cost function. This is usually done by introducing a nonnegative *penalty function* $\rho : \Theta \to \mathbb{R}_+ : \boldsymbol{\theta} \mapsto \rho(\boldsymbol{\theta})$, which intuitively captures the "complexity" of the solution $\boldsymbol{\theta}$. To penalize complicated solutions, ρ is added to the ERM cost. Problem (2.4) thus becomes

$$\min_{\boldsymbol{\theta} \in \Theta} \mathcal{R}(\boldsymbol{\theta}; \widehat{\mathcal{P}}_{\mathcal{X}}) + \lambda \rho(\boldsymbol{\theta}),$$

where the *regularization strength* $\lambda \geq 0$ controls the relative importance of the penalty function. Intuitively, increasing λ increases "the amount of prior knowledge", and it should thus be set according to the bias-complexity trade-off[12]. Note that beyond mitigating overfitting, regularization can also help to numerically *stabilize* the optimization procedure [SSBD14][13].

[11] The classical notion of "size" is the Vapnik–Chervonenkis (VC) dimension [VC71].

[12] In practice however, exact knowledge trade-off is not available, so selecting λ through *cross-validation* is good practice.

[13] As we will see, this is for example the case of *linear ridge regression*.

What's left in this section Equipped with the lens of a unifying theoretical framework—statistical learning—we now present several practical machine learning tasks and techniques. Obviously, we can only scratch the surface of this gargantuan discipline, so we biased the inevitable sampling of covered topics towards the concepts that are relevant to the following chapters. The interested reader can learn more about machine learning in general textbooks such as [FHT01] (a good starting point), [SSBD14] (focus on theoretical foundations, which served as main inspiration for this section), [Mur12] (focus on probabilistic modeling of uncertainty, *e.g.*, Bayesian methods), [SSB+02] (focus on kernel methods), [BGC17] (focus on deep learning), and probably thousands of other excellent books.

Specifically, we first review the classical machine learning tasks: the family of *supervised tasks* in Subsection 2.1.1, and the family of *unsupervised tasks* in Subsection 2.1.5 (includes k-means and Gaussian mixture modeling, two tasks of particular importance in this thesis). Those problems are presented in their "linear" version, *i.e.*, the target model somehow operates "directly" in the Euclidean space of the data samples $\boldsymbol{x}_i$. This is an important limitation, and many ML models thus work on "features", a transformed version of data $\Phi(\boldsymbol{x}_i)$. We review the two main paradigms for creating features: *neural network methods* in Subsection 2.1.2, and *kernel methods* in Subsection 2.1.3. Finally, in Subsection 2.1.4, we review *random features* constructions, a technique initially proposed to speed up kernel methods, and one of fundamental building blocks in compressive learning.

2.1.1 Linear supervised tasks

The goal of supervised learning tasks is to make predictions on unseen data. Recalling our subdivision of data vectors $\boldsymbol{x} = (\overline{\boldsymbol{x}}, y) \in \Sigma$ into inputs $\overline{\boldsymbol{x}} \in \Sigma_{\overline{x}}$ and a target $y \in \Sigma_y$, this prediction rule takes the form of a parameterized function $f_{\boldsymbol{\theta}} : \Sigma_{\overline{x}} \to \Sigma_y$. Loosely speaking, our goal is to achieve

$$f_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}}) \simeq y \quad \text{for any} \quad \boldsymbol{x} = (\overline{\boldsymbol{x}}, y) \quad \text{where} \quad \boldsymbol{x} \sim \mathcal{P}_0. \tag{2.7}$$

More formally, given a function $D : \Sigma_y \times \Sigma_y \mapsto \mathbb{R}$ that captures the dissimilarity between any prediction and its target, the associated *loss* is

$$\ell(\boldsymbol{x}, \boldsymbol{\theta}) = \ell((\overline{\boldsymbol{x}}, y), \boldsymbol{\theta}) = D(f_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}}), y), \tag{2.8}$$

and generic statistical learning formulation of supervised learning is thus

$$\widetilde{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}\in\Theta} \sum_{\boldsymbol{x}_i\in\mathcal{X}} D(f_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}}_i), y_i) + \lambda\rho(\boldsymbol{\theta}). \tag{2.9}$$

In this subsection, we focus on the class of *linear* prediction functions, *i.e.*, which rely on the linear transformation[14] $\langle\boldsymbol{\theta},\overline{\boldsymbol{x}}\rangle = \boldsymbol{\theta}^\top\overline{\boldsymbol{x}} = \sum_{i=1}^{\overline{d}}\theta_i\overline{x}_i$, where the parameter vector $\boldsymbol{\theta}\in\mathbb{R}^{\overline{d}}$ must be tuned (with $\overline{d} < d$ the dimension of $\overline{\boldsymbol{x}}$). In general, linear prediction functions can thus be written $f_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}}) = g(\langle\boldsymbol{\theta},\overline{\boldsymbol{x}}\rangle)$, where $g:\mathbb{R}\to\Sigma_y$ is an (optional) fixed mapping.

Linear regression problems In usual regression problems, the target is a single continuous attribute, *i.e.*, $y_i\in\Sigma_y = \mathbb{R}$ (we thus have the input dimension $\overline{d} = d-1$), and g is the identity, *i.e.*, $f_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}}) = \langle\boldsymbol{\theta},\overline{\boldsymbol{x}}\rangle$. To fully specify the problem, it remains to pick the dissimilarity measure D and the (optional) regularization function ρ; we discuss a few common choices below.

The simplest and most common regression problem is without doubt **linear least-squares**, where the "dissimilarity" D is the *squared difference* $D(y,y') = (y-y')^2$, *i.e.*, the empirical risk is the *mean squared error*. Introducing the matrix notation $\overline{\boldsymbol{X}} = [\overline{\boldsymbol{x}}_1,...,\overline{\boldsymbol{x}}_n]\in\mathbb{R}^{\overline{d}\times n}$ and $\boldsymbol{y} = [y_1,...,y_n]^\top\in\mathbb{R}^n$, the linear least-squares problem (without regularization) reads

$$\widetilde{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{\overline{d}}} \sum_{i=1}^{n}(\boldsymbol{\theta}^\top\overline{\boldsymbol{x}}_i - y_i)^2 = \arg\min_{\boldsymbol{\theta}} \|\overline{\boldsymbol{X}}^\top\boldsymbol{\theta} - \boldsymbol{y}\|_2^2. \tag{2.10}$$

The well-known closed-form solution to this problem is $\widetilde{\boldsymbol{\theta}} = (\overline{\boldsymbol{X}}\,\overline{\boldsymbol{X}}^\top)^{-1}\overline{\boldsymbol{X}}\boldsymbol{y}$. Note that this expression requires the *empirical covariance matrix* $\overline{\boldsymbol{X}}\,\overline{\boldsymbol{X}}^\top$ to be invertible: if it is not the case (*i.e.*, the samples $\overline{\boldsymbol{x}}_i$ do not span the whole space $\mathbb{R}^{\overline{d}}$), one can still solve for $\widetilde{\boldsymbol{\theta}}$ by resorting to the singular value decomposition [SSBD14]. Another possibility to avoid this issue is to stabilize the problem by adding regularization.

The simplest regularization penalty, sometimes called Tikhonov regularization[15], is the squared ℓ_2 norm of the parameter vector, *i.e.*, $\rho(\boldsymbol{\theta}) =$

[14] In fact, linear predictors rather typically consider the *affine transformation* $\sum_i\theta_i\overline{x}_i + b$ for some bias term $b\in\mathbb{R}$, but to simplify computations, a common trick is to incorporate b into the scalar product by adding a dummy dimension to $\overline{\boldsymbol{x}}$ which is always equal to one.

[15] In fact, Tikhonov regularization is slightly more general, as it allows to use a weighted norm, *e.g.*, $\|\boldsymbol{\theta}\|_\Gamma^2 := \boldsymbol{\theta}^\top\Gamma\boldsymbol{\theta}$ for some positive definite matrix Γ; in our case Γ is the identity $\boldsymbol{I}_{\overline{d}}$.

$\|\boldsymbol{\theta}\|_2^2 = \sum_i \theta_i^2$. When associated with the previous linear least-squares rule, this problem is known as the (linear) **Ridge Regression** problem

$$\widetilde{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{\overline{d}}} \sum_{i=1}^{n} (\boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i - y_i)^2 + \lambda\rho(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \|\overline{\boldsymbol{X}}^\top \boldsymbol{\theta} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2, \tag{2.11}$$

and the closed-form solution becomes

$$\widetilde{\boldsymbol{\theta}} = (\overline{\boldsymbol{X}}\,\overline{\boldsymbol{X}}^\top + \lambda \boldsymbol{I}_{\overline{d}})^{-1}\overline{\boldsymbol{X}}\boldsymbol{y}. \tag{2.12}$$

Notice that when $\lambda > 0$, the matrix $\overline{\boldsymbol{X}}\,\overline{\boldsymbol{X}}^\top + \lambda \boldsymbol{I}_{\overline{d}}$ is always invertible. This is known as the primal form of the ridge regression solution, which involves inverting a $\overline{d}$-by-$\overline{d}$ matrix (which takes $\mathcal{O}(\overline{d}^2)$ space and $\mathcal{O}(\overline{d}^3)$ time). By analyzing the dual optimization, or by applying a matrix inversion lemma[16], the solution $\widetilde{\boldsymbol{\theta}}$ can also be written in the so-called dual form,

$$\widetilde{\boldsymbol{\theta}} = \overline{\boldsymbol{X}}(\overline{\boldsymbol{X}}^\top\overline{\boldsymbol{X}} + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{y} = \overline{\boldsymbol{X}}\boldsymbol{\alpha} \quad \text{with} \quad \boldsymbol{\alpha} := (\overline{\boldsymbol{X}}^\top\overline{\boldsymbol{X}} + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{y} \in \mathbb{R}^n. \tag{2.13}$$

Notice that in this case, the matrix to invert is of size n-by-n, which is more efficient to compute when there are less data points than the input dimension ($\mathcal{O}(n^2)$ space and $\mathcal{O}(n^3)$ time). Another advantage of this form is that it involves the data samples only in the form of dot products: the vector $\boldsymbol{\alpha}$ requires the Gram matrix $K = \overline{\boldsymbol{X}}^\top\overline{\boldsymbol{X}}$ which has entries $K_{i,j} = \langle \overline{\boldsymbol{x}}_i, \overline{\boldsymbol{x}}_j \rangle$, and the linear predictor requires to evaluate $f_{\widetilde{\boldsymbol{\theta}}}(\boldsymbol{x}') = \widetilde{\boldsymbol{\theta}}^\top \boldsymbol{x}' = \sum_{i=1}^n \alpha_i \langle \overline{\boldsymbol{x}}_i, \boldsymbol{x}' \rangle$. This fact will be particularly important in the context of kernel methods.

Another popular regularization term, which will play an important role in the next section, is the ℓ_1 norm of the parameter vector, *i.e.*, $\rho(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1 = \sum_i |\theta_i|$. This defines the **Lasso** problem:

$$\widetilde{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^{\overline{d}}} \|\overline{\boldsymbol{X}}^\top \boldsymbol{\theta} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_1. \tag{2.14}$$

While the Lasso problem doesn't have a closed-form solution, and is not even differentiable, it is still a convex optimization program, for which efficient solvers exist (*e.g.*, subgradient and proximal methods [PB14]). The

[16] The lemma in question, which follows from equating two ways to perform blockwise matrix inversion (the same trick that gives the famous Woodbury matrix identity), is that for conformable matrices A, B, C, D, one has $(A - BD^{-1}C)^{-1}BD^{-1} = A^{-1}B(D - CA^{-1}B)^{-1}$, which we here apply to $A = \lambda \boldsymbol{I}_{\overline{d}}$, $B = C^\top = \overline{\boldsymbol{X}}$ and $D = -\boldsymbol{I}_n$.

ℓ_1 penalty is here particularly interesting because it encourages a *sparse* solution vector $\boldsymbol{\theta}$, *i.e.*, with only a few nonzero entries (more about sparsity in the next section). Among others, this makes the model more interpretable, because the prediction rule is based on only a small subset of the possible data features (the Lasso incoporates *feature selection* into the problem).

The *data fidelity term* (the ERM component of the optimization) was the squared ℓ_2 norm of the error $\overline{\boldsymbol{X}}^\top \boldsymbol{\theta} - \boldsymbol{y}$, *i.e.*, the dissimilarity is $D(y, y') = (y - y')^2$. One way to understand the popularity of this loss is through the Bayesian interpretation of (2.9), which can be seen as (the negative log of) a Maximum A Posteriori (MAP) estimation of $\boldsymbol{\theta}$, where the likelihood term is given by $p(\mathcal{X}|\boldsymbol{\theta}) \propto \exp(-\sum_i D(\overline{\boldsymbol{x}}_i^\top \boldsymbol{\theta}, y_i))$ and the prior distribution is $p(\boldsymbol{\theta}) \propto \exp(-\lambda\rho(\boldsymbol{\theta}))$; see [Mur12] for an in-depth exploration of this paradigm. The squared loss then corresponds to assuming the "observations" $y_i \simeq \overline{\boldsymbol{x}}_i^\top \boldsymbol{\theta}$ are corrupted by *Gaussian noise*, which is ubiquitous in nature (*e.g.*, think of the central limit theorem), which explains why the squared loss is very often a good choice.

However, depending on the application, other choices of data fidelity can be more adequate. One popular alternative is the absolute deviation $D(y, y') = |y - y'|$, which leads to an ERM term equal to the ℓ_1 norm the errors $\|\overline{\boldsymbol{X}}^\top \boldsymbol{\theta} - \boldsymbol{y}\|_1$. This choice is typically more robust to outliers, and is suitable to cases where data corruption may be important. In the probabilistic interpretation, it corresponds to assuming the observations are corrupted by *Laplacian noise*, *i.e.*, random variables distributed with density $p_e(e) \propto \exp(-\frac{|e|}{\sqrt{2}\sigma})$.

Linear classification problems We now turn to linear *classification problems*, where the labels $y_i \in \Sigma_y$ are from a discrete set representing $C = |\Sigma_y|$ distinct classes, *i.e.*, $y_i \neq y_j$ iff $\overline{\boldsymbol{x}}_i$ and $\overline{\boldsymbol{x}}_j$ belong to different classes. Our linear prediction function, $f_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}}) = g(\langle \boldsymbol{\theta}, \overline{\boldsymbol{x}} \rangle)$, now involves a map $g : \mathbb{R} \to \Sigma_y$ whose goal is to map the (real-valued) linear term $\langle \boldsymbol{\theta}, \overline{\boldsymbol{x}} \rangle$ to a discrete class. The natural goal in this setting is then to minimize the amount of *classification errors*. In our SL framework, this means picking the dissimilarity $D(y, y') = \iota(y \neq y')$, with $\iota(A)$ the *indicator function* of event A (equal to 1 if A is true and 0 otherwise). The associated loss, often called the 0-1 loss, is a binary value equal to one iff a classification error was committed:

$$\ell(\boldsymbol{x}, \boldsymbol{\theta}) = \iota(f_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}}) \neq y) \in \{0, 1\},$$

and the related empirical risk is the average number of such errors.

Let us first focus on the case of *binary classification*, for which the space of labels is often encoded as $\Sigma_y = \{-1, +1\}$ (we have a "positive" and a "negative" class). The obvious way to map $\langle \boldsymbol{\theta}, \overline{\boldsymbol{x}} \rangle$ to Σ_y is to keep only the sign (*i.e.*, $g = \text{sign}$), which leads to the **halfspace classifier**:

$$f_{\boldsymbol{\theta}}(\overline{\boldsymbol{x}}) = \text{sign}\left(\langle \boldsymbol{\theta}, \overline{\boldsymbol{x}} \rangle\right) \in \{\pm 1\},$$

and the related empirical risk minimization rule reads

$$\widetilde{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^{\overline{d}}} \frac{1}{n} \sum_{i=1}^{n} \iota(\text{sign}\left(\langle \boldsymbol{\theta}, \overline{\boldsymbol{x}}_i \rangle\right) \neq y_i) \tag{2.15}$$

How can we solve this classification problem? In the *linearly separable case*, where there exists a hyperplane that separates the two classes (in other words, the optimal value of problem (2.15) is zero), there exist efficient algorithms, such as Rosenblatt's celebrated perceptron algorithm [Ros58]. However, the linearly separable case is not that common in practice, and without this assumption (in the *agnostic case*) no efficient algorithms exist (the problem in fact becomes NP-hard [BDEL03]).

While the 0-1 loss and the halfspace predictor are the most natural given the nature of the classification problem, they are thus rarely used by ML practitioners, which tend to prefer convex and/or differentiable maps instead. In particular, we present two commonly used *surrogates* for the 0-1 loss function: the logistic loss (which leads to classifier known as logistic regression) and the hinge loss (which leads to the classifier known as support vector machines).

Intuitively, the halfspace classifier is difficult to train because its predictions, which are binary values, are not very informative: when the model is wrong on a given example, we do not know exactly to what extent it is wrong. If the model would output a degree of certainty of its prediction, we could penalize errors more or less severely, according to its confidence in the wrong answer. The standard way to incorporate such a mechanism is to have the model make a *probabilistic prediction*, *i.e.*, for classification with C classes, a vector in the probability simplex[17] $f_{\boldsymbol{\theta}} : \Sigma_{\overline{\boldsymbol{x}}} \to \Delta_C := \{\boldsymbol{a} \in \mathbb{R}_+^C, \sum a_k = 1\}$.

In the case of binary classification (our focus here), this can be sim-

[17] The loss should then be defined as a distance (or divergence) over the probability simplex (more on this in Section 2.4); the most popular is the *cross-entropy loss*.

plified to a single scalar output, *i.e.*, the linear transformation is mapped through some map $g : \mathbb{R} \to [0, 1]$ (*e.g.*, the sigmoid function), whose output can be associated with the probability that the predicted class is positive. When the sigmoid map is combined with the so-called logistic loss[18], we obtain the **logistic regression** task, whose loss function reads [SSBD14]

$$\ell((\overline{\boldsymbol{x}}, y), \boldsymbol{\theta}) = \log(1 + \exp(-y\langle \boldsymbol{\theta}, \overline{\boldsymbol{x}} \rangle)).$$

To interpret this expression, one can note that it is a (convex) decreasing function of the term $y\langle \boldsymbol{\theta}, \overline{\boldsymbol{x}} \rangle$, which increases whenever the prediction becomes more accurate (in particular, if $y\langle \boldsymbol{\theta}, \overline{\boldsymbol{x}} \rangle > 0$ then the prediction is correct).

Our other classification scheme of interest, **Support Vector Machines** (SVM), is similar in that its loss is also a (convex) decreasing function of $y\langle \boldsymbol{\theta}, \overline{\boldsymbol{x}} \rangle$: the *hinge loss* reads

$$\ell((\overline{\boldsymbol{x}}, y); \boldsymbol{\theta}) = \max(0, 1 - y\langle \boldsymbol{\theta}, \overline{\boldsymbol{x}} \rangle),$$

and the SVM learning problem, which considers Tikhonov regularization, is

$$\widetilde{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^{\overline{d}}} \sum_{i=1}^{n} \max(0, 1 - y_i \langle \boldsymbol{\theta}, \overline{\boldsymbol{x}}_i \rangle) + \lambda \|\boldsymbol{\theta}\|_2^2. \tag{2.16}$$

The hinge loss is similar to the logistic loss (linearly decreasing on the left, constant at zero on the right), except that the "kink" is offset by one to the right: intuitively, this forces the model to maximize the "margin", *i.e.*, the distance of the correctly classified samples to the decision boundary.

For reasons that will become clear in a moment, SVM is often formulated in the "dual" form, where the parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{\overline{d}}$ are rewritten[19] as $\boldsymbol{\theta} = \sum_{i=1}^{n} \alpha_i y_i \overline{\boldsymbol{x}}_i$; the $\boldsymbol{\alpha} \in \mathbb{R}_+^n$ are known as the "dual variables" [SSBD14].

2.1.2 Nonlinear models, features, and neural networks

Remark 2.1. From now on the distinction between input features $\overline{x} \in \Sigma_{\overline{x}}$ and data vectors $\boldsymbol{x} \in \Sigma$ (which may include labels) will be of marginal importance, so to avoid heavy expressions, whenever the meaning is unambiguous from the context, we abuse notation and denote both by $\boldsymbol{x} \in \Sigma$.

Although we have seen that they come in many flavors, linear models

[18] Which is actually equivalent to the cross-entropy loss for the case of binary predictions.

[19] This decomposition of the ERM solution as a linear combination of the input samples is allowed due to a general result known as the *representer theorem* [SHS01]. As a side note, the "Support Vectors" are the samples $\overline{\boldsymbol{x}}_i$ that contribute to the solution, *i.e.*, such that $\alpha_i \neq 0$.

are rather simple: geometrically, the prediction surface (for regression) or decision boundary (for classification) will always be a *hyperplane* in the input space $\Sigma_{\overline{x}}$. This simplicity has the advantage of making linear models efficient to learn and relatively easy analyze, but is also a severe limitation given the highly nonlinear nature of many real-world phenomena. In terms of risk (2.5) they tend to have a high *bias*, except for a minority of well-behaved cases. Most interesting machine learning models are thus nonlinear.

To do so, virtually all nonlinear models in fact "inject" nonlinearity into one of the linear models seen above, by adding a nonlinear data transformation, called *feature map*, prior to the linear relationship. For supervised learning specifically, the ML designer "crafts" a nonlinear feature map $\boldsymbol{\varphi} : \Sigma_{\overline{x}} \to \mathcal{E}$, for some relevant m-dimensional feature space $\mathcal{E}$, and the linear model $f_{\boldsymbol{\theta}}(\cdot) : \mathcal{E} \to \Sigma_y$ then operates in this augmented space; the resulting nonlinear model is $f_{\boldsymbol{\theta}}(\boldsymbol{\varphi}(\boldsymbol{x}))$. The hope is that by mapping all the data vectors $\boldsymbol{x}_i$ in the feature space representation $\boldsymbol{\varphi}(\boldsymbol{x}_i) \in \mathcal{E}$, the relevant signal structure is better exhibited, such that the problem becomes "easier" to solve, in the sense that a simple linear model in $\mathcal{E}$ performs well (*e.g.*, the classes become *linearly separable* under the map $\boldsymbol{\varphi}$).

The simplest illustration of this idea is probably the class of *polynomial models*. In dimension $d = 1$, a polynomial $f_{\boldsymbol{\theta}}(x)$ of degree p can be written

$$f_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + ... + \theta_p x^p = \boldsymbol{\theta}^\top (1, x, x^2, ..., x^p) =: \boldsymbol{\theta}^\top \boldsymbol{\varphi}(x),$$

so that $f_{\boldsymbol{\theta}}(x)$ is a linear map in the augmented space defined by $\boldsymbol{\varphi} : \mathbb{R} \to \mathbb{R}^m : x \mapsto (1, x, x^2, ..., x^p)$ with embedding dimension $m = p + 1$. In higher input dimensions d, the same trick can be applied, and $\boldsymbol{\varphi}$ contains all possible monomials terms $x_1^{p_1} \cdot x_2^{p_2} \cdot ... \cdot x_d^{p_d}$ corresponding to any combination of exponents $p_i \geq 0$ such that $\sum_i p_i \leq p$. Any of the linear models above can thus easily be transformed into a polynomial model, by simply replacing the data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$ by the feature matrix[20] $\boldsymbol{\Phi} := [\boldsymbol{\varphi}(\boldsymbol{x}_1), ..., \boldsymbol{\varphi}(\boldsymbol{x}_n)] \in \mathbb{R}^{m \times n}$.

Remark 2.2. Often, the input dimension d is quite large; since the monomial feature map has a combinatorially large embedding dimension $m = \binom{p+d-1}{p}$ [SSB+02], the polynomial model is not computationally tractable, even for low-degree polynomials. In general, although we do not explicitly focus on this aspect here, note that learning from high-dimensional

[20] For the 1-d polynomial model, $\boldsymbol{\Phi}$ is called the (transposed) *Vandermonde matrix*.

data is quite challenging—due to the *curse of dimensionality* (*e.g.*, Euclidean distances are less "comparable" in higher dimensions). Besides the exhibition of nonlinear relationships, one often also incorporates *dimensionality reduction* aspects into the feature map $\boldsymbol{\varphi}$.

With the notable exception of the polynomial model (in moderate dimension) and a few other simple cases, the design of the feature map $\boldsymbol{\varphi}$ is in general a critical yet non-trivial problem: how to ensure that $\boldsymbol{\varphi}$ correctly unfolds a nonlinear data geometry that is, most often, not well understood by the ML designers themselves? Instead of hand-crafting the features *ad hoc*, modern designers usually turn to one of two general recipes: *neural networks* (now briefly explained, for the sake of completeness), and *kernel methods* (introduced in the next subsection, more relevant to this thesis).

Neural networks are incredibly popular nowadays, which implies that they come in thousands of different flavors, but their common theme is the following: (i) the feature map $\boldsymbol{\varphi} = \boldsymbol{\varphi}^{(\boldsymbol{\theta})}$ is parameterized and learned along with the subsequent linear model $f_{\boldsymbol{\theta}_L}(\cdot) = g(\langle \boldsymbol{\theta}_L, \cdot \rangle)$ (called the "output layer") during the training phase; and (ii) the feature map is itself composed of $L-1$ successive "layers" that combine a linear mapping $\boldsymbol{\theta}_l$ followed by a point-wise nonlinearity σ (*e.g.*, the sigmoid or ReLU activations): the end-to-end model reads

$$F_{\boldsymbol{\theta}}(\boldsymbol{x}) := f_{\boldsymbol{\theta}_L}(\boldsymbol{\varphi}^{(\boldsymbol{\theta})}(\boldsymbol{x})) = g(\boldsymbol{\theta}_L^\top \sigma(\boldsymbol{\theta}_{L-1}^\top \sigma(\cdots \sigma(\boldsymbol{\theta}_1^\top \boldsymbol{x})))).$$

This general architecture can then be injected into one of the linear tasks described above. Although deep neural network often have an unreasonably large amount of tunable parameters (*i.e.*, they are *over-parameterized*), it turns out that they are very successful in practice, for reasons that are not yet completely clear to the research community. We refer the interested reader to (among others) the textbook [BGC17] for a complete description of deep learning, and now turn to kernel methods.

Remark 2.3. One can draw many connections between neural networks and kernel methods (as suggested here by presenting them as two particular cases of feature extraction); some researchers consider kernel methods to be a particular case of neural networks, while in the converse direction other researchers consider that neural network are a specific approximation to kernel methods (*e.g.*, following the neural tangent kernel line of work [JGH18]).

2.1.3 Kernel methods

The desired property of the feature map $\varphi : \Sigma \to \mathcal{E}$ is to somehow distort the space Σ such that the relevant (nonlinear) relationships between data samples are encoded in the "natural geometry" of $\mathcal{E}$. Intuitively, if two samples x and x' are semantically different for the target task, then the distance between their features $\varphi(x)$ and $\varphi(x')$ should be "large". Equivalently, the geometry in the feature space is characterized by the *dot product* $\langle \varphi(x), \varphi(x') \rangle$, which should encode a meaningful *similarity* between x and x'.

The high-level idea of kernel methods [SSB+02] is to directly specify this dot product $\langle \varphi(x), \varphi(x') \rangle$, rather than the feature map φ. More specifically, the ML designer specifies a *kernel function* $\kappa : \Sigma \times \Sigma \to \mathbb{R} : (x, x') \mapsto \kappa(x, x')$ that encodes a relevant *similarity criterion* between any data vectors x and x'. While this function represents the dot product in the feature space, *i.e.*, $\kappa(x, x') = \langle \varphi(x), \varphi(x') \rangle$, the feature map φ is however *implicit*, in the sense that its precise expression is irrelevant to the learning model.

Part of the appeal of kernel methods is that this construction principle is quite flexible. On complicated data domains, where it might be challenging to craft a relevant feature map φ, it is often still possible to directly specify a meaningful notion of "similarity" κ; in other words, kernels are a great way to incorporate *prior knowledge* into the model. For example, many natural similarities can be defined over the domain of text strings [LSST+02], graph data [BGLL+20], and structured domains in general [Gär03].

Elements of kernel theory The other main advantage of kernel methods is their well-understood theoretical properties. While a thorough presentation of this theory is beyond the scope of this manuscript (we refer once again the reader to [SSB+02] for this), we present here its main ingredients, which will support key aspects of this thesis. We begin by formally defining the notion of kernel, and that of *positive definite* (p.d.) kernel in particular (virtually all kernels used in ML are positive definite; they are sometimes called "valid" kernels).

Definition 2.4 (Kernel)**.** A kernel is a map $\kappa : \Sigma \times \Sigma \to \mathbb{C}$ which is conjugate symmetric, *i.e.*,

$$\kappa(x, x') = \kappa(x', x)^*, \quad \forall x, x' \in \Sigma$$

In the vast majority of machine learning applications, the kernel is real-

valued, *i.e.*, $\kappa : \Sigma \times \Sigma \to \mathbb{R}$ which is simply symmetric $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa(\boldsymbol{x}', \boldsymbol{x})$.

Definition 2.5 (Positive definite (p.d.) kernel)**.** A kernel κ is *positive definite* (or valid) if, for any number $n \in \mathbb{N}$, any n-by-n Gram matrix $\boldsymbol{K}$ (*i.e.*, which has entries $K_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$) is positive definite, which is equivalent to ask

$$\sum_{i,j=1}^{n} c_i c_j^* \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0, \quad \forall \boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \Sigma \text{ and } c_1, \dots, c_n \in \mathbb{C}.$$

Intuitively, positive definiteness of the kernel is important because it ensures that it can indeed be interpreted as a valid dot product. To be more specific, if the kernel is p.d., then we know that it can be associated with a (possibly implicit) feature map $\boldsymbol{\varphi} : \Sigma \to \mathcal{H}_\kappa$ for some Hilbert space[21] $\mathcal{H}_\kappa$, such that $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{\varphi}(\boldsymbol{x}') \rangle_{\mathcal{H}_\kappa}$.

The attentive reader will have noticed that this space $\mathcal{H}_\kappa$ is actually nothing but the "feature space" $\mathcal{E}$ that we defined above, but it gets a fancy notation—as well as a fancy name, the *Reproducing Kernel Hilbert Space* (RKHS)—because it enjoys some remarkable properties [Aro50].

Definition 2.6 (RKHS)**.** If $\kappa : \Sigma \times \Sigma \to \mathbb{C}$ is a p.d. kernel, it is uniquely[22] associated with a Hilbert space $\mathcal{H}_\kappa$ of functions, called Reproducing Kernel Hilbert Space, that satisfies the two following properties:

1. $\forall \boldsymbol{x} \in \Sigma$, $\kappa(\cdot, \boldsymbol{x}) \in \mathcal{H}_\kappa$;
2. (Reproducing property) $\forall f \in \mathcal{H}_\kappa$, $f(\boldsymbol{x}) = \langle f(\cdot), \kappa(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}_\kappa}$.

Another (maybe less abstract) way to describe the RKHS $\mathcal{H}_\kappa$ associated with a kernel κ is that it is the (complete) linear span of that kernel:

$$\mathcal{H}_\kappa := \overline{\operatorname{span}\left(\{\kappa(\boldsymbol{x}, \cdot) \mid \boldsymbol{x} \in \Sigma\}\right)}, \tag{2.17}$$

where $\overline{A}$ denotes the closure of the set A.

Looking at (2.17), we can intuitively think of the RKHS as the set of functions f that can be expressed as a linear combination of shifted copies of the the kernel, *i.e.*, $f(\cdot) = \sum_i \alpha_i \kappa(\cdot, \boldsymbol{a}_i)$ for some weights α_i and positions $\boldsymbol{a}_i \in \Sigma$. Note that, the RKHS being a space of functions, the features of any data sample $\boldsymbol{x}$ are also a function; we can in fact write $\boldsymbol{\varphi}(\boldsymbol{x}) = \kappa(\cdot, \boldsymbol{x}) \in \mathcal{H}_\kappa$. As we will now see, in kernel *methods*, the RKHS determines the space of possible models, *i.e.*, $f_{\boldsymbol{\theta}}(\cdot) \in \mathcal{H}_\kappa$.

[21] Intuitively, a Hilbert space $\mathbb{H}$ is simply any space that is equipped with a dot product $\langle \cdot, \cdot \rangle_{\mathbb{H}}$, and that is complete (which allows to take limits in this space).

[22] The unicity of the RKHS is a result known as the Moore-Aronszajin theorem [Aro50].

The representer theorem and the kernel trick At this point, one important question might come up to the reader's mind: how can we possibly learn models from kernels? Following our "featurization" strategy from Subsection 2.1.2, the model is linear in the feature space, so (with some waving of the hands) we should write something like $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \langle \boldsymbol{\theta}, \boldsymbol{\varphi}(\boldsymbol{x}) \rangle_{\mathcal{H}_\kappa}$. However, $\mathcal{H}_\kappa$ is a functional space, which means that $\boldsymbol{\theta} = \boldsymbol{\theta}(\cdot) \in \mathcal{H}_\kappa$ is here a *function*, living in an infinite-dimensional space, which is obviously not really practical. The solution to this issue comes from the *representer theorem*, which roughly speaking allows to write the *optimal* "parameter vector" $\widetilde{\boldsymbol{\theta}}$—the minimizer to a learning problem involving a dataset $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$—as a linear combination of the learning examples in the feature space: one could write $\widetilde{\boldsymbol{\theta}} = \sum_{i=1}^n \alpha_i \boldsymbol{\varphi}(\boldsymbol{x}_i)$. When plugged into the model $f_{\widetilde{\boldsymbol{\theta}}}$, we obtain that the optimal model will always be of the form

$$f_{\widetilde{\boldsymbol{\theta}}}(\boldsymbol{x}) = \sum_{i=1}^n \alpha_i \langle \boldsymbol{\varphi}(\boldsymbol{x}_i), \boldsymbol{\varphi}(\boldsymbol{x}) \rangle_{\mathcal{H}_\kappa} = \sum_{i=1}^n \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}).$$

Crucially, instead of solving for an infinite-dimensional parameter vector $\boldsymbol{\theta}$ (the primal variables), we can solve instead for $\boldsymbol{\alpha} \in \mathbb{R}^n$ (the dual variables).

The informal argument we just described is a (very hand-wavy) justification of a general principle known as the **kernel trick**: any machine learning algorithm that can be formulated only in terms of dot products with the learning examples $\langle \cdot, \boldsymbol{x}_i \rangle$ (*e.g.*, using the representer theorem) can be "kernelized" (*i.e.*, solved into a RKHS instead of the input space) by replacing those dot products with kernel evaluations $\kappa(\cdot, \boldsymbol{x}_i)$ instead. The kernel trick is the basis building block of any kernel method, which are all built from pairwise kernel evaluations $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ (where $\boldsymbol{x}_i, \boldsymbol{x}_j$ are elements of the training set)[23], as well as (during the inference stage) evaluations on newly observed samples $\kappa(\boldsymbol{x}_i, \boldsymbol{x})$. We illustrate this principle by two practical examples below.

Two common kernel methods One of the most popular kernel methods is *Kernel Ridge Regression* (KRR). Recalling the dual formulation of usual ridge regression (2.13), the kernelized version reads [SGV98]

$$f_{\widetilde{\boldsymbol{\theta}}}(\boldsymbol{x}) = \sum_{i=1}^n \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}) \quad \text{with} \quad \boldsymbol{\alpha} := (\boldsymbol{K} + \lambda \boldsymbol{I}_n)^{-1} \boldsymbol{y} \in \mathbb{R}^n. \tag{2.18}$$

For the classification problem, the most famous kernel method is *kernel*

[23] Usually gathered into the Gram matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$, whose has entries are $K_{i,j} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

SVM. Recall that in this case, we also had an expression of the solution in terms of dual variables $\boldsymbol{\alpha}$, and (omitting some aspects that are irrelevant to our discussion) the classifier is written as $f_{\widetilde{\boldsymbol{\theta}}}(\boldsymbol{x}) = \operatorname{sign}\big(\sum_{i=1}^{n} \alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x})\big)$. The coefficients $\boldsymbol{\alpha}$ are the solution to the dual of the SVM problem (2.16), a quadratic optimization program [SSB+02]:

$$\boldsymbol{\alpha} = \arg\max_{\mathbf{0} \leq \boldsymbol{\beta} \leq C\mathbf{1}} \ \sum_i \beta_i - \tfrac{1}{2} \sum_{i,j} \beta_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) y_j \beta_j, \tag{2.19}$$

where the hyperparameter $C = \frac{1}{2n\lambda}$ is inversely proportional to the regularization strength.

Some general-purpose kernels One last aspect we still have to discuss is how exactly to pick the kernel (*i.e.*, the "similarity") κ. As mentioned above, many kernels exists in the literature, each tailored to specific data domains and prior knowledge; if applicable, it might thus be worth seeking a kernel adapted to the problem at hand. However, some "general-purpose" kernels also exist, usually defined over the Euclidean space $\Sigma = \mathbb{R}^d$.

A first popular kernel is the *polynomial kernel* $\kappa(\boldsymbol{x}, \boldsymbol{x}') = (\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + c)^p$, for some degree p and bias c. It can be associated with an (implicit) polynomial feature map (which we used as example to introduce the notion of features).

The most popular kernel of all time is probably the *Gaussian kernel*, often called "the" radial basis function (RBF) kernel (although other types of RBF exist). It is defined, given some *scale* parameter $\sigma > 0$, as

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp\Big(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\sigma^2}\Big) =: \kappa^{\Delta}(\boldsymbol{x} - \boldsymbol{x}'),$$

where on the right we define the function $\kappa^{\Delta} : \Sigma - \Sigma \to \mathbb{R} : \boldsymbol{u} \mapsto \kappa^{\Delta}(\boldsymbol{u}) = e^{-\|\boldsymbol{u}\|_2^2/2\sigma^2}$. In general, kernels that can be written in this way, *i.e.*, as a *function of the difference of the inputs* $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa^{\Delta}(\boldsymbol{u} := \boldsymbol{x} - \boldsymbol{x}')$, are called *shift-invariant* (or *stationary*) kernels; shift-invariant kernels are a popular class of kernels, that play a special role in the next subsection. As another example of shift-invariant kernel, we have the Laplace kernel, similar to the Gaussian kernel but involving the ℓ_1 distance, *i.e.*, $\kappa^{\Delta}(\boldsymbol{u}) = e^{-\|\boldsymbol{u}\|_1/\tau}$ for some scale $\tau > 0$.

2.1.4 Random features

The kernel trick is quite powerful, as it allows to learn functions f in an *infinite-dimensional* feature space ($\mathcal{H}_\kappa$), when they can be written as a linear combination of the kernel evaluated at the learning samples, *i.e.*, $f(\cdot) = \sum_i \alpha_i \kappa(\cdot, \boldsymbol{x}_i)$. However, this same sentence also highlights the crucial weakness of kernel methods: they are parameterized by a dual vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ that has *the same size as the number of learning examples n*. Even worse, kernel methods (*i.e.*, algorithms to find $\boldsymbol{\alpha}$) typically involve the Gram matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$, which takes $\mathcal{O}(n^2)$ memory to store, and (*e.g.*, if we want to solve KRR) $\mathcal{O}(n^3)$ operations to invert. When the number of samples n is very large (as in modern large-scale machine learning), kernel methods are thus not computationally tractable, which explains (among numerous reasons) why neural networks are nowadays the preferred solution.

To circumvent this issue, several approximation strategies to kernel methods were proposed. The two most popular approaches are the Nyström method and random Fourier features. In a nutshell, the idea of **Nyström method** is to subsample $m \ll n$ data points $\widehat{\boldsymbol{x}}_i$ from the full dataset $\mathcal{X}$, and to construct a low-rank approximation to the Gram matrix based on those points. Roughly speaking, the idea is still to use the kernel trick, but to use it on a smaller amount of samples (*i.e.*, to use a restricted set of m kernel evaluations $\kappa(\cdot, \widehat{\boldsymbol{x}}_i)$ instead of the full n evaluations $\kappa(\cdot, \boldsymbol{x}_i)$). This approach recently showed state-of-the-art performance on massive-scale datasets [MCRR20].

In contrast, **random Fourier features** (RFF) [RR08], abandon the kernel trick (*i.e.*, the idea to work implicitly in an infinite-dimensional RKHS $\mathcal{H}_\kappa$), and construct instead (in a way we explain below) an *explicit* feature map $\Phi : \Sigma \to \mathbb{C}^m$ of *finite dimension*, that approximates the kernel (*i.e.*, the dot product in $\mathcal{H}_\kappa$):

$$\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_2 \simeq \kappa(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{\varphi}(\boldsymbol{x}') \rangle_{\mathcal{H}_\kappa}.$$

Because with RFF we have explicit access to the features $\Phi(\boldsymbol{x})$, one can directly kernelize any machine learning technique by replacing $\boldsymbol{x}$ by $\Phi(\boldsymbol{x})$ (or in general, the data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n]$ by the random features matrix $\boldsymbol{\Phi} = [\Phi(\boldsymbol{x}_1), ..., \Phi(\boldsymbol{x}_n)]$), even when they do not directly appear in a dot product. To illustrate the advantage of this approach, take the example of KRR: with RFF we can once again solve the problem in its primal form (2.12). Note that this formulation relies on the (feature) covariance matrix $\boldsymbol{\Phi}\boldsymbol{\Phi}^\top \in \mathbb{C}^{m \times m}$, while the dual form relies on the Gram matrix

$\boldsymbol{K} \in \mathbb{C}^{n\times n}$, which thus greatly reduces the computational complexity when $m \ll n$.

Nyström methods and random Fourier features were extensively compared in [YLM+12]. As reported in this work, Nyström methods tend to perform better (given a number of "features" m), but one has to keep in mind that such methods have access to the dataset when the approximation is constructed. On the other hand, random Fourier features are *data-independent*: they seek to approximate the kernel uniformly over a target domain. This fact is particularly important in the context of compressive learning (Section 2.5), where we design the sketch operator before accessing the data.

Constructing the Random Fourier features Random Fourier features rely on *Bochner's theorem* [Rud62], which allows to write any *p.d.* shift-invariant[24] continuous kernel[25] $\kappa(\boldsymbol{x},\boldsymbol{y}) = \kappa^\Delta(\boldsymbol{x}-\boldsymbol{y})$ as the (inverse) Fourier transform of a probability measure Λ, *i.e.*,

$$\kappa^\Delta(\boldsymbol{u}) = (\mathcal{F}^{-1}\Lambda)(\boldsymbol{u}) = \textstyle\int_{\mathbb{R}^d} e^{\mathrm{i}\boldsymbol{\omega}^\top \boldsymbol{u}} \mathrm{d}\Lambda(\boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{\omega}\sim\Lambda}\, e^{\mathrm{i}\boldsymbol{\omega}^\top \boldsymbol{u}}. \tag{2.20}$$

The key idea of random Fourier features [RR08], illustrated by Fig. 2.2, is thus to construct finite-dimensional features $\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}')$ whose inner product approximates the kernel $\kappa(\boldsymbol{x},\boldsymbol{x}')$ by Monte Carlo sampling of this expectation. Specifically, given a target shift-invariant kernel $\kappa^\Delta(\boldsymbol{x}-\boldsymbol{x}')$ with Fourier transform $\Lambda = \mathcal{F}\kappa^\Delta$, we construct m-dimensional random Fourier feature map as

$$\Phi(\boldsymbol{x}) := \tfrac{1}{\sqrt{m}} \exp(\mathrm{i}\boldsymbol{\Omega}^\top \boldsymbol{x}) \quad \in \mathbb{C}^m, \tag{2.21}$$

with random projections (or "frequencies") $\boldsymbol{\Omega} := (\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_m) \in \mathbb{R}^{d\times m}$ generated as $\boldsymbol{\Omega} \sim \Lambda^m$, *i.e.*, with $\boldsymbol{\omega}_j \sim_{\text{i.i.d.}} \Lambda$ for $j = 1, ..., m$. Indeed, by direct application of Bochner's theorem (2.20), the inner product of RFF approaches (in expectation over the draw of the frequencies $\boldsymbol{\Omega}$) the target kernel: denoting the approximatation $\widetilde{\kappa}(\boldsymbol{x},\boldsymbol{x}') := \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}')\rangle$, we have

$$\mathbb{E}\,\widetilde{\kappa}(\boldsymbol{x},\boldsymbol{x}') = \tfrac{1}{m}\textstyle\sum_{j=1}^m \mathbb{E}_{\boldsymbol{\omega}_j} e^{\mathrm{i}\boldsymbol{\omega}_j^\top(\boldsymbol{x}-\boldsymbol{x}')} = \kappa(\boldsymbol{x},\boldsymbol{x}').$$

[24] Although they mostly focus on shift-invariant kernels, random features can also approximate more general kernels, see *e.g.*, [Bac17, LHCS20].

[25] Assuming *w.l.o.g.* the normalization $\kappa(\boldsymbol{x},\boldsymbol{x}) = \kappa^\Delta(\boldsymbol{0}) = 1$.

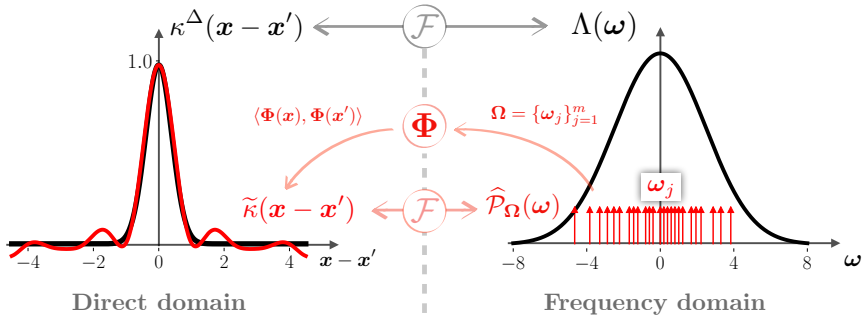


Fig. 2.2 RFF principle. Bochner's theorem (top) states that a shift-invariant kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa^{\Delta}(\boldsymbol{x} - \boldsymbol{x}')$ (left, black) can be expressed in the Fourier domain as a probability distribution $\Lambda(\boldsymbol{\omega})$ (right, black). The idea of random Fourier features is to sample m "frequencies" $\boldsymbol{\omega}_j \sim_{\text{i.i.d.}} \Lambda$, which defines an empirical probability distribution $\widehat{\mathcal{P}}_{\boldsymbol{\Omega}} = \frac{1}{m}\sum_{j=1}^{m} \delta_{\boldsymbol{\omega}_j}$ (right, red). The kernel $\widetilde{\kappa}$ computed by the random Fourier features map Φ in (2.21) is then the inverse Fourier transform of this empirical distribution (left, red), and approximates the original kernel κ.

As we explain in Chapter 3, one can also give a precise bound on the approximation error commited by RFF, which gives an idea of the required amount of features m.

Since Rahimi and Recht introduced the technique in [RR08], random Fourier features have been intensely studied. For a recent overview of this rapidly growing field, we refer the reader to the literature reviews [LHCS20] and [LTOS19].

2.1.5 Unsupervised tasks

So far, we focused (in Subsection 2.1.1) on *supervised* learning tasks, *i.e.*, "prediction" tasks such as classification and regression. In fact, most of the tasks considered in this thesis belong to the family of *unsupervised* learning. Due to the absence of target "labels", the goal of unsupervised tasks tend to be less clearly defined, but usually combine at least one (and often both) of the two following generic objectives: *understanding* the data by representing it by a simple model (*e.g.*, discovering natural clusters in the data); and *representing* the data more adequately (*e.g.*, in a smaller dimension).

Remark 2.7. The general technique of capturing nonlinear relationships by a feature map $\boldsymbol{\varphi} : \Sigma \to \mathcal{E}$, which we explored in Subsections 2.1.2 to 2.1.4,

are not specific to supervised learning. Unsupervised learning also makes use of nonlinear features (both learned as in neural networks, and fixed as in kernel methods).

In this section, we look at a shortlist of some unsupervised learning tasks of interest, with a particular focus for k-means and Gaussian mixture modeling, two tasks of specific interest in this thesis. We will cast those tasks in the SL framework, which we recall here for convenience. Given n examples $\boldsymbol{x}_i \in \Sigma$, a model parameterized by $\boldsymbol{\theta} \in \Theta$, and a loss function $\ell(\boldsymbol{x}, \boldsymbol{\theta})$ characterizing the "fitness" of the model $\boldsymbol{\theta}$ on any data vector $\boldsymbol{x}$, the goal of SL is to minimize the empirical risk:

$$\widetilde{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}\in\Theta} \mathcal{R}(\boldsymbol{\theta}; \widehat{\mathcal{P}}_{\mathcal{X}}) = \arg\min_{\boldsymbol{\theta}\in\Theta} \tfrac{1}{n} \textstyle\sum_{\boldsymbol{x}_i\in\mathcal{X}} \ell(\boldsymbol{x}_i, \boldsymbol{\theta}).$$

We find it useful to arbitrarily classify the tasks presented here into two broad families: *linear decomposition* tasks (*e.g.*, PCA, k-means) and *density fitting* tasks (*e.g.*, Gaussian mixture modeling, generative networks).

Linear decomposition tasks

We define linear decomposition tasks, or "matrix factorization-like" tasks, as seeking to express, or "explain", the learning examples $\boldsymbol{x}_i \in \mathbb{R}^d$ as a linear combination of K *factors* $\boldsymbol{f}_k \in \mathbb{R}^d$, where their *weights* g_k depend (possibly nonlinearly) on the learning example:

$$\boldsymbol{x}_i \simeq \textstyle\sum_{k=1}^K g_k(\boldsymbol{x}_i) \cdot \boldsymbol{f}_k = \sum_{k=1}^K g_k^{(\boldsymbol{\theta})}(\boldsymbol{x}_i) \cdot \boldsymbol{f}_k^{(\boldsymbol{\theta})}. \tag{2.22}$$

In order to have a proper learning problem, at least the factors $\boldsymbol{f}_k$ or the weights g_k are parameterized by tunable parameters $\boldsymbol{\theta}$, as denoted by the superscript $^{(\boldsymbol{\theta})}$. To formalize this learning problem, we define a dissimilarity function $D : \Sigma \times \Sigma \to \mathbb{R}$ such that $\ell(\boldsymbol{x}, \boldsymbol{\theta}) = D(\boldsymbol{x}, \sum_k g_k(\boldsymbol{x}) \boldsymbol{f}_k)$ captures the "error" committed by the linear decomposition. The generic linear decomposition task is thus written in the SL framework as

$$\widetilde{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}\in\Theta} \sum_{\boldsymbol{x}_i\in\mathcal{X}} D\big(\boldsymbol{x}_i, \textstyle\sum_{k=1}^K g_k^{(\boldsymbol{\theta})}(\boldsymbol{x}_i) \boldsymbol{f}_k^{(\boldsymbol{\theta})}\big).$$

This problem might seem a bit abstract for now, but as we will see, it unifies several classical learning tasks.

Principal Component Analysis In Principal Component Analysis (PCA), the goal is to find the K-dimensional subspace that best "fits" the data, or as

commonly formulated, that captures most of its variance. Formulating this in our linear decomposition framework, the "factors" $\boldsymbol{f}_k$ form the orthonormal basis of a K-dimensional subspace which must be learned. Gathering those basis vectors into a matrix $\boldsymbol{F} = [\boldsymbol{f}_1, ..., \boldsymbol{f}_K] \in \mathbb{R}^{d\times K}$, we thus have to learn $\boldsymbol{\theta} := \boldsymbol{F}$ under the constraint that $\boldsymbol{F}$ is an orthonormal basis, *i.e.*, $\boldsymbol{F}^\top \boldsymbol{F} = \boldsymbol{I}_K$ (which defines the set Θ). Moreover, the linear decomposition $\sum_k g_k(\boldsymbol{x})\boldsymbol{f}_k$ is defined to be the *orthogonal projection* of $\boldsymbol{x}$ onto that subspace, that is, $g_k(\boldsymbol{x}) := \langle \boldsymbol{x}, \boldsymbol{f}_k \rangle$. Finally, in PCA the loss is defined by the squared distance $D(\boldsymbol{u}, \boldsymbol{v}) := \|\boldsymbol{u} - \boldsymbol{v}\|_2^2$, or geometrically, $\ell(\boldsymbol{x}, \boldsymbol{\theta})$ is the (squared) distance of $\boldsymbol{x}$ to the subspace spanned by the $\boldsymbol{f}_k$. Combining these design choices yields (one of) the (many) usual formulation(s) of PCA:

$$\widetilde{\boldsymbol{F}} \in \operatorname*{arg\,min}_{\substack{\boldsymbol{F}\in\mathbb{R}^{d\times K},\\ \boldsymbol{F}^\top\boldsymbol{F}=\boldsymbol{I}_K}} \sum_{\boldsymbol{x}_i\in\mathcal{X}} \|\boldsymbol{x}_i - \boldsymbol{F}\boldsymbol{F}^\top \boldsymbol{x}_i\|_2^2 = \operatorname*{arg\,min}_{\boldsymbol{F}^\top\boldsymbol{F}=\boldsymbol{I}_K} \|\boldsymbol{X} - \boldsymbol{F}\boldsymbol{F}^\top \boldsymbol{X}\|_F^2,$$

with $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ the data matrix and $\|\cdot\|_F$ the Frobenius norm. It is well-known that the solution to PCA is given by the top K eigenvectors of the data covariance matrix $\boldsymbol{X}\boldsymbol{X}^\top$ (*e.g.*, owing to the Eckart-Young theorem).

Clustering (k-means) Another popular linear decomposition task is centroid-based clustering. Here, the "factors" $\boldsymbol{f}_k = \boldsymbol{c}_k$ are called *centroids*, and the goal is to find a set of 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 is associated to its "closest" centroid (according to a metric D) by binary weights $g_k(\boldsymbol{x}) \in \{0, 1\}$:

$$g_k(\boldsymbol{x}) := \begin{cases} 1 & \text{if } k = \arg\min_{k'} D(\boldsymbol{x}, \boldsymbol{c}_{k'}), \\ 0 & \text{else.} \end{cases}$$

In terms of statistical learning, the loss $\ell(\boldsymbol{x}, \boldsymbol{\theta}) = \min_{1\le k\le K} D(\boldsymbol{x}, \boldsymbol{c}_k)$ is thus simply the closest distance from $\boldsymbol{x}$ to a point in $\mathcal{C}$.

For example, when $D(\boldsymbol{u}, \boldsymbol{v}) = \|\boldsymbol{u} - \boldsymbol{v}\|_1$ is the ℓ_1 distance, we obtain *k-medians clustering*. A more common choice is $D(\boldsymbol{u}, \boldsymbol{v}) = \|\boldsymbol{u} - \boldsymbol{v}\|_2^2$ the squared Euclidean distance, which yields the famous *k-means clustering* problem [Ste56], widely used in, *e.g.*, data compression, pattern recognition, and bioinformatics [Jai10, Ste06]. To be perfectly clear, given K a prescribed number of clusters (groups of similar data), k-means cluster-

ing seeks the centroids $\mathcal{C} = \{\boldsymbol{c}_k \in \mathbb{R}^d\}_{k=1}^K$ minimizing the Sum of Squared Errors (SSE, the "error" is the distance between each sample $\boldsymbol{x}_i$ and the centroid closest to it):

$$\widetilde{\mathcal{C}} \in \arg\min_{\mathcal{C}} \sum_{\boldsymbol{x}_i \in \mathcal{X}} \min_{1 \leq k \leq K} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2 =: \arg\min_{\mathcal{C}} \text{SSE}(\mathcal{C}; \mathcal{X}). \tag{2.23}$$

Solving (2.23) exactly is NP-hard [ADHP09], so in practice a tractable heuristic such as the popular `k-means` algorithm [Llo82, AV07] is widely used to find an approximate solution $\mathcal{C}_{\texttt{km}}$. The idea behind the `k-means` algorithm is to observe that the problem can be broken down into two much easier subproblems: if the *cluster assignement are fixed* (*i.e.*, each data point $\boldsymbol{x}_i$ is assigned to one index k), the optimal centroids are simply found by averaging their respective cluster; while if the *centroids are fixed*, the optimal cluster assignment is obviously to assign each sample to its closest centroid. Instead of solving both subproblems at once (which is hard), `k-means` thus alternates between solving each of them in turn (in a block coordinate descent style). This local search method is not guaranteed to find the optimal solution, but when combined with (possibly several trials of) a careful initialization [AV07], it performs well enough in practice to be the most widely used method [Jai10].

Note however that the complexity of `k-means` scales poorly with the size of modern voluminous datasets, where n is typically at least $\mathcal{O}(10^3 - 10^6)$, or even grows continually as in data streams (see Section 2.3). In fact, since `k-means` repeatedly requires—at each iteration—a thorough pass over $\mathcal{X}$, this massive dataset must be stored and read several times, with prohibitive memory and time consumption. Efficient implementations on large-scale data exist, but require dedicated GPUs [JDJ19]. Paradoxically, the large dataset size (*i.e.*, dn) dwarfs, and does not affect, the number of parameters learned by `k-means` (*i.e.*, dK). *Ideally, larger datasets increase the model accuracy without requiring more training computational resources*. This motivates the use of compressive learning for the k-means problem (see Section 2.5).

Density Fitting

In density fitting tasks, the goal is to somehow approximate the true data probability density $\mathcal{P}_0$; which we access only through the empirical distribution $\widehat{\mathcal{P}}_{\mathcal{X}}$. One thus seeks a parameterized probability distribution $\mathcal{P}_{\boldsymbol{\theta}}$

that is somehow "close"[26] to $\widehat{\mathcal{P}}_{\mathcal{X}}$. We discuss here two ways to construct this distribution: as a *mixture* of simple parametric distributions, or as a *generative neural network*.

(Gaussian) mixture models A *mixture model* is defined by K *components*, which are "simple" distributions $p_{\boldsymbol{\theta}_k}$ parameterized by $\boldsymbol{\theta}_k$, that are linearly combined, with weights $w_k \geq 0$ that sum to one $\boldsymbol{w}^\top \mathbf{1} = 1$. The probabilistic interpretation of a mixture model is that the weights w_k model the probability that a sample belongs to the k-th component, while $p_{\boldsymbol{\theta}_k}$ models the distribution of a sample given the fact that it belong to the k-th component. Gathering all the parameters into $\boldsymbol{\theta} := (\boldsymbol{w}, \{\boldsymbol{\theta}_k\}_{k=1}^K)$, the mixture is thus defined by $\mathcal{P}_{\boldsymbol{\theta}} := \sum_k w_k p_{\boldsymbol{\theta}_k}$.

The most popular mixture model is, by far, the *Gaussian mixture model* (GMM). Not very surprisingly given its name, in this model the components $p_{\boldsymbol{\theta}_k}$ are Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Gamma}_k)$, *i.e.*, $\boldsymbol{\theta}_k$ is given by a center $\boldsymbol{\mu}_k \in \mathbb{R}^d$, and a (symmetric and positive semi-definite) covariance matrix $\boldsymbol{\Gamma}_k \in \mathbb{R}^{d \times d}$. The mixture thus has a probability density function[27] is given by

$$\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{k=1}^{K} w_k p_{\mathcal{N}}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Gamma}_k),$$

where $p_{\mathcal{N}}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Gamma})$ is the probability density function the multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Gamma})$.

Having defined our model $\boldsymbol{\theta}$, we turn to the loss. The most popular criterion of "closeness" between the distribution to fit $\mathcal{P}_{\boldsymbol{\theta}}$ and the dataset $\mathcal{X}$ is the log-likelihood (LL) of the dataset $\mathcal{X}$ (this is known as *maximum likelihood estimation*). In the language of SL, maximizing the likelihood of a parameterized distribution with density function $\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$ is achieved by setting the loss to $\ell(\boldsymbol{x}, \boldsymbol{\theta}) = -\log(\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{x}))$. Indeed, when plugged into the ERM (2.4), this choice of loss is equivalent to

$$\begin{aligned} \widetilde{\boldsymbol{\theta}} \in \arg\max_{\boldsymbol{\theta} \in \Theta} \sum_{\boldsymbol{x}_i \in \mathcal{X}} \log(\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) &=: \arg\max_{\boldsymbol{\theta} \in \Theta} \mathrm{LL}(\boldsymbol{\theta}; \mathcal{X}) \\ &= \arg\max_{\boldsymbol{\theta} \in \Theta} \prod_{\boldsymbol{x}_i \in \mathcal{X}} \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \end{aligned} \tag{2.24}$$

where in the second expression we see that we indeed maximize the *likeli-*

[26] The notion of "closeness" between probability distributions is formalized in Section 2.4.

[27] Abusing notations, we sometimes denote a probability distribution and its probability density function (when it exists) by the same symbol, *e.g.*, $\mathcal{P}_{\boldsymbol{\theta}}$.

hood of observing the sample $\mathcal{X}$ assuming $\boldsymbol{\theta}$. For (Gaussian) mixture models in particular, the log-likelihood criterion reads

$$\mathrm{LL}(\boldsymbol{\theta}; \mathcal{X}) = \mathrm{LL}(\{w_k, \boldsymbol{\mu}_k, \boldsymbol{\Gamma}_k\}_k; \mathcal{X}) := \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} w_k p_{\mathcal{N}}(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Gamma}_k)\right). \tag{2.25}$$

The classical technique to solve mixture model problems is the *Expectation-Maximization* (`EM`) technique [Moo96]. Without entering into the details, it iterates by alternating between two steps: the "Expectation" step that computes, for the current values of the parameters $\boldsymbol{\theta}$, the probability of each sample $\boldsymbol{x}_i$ belongs to any of the K components; and the "Maximization" step that optimizes the parameters $\boldsymbol{\theta}$ given those probabilistic assignments.

To the attentive reader, `EM` should be reminiscent of the `k-means` *algorithm* described above. In fact, `k-means` is a particular case of `EM`, where the assignments of the samples to a component are "hard" (forced to be zero or one) as opposed to the general case which allows "soft" assignments. Crucially, the `EM` algorithm suffers from the same flaws that were discussed above in the context of `k-means`: it is a local search algorithm which may be very sensitive to the particular initialization choice, and requires many passes (at each iteration) over the entire dataset $\mathcal{X}$, which motivates a compressive learning alternative.

Remark 2.8. More generally, the k-means *problem* can actually be interpreted as particular density fitting task, where the components are Dirac deltas $p_{\boldsymbol{\theta}_k} = \delta_{\boldsymbol{c}_k}$. We will come back to this interpretation when we define *compressive k-means* in Section 2.5.

Generative Networks Another density-fitting-type model that will be studied in this thesis are *generative networks* (we'll further develop this topic in Chapter 6, but give a concise introduction here). Unlike mixture models, generative networks do not seek an explicit approximation to the true data distribution $\mathcal{P}_0$, but approximate it *implicitly* by mimicking the action of sampling from $\mathcal{P}_0$. More precisely, given a low-dimensional *latent space* $\Sigma_z \subset \mathbb{R}^p$, and a "simple" distribution $\mathcal{P}_z$ to generate samples in this latent space (*e.g.*, a standard Gaussian distribution), data samples are generated by $\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{z})$ where $\mathcal{G}_{\boldsymbol{\theta}} : \Sigma_z \to \Sigma$ is a trainable neural network (with weights $\boldsymbol{\theta}$) and $\boldsymbol{z} \sim \mathcal{P}_z$. From a technical point of view, the modeled distribution $\mathcal{P}_{\boldsymbol{\theta}}$ is thus the *pushforward* of $\mathcal{P}_z$ through $\mathcal{G}_{\boldsymbol{\theta}}$.

In practice however, we don't have access to $\mathcal{P}_{\boldsymbol{\theta}}$, but can only generate an empirical distribution $\widehat{\mathcal{P}}_{\boldsymbol{\theta}}$ by sampling n' examples $\mathcal{G}_{\theta}(\boldsymbol{z}_i)$ with $\boldsymbol{z}_i \sim_{\text{i.i.d.}} \mathcal{P}_z$ from it; we can write this empirical distribution as $\widehat{\mathcal{P}}_{\theta} :=$

$\mathcal{G}_\theta(\widehat{\mathcal{P}}_Z) = \frac{1}{n'}\sum_{i=1}^{n'} \delta_{\mathcal{G}_\theta(z_i)}$. The goal of a generative network is then to find the parameters $\boldsymbol{\theta}$ such that $\widehat{\mathcal{P}}_{\boldsymbol{\theta}} \approx \widehat{\mathcal{P}}_{\mathcal{X}}$, *i.e.*, *such that the set of generated samples "look like" the set of learning examples*. The different divergences that can be used to assess the "closeness" $\widehat{\mathcal{P}}_{\boldsymbol{\theta}} \approx \widehat{\mathcal{P}}_{\mathcal{X}}$ are explored in further detail in Section 2.4; for now we simply mention that the best-known technique, generative *adversarial* networks, learn this metric through a second "discriminator" neural network [G+14].

2.2 Signal processing

When talking about datasets in this thesis, we often refer to the learning examples $\boldsymbol{x}_i \in \Sigma \subset \mathbb{R}^d$ as "signals". While this term is maybe not commonly used in the machine learning literature, it is justified in the context of this thesis, as it emphasizes the fact that we use many techniques from the *signal processing* (SP) literature to manipulate those learning examples. Indeed, the field of signal processing tackles the question of optimal acquisition, encoding, transmission, and recovery of *signals*, such as audio, images, video, multispectral volumes, etc.

A bird-eye view of signal representations While most of those signals are associated with a continuous physical process, they have to be *sampled*, or *discretized*, by some sensor (*e.g.*, according to the Shannon-Nyquist rate) in order to be handled by digital devices. Therefore, in SP (and in this thesis) the signals of interest are more often than not represented by a vector $\boldsymbol{x} \in \mathbb{R}^d$ living in a d-dimensional space[28].

However, this "canonical" representation of the signal $\boldsymbol{x}$, in the so-called *ambient space* $\mathbb{R}^d$, is quite impractical to work with directly. Typically, the dimension d might be quite large, especially for "high-definition" signals (consider for example a standard-sized image of 512-by-512 pixels; it lives in the ridiculously large space of $\mathbb{R}^{262144}$). On top of this, the space of *informative* (or *natural*) signals is *much* smaller than the ambient space (to understand this intuitively, consider again the example of a 512-by-512 image: if we assign each pixel to an independent random value, which is equivalent to performing some sight-seeing in the ambient space, the probability to obtain a "natural image" that depicts something humanly intelligible is

[28]Here, the "dimension" d refers to the total amount of entries of the recorded signal. For example, an image whose size is N-by-N pixels is of course a two-dimensional signal, but its representation as a vector $\boldsymbol{x} \in \mathbb{R}^d$ lives in dimension $d = N^2$ (assuming the image vectorized, *e.g.*, by stacking the columns).

negligibly small). Put differently, informative signals, when expressed in the ambient space, are extremely *redundant*, which suggests that it might be wasteful to meticulously store and process their d coefficients.

One of the most crucial endeavors of SP is thus dedicated to finding more appropriate representations $\Phi(\boldsymbol{x})$, called *features*[29], of signals $\boldsymbol{x} \in \mathbb{R}^d$. To achieve this, it is helpful to first identify, as precisely as possible, the set $\Sigma \subset \mathbb{R}^d$ that contains (almost) all the natural signals of interest $\boldsymbol{x}$. Because it is much smaller than the ambient space, the set Σ is often called a *low-complexity set*, and the assumption $\boldsymbol{x} \in \Sigma$ is a "low-complexity prior". Equipped with this prior assumption, we can then construct a feature map $\Phi : \Sigma \to \mathcal{E}$ that facilitates the processing of signals while accurately encoding the geometry of the low-complexity set Σ—we say that Φ "embeds" the space Σ.

What's next We present in this section a few selected topics (not meant to be exhaustive) from the SP literature (and more specifically, from the subfield of SP that studies embeddings, following the framework we just described). In Subsection 2.2.1, we present the *sparsity* principle, one of the most popular low-complexity models Σ. Next, we turn to *compressive sensing* (CS) in Subsection 2.2.2, where the feature map $\Phi : \mathbb{R}^d \to \mathbb{R}^m$ often relies on *randomness* to simultaneously achieve signal acquisition and compression provided the signals of interest follow a given low-complexity prior. CS is the main inspiration for the compressive learning framework, as we will see in Section 2.5. We complete this section with a few words on *quantized embeddings* $\Phi : \mathbb{R}^d \to \{0,1\}^m$, which are of interest for Chapter 3.

2.2.1 Sparsity

Sparsity is a simple yet often quite accurate low-complexity prior [Mal99, FR17]. Under the sparsity assumption, any signal $\boldsymbol{x} \in \Sigma$ can be explained as the *linear combination* of $N \geq d$ elementary *components* $\boldsymbol{\psi}_j \in \mathbb{R}^d$ (also called basis elements or atoms), with the specificity that only a *small number* (say, K) of components contribute to the decomposition. More specifically, given $0 < K \ll d$ and a set of N elementary components $\boldsymbol{\Psi} = [\boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_N] \in \mathbb{R}^{d \times N}$, $\boldsymbol{x}$ is said to be K-sparse (in the basis, frame, or dictionary $\boldsymbol{\Psi}$), which

[29] Note that the role of "features" is here a bit different from, but related to, the role of features in machine learning (Section 2.1), which aim at unwrapping nonlinear structures in the data. This is connection is explicitly acknowledged in Chapter 3.

we denote $\boldsymbol{x} \in \Sigma_K^{\boldsymbol{\Psi}}$, if it can be written as

$$\boldsymbol{x} = \sum_{j=1}^{N} \alpha_j \boldsymbol{\psi}_j \quad \text{such that} \quad \|\boldsymbol{\alpha}\|_0 := |\{j \in [N], \alpha_j \neq 0\}| \leq K,$$

where the ℓ_0-"norm" $\|\boldsymbol{\alpha}\|_0$ counts the number of nonzero coefficients α_j. Sometimes, the signal of interest is not exactly sparse but *compressible*, meaning that it can be closely approximated by a sparse vectors; *e.g.*, if the approximation error decays "fast enough" with increasing K [SMF10].

Classical examples of *bases* $\boldsymbol{\Psi}$ (*i.e.*, for which $N = d$) include the Fourier basis, the Discrete Cosine Transform (on which the JPEG and MP3 compression are based, for example), the (quite rich[30]) family of wavelet bases [Dau88, Mal99]. Examples of redundant *dictionaries* $\boldsymbol{\Psi}$ (*i.e.*, for which $N > d$) include among many others, curvelets and shearlets. If a dataset of signals is available, it is also possible to perform *dictionary learning*[31] and to fit $\boldsymbol{\Psi}$ to the dataset [MBP14].

Sparsity as regularizer for inverse problems An important application of low-complexity models Σ appears in the context of (linear) *inverse problems*, where the goal is to recover an estimate $\widehat{\boldsymbol{x}}$ for a signal $\boldsymbol{x}_0 \in \mathbb{R}^d$ from measurements $\boldsymbol{y} \in \mathbb{R}^m$ of the type

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{e}, \tag{2.26}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times d}$ is a known linear model for the measurement process (the "forward model"), and $\boldsymbol{e} \in \mathbb{R}^m$ models a random additive corruption (*e.g.*, Gaussian noise). Classical examples of inverse problems include denoising, deconvolution (or deblurring), and inpainting. To solve the inverse problem, one might for example be tempted to try the following procedure[32]

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

However, without additional constraints, this problem is ill-posed, and will yield a poor estimate for $\boldsymbol{x}_0$. Thus, the problem is *regularized* by means of the low-complexity set Σ *i.e.*,

[30] `http://tinyurl.com/wits-wavelets-starlet`

[31] The dictionary problem can be formulated as an instance of our generic linear decomposition task (2.22).

[32] Depending on the noise distribution, other data fidelity penalties might be considered, such as the ℓ_1 norm.

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\Sigma} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2.$$

More generally, one considers a *regularization* term $\rho(\boldsymbol{x})$ in the objective function (possibly weighted by a regularization strength λ), that encourages solutions $\boldsymbol{x} \in \Sigma$. For example, for K-sparse vectors $\Sigma_K := \Sigma_K^{\boldsymbol{I}_d}$, one would ideally set $\rho(\boldsymbol{x}) = \|\boldsymbol{x}\|_0$. However, there is a catch.

Sparse recovery methods Optimizing over the ℓ_0-norm requires, intuitively speaking, to check out all the $\binom{d}{K}$ possible supports of $\boldsymbol{x}$, which grows combinatorially with the size of the problem; more rigorously, this problem is known to be NP-hard [Nat95]. The sparse recovery problem is thus usually not solved exactly; two broad approaches are used instead: *convex relaxations* and *greedy algorithms*.

The idea of *convex relaxations* is to consider the ℓ_1-norm instead of the ℓ_0-norm, which still tends to promote sparse solutions. Typically, one might consider either the (constrained) Lasso[33]

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{x}\|_1 \leq \tau,$$

or the (Lagrangian) Basis Pursuit DeNoise (BPDN) program [CDS01]

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1.$$

The advantage of this approach is that $\|\cdot\|_1$, although nondifferentiable, is a *convex* function, which implies that there are efficient algorithms to find the global minimizer, *e.g.*, proximal methods [PB14].

On the other hand, *greedy algorithms* are local search methods (*i.e.*, that iteratively improve a candidate solution, but without concern for the global optimization landscape), that heuristically try to solve (for example) the following non-convex ℓ_0-optimization problem (where K is fixed *a priori*)

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{x}\|_0 \leq K.$$

For example, Iterative Hard Thresholding [BD09] alternates between a gra-

[33] When we defined the Lasso problem in machine learning (2.14), we actually wrote its Lagrangian formulation, which is here called BPDN. In general, the term Lasso is used in the literature to denote both the constrained form and the Lagrangian form. On the other hand, the name BPDN is sometimes used to denote a constrained version BPDN, but where the constraints are now imposed on $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2 \leq C$; this is all quite confusing, but in the end it doesn't matter, as all those approaches are roughly equivalent.

dient descent step (which decreases the norm residual $\boldsymbol{r} := \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}$) and a hard thresholding step (which keeps the K largest magnitude coefficients in $\boldsymbol{x}$, *de facto* enforcing K-sparsity).

Another family of popular greedy algorithms for sparse recovery are Matching Pursuit [MZ93] (MP) and its further improvements, such as Orthogonal Matching Pursuit [PRK93] (OMP), and Orthogonal Matching Pursui with Replacement (OMPR) [JTD11]. The idea of MP-based approaches is to initialize a solution with an empty support $\text{supp}\,(\boldsymbol{x}^{(0)}) = \emptyset$, which is then iteratively refined. In particular, at each iteration t, we greedily select the most promising index i_t (*i.e.*, the atom $\boldsymbol{a}_{i_t}$ that is the most correlated with the residual), and add it to the support of our solution, *i.e.*, $\text{supp}\,(\boldsymbol{x}^{(t)}) = \{i_t, i_{t-1}, ...\}$; the best approximation within this given support can then be computed efficiently. As we will see in Section 2.5, the main compressive learning algorithm, `CL-OMP(R)`, is based on this strategy.

2.2.2 Compressive Sensing

We have established that natural signals belong to a low-complexity set Σ (*e.g.*, the set of K-sparse signals in some appropriate representation), *i.e.*, have a small *intrinsic dimension*, which means that they can be represented by a relatively small number of coefficient (*e.g.*, $\boldsymbol{\alpha}$ with $\|\boldsymbol{\alpha}\|_0 \leq K \ll d$). The main motivation for Compressive Sensing (CS) [CRT06, Don06, FR17] is the following observation: since such signals can be summarized as a few coefficients, it is somehow wasteful to invest a lot of "resources" (see Remark 2.9) in the high-definition acquisition of $\boldsymbol{x}_0 \in \mathbb{R}^d$, as most of those coefficients are redundant and will be thrown away by a compression procedure.

CS thus aims at *simultaneously compressing and sensing* the signal of interest. More precisely, instead of acquiring $\boldsymbol{x}_0$ directly, we collect *compressive measurements* $\boldsymbol{y} \in \mathbb{R}^m$, whose forward model reads as in (2.26), *i.e.*,

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{e},$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times d}$ is a sensing operation that must be properly designed, which should satisfy $m \ll d$. The philosophy is to thus *sample only what you need*, *i.e.*, sampling at m proportional to the intrinsic "information rate" of $\boldsymbol{x}_0$, which might be much smaller than its Nyquist rate d.

Remark 2.9. The term "resources" is here a big vague, and can translate to different physical quantities depending on the specific CS application (*e.g.*,

the energy, time, hardware, etc. required for acquisition). Crucially, such gains must be balanced with the drawback that the signal reconstruction procedure (*e.g.*, sparse recovery) from the compressed measurements is often quite demanding, which could in some cases nullify the gains of CS at acquisition time. One typical example where CS has a clear advantage is compressive sensing MRI [LDSP08, JFL15], where the saved "resource" of interest is the MRI scan time, which is particularly costly, and justifies a longer reconstruction time. We find it important to emphasize this tradeoff underlying any CS application, because it similarly applies to compressive learning as well.

The power of randomness The main question becomes: how should one design the sensing operation $\boldsymbol{A}$? One of the most striking messages of compressive sensing is that this operation should be *randomly generated* (as a typical example, the entries of the matrix are Gaussian $A_{i,j} \sim_{\text{i.i.d.}} \mathcal{N}(0, \frac{1}{m})$). The obtained measurements, $y_j = \langle \boldsymbol{a}_j, \boldsymbol{x}_0 \rangle$ are also called *random projections*.

This fact can be quite intriguing; why would it not be possible to construct $\boldsymbol{A}$ deterministically, where a randomized approach works? Referring the reader to [FR17] for an in-depth explanation of this counter-intuitive fact, we here provide an instructive illustration, coming from a classical result closely related to CS: the *Johnson–Lindenstrauss (JL) lemma* [JL84].

Theorem 2.10 (JL lemma)**.** *Given a point cloud* $\Sigma = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\} \subset \mathbb{R}^d$, i.e., *a finite set of N points in* $\mathbb{R}^d$*, and a multiplicative distortion* $\epsilon \in (0, 1)$*, if the target embedding dimension m satisfies* $m \geq C\frac{\log(N)}{\epsilon^2}$*, there exists a linear map* $f : \mathbb{R}^d \to \mathbb{R}^m$ *that preserves the relative distances in* Σ *up to a distortion* ϵ, i.e.,

$$(1-\epsilon)\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \leq \|f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)\|_2 \leq (1+\epsilon)\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2, \quad \forall \boldsymbol{x}_i, \boldsymbol{x}_j \in \Sigma.$$

The detail that is of importance here is that the JL lemma only states the *existence* of a map f, but not *what* this map actually is. More specifically, the existence of f is shown by the *probabilistic method* [AS04]. Unlike *constructive* methods that show the existence of f by providing a method to construct it, the probabilistic method rather shows that, if f is randomly generated from some well-crafted probability distribution (*e.g.*, $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ with $A_{i,j} \sim_{\text{i.i.d.}} \mathcal{N}(0, \frac{1}{m})$), then there is a nonzero probability that f satisfies the desired property (here, the embedding of a point cloud up to a distortion ϵ), which implies that such an f must necessarily exist. Similarly, Compressive Sensing results are "nonconstructive", in the sense that they

do not provide an explicit construction for $\boldsymbol{A}$, but a random generation procedure.

A nice property of the JL lemma is that the embedding dimension m does *not* depend on the ambient space d, but only the "complexity" of the set Σ, here measured by its cardinality (which related to its Kolmogorov entropy, see Chapter 3). The idea of CS is to extend this fact to other low-complexity sets Σ such as the set of K-sparse vectors; this is (often) formalized as a Restricted Isometry Property, as explained below.

Recovery guarantees Let us summarize the main ingredients of CS: we design a *random sensing operator* $\boldsymbol{A}$ (*e.g.*, a Gaussian matrix), and obtain compressed measurements $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{e}$. To obtain a reconstruction $\widehat{\boldsymbol{x}} \simeq \boldsymbol{x}_0$, we find an appropriate *low-complexity prior* Σ, and obtain $\widehat{\boldsymbol{x}} := \Delta(\boldsymbol{y}, \boldsymbol{A}, \Sigma)$. The "decoder" Δ is an algorithm that (approximately) solves the inverse problem regularized by Σ (*e.g.*, using one of the sparse recovery techniques from Subsec. 2.2.1). The main objective of CS theory is then to find *guarantees* that quantitatively explain how close the reconstruction $\widehat{\boldsymbol{x}}$ is from $\boldsymbol{x}_0$ (as a function of the number of measurements m). In general, such guarantees take the form of an Instance Optimality Property [BDP+14, KG18]:

$$\|\widehat{\boldsymbol{x}} - \boldsymbol{x}_0\| \leq c_1 D(\boldsymbol{x}_0, \Sigma) + c_2 \|\boldsymbol{e}\|, \tag{2.27}$$

where $c_1, c_2 > 0$ are constants, and $\|\cdot\|$ are (possibly two different) norms (or pseudometrics). The reconstruction error is thus the sum of two contributions:, a *noise term* proportional to $\|\boldsymbol{e}\|$ (robustness), and a *modeling error* $D(\boldsymbol{x}_0, \Sigma)$ (stability). The modeling error captures the "distance" between $\boldsymbol{x}_0$ and the model set Σ, which satisfies in particular $D(\boldsymbol{x}_0, \Sigma) = 0$ if the modeling is exact, *i.e.*, $\boldsymbol{x}_0 \in \Sigma$ (*e.g.*, for sparse signals $\Sigma = \Sigma_K$, $D(\boldsymbol{x}_0, \Sigma) = \|\boldsymbol{x}_0 - \boldsymbol{x}_K\|$ with $\boldsymbol{x}_K$ is the best K-sparse approximation to $\boldsymbol{x}_0$).

Obtaining guarantees through the (L)RIP Such guarantees can usually be established by proving that the random operator $\boldsymbol{A}$ satisfies—with high probability—a specific property known as the Restricted Isometry Property (RIP). In a nutshell, the RIP ensures that the distances between points of Σ (where traditionally $\Sigma = \Sigma_K$) are preserved under the map $\boldsymbol{A}$ (similarly to the JL lemma, Thm. 2.10), *i.e.*, $\boldsymbol{A}$ is an embedding of Σ. In fact, it was shown [BDP+14, KG18] that the instance optimality recovery property is guaranteed[34] whenever $\boldsymbol{A}$ satisfies the *Lower RIP* (LRIP), defined as

[34] Although the efficient implementation of the decoder Δ can still be an issue.

follows[35]: we say that $\boldsymbol{A}$ has the Lower RIP over the set Σ, with constant $\gamma > 0$, if

$$\|\boldsymbol{x} - \boldsymbol{x}'\| \leq \gamma \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{x}'\|, \quad \forall \boldsymbol{x}, \boldsymbol{x}' \in \Sigma. \tag{2.28}$$

This can be intuitively understood as the fact that two signals $\boldsymbol{x}, \boldsymbol{x}'$ that are distinguishable in the ambient space remains distinguishable after being embedded by $\boldsymbol{A}$ (a smaller constant γ ensures a better control of the distance distortion, which translates in lower constants c_1 and c_2 in (2.27)).

To guarantee successful CS recovery, it thus remains to prove that $\boldsymbol{A}$ has the (L)RIP. One classical way to achieve this, as explained in [BDDW08], is to first prove the RIP for a finite set of points (*e.g.*, as in the JL lemma), and to then extend this property by continuity; this strategy is further explained in Chapter 3. In the end, one obtains *probabilistic guarantees*, where the RIP hold with probability $1 - \delta$ for some small probability of failure δ, provided that the number of measurements grows as $m \geq m(\delta, \gamma, \dim \Sigma)$, where $\dim \Sigma$ is a quantity that captures the intrinsic dimension of the low-complexity set Σ (*e.g.*, Kolmogorov entropy [Pis99], upper box-counting dimension [PDG17], Gaussian mean width [CRPW12]).

Let us finally mention that many sensing operations $\boldsymbol{A}$ exist beyond the dense Gaussian case, which can be quite cumbersome in practice: *e.g.*, generic i.i.d. sub-Gaussian entries [MPTJ08], sub-sampled Fourier [Rau10] or Hadamard [MBDJ20], etc. Of particular interest sensing operations that can be (at least conceptually) be implemented in *hardware*, such as the one-pixel camera [DDT+08], random convolutions CMOS imagers [JVB+09], as well as the very promising Optical Processing Unit that performs random projections optically in a scattering medium [SCC+16].

2.2.3 Quantized embeddings

We opened this section by explaining that, to be processed on a digital computer, signals had to be *discretized* (*i.e.*, sampled at a finite number of points), *e.g.*, by an Analog-to-Digital Converter (ADC), possibly combined with a compressive sensing mechanism, which yields measurements $\boldsymbol{y} \in \mathbb{R}^m$. However, computers also have to work with a *finite-precision* representation of numbers, which means that those measurements must further be *quantized* to be stored and processed, which can be modeled as a map $q : \mathbb{R}^m \to \mathcal{E}$ where $\mathcal{E}$ is a finite set (in particular, to fit onto a b-bit

[35] Following [KG18], the expression provided here is *not at all* (but of course still related to) the "classical" formulation of the RIP, which is usually defined on Σ_K and not pairs in $\Sigma \times \Sigma$, with squared distances, and involves a multiplicative distortion $(1 \pm \delta)$.

representation, the encoding must satisfy $|\mathcal{E}| = 2^b$). Of course quantization induces an approximation error, which can be mitigated at the price of increasing the amount of bits b (according to the *rate-distortion* curve). In order to get the most out of the b-bit representation, a careful design and study of the quantization operation q is required, which is thus an important research area [Sha48, GN98].

Quantized CS Reconciling this quantization with compressive sensing theory, *i.e.*, *quantized compressive sensing* (QCS), is a problem that has generated a lot of interest. In particular, one-bit CS, where $\mathcal{Q} : \mathbb{R}^m \to \{0,1\}^m$ (*e.g.*, up to a rescaling, $q(\cdot) = \text{sign}(\cdot)$), has been particularly intensively. We refer the reader to [BJKS15, Dir19] for broad overviews of this field.

One important trick that we will borrow from this literature, is that of *dithering* (see, *e.g.*, [XJ20, DS20]). This idea states that, to mitigate the discontinuous nature of the quantizer q, it might in some cases be useful to add a controlled random vector $\boldsymbol{\xi} \in \mathbb{R}^m$, called the *dither*, prior to the quantizer. The main idea is that, if the distribution of $\boldsymbol{\xi} \in \mathbb{R}^m$ is well-chosen, then it allows to *mitigate the effect of quantization on average*.

Universal Quantized Embeddings Among QCS approaches, of particular interest for this thesis (*i.e.*, for Part I) is the so-called *(one-bit) universal embedding*[36], studied by Boufounos et al. in [Bou12, BM15, BRM17]. This approach considers a *periodic* quantizer q, corresponding to the least significant bit of a standard uniform quantizer. Without loss of generality, we can assume that this quantizer has a fixed step-size of π (*i.e.*, it is periodic with period 2π) and is re-scaled in $\{\pm 1\}$, and define it as

$$q(\cdot) := \text{sign}(\cos(\cdot)).$$

Boufounos et al. showed that the resulting embedding, when combined with Gaussian random projections and a uniform dithering, *i.e.*,

$$\Phi(\boldsymbol{x}) := q(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{\xi}) \quad \text{where} \quad A_{i,j} \sim_{\text{i.i.d.}} \mathcal{N}(0, \sigma^{-2}),\ \xi_j \sim_{\text{i.i.d.}} \mathcal{U}([0, 2\pi)),$$

preserves the *local* distances in the ambient space, that is, the distances up to a given threshold $D_0 \propto \sigma$. To be more precise, this embedding satis-

[36] The name "universal" here refers to the fact that the quantization does not depend on the given signal, as opposed to *e.g.*, $\Sigma\Delta$ quantization [STN96]. As there are many other "universal" quantized embeddings, in my humble opinion, that name could have been better, such as maybe "local embedding"?

fies [Bou12]

$$\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}')\|_2 \simeq c\|\boldsymbol{x} - \boldsymbol{x}'\|_2 \quad \text{provided} \quad \|\boldsymbol{x} - \boldsymbol{x}'\|_2 \leq D_0 = c'\sigma,$$

where $c, c' > 0$ are absolute constants, and we note that $\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}')\|_2$ is here equivalent to the *Hamming distance* in the embedded space. Note that the scale σ generating the random projections directly controls the radius up to which the distances are preserved. Assuming an application where only the local distances are important, the intuitive advantage of this approach is that the bit budget is allocated to encoding *only the distances that matter*.

In order to study the precise distance-preserving capabilities of this embedding (*e.g.*, for a finite value of m), the authors of [BRM17] actually study a generalization of this scheme, where q is *any* periodic function f, *i.e.*,

$$\Phi_f(\boldsymbol{x}) = f(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{\xi}).$$

Note that when $f(\cdot) = \exp(\mathrm{i}\cdot)$, this approach recovers the random Fourier features (see Subsection 2.1.4); in this work, we thus dub this approach *random periodic features* (RPF). We study RPF extensively in Chapter 3, and sketches obtained by averaging RPF in Chapter 4.

2.3 Massive data synopses

As explained in Chapter 1, *massive data* fuel an increasingly large fraction of modern systems. These applications must thus be able to handle *large-scale datasets* $\mathcal{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n \,|\, \boldsymbol{x}_i \in \mathbb{R}^d\}$, *i.e.*, that present at least one, and possibly both, of the following characteristics: *(i)* the samples are high-dimensional, $d \gg 1$ (*e.g.*, high-definition images count millions of pixels); and *(ii)* the amount of learning samples is very large[37], $n \gg 1$ (*e.g.*, an image recognition software might have to handle millions of pictures). The exact definition of how "large" d or n should be to qualify as a "large-scale dataset" varies according to the relative amount of resources available to process the data, but to give a rough idea, one would be justified to call data large-scale as soon it cannot fit in memory, *e.g.*, if $d \geq 10^3$ or $n \geq 10^6$.

The challenges of massive data When presented with such massive data, traditional methods (*i.e.*, designed for data of small to moderate scale) of-

[37] As we will see, Compressive Learning is especially promising for this case in particular.

ten fail spectacularly. For example, the amount of memory required by the data-processing routine might exceed by far the physically available amount. Even if memory constraints can be met, the computing time required to process the data volume might be exceedingly large (say, months or years). Algorithms for massive data are thus subject to *strict limits on the allowed computational (*i.e., *time- and space-) complexities*. Focusing only on the number of samples n for the moment, polynomial complexities such as $\mathcal{O}(n^p)$ with power $p > 1$ are often prohibited in this context, although they are considered benign in other areas of computer science [AB09]. Linear complexity methods, *i.e.*, $\mathcal{O}(n)$, might not even be feasible if they require multiple "passes" over the dataset; sublinear complexity being then a *sine qua non* requirement to handle the data. Similar problems apply with respect to the dimension d, when dealing with high-dimensional data[38].

While the raw computational complexity is the quintessential challenge of large-scale data, other difficulties can also typically arise in this context. A first one is the case of *distributed data* [PKP06, ALNT14], where the dataset $\mathcal{X}$ is not stored at one single location but shared across $p > 1$ distinct devices, *i.e.*, $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup ... \cup \mathcal{X}_p$, where $\mathcal{X}_i$ is the dataset owned by the i-th data holder. For example, the whole dataset might simply be too big to fit in the memory of the device that has to do the computations, or the sub-datasets $\mathcal{X}_i$ are too costly to be transmitted (*e.g.*, through a communication network) to that device, or maybe those sub-datasets cannot be shared directly due to privacy concerns (this scenario is discussed in Chapter 5). In any case, the data analysis must deal with the distributed nature of data, *e.g.*, by several rounds of communication with the different data holders.

Another important massive data setting is that of *data streams* [GZK05, Agg07], where new data samples are perpetually collected; typical examples of this setting include network traffic analysis, sensor networks, web queries, the entire collection of Youtube videos [CAS16], and in general any application that requires monitoring some form of perpetual data flow. The dataset $\mathcal{X}$ is thus not fixed but always growing, which can be thought of as $n \to \infty$ (an algorithm that scales with $\mathcal{O}(n)$ is therefore not an option). Of course, in practice $\mathcal{X}$ cannot ever be stored at all, and the new examples must be handled "on the fly" as soon as they become available, before being discarded (note that this constitutes a *single pass* over the dataset).

[38] Note that this computational problem is distinct from the *curse of dimensionality*, discussed in Section 2.1, although they of course often appear together.

Data synopses Since traditional approaches fail on large-scale data, numerous and diverse research efforts are dedicated to design alternative methods, specialized to handle the large-scale setting. Their common[39] basic idea is the following: since the dataset $\mathcal{X}$ is too large to solve the target task directly, break the problem down into two easier steps. First, we *summarize* (or *compress*) $\mathcal{X}$ to a much smaller data structure $S(\mathcal{X})$, called the *data synopsis*[40]. Then, *the desired problem is solved using only the synopsis*, which is significantly cheaper if $S(\mathcal{X})$ is much "smaller" than $\mathcal{X}$.

For this generic approach to be useful, two conditions have to be met: *(i)* the synopsis $S(\mathcal{X})$ must be very *efficient* to compute and store (*e.g.*, it should not require more than a single pass on $\mathcal{X}$); and *(ii)* the synopsis must *accurately* encode the information that we seek to recover. To tradeoff between both of these conflicting objectives, a recurring theme in data synopses is to introduce *randomness* in S: by sacrificing a bit of accuracy, in the form of a *probabilistic error*, the compression procedure can be made more efficient.

As illustrated Fig. 2.3, the synopsis operation S usually applies one (or a combination) of three basic techniques: *dimensionality reduction*, *subsampling*, and *sketching*. We briefly describe those techniques in Subsections 2.3.1, 2.3.2, and 2.3.3, respectively. More information on those subject can be found, among others, in [CDK17, BHK20, Phi16].

2.3.1 Dimensionality reduction

The idea of dimensionality reduction is to *compress each sample* in the dataset individually, which provides a new dataset of size $n \times d'$ for some embedding dimension $d' \ll d$. Given a dimensionality reduction map $\Phi : \mathbb{R}^d \to \mathbb{R}^{d'}$, the synopsis is thus given by

$$S(\mathcal{X}) = \mathcal{Y} := (\boldsymbol{y}_1, ..., \boldsymbol{y}_n) \quad \text{where} \quad \boldsymbol{y}_i = \Phi(\boldsymbol{x}_i) \in \mathbb{R}^{d'}.$$

Dimensionality reduction techniques are often further classified into *data-independent* and *data-dependent* approaches, depending on whether or

[39] Some techniques, such as MapReduce [DG10], do not follow the "synopsis" idea presented here. Instead, they seek efficient data communication protocols to still handle the full-scale dataset, by relying on the "brute force" of cluster computing. Those approaches are however out of the scope of this work.

[40] We borrow the term "synopsis" from [CGHJ12], which reviews the field of *Approximate Query Processing* (AQP). AQP mainly tackles SQL-like queries (*e.g.*, counting the data samples that share some specific attributes) on large-scale data; here the context is slightly more general, and "data synopses" also includes other tools for massive data, such as coresets.

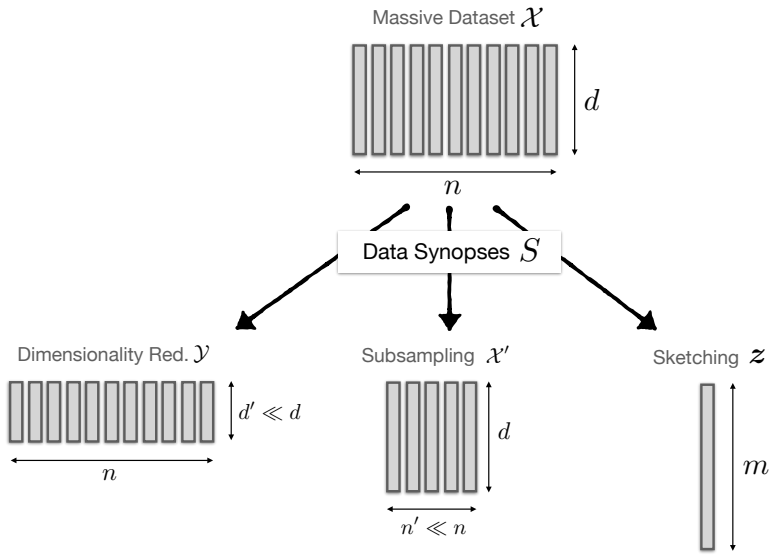


Fig. 2.3 Three basic techniques to construct synopses of massive data (n samples in dimension d): *dimensionality reduction* which reduces the dimension to $d' \ll d$ for each data sample individually, *subsampling* which represents the entire dataset by only $n' \ll n$ a few representative samples, and *sketching* which constructs an easy-to-update data summary of size m.

not the dataset $\mathcal{X}$ is used in the design the map Φ.

Data-independent approaches We already encountered the most prominent data-independent dimensionality reduction technique, the Johnson-Lindenstrauss lemma (Theorem 2.10), and its implementation by *random projections* [XLX17] in particular: *i.e.*, $\Phi(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ for $\boldsymbol{A} \in \mathbb{R}^{d' \times d}$ a random matrix, possibly structured to ensure fast transforms [CRW17]. To further compress the representation, this is sometimes combined with quantization (in the spirit of QCS), possibly involving Locally Sensitive Hashing techniques [IM98].

Many signal processing tasks (*e.g.*, detection, filtering) can be performed *directly* in the compressed domain (*i.e.*, on the random projections $\boldsymbol{y}_i$ instead of the high-dimensional signals $\boldsymbol{x}_i$) [DBWB10]. This idea was also applied in the context of machine learning, *e.g.*, for SVM[41] [CJS09] or regression [MM09]. Solving k-means clustering from compressed signals also received a lot of attention, see *e.g.*, [BZD10, NSW17, Dup18].

Data-dependent approaches Fitting Φ to the specific dataset at hand is done by solving an *unsupervised learning problem*, such as those discussed in Subsection 2.1.5. Typically, PCA is used to project the data to a lower dimension while preserving most of its variance. More sophisticated, non-linear dimensionality reduction approaches can also be considered, such as t-SNE [VdMH08], manifold unfolding techniques [LV07], or autoencoder networks [BGC17]. Although data-dependent approaches tend to better capture the relevant data geometry (given a fixed target dimension d'), they can be infeasible, *e.g.*, if the dataset is not available in advance or cannot be processed efficiently—which is here the point of computing a synopsis in the first place.

Obviously, dimensionality reduction techniques are especially useful when the large-scale component of the dataset is primarily the dimension d. Conversely, since the synopsis is still constituted of n distinct data vectors, this approach is ill-suited to problems where the number of samples n is particularly large—which is remedied by the two other basic techniques.

[41] The idea of learning from compressed signals is sometimes (among others, in [CJS09, TYD+20]) called "compressive learning". That line of work should not be confused with "our" notion of compressive (statistical) learning, where (as explained in Sec. 2.5) the linear compression operation acts not on individual signals but on entire probability distributions (*e.g.*, the empirical distribution associated with the dataset).

2.3.2 Subsampling

Generally speaking, subsampling is the converse idea of dimensionality reduction: to summarize the dataset, represent it by a smaller subset of $n' \ll n$ examples:

$$S(\mathcal{X}) = \mathcal{X}' := (\boldsymbol{x}'_1, ..., \boldsymbol{x}'_{n'}) \quad \text{where} \quad \boldsymbol{x}'_i \in \mathbb{R}^d.$$

To enrich this type of representation, *weights* $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_{n'}) \in \Delta_{n'}$ are also often assigned to the samples, depending on their relative importance.

Random sampling The most direct method to produce such a synopsis is to *sample at random* from the original dataset $\mathcal{X}$, *i.e.*, $\boldsymbol{x}'_i = \boldsymbol{x}_{j_i}$ where j_i denotes the index that was chosen for the ith subsample, here randomly generated (*e.g.*, uniformly, $j_i \sim_{\text{i.i.d.}} \mathcal{U}([n])$). While this idea sounds simple, it can already lead to nontrivial tradeoffs (*e.g.*, should one sample with or without replacement?), and can be refined by more sophisticated techniques such as stratified sampling; see [CGHJ12, Chapter 2] for a review of "direct" subsampling in AQP.

Coresets Another important type of "subsampling-based" synopses are *coresets*; we present the idea only briefly here, referring the interested reader to detailed introductions to the subject in *e.g.*, [Phi16, BLK17, JMF19]. In contrast with the generic random subsampling approach, the philosophy of coresets is to somehow tailor the subset $\mathcal{X}'$ towards a specific application (typically one machine learning task in particular). In other words, *from the point of view of the machine learning task*, the datasets $\mathcal{X}'$ and $\mathcal{X}$ should be almost identical. Given a task defined by a loss $\ell(\boldsymbol{\theta}, \boldsymbol{x})$, a good coreset should for example aim at approximating the empirical risk objective

$$\textstyle\sum_{i=1}^{n'} \alpha_i \ell(\boldsymbol{\theta}, \boldsymbol{x}'_i) \simeq \sum_{i=1}^{n} \frac{1}{n} \ell(\boldsymbol{\theta}, \boldsymbol{x}_i).$$

The subsamples are either elements from the original dataset $\boldsymbol{x}'_i \in \mathcal{X}$ (*e.g.*, by a random sampling biased towards samples that contribute significantly to the risk objective), or are "free parameters" $\boldsymbol{x}'_i \in \mathbb{R}^d$ that can be tuned during the coreset construction (*i.e.*, generalized coresets).

Because they are specialized to the task at hand, coresets can often lead to much smaller subsamples while incurring only a small loss of learning performance—although this must be carefully balanced with the complexity of building the coreset in the first place. Another family of synopsis

constructions which tends to be more efficient with respect to this aspect is the one of sketching methods.

2.3.3 Sketching

Both dimensionality reduction and subsampling synopses approximate $\mathcal{X}$ by a "smaller dataset" (by acting on the dimension d or number of samples n, respectively), that can then (almost) directly be fed to a classical learning algorithm. In contrast, the *sketching* approach compresses the dataset as a more generic structure[42], here *w.l.o.g.* presented as a vector $\boldsymbol{z}$, from which the desired parameters $\boldsymbol{\theta}$ must be extracted by a different algorithm Δ:

$$S(\mathcal{X}) = \boldsymbol{z} := \boldsymbol{z}(\mathcal{X}) \in \mathbb{R}^m \quad \text{where} \quad \boldsymbol{\theta} = \Delta(\boldsymbol{z}).$$

Many different definitions of "sketches" exist in the literature, as illustrated by several reviews [CGHJ12, ACH+13, Phi16, BHK20], but it is commonly agreed upon that they should satisfy (some of) the following properties:

- *easy to store and process*: the sketch should have a small size with respect to the dataset size $m \ll nd$, such that $\boldsymbol{z}$ takes little space in memory, and queries for $\Delta(\boldsymbol{z})$ are fast;
- *easy to update*: when presented with a new learning example $\boldsymbol{x}'$, the *updated sketch* $\boldsymbol{z}(\mathcal{X} \cup \{\boldsymbol{x}'\})$ can be cheaply computed from $\boldsymbol{z}(\mathcal{X})$ and $\boldsymbol{x}'$, which allows to handle *data streams* efficiently;
- *easy to merge*: when two datasets $\mathcal{X}_1$ and $\mathcal{X}_2$ must be combined, the *merged sketch* $\boldsymbol{z}(\mathcal{X}_1 \cup \mathcal{X}_2)$ can be cheaply computed from their respective sketches $\boldsymbol{z}(\mathcal{X}_1)$ and $\boldsymbol{z}(\mathcal{X}_2)$, which allows to handle *distributed data* efficiently.

Historically, sketching methods focused on simple queries, as illustrated by their classic examples: the Morris sketch for counting the size of the dataset n with space complexity $\mathcal{O}(\log(\log(n)))$ [Mor78], the Bloom filter for testing whether or not one specific example is in the database [Blo70], the Flajolet-Martin sketch and its HyperLogLog improvement for counting the number of distinct elements in the dataset [FM85, FFGM07], the Count-Min sketch for detecting "heavy hitters" (elements in the database that are the most frequent) [CM05], etc.

[42] Although we here give a specific meaning to it, "sketching" is a vague term used quite inconsistently in the literature to denote many different techniques, and should thus be interpreted carefully (*e.g.*, it sometimes denotes what we here called dimensionality reduction, but is also sometimes a synonym for "streaming algorithms", two fairly different concepts).

Linear sketches A generic class of sketches that are both easy to update and easy to merge are *linear sketches*, defined by[43] the property that the merged sketch is the sum of its sub-sketches, *i.e.*, $\boldsymbol{z}(\mathcal{X}_1 \cup \mathcal{X}_2) = \boldsymbol{z}(\mathcal{X}_1) + \boldsymbol{z}(\mathcal{X}_2)$. This sketch is necessarily a "sum-of-features" operation: for some feature map $\Phi : \mathbb{R}^d \mapsto \mathbb{R}^m$, the associated linear sketch is

$$\boldsymbol{z}(\mathcal{X}) = \sum_{i=1}^{n} \Phi(\boldsymbol{x}_i). \tag{2.29}$$

A well-known example of linear sketch is the *histogram* of the dataset, which is obtained by setting (assuming $d = 1$ for simplicity) $\Phi_j(x) = \iota(b_j \leq x < b_{j+1}) \in \{0, 1\}$ for b_j the edges of the histogram bins (assumed fixed in advance); see [CGHJ12, Chapter 3]. Another popular example are wavelet synopses of the dataset, where $\Phi_j(\boldsymbol{x})$ are wavelet basis elements[44] over $\mathbb{R}^d$; see [CGHJ12, Chapter 4].

However, such sketches scale exponentially with the dimensionality d. One way to circumvent such scalability issues is to consider a *randomized map* Φ. For example, the recently proposed RACE sketch considers locality-sensitive hashing features [CS20b]. This type of sketch construction is in fact exactly the paradigm followed by compressive learning—but before delving into the details of CL, remark how the linear sketch constructions discussed here all aim at somehow representing the *data distribution* $\mathcal{P}_0$ instead of the dataset $\mathcal{X}$. In the next section, we thus first detail the meaning of "approximating a probability distribution".

2.4 Probability measures geometry

The philosophy of compressive learning is to focus on accurately "representing" the probability distribution of the data instead of the individual signals in the dataset. A last broad topic of interest to understand CL is thus what we will call[45] the *geometry of probability measures*, which seeks to answer the following questions: given two probability distributions, how "similar" are they to one another? Pragmatically, how can we encode (or embed) those distributions such that this similarity, or "geometry", is pre-

[43] We give here again one specific meaning to "linear sketch", but this term can have different meanings in the literature.

[44] Anticipating connections from Section 2.5, wavelet sketches are nothing but the wavelet representation of a signal, where the signal in question is the (empirical) probability distribution of the data, considered up to some finite resolution.

[45] There isn't really a cohesive "geometry of probability measures" research field; what we present here are broad questions studied in statistics, applied mathematics, and data science.

served?

In Subsection 2.4.1, after formalizing this question, we present two broad families of *divergences and metrics* on probability measures (*i.e.*, ways to encode a "dissimilarity"), illustrated by a few classic examples. In Subsection 2.4.2 we take a closer look at one specific metric of interest for CL, namely the *Maximum Mean Discrepancy*. To complete our tour of probability measures geometry, we discuss a few related approaches in Subsection 2.4.3, *i.e.*, *information geometry*, *optimal transport*, and the *generalized method of moments*.

2.4.1 Problem statement and overview

Probability measures Most probability courses start in the *discrete* world, inhabited by coins, dices and cards: random variables X that can take values only from a *finite*[46] set, $\{\boldsymbol{s}_1, ..., \boldsymbol{s}_N\}$. Such random variables are fully characterized by their *probability mass function*, or simply *probability distribution*, a vector $\boldsymbol{p}$ whose entries correspond to the probability that each possible outcome happens, *i.e.*, $p_i = \mathbb{P}[X = \boldsymbol{s}_i]$. Owing to the probability axioms, the entries of $\boldsymbol{p}$ are nonnegative and must sum to one; the space of all possible probability distribution is called the *probability simplex* Δ_N,

$$\boldsymbol{p} \in \Delta_N := \{\boldsymbol{\alpha} \in \mathbb{R}^N, \alpha_i \geq 0, \textstyle\sum_{i=1}^N \alpha_i = 1\}.$$

When the random variable X is *continuous*, which can take values in some infinite set Σ, it can sometimes still be represented by a probability density function (PDF) $p_X(\boldsymbol{x})$, which can then be integrated (with respect to the Lebesgue measure) to give the probability that X belongs to a specific set $S \subseteq \Sigma$, *i.e.*, $\int_S p_X(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \mathbb{P}[X \in S]$.

In general, not all distributions have a PDF, but are instead characterized by the more general notion of *probability measure*. Roughly speaking, a measure $\mathcal{P} \in \mathcal{M}(\Sigma)$ defined on the set Σ is a mathematical object such that one can integate (in the Lebesgue sense) any (well-behaved) continuous function $f : \Sigma \to \mathbb{R}$ with respect to it [PC+19], *i.e.*,

$$\textstyle\int_\Sigma f(\boldsymbol{x})\mathrm{d}\mathcal{P}(\boldsymbol{x}) \in \mathbb{R}. \tag{2.30}$$

Similarly to the definition of the probability simplex, a measure is a *probability* measure if it is moreover *(i)* positive, *i.e.*, for all $S \subseteq \Sigma$, $\mathcal{P}(S) =$

[46] To be correct, discrete random variables can also be defined over infinite but countable sets, but we focus on the finite case for now.

$\int_S \mathrm{d}\mathcal{P}(\boldsymbol{x}) = \mathbb{P}[X \in S] \geq 0$ (the probability of any event S is non-negative), and *(ii)* normalized such that $\int_\Sigma \mathrm{d}\mathcal{P}(\boldsymbol{x}) = \mathbb{P}[X \in \Sigma] = 1$ (the total "probability mass" is one). The set of all probability measures on Σ is then noted $\mathcal{M}_+^1(\Sigma)$. Note that if the distribution has a PDF $p_X(\boldsymbol{x})$, we can write $\mathrm{d}\mathcal{P}(\boldsymbol{x}) = p_X(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$.

One elementary probability measure of great interest is the *Dirac measure*, an infinitely concentrated probability mass. The Dirac measure located at $\boldsymbol{c} \in \Sigma$, noted $\delta_{\boldsymbol{c}} \in \mathcal{M}_+^1(\Sigma)$, is such that for all continuous f, we have $\int_\Sigma f(\boldsymbol{x})\mathrm{d}\delta_{\boldsymbol{c}}(\boldsymbol{x}) = f(\boldsymbol{c})$. We already encountered it in the context of the empirical distribution of a dataset $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$, namely $\widehat{\mathcal{P}}_{\mathcal{X}} := \frac{1}{n}\sum_{i=1}^n \delta_{\boldsymbol{x}_i}$.

Divergences and metrics on probability measures Given two probability measures, say, $\mathcal{P}$ and $\mathcal{Q}$, the broad question of "probability measures geometry" is to quantify how different $\mathcal{P}$ and $\mathcal{Q}$ are: we are interested in a notion of "dissimilarity" $D : \mathcal{M}_+^1(\Sigma) \times \mathcal{M}_+^1(\Sigma) \to \mathbb{R}_+ : (\mathcal{P}, \mathcal{Q}) \mapsto D(\mathcal{P}, \mathcal{Q})$.

To be more precise, the function D is usually either a (statistical) *divergence* [PC+19], for which $D(\mathcal{P}, \mathcal{Q}) = 0 \Leftrightarrow \mathcal{P} = \mathcal{Q}$ but that do not necessarily satisfy symmetry (in general, $D(\mathcal{P}, \mathcal{Q}) \neq D(\mathcal{Q}, \mathcal{P})$) nor the triangle inequality; or a *(pseudo)metric* [S+10], which is symmetric, satisfies the triangle inequality, and for which $\mathcal{P} = \mathcal{Q} \Rightarrow D(\mathcal{P}, \mathcal{Q}) = 0$, but which does not necessarily satisfy $D(\mathcal{P}, \mathcal{Q}) = 0 \Rightarrow \mathcal{P} = \mathcal{Q}$ (only if D is a "true" metric).

Two important families Many popular statistical divergences belong to the family of *ϕ-divergences*, also known as Csiszar divergences [Csi67] or Ali-Silvey distances [AS66]. Intuitively speaking, such divergences compare the probability measures "point-wise" by averaging their "odds ratio"[47] $\frac{\mathrm{d}\mathcal{P}}{\mathrm{d}\mathcal{Q}}$. If the measures $\mathcal{P}$, $\mathcal{Q}$ have PDFs p_X and p_Y, this amounts to compare $p_X(\boldsymbol{x})$ and $p_Y(\boldsymbol{x})$ at every $\boldsymbol{x}$, by averaging the ratio of the densities $\frac{p_X(\boldsymbol{x})}{p_Y(\boldsymbol{x})}$. Moreover, this ratio is weighted by an *entropy function* ϕ, which is required to be convex and such that $\phi(1) = 0$ (when the odds ratio is one, there is no contribution to the divergence). The ϕ-divergence thus reads

$$D_\phi(\mathcal{P}, \mathcal{Q}) := \int_\Sigma \phi\left(\frac{\mathrm{d}\mathcal{P}}{\mathrm{d}\mathcal{Q}}\right)\mathrm{d}\mathcal{Q}, \tag{2.31}$$

where, to ensure $\frac{\mathrm{d}\mathcal{P}}{\mathrm{d}\mathcal{Q}}$ is well-defined, one usually sets $D_\phi(\mathcal{P}, \mathcal{Q}) = +\infty$ whenever $\mathcal{P}$ is not absolutely continuous with respect to $\mathcal{Q}$[48]. If the mea-

[47] This quantity is known as the Radon–Nikodym derivative.
[48] See [PC+19, Chapter 8] for a more formal discussion.

sures $\mathcal{P}$, $\mathcal{Q}$ have PDFs p_X and p_Y, this happens if the support of p_X is not a subset of that of p_Y, *i.e.*, $\exists \boldsymbol{x}$ such that $p_X(\boldsymbol{x}) > p_Y(\boldsymbol{x}) = 0$.

Remark 2.11. In the better-known discrete setting, if we have two probability distributions $\boldsymbol{p}, \boldsymbol{q} \in \Delta_N$, the ϕ-divergence is given by

$$D_\phi(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^N \phi(\tfrac{p_i}{q_i}) q_i,$$

with $D_\phi(\boldsymbol{p}, \boldsymbol{q}) = +\infty$ if for some i, $p_i > q_i = 0$.

Another important family of dissimilarities on probability measures $\mathcal{M}_+^1(\Sigma)$ is the one of *Integral Probability Metrics* (IPM) [Mül97]. Intuitively, the underlying idea is to use (2.30), the fact that measures can be integrated against any continuous functions f, and to compare the integrals obtained with both $\mathcal{P}$ and $\mathcal{Q}$, for all "test" functions f in a well-chosen family. More precisely, given a class $\mathcal{F}$ of bounded and measurable functions $f : \Sigma \to \mathbb{R}$, the related integral probability metric $\gamma_\mathcal{F}$ is

$$\gamma_\mathcal{F}(\mathcal{P}, \mathcal{Q}) := \sup_{f \in \mathcal{F}} \left| \int_\Sigma f \mathrm{d}\mathcal{P} - \int_\Sigma f \mathrm{d}\mathcal{Q} \right|. \tag{2.32}$$

As explained in [S+10], integral probability metrics are in general *pseudo-metrics*, but there are many choices of the family of functions $\mathcal{F}$ that ensure $\gamma_\mathcal{F}$ is a proper metric (such that $\gamma_\mathcal{F}(\mathcal{P}, \mathcal{Q}) = 0 \Rightarrow \mathcal{P} = \mathcal{Q}$). The relationship between ϕ-divergences and integral probability metrics is explored in [SFG+09].

Classic examples One of the most famous divergences is the *Kullback-Leibler* (KL) divergence $D_{KL}(\mathcal{P}, \mathcal{Q})$, also known as relative entropy[49], obtained by picking the usual Shannon entropy function [Sha48], *i.e.*, $\phi(t) = t \log(t)$. A "symmetrized" version of the KL divergence (which is itself a ϕ-divergence for an appropriate ϕ) that is often used is the Jensen-Shannon divergence $D_{JS}(\mathcal{P}, \mathcal{Q}) = \frac{1}{2}\left(D_{KL}(\mathcal{P}, \mathcal{R}) + D_{KL}(\mathcal{Q}, \mathcal{R})\right)$, where $\mathcal{R} = \frac{\mathcal{P}+\mathcal{Q}}{2}$.

An often encountered distance (true metric) between probability measures is the *total variation*, which is intuitively the largest difference between probabilities that any event S occurs under both probability distributions, *i.e.*, $D_{TV}(\mathcal{P}, \mathcal{Q}) = \sup_{S \subset \Sigma} |\mathcal{P}(S) - \mathcal{Q}(S)|$. It belongs both to the family of ϕ-divergences (with $\phi(t) = |t - 1|$) and integral probability metrics (with $\mathcal{F} = \{f, \|f\|_\infty \leq 1\}$).

[49] In information theoretic terms, this is the difference between the cross-entropy of $\mathcal{P}, \mathcal{Q}$ and the entropy of $\mathcal{P}$, *i.e.*, $H(\mathcal{P}, \mathcal{Q}) - H(\mathcal{P})$.

2.4.2 Maximum Mean Discrepancy

One metric of specific interest to compressive learning is the *Maximum Mean Discrepancy* (MMD) metric, introduced in [GBR+12]. The high-level idea of the MMD is to specify a kernel κ to compare points in the signal space Σ (see Subsection 2.1.3), and to assign a distance $D_\kappa(\mathcal{P}, \mathcal{Q})$ measuring how much pairs of signals differ (as captured by κ) on average (defined by $\mathcal{P}, \mathcal{Q}$). Mathematically, the MMD $D_\kappa(\mathcal{P}, \mathcal{Q})$ is obtained as an instance of the IPM family (2.32), where the family $\mathcal{F} = \{f \in \mathcal{H}_\kappa, \|f\|_{\mathcal{H}_\kappa} \leq 1\}$ are bounded functions in the RKHS $\mathcal{H}_\kappa$ associated with κ [GBR+12].

Kernel Mean Embeddings The MMD is actually best understood when expressed differently: intuitively, it maps the probability distributions to a nicer space, the *reproducible kernel Hilbert space* $\mathcal{H}_\kappa$, and to compares them in this embedded space. More formally, given a kernel κ, the associated *kernel mean embedding* is the map $\mu_\kappa : \mathcal{M}_+^1(\Sigma) \to \mathcal{H}_\kappa$ defined by [SGSS07]

$$\mu_\kappa(\mathcal{P}) := \int_\Sigma \kappa(\boldsymbol{x}, \cdot)\, \mathrm{d}\mathcal{P}(\boldsymbol{x}). \tag{2.33}$$

The associated maximum mean discrepancy can then be written as the distance in the embedded space [GBR+12, Lemma 4], which is easier to interpret as averages of pointwise kernel evaluations [GBR+12, Lemma 6]:

$$\begin{aligned} D_\kappa^2(\mathcal{P}, \mathcal{Q}) &= \|\mu_\kappa(\mathcal{P}) - \mu_\kappa(\mathcal{Q})\|_{\mathcal{H}_\kappa}^2 \\ &= \mathop{\mathbb{E}}_{\substack{\boldsymbol{x}\sim\mathcal{P} \\ \boldsymbol{x}'\sim\mathcal{P}}} \kappa(\boldsymbol{x}, \boldsymbol{x}') - 2 \mathop{\mathbb{E}}_{\substack{\boldsymbol{x}\sim\mathcal{P} \\ \boldsymbol{y}\sim\mathcal{Q}}} \kappa(\boldsymbol{x}, \boldsymbol{y}) + \mathop{\mathbb{E}}_{\substack{\boldsymbol{y}\sim\mathcal{Q} \\ \boldsymbol{y}'\sim\mathcal{Q}}} \kappa(\boldsymbol{y}, \boldsymbol{y}'). \end{aligned} \tag{2.34}$$

Remark 2.12. When $\Sigma = \mathbb{R}^d$ and the kernel is shift-invariant $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa^\Delta(\boldsymbol{x} - \boldsymbol{x}')$, there is a particularly intuitive interpretation of the MMD: it corresponds to the plain L_2 distance of the *convolutions* of the two probability measures by $\kappa^\Delta(\boldsymbol{u})$, *i.e.*,

$$D_\kappa^2(\mathcal{P}, \mathcal{Q}) = \int |\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{P}} \kappa^\Delta(\boldsymbol{x} - \boldsymbol{u}) - \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{P}} \kappa^\Delta(\boldsymbol{y} - \boldsymbol{u})|^2\, \mathrm{d}\boldsymbol{u} = \|\kappa^\Delta * \mathcal{P} - \kappa^\Delta * \mathcal{Q}\|_{L^2}^2.$$

Also, due to Bochner's theorem $\kappa^\Delta = \mathcal{F}^{-1}\Lambda$, the MMD can be expressed in the Fourier domain: writing $\varphi_\mathcal{P}(\boldsymbol{\omega}) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\, e^{\mathrm{i}\boldsymbol{\omega}^\top \boldsymbol{x}}$ the characteristic

function (inverse Fourier transform) of the distribution $\mathcal{P}$,

$$D_\kappa^2(\mathcal{P}, \mathcal{Q}) = \int |\varphi_\mathcal{P}(\boldsymbol{\omega}) - \varphi_\mathcal{Q}(\boldsymbol{\omega})|^2 \, \mathrm{d}\Lambda(\boldsymbol{\omega}). \tag{2.35}$$

The MMD thus captures the (pointwise) difference between the characteristic functions $\varphi_\mathcal{P}(\boldsymbol{\omega})$ and $\varphi_\mathcal{Q}(\boldsymbol{\omega})$, weighted by the distribution $\Lambda(\boldsymbol{\omega})$. In short, from a signal processing point of view, the MMD compares *low-pass filtered* versions of the probability distributions $\mathcal{P}$ and $\mathcal{Q}$.

Characteristic kernels Note that in general, the MMD is a *pseudo*metric. Kernels κ which ensure the D_κ is a proper metric (*i.e.*, $D_\kappa(\mathcal{P}, \mathcal{Q}) = 0 \Rightarrow \mathcal{P} = \mathcal{Q}$) are called *characteristic kernels* [FSGS08]. For example, in $\Sigma = \mathbb{R}^d$, if the kernel is shift-invariant $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa^\Delta(\boldsymbol{x} - \boldsymbol{x}')$, it is characteristic if and only if its Fourier transform $\Lambda = \mathcal{F}\kappa^\Delta$ is supported on the whole space $\mathbb{R}^d$ [S+10].

We refer the interested reader to [MFSS17] for an complete introduction and review on the subject of kernel mean embeddings and the MMD.

2.4.3 Other topics

For completeness, we briefly mention a few other techniques to deal with the generic problem of "probability measure geometry".

Information Geometry The field of *Information Geometry* [Nie20] formally studies the geometry of probability measures using tools from differential geometry, *i.e.*, it explicitly acknowledges, and deals with, the manifold structure of $\mathcal{M}_+^1(\Sigma)$ (possibly with some additional parameterization). This approach leads to a natural notion distance, the *Fisher information metric* [Rao45], and other useful notions such as gradients on $\mathcal{M}_+^1(\Sigma)$.

Optimal Transport In a nutshell, the goal of *Optimal Transport* [Vil08, PC+19] is to solve *mass transportation problems*, specified by a notion of "cost" $c(\boldsymbol{x}, \boldsymbol{x}')$ which represents the price to move one unit of mass from $\boldsymbol{x}$ to $\boldsymbol{x}'$. This approach leads to several important and elegant definitions of distances on (probability) measures; the most popular is undoubtedly the *Wasserstein distance*, also known as *earth mover's distance*. They are an instance of IPM, where $\mathcal{F}$ is the set of $1-$Lipschitz functions.

Generalized Method of Moments We are given a parametric distribution $\mathcal{P}_{\boldsymbol{\theta}} \in \mathcal{M}_+^1(\Sigma)$ that must be fitted to a dataset $\mathcal{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\} \subset \mathbb{R}^d$. To estimate the parameter $\boldsymbol{\theta}$, a well-known alternative to Maximum Likelihood (*e.g.*, when the likelihood is not available) is the *Generalized Method of Moments* (GeMM) [Hal05]. The main idea of this technique is to craft a vector-valued function $\Phi : \mathbb{R}^d \to \mathbb{R}^m$ and to select the parameters that best match the empirical generalized moments[50] defined by Φ, *i.e.*,

$$\widehat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}\in\Theta} \| \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}_{\boldsymbol{\theta}}} \Phi(\boldsymbol{x}) - \tfrac{1}{n}\sum_{i=1}^n \Phi(\boldsymbol{x}_i)\|_2^2 \tag{2.36}$$

This amounts to pick the distribution $\mathcal{P}_{\boldsymbol{\theta}}$ that is the *closest* to the empirical distribution $\widehat{\mathcal{P}}_{\mathcal{X}}$ in the sense defined by the "distance" $\|\mathcal{P}_{\boldsymbol{\theta}} - \widehat{\mathcal{P}}_{\mathcal{X}}\|_\Phi := \|\mathcal{A}_\Phi(\mathcal{P}_{\boldsymbol{\theta}}) - \mathcal{A}_\Phi(\widehat{\mathcal{P}}_{\mathcal{X}})\|_2$, where $\mathcal{A}_\Phi(\mathcal{P}) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}} \Phi(\boldsymbol{x})$ is the operation that computes the generalized moments of $\mathcal{P}$. This will become clearer in the next subsection, where we will see that compressive learning follows almost exactly the same prinicple, with the twist that Φ is *randomly generated*—in contrast with GeMM that considers a carefully crafted Φ.

2.5 Compressive Learning

Having discussed the various disciplines that inspired compressive learning, we can now fulfill the promise from Fig. 2.1 and present CL while drawing insights from those disciplines. We describe the general framework in Subsection 2.5.1, give practical instances of it in Subsection 2.5.2, and discuss theoretical learning guarantees in Subsection 2.5.3

2.5.1 The general compressive statistical learning framework

Sketching phase The general compressive learning framework was introduced in [KBGP18] and further developed in [GBKT17] by Keriven, Gribonval, and coauthors. Their main inspiration is to generalize the framework of compressive *sensing* (see Subsec. 2.2.2), which compressively senses a *vector signal* $\boldsymbol{x}_0 \in \mathbb{R}^d$ by a random linear operation $\boldsymbol{A} : \mathbb{R}^d \to \mathbb{R}^m$, to the case where the sensed signal of interest is a *probability measure* $\mathcal{P}_0 \in \mathcal{M}_+^1(\Sigma)$ (see Subsec. 2.4.1). The random linear "sensing operator" is thus a map $\mathcal{A} : \mathcal{M}_+^1(\Sigma) \to \mathbb{C}^m$ that "compresses" any probability distribu-

[50] The "usual" moments of a (one-dimensional) distribution are $\mathbb{E}_{X\sim\mathcal{P}} X^p$ for $p = 1, 2, ...$, which are a particular case of the generalized moments, with $\Phi(\cdot) = (\cdot)^p$.

tion[51] $\mathcal{P} \in \mathcal{M}_+^1(\Sigma)$ as $\mathcal{A}(\mathcal{P}) \in \mathbb{C}^m$, a finite-dimensional vector of measurements[52]. By analogy with the literature on sketching methods in data synopses (see Subsec. 2.3.3), the vector of compressive measurements $\mathcal{A}(\mathcal{P})$ is called the *sketch* of $\mathcal{P}$, and $\mathcal{A}$ is coined the sketching operator. More precisely, this operator is defined as the *average of some feature map* $\Phi : \Sigma \to \mathbb{C}^m$.

Definition 2.13 (Sketching operator)**.** Given a *feature map* $\Phi : \Sigma \to \mathbb{C}^m$, the related *sketching operator* $\mathcal{A}_\Phi : \mathcal{M}_+^1(\Sigma) \to \mathbb{C}^m$ is defined as

$$\mathcal{A}_\Phi(\mathcal{P}) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\, \Phi(\boldsymbol{x}) = \int_\Sigma \Phi(\boldsymbol{x}) \mathrm{d}\mathcal{P}(\boldsymbol{x}). \tag{2.37}$$

We mentioned above that, by analogy with CS, the sketching operator $\mathcal{A}_\Phi$ is linear and random. Here, *linearity* refers to the fact that, by linearity of the expectation, $\mathcal{A}_\Phi$ is linear in the probability distributions[53], *i.e.*, $\mathcal{A}_\Phi(\alpha\mathcal{P}_1 + (1-\alpha)\mathcal{P}_2) = \alpha\mathcal{A}_\Phi(\mathcal{P}_1) + (1-\alpha)\mathcal{A}_\Phi(\mathcal{P}_2)$ for any two distributions $\mathcal{P}_1, \mathcal{P}_2$ and mixture coefficient $\alpha \in [0,1]$. Moreover, *randomness* refers to the fact that in CL the map Φ is randomly generated (unlike in GeMM [Hal05] for example), as will be clarified in the next section. We say that the sketch computes *random feature moments* of the distribution.

The goal of compressive learning is to solve machine learning tasks (Section 2.1), *i.e.*, to infer parameters $\boldsymbol{\theta}$ from a (compressed version of) *datasets* $\mathcal{X}$. The *sketch of the dataset* is defined as the sketch operator acting on the empirical distribution $\widehat{\mathcal{P}}_\mathcal{X}$ associated with this dataset.

Definition 2.14 (Sketch of a dataset)**.** Given a feature map $\Phi : \Sigma \to \mathbb{C}^m$ and a dataset $\mathcal{X} = \{\boldsymbol{x}_i \in \Sigma\}_{i=1}^n$, the sketch of this dataset, noted[54] $\boldsymbol{z}_{\Phi,\mathcal{X}}$, is

$$\boldsymbol{z}_{\Phi,\mathcal{X}} := \mathcal{A}_\Phi(\widehat{\mathcal{P}}_\mathcal{X}) = \tfrac{1}{n}\textstyle\sum_{i=1}^n \Phi(\boldsymbol{x}_i) \in \mathbb{C}^m, \tag{2.38}$$

i.e., the empirical average of the features $\Phi(\boldsymbol{x}_i)$ of the dataset.

The computation of $\boldsymbol{z}_{\Phi,\mathcal{X}}$ can be performed very efficiently on large-scale datasets, thanks to the independence of the contribution by each data sample. Indeed, this computation requires only a *single pass* over the dataset (after which $\mathcal{X}$ can be discarded), and is highly ("embarassingly")

[51] Omitted here is the fact that more generally, the sketch operator can be defined on arbitrary signed measures, see [KBGP18].

[52] While here for convenience the resulting sketch is a complex vector $\boldsymbol{z} \in \mathbb{C}^m$, this is not necessarily always the case, as real-valued sketches $\boldsymbol{z} \in \mathbb{R}^m$ can be considered as well.

[53] In particular, the feature map Φ is allowed to (and will indeed) be nonlinear.

[54] When there is no ambiguity, we drop the subscripts Φ or $\mathcal{X}$ from $\mathcal{A}_\Phi$ and $\boldsymbol{z}_{\Phi,\mathcal{X}}$.

parallelizable. Moreover, it is especially well-suited to deal with *streaming* and *distributed datasets*, as explained in Subsection 2.3.3.

Remark 2.15 ($z_{\Phi,\mathcal{X}}$ as a "linear sketch"). The sketch defined by (2.38) is not, strictly speaking, a *linear sketch* in the sense defined in Subsection 2.3.3, since it is the *average* of features instead of the *sum* as required by (2.29). It can however still be assimilated to a linear sketch—and its advantages, such as easy *update* and *merging* operations—if we consider the map $\overline{\Phi}(\boldsymbol{x}) := (\Phi(\boldsymbol{x}), 1)$. This gives after summation (2.29), a tuple $(\sum_i \Phi(\boldsymbol{x}_i), n)$, which is indeed a linear sketch, from which the "average" sketch (2.38) can later be computed[55].

Learning phase The ultimate objective in compressive learning is to solve a given machine learning task using only the sketch $\boldsymbol{z}$. As explained by (2.3) in Section 2.1, ML tasks are often formulated as the fitting of some parameter vector $\boldsymbol{\theta} \in \Theta$ to the true data-generating distribution $\mathcal{P}_0$, following the *risk minimization principle*. Given a loss $\ell(\boldsymbol{\theta}, \boldsymbol{x})$, we seek the parameters $\boldsymbol{\theta}^*$ which minimize the average loss, or risk, according to $\mathcal{P}_0$, *i.e.*,

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}\in\Theta} \mathcal{R}(\boldsymbol{\theta}; \mathcal{P}_0), \quad \text{where} \quad \mathcal{R}(\boldsymbol{\theta}; \mathcal{P}_0) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}_0} \ell(\boldsymbol{\theta}, \boldsymbol{x}).$$

Following through with the compressive sensing analogy, CL formulates this learning phase as an *inverse problem* (see Subsec. 2.2.1). Roughly speaking, the idea is to assimilate[56] a distribution $\mathcal{P}_{\boldsymbol{\theta}} \in \mathcal{M}_+^1(\Sigma)$ to any parameter vector $\boldsymbol{\theta} \in \Theta$. Intuitively, $\mathcal{P}_{\boldsymbol{\theta}}$ describes what the data distribution should look like under the assumption that $\boldsymbol{\theta}$ models the data perfectly.

As in a usual inverse problem, compressive learning then selects the parameters $\widehat{\boldsymbol{\theta}}$ that best fits the "measurements" $\boldsymbol{z}$, as captured by "sketch matching" cost function $\mathcal{C}(\boldsymbol{\theta}; \boldsymbol{z})$:

$$\widehat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}\in\Theta} \mathcal{C}(\boldsymbol{\theta}; \boldsymbol{z}), \quad \text{where} \quad \mathcal{C}(\boldsymbol{\theta}; \boldsymbol{z}) := \|\boldsymbol{z} - \mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}})\|_2. \tag{2.39}$$

Why should this principle work? As will be further developed in Subsec. 2.5.3, the main explanation is that the sketch matching cost $\mathcal{C}(\boldsymbol{\theta}; \boldsymbol{z})$ can be seen as a *surrogate* to the risk objective $\mathcal{R}(\boldsymbol{\theta}; \mathcal{P}_0)$. If the sketch is well-designed and is sufficiently large, we can expect that (roughly speaking) $\mathcal{C}(\boldsymbol{\theta}; \boldsymbol{z}) \simeq \mathcal{R}(\boldsymbol{\theta}; \mathcal{P}_0)$, which ensures that the compressive learning solution

[55] A similar trick will be employed in Chapter 5 to decouple the impact of a new sample as one contribution to the sum-of-features and one to the dataset size.

[56] In general the idea is in fact a bit subtler than that, as explained in Remark 2.16.

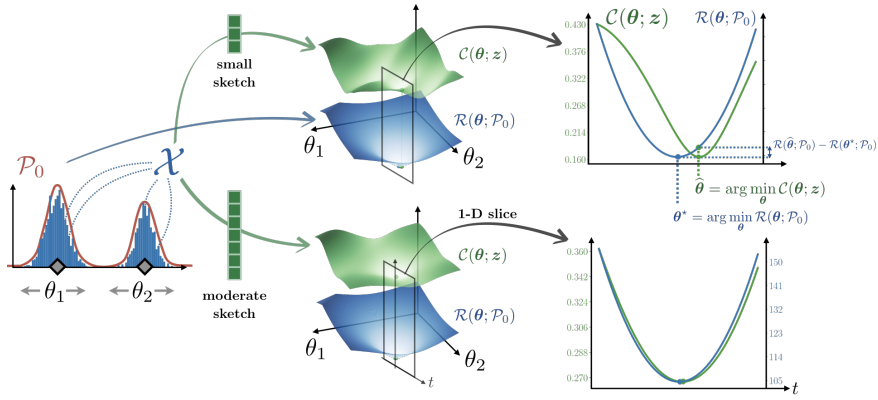


Fig. 2.4 The sketch matching cost $\mathcal{C}(\boldsymbol{\theta}; \boldsymbol{z})$ as surrogate for the true risk $\mathcal{R}(\boldsymbol{\theta}; \mathcal{P}_0)$. Here, the k-means problem with $k = 2$ centroids $\{\theta_1, \theta_2\}$ is considered on a 1-d dataset $\mathcal{X}$ (left). The risk $\mathcal{R}(\boldsymbol{\theta}; \mathcal{P}_0)$ (which defines the true optimal centroids $\boldsymbol{\theta}^*$) is shown in blue. The sketch matching cost $\mathcal{C}(\boldsymbol{\theta}; \boldsymbol{z})$ is shown in green, for a small (top row) and moderate (bottom row) sketch size m. As can be seen on the slice of the objective functions on the right, when the sketch size increases, the approximation $\mathcal{C}(\boldsymbol{\theta}; \boldsymbol{z}) \simeq \mathcal{R}(\boldsymbol{\theta}; \mathcal{P}_0)$ improves, and the compressive learning solution $\widehat{\boldsymbol{\theta}}$ approaches $\boldsymbol{\theta}^*$.

$\widehat{\boldsymbol{\theta}}$ is a good estimate of the desired true risk minimizer $\boldsymbol{\theta}^*$. This idea is illustrated by Fig. 2.4.

Remark 2.16 (Semi-parametric compressive learning). As explained by Sheehan et al. in [SGD19], for *semi-parametric modeling* tasks (such as PCA), the map $\boldsymbol{\theta} \mapsto \mathcal{P}_{\boldsymbol{\theta}}$ is not necessarily well-defined (*e.g.*, many densities $\mathcal{P}_{\boldsymbol{\theta}}$ may equivalently be considered for a given model $\boldsymbol{\theta}$). This is not *per se* an issue, as in practice we do not actually need $\boldsymbol{\theta} \mapsto \mathcal{P}_{\boldsymbol{\theta}}$, but rather $\boldsymbol{\theta} \mapsto \mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}})$, which is indeed (*e.g.*, for PCA) a proper function [GBKT17, Remark 2.1]. The reformulation proposed in [SGD19] clarifies this issue by mapping $\boldsymbol{\theta}$ to a unique *statistic* $\boldsymbol{\theta} \mapsto \Sigma_{\boldsymbol{\theta}}$ instead of $\mathcal{P}_{\boldsymbol{\theta}}$.

2.5.2 Practical applications: sketch constructions and learning algorithms

We now instantiate the generic framework described above to the two most important tasks considered in this thesis: compressive k-means [KTTG17] and compressive Gaussian mixture modeling [KBGP18], which both rely on random Fourier features to build the sketch. Other tasks and sketch constructions are discussed at the end of this subsection.

Sketching with random Fourier features Recall that random Fourier features (see Subsec. 2.1.4) are constructed by the projection on a matrix $\boldsymbol{\Omega} = (\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_m)$ of random vectors $\boldsymbol{\omega}_j \sim_{\text{i.i.d.}} \Lambda$[57], followed by complex exponentiation, *i.e.*,

$$\Phi_{\text{RFF}}(\boldsymbol{x}) := \tfrac{1}{\sqrt{m}} \exp(\mathrm{i}\boldsymbol{\Omega}^\top \boldsymbol{x}).$$

In [KBGP18], the authors propose to use $\Phi = \Phi_{\text{RFF}}$ to construct the sketch operator, which gives

$$\mathcal{A}(\mathcal{P}) = \tfrac{1}{\sqrt{m}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\, e^{\mathrm{i}\boldsymbol{\Omega}^\top \boldsymbol{x}} = \tfrac{1}{\sqrt{m}} [\varphi_{\mathcal{P}}(\boldsymbol{\omega}_j)]_{j=1}^m,$$

where $\varphi_{\mathcal{P}}(\boldsymbol{\omega}) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\, e^{\mathrm{i}\boldsymbol{\omega}^\top \boldsymbol{x}}$ is the *characteristic function* of $\mathcal{P}$. We observe that the RFF sketch thus corresponds to random sampling of $\mathcal{P}$ in the Fourier domain, which is sampled at the m frequencies $\boldsymbol{\omega}_j \sim_{\text{i.i.d.}} \Lambda$.

Remark 2.17 (RFF sketch as an embedding that preserves the MMD distance). The random Fourier features sketch operator $\mathcal{A}$ can be seen as a finite-dimensional approximation to the *kernel mean embedding* (2.33) that is defined by the shift-invariant kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa^\Delta(\boldsymbol{x} - \boldsymbol{x}') = (\mathcal{F}^{-1}\Lambda)(\boldsymbol{x} -$

[57] Where we recall that the distribution Λ can be related to a shift-invariant kernel κ through Bochner's theorem.

$\boldsymbol{x}'$) [KBGP18, Section 4]. In particular, by virtue of (2.35), the Euclidean distance between RFF sketches $\|\mathcal{A}(\mathcal{P}) - \mathcal{A}(\mathcal{Q})\|_2$ approximates ("embeds") the MMD metric $D_\kappa(\mathcal{P}, \mathcal{Q})$, as

$$\begin{aligned}\|\mathcal{A}(\mathcal{P}) - \mathcal{A}(\mathcal{Q})\|_2^2 &= \frac{1}{m}\sum_{j=1}^{m} |\varphi_{\mathcal{P}}(\boldsymbol{\omega}_j) - \varphi_{\mathcal{Q}}(\boldsymbol{\omega}_j)|^2 \\ &\simeq \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim\Lambda} |\varphi_{\mathcal{P}}(\boldsymbol{\omega}) - \varphi_{\mathcal{Q}}(\boldsymbol{\omega})|^2 = D_\kappa^2(\mathcal{P}, \mathcal{Q}).\end{aligned}$$

Compressive learning of mixture models Consider a generic mixture model fitting problem (see Subsec. 2.1.5), where the density to fit is a mixture of K simple components $p_{\boldsymbol{\theta}_k}$ weighted by $\alpha_k \geq 0$; the parameter vector is thus $\boldsymbol{\theta} := (\boldsymbol{\alpha}, \{\boldsymbol{\theta}_k\}_{k=1}^K)$. In this case, the map $\boldsymbol{\theta} \mapsto \mathcal{P}_{\boldsymbol{\theta}}$ (which should assign a distribution to the model $\boldsymbol{\theta}$) is simply the mixture itself, *i.e.*,

$$\mathcal{P}_{\boldsymbol{\theta}} = \textstyle\sum_{k=1}^K \alpha_k p_{\boldsymbol{\theta}_k}. \tag{2.40}$$

This setting covers both *compressive Gaussian mixture modeling* [KBGP18], where $p_{\boldsymbol{\theta}_k} = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Gamma}_k)$, and *compressive k-means* [KTTG17], where $p_{\boldsymbol{\theta}_k} = \delta_{\boldsymbol{c}_k}$ (*i.e.*, a weighted mixture of Dirac deltas located at the centroid positions). Note that for compressive k-means, the sketch of each component is given directly by $\mathcal{A}(p_{\boldsymbol{\theta}_k}) = \Phi(\boldsymbol{c}_k)$. Another instance of this setting are mixtures of alpha-stable distributions [KDL18].

Note that (2.40) is a *sparsity assumption* (see Subsec. 2.2.1), as it assumes that the signal of interest (the probability distribution) is a linear combination of a small number K of elementary contributions $p_{\boldsymbol{\theta}_k}$.

In this case, using the linearity of the sketching operator, the compressive learning problem (2.39) becomes

$$\widehat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{z} - \textstyle\sum_{k=1}^K \alpha_k \mathcal{A}(p_{\boldsymbol{\theta}_k})\|_2.$$

This non-convex problem is hard to solve exactly, but the `CL-OMPR` algorithm (based on the Matching Pursuit methods discussed in Subsec. 2.2.1) was proposed in [KBGP18] to solve it approximately. More precisely, this method greedily selects new atoms $\boldsymbol{\theta}'$ (that are the most promising in reducing a residual $\boldsymbol{r}$, *i.e.*, by approximately maximizing (*e.g.*, using quasi-Newton optimization schemes) the non-convex criterion $\langle \frac{\mathcal{A}(p_{\boldsymbol{\theta}'})}{\|\mathcal{A}(p_{\boldsymbol{\theta}'})\|}, \boldsymbol{r}\rangle$. The method then alternates between adding new atoms and further decreasing the cost function by local minimization initialized at the current solution.

Note that this heuristic algorithm ("decoder") is not guaranteed to find the global minimizer $\widehat{\boldsymbol{\theta}}$, as further discussed in Chapter 6.

Designing the sketch operator To fully specify the CL scheme, it remains to specify the parameters defining the sketch operator $\mathcal{A}$, *i.e.*, in the RFF case, the frequency sampling pattern Λ, and the sketch size m.

Several *frequency sampling patterns* Λ have been discussed in [KBGP18], which we quickly recall here. A first straightforward choice is the Gaussian sampling strategy $\Lambda = \mathcal{N}(\mathbf{0}, \sigma^{-2}\boldsymbol{I}_d)$, which means that the associated RFF approximate a Gaussian kernel with scale σ, *i.e.*, $\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle \simeq \kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{2\sigma^2})$. As argued in [KBGP18], this pattern under-explores the low frequencies (due to the curse of dimensionality), so another possibility is to decompose $\boldsymbol{\omega} = R\boldsymbol{\varphi}$, where $\boldsymbol{\varphi} \sim \mathcal{U}(S^{d-1})$ is a normed random direction and $R \sim p_R(R;\sigma)$ is the norm of $\boldsymbol{\omega}$, which can thus be directly controlled. Two choices for this latter distribution were proposed: either (FG) a folded Gaussian $p_R \propto e^{-(\sigma R)^2}$, or (AR) the adapted radius distribution defined as $p_R \propto \left((\sigma R)^2 + \frac{(\sigma R)^4}{4}\right)^{\frac{1}{2}} e^{-(\sigma R)^2}$ [KBGP18].

In any case, given a frequency sampling pattern, one still has to select the *scale parameter* $\sigma > 0$, which should be adjusted to the current dataset either by prior knowledge or from a some heuristic (*e.g.*, [KBGP18,BCGS19]). Fig. 2.5 illustrates the important role of this parameter: if it is too large or too small, the estimated data distribution will not be accurate.

As for the *sketch size*, the analogy with compressive sensing suggests that m should scale with the "sparsity" of the considered "low-complexity" candidates $\mathcal{P}_{\boldsymbol{\theta}}$, *e.g.*, $m \gtrsim p$ if there are p parameters to estimate ($\boldsymbol{\theta} \in \mathbb{R}^p$). Numerical simulations validated that for GMM [KBGP18] and for k-means [KTTG17] this is indeed the case. For example, compressive k-means succeeds as soon as $m \gtrsim Kd$, where Kd is the amount of parameters we seek (d coordinates of K centroids).

Other tasks and sketch constructions In this thesis we will focus on compressive k-means and compressive GMM with random Fourier features sketches, but for completeness, we briefly mention other tasks and feature maps that have been considered in the literature. Those are semi-parametric models, to which Remark 2.16 applies.

Compressive PCA was proposed in [GBKT17], and numerically validated in [Cha20]. Instead of RFF, this approach relies on *random quadratic features*, where complex exponentiation is replaced by $t \mapsto t^2$, *i.e.*, $\Phi_{\text{RQF}}(\boldsymbol{x}) :=$

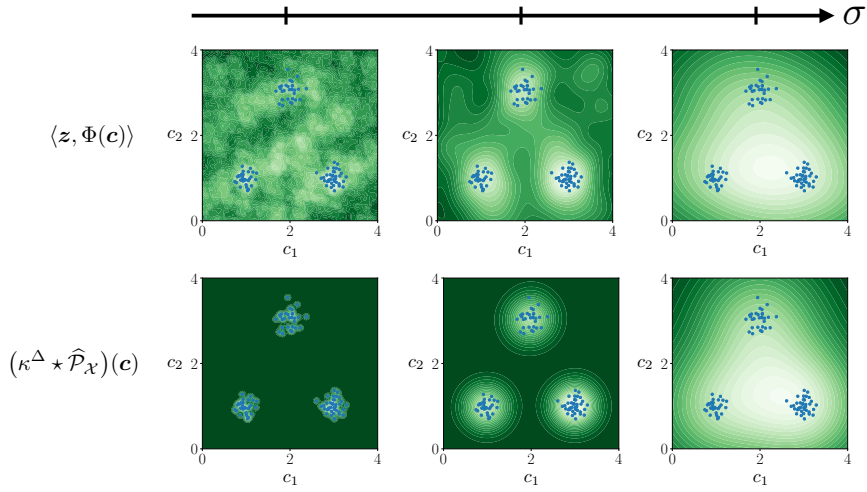


Fig. 2.5 We consider a 2-d dataset of three clusters (blue points), sketched with RFF associated to a Gaussian kernel with scale σ. **Top row**: for a fixed finite sketch size m, we plot the landscape $\langle \boldsymbol{z}, \Phi(\boldsymbol{c}) \rangle$ as function of a test vector $\boldsymbol{c}$, *i.e.*, the criterion used to select the first centroid $\boldsymbol{c}$ in CLOMP for k-means. We plot this for three varying *sketch scales* σ (with σ that increases from left to right). This intuitively illustrates how the sketch "views" the data distribution. **Bottom row**: the related kernel mean embedding (2.33), $\mu_{\kappa}(\widehat{\mathcal{P}}_{\mathcal{X}}) = \kappa^{\Delta} \star \widehat{\mathcal{P}}_{\mathcal{X}}$, *i.e.*, what the top row approximates as $m \to \infty$.

$[(\boldsymbol{\omega}_j^\top \boldsymbol{x})^2]_{j=1}^m$. The resulting sketch can also be seen as a random compression operation $\widetilde{\mathcal{A}}$ acting on the data covariance matrix, $\boldsymbol{z} = \widetilde{\mathcal{A}}(\boldsymbol{X}\boldsymbol{X}^\top)$.

In [SKD19], the authors consider *compressive Independent Component Analysis*, which follows a similar approach, but where the fourth order cumulant tensor is compressively sensed instead of the covariance matrix. In [SGD19], a scheme was proposed to perform *compressive subspace clustering* by sensing the correlation matrix of a Veronese embedding of the data, although computational gains are possible only in small dimensions.

2.5.3 Theoretical guarantees

One important achievement of the compressive statistical learning framework is the development of strong *statistical learning guarantees*, in the form of probabilistic bounds on the excess risk (2.6), as usually considered in the SL framework (see Sec. 2.1). Those bounds guarantee that, with high probability $1-\delta$, the compressive learning solution $\widehat{\boldsymbol{\theta}}$ from (2.39) is not much worse than the optimal solution $\boldsymbol{\theta}^*$ (*i.e.*, the minimizer to the true risk $\mathcal{R}(\boldsymbol{\theta}; \mathcal{P}_0)$) by an error η, *i.e.*,

$$\mathbb{P}[\mathcal{R}(\widehat{\boldsymbol{\theta}}; \mathcal{P}_0) - \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0) \leq \eta] \geq 1 - \delta.$$

By another analogy with compressive sensing (see Subsec. 2.2.2), the strategy deployed in [GBKT17] to prove such a bound is to relate it to a form of Lower Restricted Isometry Property (LRIP). Similarly to (2.28), the LRIP is defined as, for some constant $\gamma > 0$,

$$\|\mathcal{P} - \mathcal{Q}\|_{\mathcal{R}} \leq \gamma \|\mathcal{A}(\mathcal{P}) - \mathcal{A}(\mathcal{Q})\|_2, \quad \forall \mathcal{P}, \mathcal{Q} \in \mathcal{G}. \tag{2.41}$$

On the left, the *risk-induced metric* $\|\mathcal{P} - \mathcal{Q}\|_{\mathcal{R}}$ is an instance of an Integral Probability Metric (2.32), where the family of test functions are the loss functions for all possible parameter vectors, *i.e.*, $\mathcal{F} = \{\ell(\boldsymbol{\theta}, \cdot), \boldsymbol{\theta} \in \Theta\}$. Intuitively, this quantity captures how different the risk with respect to $\mathcal{P}$ and with respect to $\mathcal{Q}$ is: if $\|\mathcal{P} - \mathcal{Q}\|_{\mathcal{R}}$ is small, then we expect that $\mathcal{R}(\boldsymbol{\theta}; \mathcal{P}) \simeq \mathcal{R}(\boldsymbol{\theta}; \mathcal{Q})$, which means that solving a machine learning task with respect to $\mathcal{P}$ or with respect to $\mathcal{Q}$ is roughly equivalent. On the right, the low-complexity *model set* $\mathcal{G}$ is the set of all possible model distributions $\mathcal{P}_{\boldsymbol{\theta}}$, *i.e.*, $\mathcal{G} := \{\mathcal{P}_{\boldsymbol{\theta}}, \boldsymbol{\theta} \in \Theta\}$.

Intuitively, the LRIP thus ensures that the sketch accurately *encodes* the machine learning task: for any pair of distributions $\mathcal{P}, \mathcal{Q}$ that are of interest in solving the task, if the sketch of those distributions are close, the

LRIP guarantees that those distributions are indeed roughly equivalent from the point of view of the risk objective. More precisely, the results from [GBKT17] ensures that if the LRIP holds, then the excess risk is bound given by

$$\eta = D(\mathcal{P}_0, \mathcal{G}) + c\|\mathcal{A}(\mathcal{P}_0) - \mathcal{A}(\widehat{\mathcal{P}}_{\mathcal{X}})\|_2,$$

where the first term is a *modeling error*, a "distance" between the true data distribution $\mathcal{P}_0$ and the model set $\mathcal{G}$, and the second term is a *sampling error*. This bound can be put in correspondence with (2.27) in usual CS.

It then remains to prove the LRIP holds, which typically holds with high probability over the draw of the feature map Φ, thus defining the probability of failure δ in (2.41). For k-means and GMM with RFF sketches, this is done in [GBKT20]. Without entering into the details, the strategy relies on using the MMD (6.11) $D_\kappa(\mathcal{P}, \mathcal{Q})$ as the intermediary between $\|\mathcal{A}(\mathcal{P}) - \mathcal{A}(\mathcal{Q})\|_2$ and $\|\mathcal{P} - \mathcal{Q}\|_{\mathcal{R}}$.

The theoretical guarantees of CL are further detailed in Chapter 4.

PART I
Quantized Sketches

3

Asymmetric Random Periodic Features

THIS chapter is a bit special in that it is (despite the title of this thesis) *not* about the compressive learning (CL) framework. To be more precise, recall that in compressive learning we study the sketch of a dataset given by *the average of random features* $\boldsymbol{z}_{\mathcal{X}} := \frac{1}{n}\sum_i \Phi(\boldsymbol{x}_i)$. In this chapter, we temporarily "forget" about the averaging operation, and focus on *the individual random features* $\boldsymbol{z}(\boldsymbol{x}_i) := \Phi(\boldsymbol{x}_i)$. The obtained results will serve as main building block in Chapter 4, when we re-introduce the averaging operation: from a "macro" point of view, the role of this chapter is thus to establish preliminary results for the next one (which is about quantized compressive learning). However, as argued below, the setting studied in this chapter is also of independent interest.

Concretely, this chapter formally introduces the general framework of *asymmetric random periodic features*, where two signals of interest $\boldsymbol{x}, \boldsymbol{y}$ are observed through random periodic features $\boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_g(\boldsymbol{y})$: random projections followed by a general periodic map f or g, which is allowed to be different for both signals. We derive the influence of those periodic maps on the *kernel* (*i.e.*, the similarity measure) $\kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y})$ that is approximated by their dot product $\langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_g(\boldsymbol{y}) \rangle \simeq \kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y})$. This result generalizes earlier results from (symmetric) random periodic features (introduced in Section 2.2.3) which showed in particular that a simple quantization of ran-

dom Fourier features (corresponding to replacing the complex exponential by a different periodic map that takes binary values, which is appealing for their transmission and storage), distorts the approximated kernel. As we will see, a remarkable consequence of our analysis is that when the features of *only one* of the two signals are quantized, *the original kernel is recovered without distortion*.

The main achievement of this chapter is to prove *uniform (probabilistic) error bounds* (that is, bounds on the deviation $|\langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_g(\boldsymbol{y})\rangle - \kappa_{f,g}(\boldsymbol{x},\boldsymbol{y})|$) holding for all pair of signals picked in an infinite low-complexity set. Interestingly, our results allow the periodic maps to be discontinuous, thanks to a new mathematical tool, *i.e.*, the mean Lipschitz smoothness. We then apply this generic framework to *semi-quantized kernel machines* (where only one of the signals has quantized features and the other has classical random Fourier features), for which we show theoretically that the approximated kernel remains unchanged (with the associated error bound), and confirm the power of the approach with numerical simulations.

This chapter mostly coincides (often verbatim) with the content of our publication "Breaking the waves: asymmetric random periodic features for low-bitrate kernel machines" [SJ20a] (in *Information and Inference*), with the notable exception of an extra section at the end (indicated by a star *).

3.1 Introduction

3.1.1 Motivation

Rather than to directly process high-dimensional signals, it is often more efficient to first summarize them to their main *features*. This assumes that these capture essential information for the considered processing, such as the proximity of any pair of signals. Mathematically, the signal summarization is modeled by a feature map $\boldsymbol{\varphi}$ from the signal space Σ to the feature (or embedding) space $\mathcal{E}$. This map $\boldsymbol{\varphi}$ transforms the representation of signals while encoding some aspects of their geometry; loosely speaking, this can be written as $\mathcal{D}_{\mathcal{E}}(\boldsymbol{\varphi}(\boldsymbol{x}),\boldsymbol{\varphi}(\boldsymbol{y})) \approx \mathcal{D}_{\Sigma}(\boldsymbol{x},\boldsymbol{y})$ for any pair of signals $\boldsymbol{x},\boldsymbol{y} \in \Sigma$, where $\mathcal{D}_{\Sigma}$ is the preserved geometric quantity (such as an inner product or a distance), and $\mathcal{D}_{\mathcal{E}}$ is an evaluation procedure acting only on the signal features. This approach is useful whenever the features $\boldsymbol{\varphi}(\boldsymbol{x})$ are easier to process with respect to some critical computational resource (*e.g.*, memory usage, computing time)—often at the price of an approximation error (as suggested by the approximation symbol $\mathcal{D}_{\mathcal{E}} \approx \mathcal{D}_{\Sigma}$ above). The

map $\boldsymbol{\varphi}$ is most often than not a randomized function (drawn from a distribution). There are essentially three main ways to save computational resources with features: *(i)* leveraging *dimensionality reduction* (*i.e.*, $\boldsymbol{\varphi}(\boldsymbol{x})$ is encoded by much less coefficients than the dimension of $\boldsymbol{x}$), such as in compressive sensing techniques [FR17], where typically the "embedding" $\boldsymbol{\varphi}$ is linear and $\mathcal{D}_\Sigma$ and $\mathcal{D}_\mathcal{E}$ are the Euclidean distances (as explained in Section 2.2.2); *(ii)* using mappings that *linearize* the evaluation of an otherwise nonlinear quantity, such as random Fourier features (RFF) [RR08], where $\mathcal{D}_\Sigma$ is a kernel κ and $\mathcal{D}_\mathcal{E}$ is simply the inner product (as explained in Section 2.1.4); and *(iii) quantizing features*, where $\boldsymbol{\varphi}(\boldsymbol{x})$ produces a quantized output that can be encoded with a highly reduced bitrate compared to the initial signal, allowing reducing the memory and/or transmission load, as well as paving the way for hardware-based procedures (as explained in Section 2.2.3). This quantization objective is often combined with either *(i)*, as in quantized compressive sensing [BJKS15, GLP+13, Dir19], or *(ii)*, as in one-bit universal embeddings [BM15]. As made clear below, our contributions also target the combination of *(ii)* with *(iii)*.

In all those applications, it is almost always assumed that the available features for the two signals $\boldsymbol{x}, \boldsymbol{y} \in \Sigma$ come from the *same* feature map $\boldsymbol{\varphi}$ (we say the features are *symmetric*). However, we can legitimately wonder if removing this assumption, *i.e.*, accessing the signals through features $\boldsymbol{\varphi}(\boldsymbol{x})$ and $\boldsymbol{\psi}(\boldsymbol{y})$, where we have the freedom to set $\boldsymbol{\varphi} \neq \boldsymbol{\psi}$, can further reduce specific computational resources. The practical interest of this relaxation—that we call *asymmetric features*—arises when the two signals come from different sources (*i.e.*, when the setting is intrinsically asymmetric): for example, those sources might have different resources (such as memory or power) at their disposal, and will therefore benefit differently from techniques *(i)-(iii)*.

Here, we are interested in asymmetric features for linearizing kernel estimations, as explained in *(ii)*. In particular, we work with *random periodic features* (introduced in Subsec. 2.2.3, and further detailed in Sec. 3.2); those are defined as $\boldsymbol{\varphi}(\boldsymbol{x}) := f(\boldsymbol{\Omega}^T\boldsymbol{x} + \boldsymbol{\xi})$ and $\boldsymbol{\psi}(\boldsymbol{y}) := g(\boldsymbol{\Omega}^T\boldsymbol{y} + \boldsymbol{\xi})$ with $\boldsymbol{\Omega}$ a random projection matrix, $\boldsymbol{\xi}$ a random dither, and f, g two periodic functions. We thus generalize the context of random Fourier features [RR08], where $f(\cdot) = g(\cdot) = \exp(\mathrm{i}\cdot)$ (the complex exponential), by "*breaking the waves*" with possibly discontinuous, distinct functions f and g (as described in Sec. 3.4). We show how those features can be used to approximate shift-invariant kernels κ, *i.e.*, $\langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{\psi}(\boldsymbol{y}) \rangle \approx \kappa(\boldsymbol{x}, \boldsymbol{y})$, in expectation over the random quantities $\boldsymbol{\Omega}, \boldsymbol{\xi}$. Our motivating use-case is to combine this approach

with harsh quantization of some features, objective *(iii)*, as we explain in the next paragraph. However, all our developments are generic, and of interest for any machine learning algorithm that processes or takes decisions from the local geometry of data.

Semi-binary kernel machines as a motivating application: If one of the periodic functions incorporates the quantization of the feature vector (say, the one-bit universal quantization, or square wave function $q : \mathbb{R} \to \{0, 1\}$ [BR13]; see Fig. 3.2), then that feature vector can be stored or transmitted (or both) much more efficiently than the usual (infinite-precision) random Fourier features. Consider for example a machine learning context, where a kernel method [SSB+02] such as a Support Vector Machine (SVM) [BGV92] has been trained in advance on some dataset $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n \subset \Sigma$ (see Section 2.1 for details on kernel methods and SVM). To actually use this model for prediction on a new signal $\boldsymbol{x}' \in \Sigma$, the physical device that records it must either communicate with a server where the inference is performed remotely (Fig. 3.1a), or implement this model directly (Fig. 3.1b); in either case, this is an expensive operation whenever this device is under tight computational resources constraints, and quantization of feature vectors is potentially very helpful.

In the first scenario (inference done remotely on a server), we might quantize the feature vector of the query signal, $\boldsymbol{\varphi}(\boldsymbol{x}') \in \{0, 1\}^m$ (but not the feature vectors of the dataset $\boldsymbol{\psi}(\boldsymbol{x}_i)$). This allows to heavily reduce the bitrate when communicating this vector to the server, and even paves the way for computing those features directly in hardware, *e.g.*, relying on voltage-controlled oscillators [YKJC08]. In the second context, we could conversely binarize the feature vectors of the dataset so that $\boldsymbol{\psi}(\boldsymbol{x}_i) \in \{0, 1\}^m$ for all $\boldsymbol{x}_i \in \mathcal{X}$, but not the incoming query vector $\boldsymbol{\varphi}(\boldsymbol{x}')$[1]. The advantage here is that the memory needed to store the model is heavily reduced, with additional computational benefits coming from the embedded processing of binary values. This idea has received significant attention in the literature, *e.g.*, following [JDS10] for nearest-neighbor search.

For both of those examples, the main question that we seek to answer is to quantify the loss of accuracy (induced by quantization) as a function of the feature vector length m. More precisely, our goal is to obtain (probabilistic) guarantees on the decay of the kernel approximation error as a function of m, that hold uniformly for any pairwise comparisons of signals

[1] It is even possible to encode only a subset of the dataset features for models that only need to access some entries, such as SVM with the support vectors.

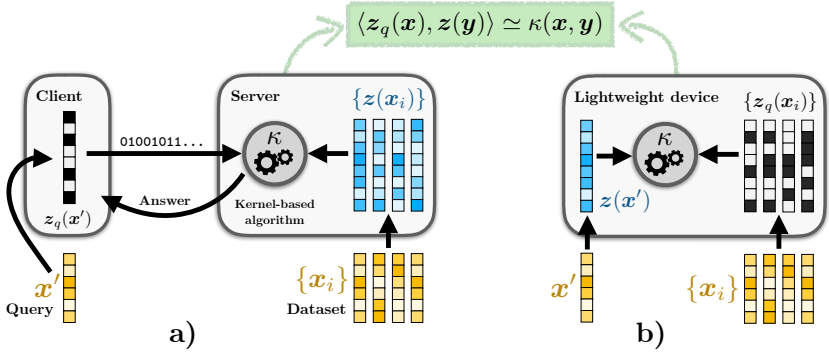


Fig. 3.1 Two motivating applications of our results (from Sec. 3.5, in green): combining one-bit universal features with usual random Fourier features yields the same desired kernel κ. (a) A "client" device records a "query" signal x', and transmits its quantized features $z_q(x')$, encoded efficiently as only m bits, to a "server" device that can evaluate the kernel similarity with the rest of a dataset from their usual n full-precision RFF $\{z(x_i)\}_{i=1}^n$. (b) A lightweight device implements a kernel method with very low memory requirements, only having to store $\{z_q(x_i)\}_{i=1}^n$ the n one-bit feature vectors of the dataset instead of the full-precision ones, provided the usual RFF $z(x')$ are used for the incoming query vectors.

taken an infinite (but compact) set Σ. In this case, the main challenge lies in dealing with the discontinuous nature of the quantization operation—handling discontinuities is thus one of the key features of this work.

3.1.2 Related work

Before detailing the elements of our approach, we find useful to mention a few related works, showing how they inspired us, and stressing their connections and differences with our contributions.

Quantization of (symmetric) random Fourier features: The construction of the general random periodic features considered in this work is instantiated in Section 3.5 to the case where the corresponding periodic map is the universal quantizer (or square wave function). As mentioned in Subsection 2.2.3, this approach was introduced in [RL09, BR13] as a binary map preserving local distances (*i.e.*, up to a given radius), under the name universal quantized embedding. Those features have subsequently been used for kernel methods in [BM15], which is similar to the framework we propose but where not one but both signal features being compared are quantized in a symmetric fashion, which distorts the kernel to be recovered. This line of work was further generalized is [BRM17], where uniform guarantees are derived for generic periodic function (possibly discontinuous) instead of the one-bit universal quantizer, holding on infinite signal sets. This defines the random periodic features approach (see Section 3.2 for details) that we also consider; we provide an in-depth description of how our results relate to (and complement) those from [BRM17] in Appendix C.

Back to the particular problem of quantizing random Fourier features, another line of work [ZMDR18] shows that a specific stochastic quantization hardly harms the generalization performance of RFF-based algorithms. The ultimate objective of this last work is, however, fairly different from ours: the authors seek to reduce the memory requirements *during training* by performing a more sophisticated quantization, and then use the full-precision RFF for the subsequent inference stage; on the other hand, our objective is to provide a simple quantization scheme to reduce the resources *during the inference stage*, without concerns for how the training was performed.

Asymmetric features and quantizations: The possibility to use asymmetric features has been explored for linear embeddings in [RKL19], as

an additional degree of freedom to minimize (in a data-dependent fashion) the average error of the distance estimation. In [DCL08], weighted universal embeddings are used for distance estimation, where the weights depend upon one of the two signals (which makes the scheme asymmetric) to decrease the error on this estimation. Closer to our context, in [GPGL13], it is experimentally shown for a broad set of feature maps[2] that quantizing the features of the dataset but not of the query (as in scenario Fig. 3.1b) significantly improves the performance compared to quantizing both features. Similarly, the authors of [LL19, Li19] recently considered linear random projections (with the same matrix) of two signals that have been quantized with different quantization levels.

3.1.3 Chapter contributions

We provide in Sec. 3.2 several preliminary elements as well as important concepts of the relevant literature: random Fourier features and their (possibly quantized) extension to any periodic nonlinearity. We then start by analyzing how the kernel approached by asymmetric random periodic features behaves in expectation (in the asymptotic case), which is proved in Sec. 3.3. Our main results come in Sec. 3.4, where we prove uniform error bounds of the kernel approximation for infinite signal sets. In order to do so, we introduce a new tool, the mean Lipschitz smoothness property. Next, we apply our general results to the semi-quantized setting motivated above in Sec. 3.5, and illustrate with numerical experiments in Sec. 3.6. To further establish the generic nature of our results, we also instantiate them on modulo random features and complex extensions in Sec. 3.7, before concluding in Sec. 3.8.

For the interested reader, Appendix C relates our approach to the context of geometry-preserving embedding (or coding) developed in [BRM17], solving in the same time an error in the proof of one of their results

3.1.4 Notations specific to this chapter

We will often consider bounded 2π-periodic functions $f, g : \mathbb{R} \to \mathbb{C}$ for which the 2-norm and the infinity norm read $\|f\|^2 = \frac{1}{2\pi}\int_0^{2\pi} |f(t)|^2 \mathrm{d}t$ and $\|f\|_\infty := \sup_{t\in[0,2\pi]} |f(t)|$, respectively, and the inner product of f and g is $\langle f, g\rangle = \frac{1}{2\pi}\int_0^{2\pi} f(t)g^*(t)\mathrm{d}t$. For brevity and clarity, we will sometimes refer to a function using the "dot" notation, *e.g.*, $\exp(\mathrm{i}\cdot)$ for the function

[2] Such as Locality Sensitive Hashing, universal embeddings, and several variants of PCA.

$t \mapsto \exp(\mathrm{i}t) \in \mathbb{C}$ for $t \in \mathbb{R}$.

We use the convention where the (d-dimensional) Fourier transform of a function $f : \mathbb{R}^d \to \mathbb{C}$ reads $\hat{f}(\boldsymbol{\omega}) = (\mathcal{F}f)(\boldsymbol{\omega}) := \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} e^{-\mathrm{i}\boldsymbol{u}^\top \boldsymbol{\omega}} f(\boldsymbol{u}) \mathrm{d}\boldsymbol{u}$, with inverse $(\mathcal{F}^{-1}\hat{f})(\boldsymbol{u}) := \int_{\mathbb{R}^d} e^{\mathrm{i}\boldsymbol{u}^\top \boldsymbol{\omega}} \hat{f}(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}$. The same convention is used for the Fourier transform of finite measures on $\mathbb{R}^d$. In all our developments, except if specified differently, $C, C', \ldots, c, c', \ldots > 0$ denote *universal* constants whose value may change from one instance to the other.

3.2 Preliminaries

We introduce here several fundamental concepts supporting our approach. We first precise the kind of signal space we consider, as well as how signals are compared through a kernel, before to briefly explain the principles sustaining the definition of random Fourier features (RFF). Next, we generalize RFF to any random periodic features for a family of bounded periodic functions.

3.2.1 Signals and kernels

In this work, we focus on signals belonging to a bounded signal space $\Sigma \subset \mathbb{R}^d$ having finite Kolmogorov η-entropy $\mathcal{H}_\eta(\Sigma)$ for any radius $\eta > 0$ [KT61]. This entropy, defined as $\mathcal{H}_\eta(\Sigma) := \log \mathcal{C}_\eta(\Sigma)$, is related to the covering number $\mathcal{C}_\eta(\Sigma)$ of Σ, the cardinality of the smallest finite subset of Σ that *covers* it with balls of radius η. Using the Minkowski sum, this means that

$$\mathcal{C}_\eta(\Sigma) := \min\{|\mathcal{S}| : \mathcal{S} \subset \Sigma \subset \mathcal{S} + \eta \mathbb{B}_2^d\},$$

which is finite for any compact set Σ.

The Kolmogorov entropy measures the intrinsic dimension of Σ in $\mathbb{R}^d$. In particular, $\mathcal{H}_\eta(\mathcal{V} \cap \mathbb{B}_2^d) \leq Cd' \log(1 + \frac{1}{\eta})$ for any subspace $\mathcal{V} \subset \mathbb{R}^d$ of dimension $d' \leq d$ [Pis99], and the set of s-sparse vectors $\Sigma_s := \{\boldsymbol{x} \in \mathbb{R}^d, \|\boldsymbol{x}\|_0 \leq s\}$ restricted to the unit ball has entropy bounded by $\mathcal{H}_\eta(\Sigma_s \cap \mathbb{B}_2^d) \leq C \cdot s \log(\frac{d}{s}) \log(1 + \frac{1}{\eta})$, see for example [BDDW08]. Other bounds exist for, *e.g.*, the set of bounded group sparse signals [ADR16], bounded low-rank matrices [CP11], or for specific low-dimensional manifolds [EW15].

At the heart of our study is the comparison of two signals through a *kernel*, *i.e.*, a function over pairs of signals $\kappa : \Sigma \times \Sigma \to \mathbb{C}$ (in the machine learning literature, kernels are usually real-valued). Typically, invoking the so-called "kernel trick" [BGV92], κ represents the inner product be-

tween the input signals x, x' when they are mapped in some *implicit* feature space by an appropriate map $\phi : \Sigma \to \mathbb{H}$: $\kappa(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathbb{H}}$ for some Hilbert space $\mathbb{H}$. By definition of the inner product, the kernel κ must then necessarily be conjugate symmetric ($\kappa(x, x') = \kappa^*(x', x)$), and positive definite (p.d.), *i.e.*, for any number n, $\sum_{i,j=1}^{n} c_i c_j^* \kappa(x_i, x_j) \geq 0$ for all $x_1, \ldots, x_n \in \Sigma$ and $c_1, \ldots, c_n \in \mathbb{C}$.

3.2.2 Random Fourier features

We here succinctly remind the context of random Fourier features (RFF); see Subsection 2.1.4 for a gentle introduction. RFF are implicitly built on *Bochner's theorem* [Rud62]. This theorem states that a shift-invariant continuous kernel $\kappa(x, y) = \kappa^{\Delta}(x - y)$ (for some $\kappa^{\Delta} : \Sigma - \Sigma \to \mathbb{C}$) is positive definite if and only if it is the (inverse) Fourier transform of a nonnegative finite measure Λ, *i.e.*,

$$\kappa \text{ positive definite} \quad \Leftrightarrow \quad \kappa^{\Delta}(u) = (\mathcal{F}^{-1}\Lambda)(u) = \int_{\mathbb{R}^d} e^{\mathrm{i}\omega^\top u} \mathrm{d}\Lambda(\omega). \quad (3.1)$$

In particular, assuming *w.l.o.g.* the normalization $\kappa(x, x) = \kappa^{\Delta}(0) = 1$, Λ is a probability distribution over $\mathbb{R}^d$, and the kernel can be written $\kappa^{\Delta}(u) = \mathbb{E}_{\omega \sim \Lambda} e^{\mathrm{i}\omega^\top u}$. The key idea of random Fourier features [RR08] is thus to construct low-dimensional features $z(x), z(y)$ whose inner product approximates the kernel $\kappa(x, y)$ by Monte Carlo sampling of this expectation.

Definition 3.1 (Random Fourier features)**.** Let $\kappa(x, y) = \kappa^{\Delta}(x - y)$ be a shift-invariant p.d. kernel, normalized such that $\kappa^{\Delta}(0) = 1$, with Fourier transform $\Lambda = \mathcal{F}\kappa^{\Delta}$. Given a target dimension m, the associated "complex" random Fourier features are

$$z(x) := \tfrac{1}{\sqrt{m}} \exp\left(\mathrm{i}(\Omega^\top x + \xi)\right) \quad \in \mathbb{C}^m, \quad (3.2)$$

with random projections (or "frequencies") $\Omega := (\omega_1, \cdots, \omega_m) \in \mathbb{R}^{d \times m}$ generated as $\Omega \sim \Lambda^m$, *i.e.*, with $\omega_j \sim_{\text{i.i.d.}} \Lambda$ for $j \in [m]$, and a random dither $\xi \in \mathbb{R}^m$ generated as $\xi \sim \mathcal{U}^m([0, 2\pi))$, *i.e.*, with $\xi_j \sim_{\text{i.i.d.}} \mathcal{U}([0, 2\pi))$ for $j \in [m]$. We also define the "real" random Fourier features $z_{\cos}(x)$ as $\Re[z(x)]$, the real part of those features:

$$z_{\cos}(x) := \tfrac{1}{\sqrt{m}} \cos\left(\Omega^\top x + \xi\right) \quad \in \mathbb{R}^m. \quad (3.3)$$

Remark 3.2. The dither ξ was initially introduced in [RR08] when only the

real RFF $z_{\cos}(\boldsymbol{x})$ were used; in the (more widely used) complex case $\boldsymbol{\xi}$ is not necessary (see [SS15b] for an in-depth comparison of the "real" versus "complex" random Fourier features). We still included it in this definition for the sake of consistency with Def. 3.5 below.

By direct application of Bochner's theorem, the inner product of RFF indeed approaches (in expectation over the draw of the frequencies $\boldsymbol{\Omega}$) the target kernel: $\mathbb{E}\langle \boldsymbol{z}(\boldsymbol{x}), \boldsymbol{z}(\boldsymbol{y})\rangle = \kappa(\boldsymbol{x}, \boldsymbol{y})$. Moreover, for a finite feature dimension m, the error of the kernel approximation $\widehat{\kappa}(\boldsymbol{x}, \boldsymbol{y}) := \langle \boldsymbol{z}(\boldsymbol{x}), \boldsymbol{z}(\boldsymbol{y})\rangle$ can be uniformly bounded (*i.e.*, bound the absolute error $|\widehat{\kappa}(\boldsymbol{x}, \boldsymbol{y}) - \kappa(\boldsymbol{x}, \boldsymbol{y})|$ for all values $\boldsymbol{x}, \boldsymbol{y}$ in Σ), with high probability on the draw of $\boldsymbol{\Omega}$ (we work with different normalization choices, so the result we present here differs slightly from the initial bound [RR08, Claim 1]). Finer bounds can be found, among others, in [SS15b, SS15a].

Proposition 3.3 (Uniform kernel approximation error for RFF)**.** *Let Σ be a compact set, and $\boldsymbol{z}(\boldsymbol{x})$ be the RFF defined above. Assume that there exists an associated constant C_Λ, such that*

$$\mathbb{E}_{\boldsymbol{\omega}\sim\Lambda} |\boldsymbol{\omega}^\top \boldsymbol{a}| \leq C_\Lambda \|\boldsymbol{a}\|_2, \quad \forall \boldsymbol{a} \in \mathbb{R}^d. \tag{3.4}$$

Provided that, for $\epsilon > 0$,

$$m \geq C\epsilon^{-2}\mathcal{H}_{c\epsilon/C_\Lambda}(\Sigma),$$

the kernel approximation $\widehat{\kappa}(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{z}(\boldsymbol{x}), \boldsymbol{z}(\boldsymbol{y})\rangle$ has error uniformly bounded by

$$\left|\widehat{\kappa}(\boldsymbol{x}, \boldsymbol{y}) - \kappa(\boldsymbol{x}, \boldsymbol{y})\right| \leq \epsilon, \quad \forall\, \boldsymbol{x}, \boldsymbol{y} \in \Sigma,$$

with probability exceeding $1 - C'e^{-c'm\epsilon^2}$.

Proof. This version of the RFF approximation error is obtained as a particular case of our Prop. 3.15; see [RR08] for the initial result. □

The constant C_Λ defined in (3.4) characterizes the smoothness of the kernel (if the kernel is smoother, it exhibits less high-frequency content, and C_Λ will be lower). In most of the RFF literature, this constant is bounded by the Cauchy-Schwarz inequality as $C_\Lambda = \mathbb{E}_{\boldsymbol{\omega}\sim\Lambda} \|\boldsymbol{\omega}\|_2$. Then, one can (as done in [RR08]) further bound $\mathbb{E}_{\boldsymbol{\omega}\sim\Lambda} \|\boldsymbol{\omega}\|_2 \leq \sigma_\Lambda$ where σ_Λ^2 is the second moment of Λ, equivalent to the kernel curvature at the origin, *i.e.*, $C_\Lambda^2 \leq \sigma_\Lambda^2 := \mathbb{E}_{\boldsymbol{\omega}\sim\Lambda} \|\boldsymbol{\omega}\|_2^2 = \nabla^2 \kappa^\Delta|_{\boldsymbol{u}=\boldsymbol{0}}$, with ∇^2 the Laplacian operator.

However, for specific distributions, using the Cauchy-Schwarz inequality in high dimension leads to a loose bound of C_Λ. For example, if the covariance matrix of $\boldsymbol{\omega}$ is upper bounded by $\widetilde{\sigma}_\Lambda^2 \boldsymbol{I}_d$ for some $\widetilde{\sigma}_\Lambda^2 > 0$ (if, *e.g.*, $\mathcal{F}^{-1}\Lambda$ is the Gaussian RBF ("radial basis function") kernel with radius $1/\widetilde{\sigma}_\Lambda^2$, Λ is isotropic [Ver12], or if each component of $\boldsymbol{\omega}$ are i.i.d. with variance bounded by $\widetilde{\sigma}_\Lambda^2$) then

$$\begin{aligned}\left(\mathbb{E}_{\boldsymbol{\omega}\sim\Lambda} |\boldsymbol{\omega}^\top \boldsymbol{a}|\right)^2 &\leq \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim\Lambda} |\boldsymbol{\omega}^\top \boldsymbol{a}|^2 = \boldsymbol{a}^\top \left(\mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim\Lambda} \boldsymbol{\omega}\boldsymbol{\omega}^\top\right) \boldsymbol{a} \\ &\leq \boldsymbol{a}^\top \cdot \widetilde{\sigma}_\Lambda^2 \boldsymbol{I}_d \cdot \boldsymbol{a} = \widetilde{\sigma}_\Lambda^2 \|\boldsymbol{a}\|_2^2.\end{aligned}$$

In this case we obtain $C_\Lambda = \widetilde{\sigma}_\Lambda$, while Cauchy-Schwarz gives $\sqrt{d} \cdot \widetilde{\sigma}_\Lambda$, hence overestimating the constant by a factor $\sqrt{d}$.

Example 3.4. Consider the simple case where the signals of interest have an ℓ_2-norm smaller than 1 and lie in a union of S subspaces of $\mathbb{R}^d$, each with dimension s. This signal space model encompasses, for instance, $\Sigma = \mathbb{B}_2^d$ (that is, where $S = 1$ and $s = d$), the set of bounded s-sparse signals in $\mathbb{R}^d$ for which $S = \binom{d}{s} \leq (\frac{ed}{s})^s$ and each subspace (one per fixed sparse signal support) has dimension s, or more advanced models with structured sparsity [BCDH10, ADR16]. For such a model, the Kolmogorov entropy is bounded by $Cs \cdot \log\left(\frac{1}{\eta}\right) \leq \mathcal{H}_\eta(\Sigma) \leq C's \cdot \log\left(1 + \frac{2}{\eta}\right) + \log S$ (see, *e.g.*, [JC17, Lemma 10]). Assume that we target the usual Gaussian kernel with unit bandwidth, hence $C_\Lambda = 1$. In this case, the RFF kernel approximation error is uniformly bounded over Σ, with high probability, provided that the number of features satisfies $m \geq C\epsilon^{-2}\left(s \log(\frac{1}{c\epsilon}) + \log S\right)$. For instance, for bounded s-sparse signals we need $m \geq C\epsilon^{-2} s \log(\frac{ed}{cs\,\epsilon})$.

3.2.3 Random periodic features

A crucial generalization to RFF has been proposed in [BRM17], where the complex exponential is replaced by a *generic periodic function* f. We refer to this approach as *random periodic features* (RPF). Without loss of generality, we make the following normalization assumptions throughout this work: f has period given by 2π, is centered (zero mean), and takes (absolute) values bounded by one. We note this compactly as $f \in \mathrm{PF}$, with

$$\mathrm{PF} := \{f : \mathbb{R} \to \mathbb{C} \mid f \text{ is } 2\pi\text{-periodic}, \textstyle\int_0^{2\pi} f(t)\mathrm{d}t = 0, \|f\|_\infty \leq 1\}.$$

Functions of PF can be expressed as a Fourier series of the following form

$$f(t) = \sum_{k\in\mathbb{Z}} F_k e^{\mathrm{i}kt}, \quad \text{where} \quad F_k := \tfrac{1}{2\pi}\int_0^{2\pi} f(t)e^{-\mathrm{i}kt}\mathrm{d}t. \tag{3.5}$$

Note that $f \in \mathrm{PF}$ implies $F_0 = 0$ (because f is centered) and $|F_k| \leq 1$ (because f is bounded).

Definition 3.5 (Random periodic features)**.** Let f be a generic periodic function, normalized such that $f \in \mathrm{PF}$, and Λ a probability distribution on $\mathbb{R}^d$. Given a target dimension m, the associated *random periodic features* are

$$\boldsymbol{z}_f(\boldsymbol{x}) := \tfrac{1}{\sqrt{m}} f(\boldsymbol{\Omega}^\top \boldsymbol{x} + \boldsymbol{\xi}) \quad \in \mathbb{C}^m, \tag{3.6}$$

with a $d \times m$ random projection matrix $\boldsymbol{\Omega} := (\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_m) \sim \Lambda^m$, and a random dither $\boldsymbol{\xi} \sim \mathcal{U}^m([0, 2\pi))$.

Remark 3.6. As the complex exponentiation satisfies $\exp(\mathrm{i}\cdot) \in \mathrm{PF}$, this definition includes the classical RFF, with $\boldsymbol{z}(\boldsymbol{x}) = \boldsymbol{z}_{\exp(\mathrm{i}\cdot)}(\boldsymbol{x})$. The real RFF $\boldsymbol{z}_{\cos}(\boldsymbol{x})$ are also a particular case of this definition.

The geometry induced by such generic features can be characterized the inner product $\widehat{\kappa}_{f,f}(\boldsymbol{x}, \boldsymbol{y}) := \langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_f(\boldsymbol{y})\rangle$. As explained by the following result (adapted from [BRM17, Theorem 4.4]), this product is associated with a modified kernel $\kappa_{f,f}(\boldsymbol{x}, \boldsymbol{y})$ (the rationale for these notations is clarified in the next section).

Proposition 3.7 (Kernel from symmetric RPF)**.** *The inner product of random periodic features* (3.6) *approaches, on average, a kernel*

$$\kappa_{f,f}(\boldsymbol{x}, \boldsymbol{y}) := \mathbb{E}\langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_f(\boldsymbol{y})\rangle$$

that is shift-invariant and given by

$$\kappa_{f,f}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{k\in\mathbb{Z}} |F_k|^2 \kappa^\Delta(k(\boldsymbol{x} - \boldsymbol{y})) =: \kappa^\Delta_{f,f}(\boldsymbol{x} - \boldsymbol{y}), \tag{3.7}$$

where $\kappa^\Delta(\boldsymbol{u}) = (\mathcal{F}^{-1}\Lambda)(\boldsymbol{u})$ is the shift-invariant kernel associated with the distribution of $\boldsymbol{\Omega}$ in the RPF.

Proof. This version is obtained as a particular case of Prop. 3.8; see [BRM17] for the initial result. □

The modified kernel $\kappa_{f,f}$ is thus a *scale mixture* of the initial kernel κ (that is approached by the "classical" RFF), where the weight of scale k is

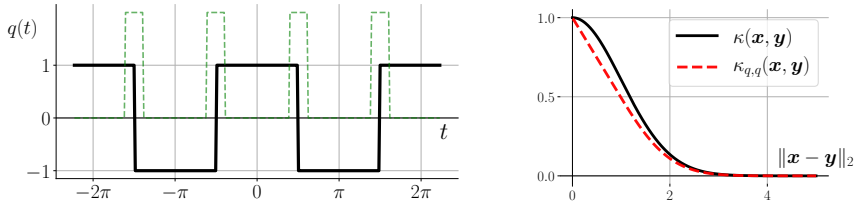


Fig. 3.2 (Left) The solid black curve represents the universal quantization function $q(t)$ (with $q \in \mathrm{PF}$) defined in (3.8). Up to a shift and rescaling, this function corresponds to the *least significant bit* of a standard uniform scalar quantizer. In dashed green, we display the related integrand $I_\delta(t)$, with $\delta = 0.35$. This quantity refers to the proof of the mean smoothness (Def. 3.13) of q in Prop. 3.19 (Sec. 3.5). (Right) When drawing $\boldsymbol{\Omega}$ from a Gaussian distribution $\Lambda = \mathcal{N}(0, \boldsymbol{I}_d)$, the associated RFF recover the Gaussian kernel κ (in black), but the RPF with universal quantization approximate a "distorted" kernel $\kappa_{q,q}$ (dashed red).

given by $|F_k|^2$. In the non-asymptotic case, the authors of [BRM17] show that, for all pairs of vectors taken in a finite set Σ of size N, $\widehat{\kappa}_{f,f}(\boldsymbol{x}, \boldsymbol{y})$ quickly concentrates around $\kappa_{f,f}(\boldsymbol{x}, \boldsymbol{y})$ when m is large compared to $\log N$; the deviation error scaling as $\mathcal{O}(\sqrt{\log N / m})$ when m increases. Our result in Prop. 3.15 provides a uniform approximation bound valid for infinite sets.

Random periodic features were introduced as a general theoretical framework to analyze the so-called universal quantization embeddings [BR13]; those binary embeddings encode the local distances (*i.e.*, the distances below a given threshold) on an efficiently small number of bits. This embedding relies on the "one-bit universal quantization" given by $\mathcal{Q}_\Delta : \mathbb{R} \to \{0, 1\} : t \mapsto \mathcal{Q}_\Delta(t) = 1$ if $(2k-1) \leq t/\Delta \leq 2k$ for any $k \in \mathbb{Z}$ and 0 otherwise. It can be interpreted as the least significant bit of a usual, plain scalar quantizer with stepsize Δ, and visualized as a square wave: see Fig. 3.2, left. Here, we will for convenience use q instead, its normalized equivalent in PF,

$$q(t) := \mathrm{sign} \circ \cos(t) = \sum_{k \in \mathbb{Z}} Q_k e^{ikt}, \quad \text{with} \quad Q_k = \begin{cases} \frac{2}{k\pi}(-1)^{\frac{k-1}{2}} & \text{if } k \text{ odd}, \\ 0 & \text{if } k \text{ even}. \end{cases} \tag{3.8}$$

Using the universal quantization as periodic nonlinearity is appealing because $\boldsymbol{z}_q(\boldsymbol{x}) \in \{-1, +1\}^m$, which can thus be encoded/transmitted by only m bits. However, as predicted by (3.7), the approximated kernel is

modified, as illustrated for the Gaussian kernel Fig. 3.2, right. Moreover, proving uniform kernel approximation bounds (as in Prop. 3.3) for infinite sets Σ is specially challenging when the nonlinearity f presents discontinuities (which is the case when $f = q$, for example). In [BRM17], the authors introduced a formalism (the T-part Lipschitz functions) to deal with this problem and to obtain uniform approximation bounds on infinite signal sets for the universal embeddings. As we explain in Appendix C, the proof relying on this approach is however wrong, which motivates us to introduce another tool, the mean Lipschitz smoothness, to deal with discontinuous maps.

3.3 Expected kernel (asymptotic case)

Following the considerations of the Introduction, let us now consider the *asymmetric features* setting where a pair of signals of interest, $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, are available only through their random periodic features, $\boldsymbol{z}_f(\boldsymbol{x})$ and $\boldsymbol{z}_g(\boldsymbol{y})$, as defined in (3.6). Those features are allowed to result from different periodic maps $f, g \in \mathrm{PF}$, but the preceding projection $\boldsymbol{\Omega}$ and dithering $\boldsymbol{\xi}$ are kept identical.

In this section, we characterize the properties of the *expected kernel* yielded by the expectation, over the draw of $\boldsymbol{\Omega}$ and $\boldsymbol{\xi}$, of the following "asymmetric" inner product:

$$\widehat{\kappa}_{f,g}(\boldsymbol{x}, \boldsymbol{y}) := \langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_g(\boldsymbol{y}) \rangle. \tag{3.9}$$

This asymmetric RPF kernel is defined from

$$\begin{aligned} \kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y}) &:= \mathbb{E}_{\boldsymbol{\Omega}, \boldsymbol{\xi}} \, \langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_g(\boldsymbol{y}) \rangle \\ &= \tfrac{1}{m} \textstyle\sum_{j=1}^{m} \mathbb{E}_{\boldsymbol{\omega}_j, \xi_j} f(\boldsymbol{\omega}_j^\top \boldsymbol{x} + \xi_j) \, g^*(\boldsymbol{\omega}_j^\top \boldsymbol{y} + \xi_j) \\ &= \mathbb{E}_{\boldsymbol{\omega} \sim \Lambda, \, \xi \sim \mathcal{U}([0, 2\pi))} f(\boldsymbol{\omega}^\top \boldsymbol{x} + \xi) \, g^*(\boldsymbol{\omega}^\top \boldsymbol{y} + \xi). \end{aligned} \tag{3.10}$$

In the two bottom lines, we used the fact that $\boldsymbol{\omega}_j$ (and ξ_j) are independently and identically distributed, for all j, with $\boldsymbol{\omega}_j$ and ξ_j mutually independent. Remark that by the law of large numbers, $\kappa_{f,g}$ thus corresponds to the kernel that the asymmetric inner product $\widehat{\kappa}_{f,g}$ approximates when we let the feature space dimension m grow to infinity.

Proposition 3.8 (Expected kernel for asymmetric periodic random features)**.** *Let $\boldsymbol{z}_f$ and $\boldsymbol{z}_g$ be random periodic features, associated with functions $f, g \in \mathrm{PF}$,*

frequencies $\boldsymbol{\omega}_j \sim_{\text{i.i.d.}} \Lambda = \mathcal{F}\kappa^\Delta$ *and* $\xi_j \sim_{\text{i.i.d.}} \mathcal{U}([0, 2\pi))$. *For any pair* $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, *the expected kernel* $\kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}_{\boldsymbol{\Omega}, \boldsymbol{\xi}} \langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_g(\boldsymbol{y}) \rangle$ *satisfies*

$$\kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y}) = \textstyle\sum_{k \in \mathbb{Z}} F_k G_k^* \, \kappa^\Delta(k(\boldsymbol{x} - \boldsymbol{y})) =: \kappa^\Delta_{f,g}(\boldsymbol{x} - \boldsymbol{y}). \tag{3.11}$$

Although here expanded as an infinite series, this kernel is bounded $|\kappa_{f,g}| \leq 1$ *since* $f, g \in \mathrm{PF}$.

Proof. Starting from the last line of (3.10), and decomposing f and g as their Fourier series,

$$\begin{aligned}
\kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y}) &= \mathbb{E}_{\boldsymbol{\omega}, \xi} \textstyle\sum_{k \in \mathbb{Z}} \sum_{k' \in \mathbb{Z}} F_k G_{k'}^* \, e^{\mathrm{i}k(\boldsymbol{\omega}^\top \boldsymbol{x} + \xi)} e^{-\mathrm{i}k'(\boldsymbol{\omega}^\top \boldsymbol{y} + \xi)} \\
&= \textstyle\sum_{k,k'} F_k G_{k'}^* \; \mathbb{E}_{\boldsymbol{\omega} \sim \Lambda} \, e^{\mathrm{i}\boldsymbol{\omega}^\top (k\boldsymbol{x} - k'\boldsymbol{y})} \; \mathbb{E}_{\xi \sim \mathcal{U}([0,2\pi))} \, e^{\mathrm{i}(k-k')\xi} \\
&= \textstyle\sum_{k,k'} F_k G_{k'}^* \, \kappa^\Delta(k\boldsymbol{x} - k'\boldsymbol{y}) \, \delta_{k,k'} \\
&= \textstyle\sum_k F_k G_k^* \, \kappa^\Delta(k(\boldsymbol{x} - \boldsymbol{y})),
\end{aligned} \tag{3.12}$$

where in the third line we used Bochner's theorem (3.1) and the orthogonality of complex exponentials on one period: $\frac{1}{2\pi} \int_0^{2\pi} e^{\mathrm{i}kt} e^{-\mathrm{i}k't} \mathrm{d}t = \delta_{k,k'}$. □

Example 3.9. As will be further developed in Sec. 3.5, when $f(\cdot) = \cos(\cdot)$ and $g(\cdot) = q(\cdot)$ the universal quantization defined in (3.8), we observe that the expected kernel is (up to a proportionality constant) exactly the "initial" kernel approximated by the RFF, *i.e.*, $\kappa_{\cos,q}(\boldsymbol{x}, \boldsymbol{y}) = \frac{2}{\pi} \kappa(\boldsymbol{x}, \boldsymbol{y})$.

The dither $\boldsymbol{\xi}$ plays here a crucial role: it cancels out (in expectation) the "cross-terms" in (3.12), each related to $F_k G_{k'}^* \, \kappa^\Delta(k\boldsymbol{x} - k'\boldsymbol{y}) = F_k G_{k'}^* \, \kappa(k\boldsymbol{x}, k'\boldsymbol{y})$, that have different scales $k \neq k'$ for $\boldsymbol{x}$ and $\boldsymbol{y}$. As a consequence, the expected kernel is—as any kernel should be—conjugate symmetric, *i.e.*, $\kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y}) = \kappa^*_{f,g}(\boldsymbol{y}, \boldsymbol{x})$, despite the asymmetry of its empirical approximation, *i.e.*, despite that $\widehat{\kappa}_{f,g}(\boldsymbol{x}, \boldsymbol{y}) \neq \widehat{\kappa}^*_{f,g}(\boldsymbol{y}, \boldsymbol{x})$. The dithering can thus be thought of as a means to symmetrize, through expectation, the kernel associated with the asymmetric features inner product.

For the same reason, the dithering ensures that the expected kernel remains shift-invariant; Prop. 3.8 provides $\kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y}) = \kappa^\Delta_{f,g}(\boldsymbol{x} - \boldsymbol{y})$, where κ^Δ in (3.11) is the kernel related to the frequency sampling pattern Λ. The expected kernel is thus a *scale mixture*, a linear combination of copies of κ^Δ, scaled (actually contracted) by an integer factor k (which is non-zero, since $F_0 = G_0 = 0$), and weighted by coefficients $F_k G_k^*$. In general, we expect this

scale mixture $\kappa^{\Delta}_{f,g}$ to be narrower than the initial kernel κ^{Δ} (or more spread out in the frequency domain).

In general, however, the positive definiteness of κ does not imply that $\kappa_{f,g}$ is p.d., since taking, for instance, $g = -f$ (*i.e.*, $F_k G_k^* = -|F_k|^2 < 0$) induces that $\kappa_{f,g}(\boldsymbol{x}, \boldsymbol{x}) = -\sum_k |F_k|^2 \kappa^{\Delta}(\boldsymbol{0}) < 0$ for all $\boldsymbol{x} \in \mathbb{R}^d$. Whether $\kappa_{f,g}$ is a positive definite kernel or not depends on the phase synchronization between the Fourier coefficients of f and g. A sufficient condition for $\kappa_{f,g}$ to be p.d. is to ensure that $F_k G_k^* \in \mathbb{R}^+$ for all k, as verified by taking $f = g$ (as explained in Appendix C), or the combination $f = q$, $g(\cdot) = \cos(\cdot)$ in Sec. 3.5 and Sec. 3.6.

Remark 3.10. In light of (3.11), we could decide to normalize our approach differently. Assuming that $f, g \in \mathrm{PF}$ are not orthogonal, *i.e.*, $\langle f, g \rangle \neq 0$, we can define, for $\boldsymbol{x}, \boldsymbol{y} \in \Sigma$, the normalized kernels

$$\begin{aligned} \widetilde{\kappa}_{f,g}(\boldsymbol{x}, \boldsymbol{y}) &:= \tfrac{1}{\langle f,g \rangle} \langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_g(\boldsymbol{y}) \rangle, \\ \dot{\kappa}_{f,g}(\boldsymbol{x}, \boldsymbol{y}) &:= \tfrac{1}{\langle f,g \rangle} \kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y}), \\ \dot{\kappa}^{\Delta}_{f,g}(\boldsymbol{u}) &:= \tfrac{1}{\langle f,g \rangle} \kappa^{\Delta}_{f,g}(\boldsymbol{u}). \end{aligned} \tag{3.13}$$

Since $\kappa^{\Delta}(\boldsymbol{0}) = 1$ and, from (3.11), $\langle f, g \rangle = \sum_k F_k G_k^* = \kappa^{\Delta}_{f,g}(\boldsymbol{0})$, (3.13) ensures that, for any $\boldsymbol{x} \in \Sigma$, $\mathbb{E}\, \widetilde{\kappa}_{f,g}(\boldsymbol{x}, \boldsymbol{x}) = \dot{\kappa}_{f,g}(\boldsymbol{x}, \boldsymbol{x}) = \dot{\kappa}^{\Delta}_{f,g}(\boldsymbol{0}) = 1 = \kappa^{\Delta}(\boldsymbol{0})$. Without guaranteeing that $\dot{\kappa}_{f,g}$ is p.d., this normalization prevents the counterexample $f = -g$ to lead to a kernel with negative value on the origin. For clarity, we do not base our following developments on $\dot{\kappa}_{f,g}(\boldsymbol{x}, \boldsymbol{y})$ but we will refer to this useful quantity in Sec. 3.5 when, for $f = q$ and $g(\cdot) = \cos(\cdot)$, we will need to compare $\dot{\kappa}^{\Delta}_{f,g}$ to the RFF kernel κ^{Δ}.

Let us now provide an alternative expression of the expected kernel $\kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y}) = \kappa^{\Delta}_{f,g}(\boldsymbol{x} - \boldsymbol{y})$, that will prove to be useful in the next section.

Lemma 3.11. *Define the* correlation *h between f and g,*

$$h(t) := (f * \bar{g})(t) = \tfrac{1}{2\pi} \textstyle\int_0^{2\pi} f(\tau) g^*(\tau - t)\, \mathrm{d}\tau, \tag{3.14}$$

where $\bar{g}(t) := g^*(-t)$ denotes the conjugate reverse of g, and $*$ the convolution operator on $[0, 2\pi]$. The expected (shift-invariant) kernel $\kappa^{\Delta}_{f,g}$ can be expressed by

$$\kappa^{\Delta}_{f,g}(\boldsymbol{u}) = \mathbb{E}_{\boldsymbol{\omega} \sim \Lambda}\, h(\boldsymbol{\omega}^\top \boldsymbol{u}) = \mathbb{E}_{\boldsymbol{\omega} \sim \Lambda} (f * \bar{g})(\boldsymbol{\omega}^\top \boldsymbol{u}). \tag{3.15}$$

Proof. By the convolution theorem, the Fourier series coefficients of h are

given by $H_k = F_k G_k^*$. The result follows from plugging this fact into the proof of Prop. 3.8. □

Lemma 3.11 can be interpreted as an expansion similar to the one provided by Bochner's theorem: whereas the initial kernel $\kappa^\Delta(\boldsymbol{u}) = \int e^{\mathrm{i}\boldsymbol{\omega}^\top \boldsymbol{u}} \mathrm{d}\Lambda(\boldsymbol{\omega})$ can be expressed in a basis that is a family complex exponentials $e^{\mathrm{i}\boldsymbol{\omega}^\top \boldsymbol{u}}$ with "coordinates" given by Λ, $\kappa^\Delta_{f,g}(\boldsymbol{u}) = \int h(\boldsymbol{\omega}^\top \boldsymbol{u}) \mathrm{d}\Lambda(\boldsymbol{\omega})$ can be expressed in a basis that is the family of functions $\{\boldsymbol{u} \to h(\boldsymbol{\omega}^\top \boldsymbol{u}) : \boldsymbol{\omega} \in \mathbb{R}^d\}$, another type of $2\pi-$periodic functions (replacing the complex exponential $\exp(\mathrm{i}\cdot)$ with $h(\cdot)$).

3.4 Approximation error analysis (non-asymptotic case)

In the practical setting where the vectors $\boldsymbol{z}_f(\boldsymbol{x})$ and $\boldsymbol{z}_g(\boldsymbol{y})$ must be quickly processed or stored in memory, their size m must be as small as possible. On the other hand, setting m too small hurts the empirical estimation $\widehat{\kappa}_{f,g}(\boldsymbol{x}, \boldsymbol{y})$ of the expected kernel $\kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}\, \widehat{\kappa}_{f,g}(\boldsymbol{x}, \boldsymbol{y})$. To understand this trade-off, we are thus interested in a probabilistic bound for the (absolute) kernel approximation error $|\widehat{\kappa}_{f,g} - \kappa_{f,g}|$, as a function of the RPF dimension m. We give here an answer to this question under generic assumptions, and show how to apply it in a concrete situation—for asymmetric kernel estimation with one-bit quantized RFF—in Sec. 3.5.

3.4.1 Non-uniform approximation error

Ultimately, we want to obtain a (probabilistic) bound for the kernel approximation error that holds uniformly over all $\boldsymbol{x}, \boldsymbol{y} \in \Sigma$. First bounding the error for one *fixed* pair $(\boldsymbol{x}, \boldsymbol{y})$ is often used as an easier intermediary step. This is provided by the following proposition.

Proposition 3.12 (Non-uniform kernel approximation error from asymmetric periodic random features)**.** *For two functions $f, g \in \mathrm{PF}$, let $\boldsymbol{z}_f, \boldsymbol{z}_g$ be random periodic features associated with frequencies $\boldsymbol{\Omega}$ and a dither $\boldsymbol{\xi}$. For any* fixed *pair $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^d \times \mathbb{R}^d$, the inner product $\widehat{\kappa}_{f,g}(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_g(\boldsymbol{y}) \rangle$ concentrates, in probability over the draw of $\boldsymbol{\Omega} \sim \Lambda^m, \boldsymbol{\xi} \sim \mathcal{U}^m([0, 2\pi))$, around $\kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}_{\boldsymbol{\Omega}, \boldsymbol{\xi}} \langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_g(\boldsymbol{y}) \rangle$ as*

$$\mathbb{P}\left[\left|\widehat{\kappa}_{f,g}(\boldsymbol{x}, \boldsymbol{y}) - \kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y})\right| \leq \epsilon\right] \geq 1 - 2e^{-m\epsilon^2/2}. \tag{3.16}$$

Proof. We rewrite $\langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_g(\boldsymbol{y})\rangle = \frac{1}{m}\sum_j Z_j$, with the random variables $Z_j := f(\boldsymbol{\omega}_j^\top \boldsymbol{x} + \xi_j) g^*(\boldsymbol{\omega}_j^\top \boldsymbol{y} + \xi_j)$. The Z_j variables are i.i.d., have mean $\kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y})$ by definition of the expected kernel (3.10), and are bounded by $|Z_j| \leq \|f\|_\infty \|g\|_\infty \leq 1$ (because $f, g \in \mathrm{PF}$). The result follows by Hoeffding's inequality. □

3.4.2 Uniform approximation error

We now want to extend the error bound in Prop. 3.12 to hold not only for one fixed pair $(\boldsymbol{x}, \boldsymbol{y})$ but simultaneously over all pairs $(\boldsymbol{x}, \boldsymbol{y}) \in \Sigma \times \Sigma$; this is called a *uniform bound*. The classical argument invoked in this type of proofs (*e.g.*, [RR08, BDDW08, PDG17]) goes as follows. If Σ is a finite set (of finite cardinality $|\Sigma|$), the uniform bound is obtained by applying a union bound over $|\Sigma|^2$ instances of Prop. 3.12 (one for each pair in $\Sigma \times \Sigma$). In the case where $\Sigma \subset \mathbb{R}^d$ is an infinite but compact set, the strategy is to bound the approximation error on a finite set Σ_η that covers Σ by balls of some radius $\eta > 0$, then to extend this bound by some notion of continuity (smoothness) over the $\eta-$balls. We then obtain a bound which holds over $\Sigma_\eta + \eta\mathbb{B}_2^d \supseteq \Sigma$, which concludes the proof.

In our setting, the last step of this proof technique would ideally use Lipschitz continuity; we say that a function $f : \mathbb{R} \to \mathbb{C}$ is *Lipschitz continuous* with constant L_f if, for all $t, t' \in \mathbb{R}$, $|f(t) - f(t')| \leq L_f |t - t'|$, which is equivalent to

$$\forall t \in \mathbb{R}, \forall \delta > 0, \quad \sup_{r \in [-\delta, \delta]} \{|f(t + r) - f(t)|\} \leq L_f \cdot \delta. \tag{3.17}$$

However, this strategy fails when any of the maps f or g is not Lipschitz continuous (*e.g.*, when they present discontinuities, such as the "square wave" universal quantization map q from (3.8)). To be able to include such maps in our analysis, we must define a more permissive notion of smoothness, just as the T-part Lipschitz property defined in [BRM17] (but without the limitations explained in Appendix C). In this work, we rather introduce the concept of *mean Lipschitz smoothness property* for periodic function in PF. Intuitively, a periodic function is smooth in the mean Lipschitz sense if its largest local deviation is small *on average*.

Definition 3.13 (Mean Lipschitz property)**.** Let $f : \mathbb{R} \to \mathbb{C}$ be a generic periodic function (here *w.l.o.g.* assumed of period 2π). We say it is *mean Lipschitz smooth* with mean Lipschitz constant L_f^μ if for all radii $\delta \in (0, \pi]$,

the average maximum deviation of f in $[-\delta, \delta]$ is bounded by $L_f^\mu \cdot \delta$:

$$\begin{aligned}&\mathbb{E}_{t\sim\mathcal{U}([0,2\pi))} \sup_{r\in[-\delta,\delta]}\{|f(t+r)-f(t)|\}\\&= \tfrac{1}{2\pi}\int_0^{2\pi} \sup_{r\in[-\delta,\delta]}\{|f(t+r)-f(t)|\}\,\mathrm{d}t \leq L_f^\mu\cdot\delta.\end{aligned} \tag{3.18}$$

The mean Lipschitz property (that we will refer to as "mean smoothness" to avoid confusion with the usual Lipschitz continuity when necessary) can truly be understood as the Lipschitz continuity after an averaging. It is reminiscent of the mean modulus of continuity from [Szá37, Wik72] but where the order of the supremum and averaging operations are reversed (which is less restrictive). If f is Lipschitz continuous with Lipschitz constant L_f, then it has necessarily also the mean smoothness property with constant $L_f^\mu \leq L_f$. However, it is possible that $L_f^\mu \ll L_f$ (if the large slopes of f are concentrated on a small portion of $[0, 2\pi]$), and discontinuous function can have a finite L_f^μ constant—for example, the square wave q representing the universal quantization is mean smooth with constant $L_q^\mu = \frac{4}{\pi}$ (see Prop. 3.19), although it is not a Lipschitz continuous function. Leaving the detailed proof to Sec. 3.5, the trick is to observe that the integrand $I_\delta(t) = \sup_{r\in[-\delta,\delta]}\{|f(t+r)-f(t)|\}$ is supported on an interval whose length is proportional to δ, as shown Fig. 3.2 left. Moreover, the convolution of any PF function with a mean smooth PF function yields a Lipschitz continuous one.

Lemma 3.14. *Given two functions $f, g \in$ PF, among which f is mean smooth with constant L_f^μ, their convolution $(f * g)$ is Lipschitz continuous with constant*

$$L_{f*g} \leq L_f^\mu.$$

Proof. Re-writing (3.17) for $(f*g)(t) = \frac{1}{2\pi}\int_0^{2\pi} f(t-\tau)g(\tau)\,\mathrm{d}\tau$ gives, since $\|g\|_\infty \leq 1$,

$$\begin{aligned}&\sup_{r\in[-\delta,\delta]}\{|(f*g)(t+r)-(f*g)(t)|\}\\&\quad= \sup_{|r|\leq\delta}\left|\tfrac{1}{2\pi}\int_0^{2\pi}[f(t+r-\tau)-f(t-\tau)]\,g(\tau)\,\mathrm{d}\tau\right|\\&\quad\leq \tfrac{1}{2\pi}\sup_{|r|\leq\delta}\int_0^{2\pi}|f(t+r-\tau)-f(t-\tau)|\,\mathrm{d}\tau\\&\quad\leq \tfrac{1}{2\pi}\int_0^{2\pi}\sup_{|r|\leq\delta}|f(\tau'+r)-f(\tau')|\,\mathrm{d}\tau' \leq L_f^\mu\delta.\end{aligned}$$

□

In other words, the convolution of two PF functions, among whom one of them is mean Lipschitz, is "smoother" than its factors, a property that comes from the convolution itself (which is to be put in correspondence with the fact that for f differentiable and g discontinuous, $f * g$ is differentiable). In particular, if *both* f and g are mean smooth with constants L_f^μ and L_g^μ respectively, their *correlation* $h = f \star \bar{g}$ is Lipschitz with $L_h \leq \min(L_f^\mu, L_g^\mu)$. Coming back to our setting, this fact allows us (by using Lemma 3.11) to characterize the Lipschitz continuity of the expected kernel $\kappa_{f,g}^\Lambda$. With that, we have all the tools to prove our main result, a uniform bound on the kernel approximation error obtained with possibly discontinuous (but mean smooth) maps.

Proposition 3.15 (Uniform kernel approximation error from asymmetric periodic random features)**.** *Let Σ be a compact set and $f, g \in \mathrm{PF}$ periodic functions with finite mean smoothness constants L_f^μ and L_g^μ, respectively, and let $C_\Lambda < \infty$ such that $\mathbb{E}_{\boldsymbol{\omega}\sim\Lambda} |\boldsymbol{\omega}^\top \boldsymbol{a}| \leq C_\Lambda \|\boldsymbol{a}\|_2$ for all $\boldsymbol{a}$ (the kernel smoothness constant).*

For all error level $\epsilon > 0$, provided the feature dimension is larger than

$$m \geq 128 \cdot \tfrac{1}{\epsilon^2} \cdot \mathcal{H}_{\epsilon/c}(\Sigma), \tag{3.19}$$

with the constant $c = 4C_\Lambda(L_f^\mu + L_g^\mu + 2\min(L_f^\mu, L_g^\mu))$, the following kernel approximation bounds holds uniformly:

$$\left|\widehat{\kappa}_{f,g}(\boldsymbol{x}, \boldsymbol{y}) - \kappa_{f,g}(\boldsymbol{x}, \boldsymbol{y})\right| \leq \epsilon, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \Sigma, \tag{3.20}$$

with probability exceeding $1 - 3\exp(-\frac{m\epsilon^2}{64})$.

Proof. With Σ_η a finite optimal $\eta-$covering of Σ, any $\boldsymbol{x}' \in \Sigma$ (resp. $\boldsymbol{y}'$) can be written $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{r}_{\boldsymbol{x}}$ (resp. $\boldsymbol{y}' = \boldsymbol{y} + \boldsymbol{r}_{\boldsymbol{y}}$) for centers $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_\eta$, and $\boldsymbol{r}_{\boldsymbol{x}}, \boldsymbol{r}_{\boldsymbol{y}} \in \eta\mathbb{B}_2^d$. The proof proceeds by defining three events $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$ from which Prop. 3.15 follows, and then by bounding the failure probability of their joint occurrence. First, for any covering center $\boldsymbol{x} \in \Sigma_\eta$, we can expect that the set of m functions $h_j^f : \eta\mathbb{B}_2^d \to \mathbb{C}$ defined for $j \in [m]$ as $h_j^f(\boldsymbol{r}; \boldsymbol{x}) := f(\boldsymbol{\omega}_j^\top \boldsymbol{x} + \boldsymbol{\omega}_j^\top \boldsymbol{r} + \xi_j)$ contains, on average, few "variations" over the $\eta-$ball.

More precisely, defining the largest variation of $h_j^f(\boldsymbol{r};\boldsymbol{x})$ over the $\eta-$ball as

$$\begin{aligned} H_j^f(\eta;\boldsymbol{x}) &:= \sup_{\boldsymbol{r}\in\eta\mathbb{B}_2^d} |h_j^f(\boldsymbol{r};\boldsymbol{x}) - h_j^f(\boldsymbol{0};\boldsymbol{x})| \\ &= \sup_{\boldsymbol{r}\in\eta\mathbb{B}_2^d} |f(\boldsymbol{\omega}_j^\top \boldsymbol{x} + \boldsymbol{\omega}_j^\top \boldsymbol{r} + \xi_j) - f(\boldsymbol{\omega}_j^\top \boldsymbol{x} + \xi_j)|, \end{aligned}$$

we first assume that, given $\epsilon_1 > 0$, the event $\mathcal{E}_1$ holds, with

$$\mathcal{E}_1 : \sup_{\boldsymbol{x}\in\Sigma_\eta} \tfrac{1}{m}\textstyle\sum_{j=1}^m H_j^f(\eta;\boldsymbol{x}) \leq L_f^\mu \eta C_\Lambda + \epsilon_1.$$

Similarly for g, we define the event $\mathcal{E}_2$ such that

$$\mathcal{E}_2 : \sup_{\boldsymbol{y}\in\Sigma_\eta} \tfrac{1}{m}\textstyle\sum_{j=1}^m H_j^g(\eta;\boldsymbol{y}) \leq L_g^\mu \eta C_\Lambda + \epsilon_2.$$

Next, we suppose that the kernel approximation has error bounded by ϵ_3 for all the covering centers $\boldsymbol{x},\boldsymbol{y} \in \Sigma_\eta$, *i.e.*,

$$\mathcal{E}_3 : \sup_{\boldsymbol{x},\boldsymbol{y}\in\Sigma_\eta} \left|\widehat{\kappa}_{f,g}(\boldsymbol{x},\boldsymbol{y}) - \kappa_{f,g}(\boldsymbol{x},\boldsymbol{y})\right| \leq \epsilon_3.$$

Under those events, we establish a deterministic bound for all $\boldsymbol{x}',\boldsymbol{y}'$ using a chain of triangle inequalities:

$$\begin{aligned} |\widehat{\kappa}_{f,g}(\boldsymbol{x}',\boldsymbol{y}') - \kappa_{f,g}(\boldsymbol{x}',\boldsymbol{y}')| &= |\widehat{\kappa}_{f,g}(\boldsymbol{x}+\boldsymbol{r}_x,\boldsymbol{y}+\boldsymbol{r}_y) - \kappa_{f,g}(\boldsymbol{x}+\boldsymbol{r}_x,\boldsymbol{y}+\boldsymbol{r}_y)| \\ &\leq \delta_1 + \delta_2 + \delta_3 + \delta_4, \end{aligned}$$

where the error terms are defined as

$$\begin{aligned} \delta_1 &:= |\widehat{\kappa}_{f,g}(\boldsymbol{x}+\boldsymbol{r}_x,\boldsymbol{y}+\boldsymbol{r}_y) - \widehat{\kappa}_{f,g}(\boldsymbol{x},\boldsymbol{y}+\boldsymbol{r}_y)|, \\ \delta_2 &:= |\widehat{\kappa}_{f,g}(\boldsymbol{x},\boldsymbol{y}+\boldsymbol{r}_y) - \widehat{\kappa}_{f,g}(\boldsymbol{x},\boldsymbol{y})|, \\ \delta_3 &:= |\widehat{\kappa}_{f,g}(\boldsymbol{x},\boldsymbol{y}) - \kappa_{f,g}(\boldsymbol{x},\boldsymbol{y})|, \\ \delta_4 &:= |\kappa_{f,g}(\boldsymbol{x},\boldsymbol{y}) - \kappa_{f,g}(\boldsymbol{x}+\boldsymbol{r}_x,\boldsymbol{y}+\boldsymbol{r}_y)|. \end{aligned}$$

First, we observe that, thanks to $\mathcal{E}_1$:

$$\begin{aligned} \delta_1 &= \tfrac{1}{m}\left|\textstyle\sum_{j=1}^m \left[f(\boldsymbol{\omega}_j^\top(\boldsymbol{x}+\boldsymbol{r}_x)+\xi_j) - f(\boldsymbol{\omega}_j^\top\boldsymbol{x}+\xi_j)\right] g^*(\boldsymbol{\omega}_j^\top\boldsymbol{y}'+\xi_j)\right| \\ &\leq \|g\|_\infty \cdot \tfrac{1}{m}\textstyle\sum_{j=1}^m \left|f(\boldsymbol{\omega}_j^\top\boldsymbol{x}+\boldsymbol{\omega}_j^\top\boldsymbol{r}_x+\xi_j) - f(\boldsymbol{\omega}_j^\top\boldsymbol{x}+\xi_j)\right| \leq \eta L_f^\mu C_\Lambda + \epsilon_1. \end{aligned}$$

Similarly, using $\mathcal{E}_2$, we get for δ_2:

$$\begin{aligned}\delta_2 &= \tfrac{1}{m}\left|\textstyle\sum_{j=1}^m f(\boldsymbol{\omega}_j^\top \boldsymbol{x} + \xi_j)\left[g^*(\boldsymbol{\omega}_j^\top \boldsymbol{y} + \boldsymbol{\omega}_j^\top \boldsymbol{r}_{\boldsymbol{y}} + \xi_j) - g^*(\boldsymbol{\omega}_j^\top \boldsymbol{y} + \xi_j)\right]\right| \\ &\leq \eta L_g^\mu C_\Lambda + \epsilon_2.\end{aligned}$$

Regarding δ_3, we directly get from $\mathcal{E}_3$:

$$\delta_3 \leq \sup_{\boldsymbol{x},\boldsymbol{y}\in\Sigma_\eta} |\widehat{\kappa}_{f,g}(\boldsymbol{x},\boldsymbol{y}) - \kappa_{f,g}(\boldsymbol{x},\boldsymbol{y})| \leq \epsilon_3.$$

Finally, for δ_4, denoting $\boldsymbol{u} := \boldsymbol{x} - \boldsymbol{y}$, $\boldsymbol{r}_{\boldsymbol{u}} := \boldsymbol{r}_{\boldsymbol{x}} - \boldsymbol{r}_{\boldsymbol{y}} \in 2\eta\mathbb{B}_2^d$ and using Lemma 3.11 (also recall that $h = f * \bar{g}$ is Lipschitz continuous with $L_h \leq \min(L_f^\mu, L_g^\mu)$ by Lemma 3.14), as well as the definition of C_Λ in (3.4),

$$\begin{aligned}\delta_4 &= \left|\kappa_{f,g}^\Delta(\boldsymbol{u}) - \kappa_{f,g}^\Delta(\boldsymbol{u}+\boldsymbol{r}_{\boldsymbol{u}})\right| = \left|\mathbb{E}_{\omega\sim\Lambda}\, h(\boldsymbol{\omega}^\top \boldsymbol{u}) - h(\boldsymbol{\omega}^\top \boldsymbol{u} + \boldsymbol{\omega}^\top \boldsymbol{r}_{\boldsymbol{u}})\right| \\ &\leq \mathbb{E}_{\boldsymbol{\omega}}\left|h(\boldsymbol{\omega}^\top \boldsymbol{u}) - h(\boldsymbol{\omega}^\top \boldsymbol{u} + \boldsymbol{\omega}^\top \boldsymbol{r}_{\boldsymbol{u}})\right| \\ &\leq \mathop{\mathbb{E}}_{\boldsymbol{\omega}} L_h \cdot \left|\boldsymbol{\omega}^\top \boldsymbol{r}_{\boldsymbol{u}}\right| \leq L_h C_\Lambda \|\boldsymbol{r}_{\boldsymbol{u}}\|_2 \leq 2\eta C_\Lambda \cdot \min(L_f^\mu, L_g^\mu).\end{aligned}$$

Putting everything back together, under $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_3$, for any $\boldsymbol{x}', \boldsymbol{y}' \in \Sigma$:

$$\left|\widehat{\kappa}_{f,g}(\boldsymbol{x}',\boldsymbol{y}') - \kappa_{f,g}(\boldsymbol{x}',\boldsymbol{y}')\right| \leq \epsilon_1 + \epsilon_2 + \epsilon_3 + \eta C_\Lambda\left(L_f^\mu + L_g^\mu + 2\cdot\min(L_f^\mu, L_g^\mu)\right). \tag{3.21}$$

It remains to bound the failure probability for each event. For event $\mathcal{E}_1$, we have to bound the probability

$$\mathbb{P}\left[\bar{\mathcal{E}}_1\right] = \mathbb{P}\left[\exists \boldsymbol{x} \in \Sigma_\eta \text{ s.t. } \tfrac{1}{m}\textstyle\sum_{j=1}^m H_j^f(\eta;\boldsymbol{x}) \geq L_f^\mu \eta C_\Lambda + \epsilon_1\right].$$

We first focus on one single center $\boldsymbol{x} \in \Sigma_\eta$. Each associated $H_j^f(\eta;\boldsymbol{x}) = \sup_{\boldsymbol{r}\in\eta\mathbb{B}_2^d} |f(\boldsymbol{\omega}_j^\top \boldsymbol{x} + \boldsymbol{\omega}_j^\top \boldsymbol{r} + \xi_j) - f(\boldsymbol{\omega}_j^\top \boldsymbol{x} + \xi_j)|$ is a random variable identically and independently distributed (where the randomness is due to the draw of $\boldsymbol{\omega}_j$ and ξ_j). The expectation $\mathbb{E}\, H_j^f$ of those variables is bounded by (we use the mean Lipschitz smoothness (3.18) with $t = \xi_j$, $r = \boldsymbol{\omega}_j^\top \boldsymbol{r}$ and $\delta = |\boldsymbol{\omega}_j^\top \boldsymbol{r}|$):

$$\begin{aligned}\mathbb{E}\, H_j^f &= \mathbb{E}_{\boldsymbol{\omega}_j}\mathbb{E}_{\xi_j}\sup_{\boldsymbol{r}\in\eta\mathbb{B}_2^d}|f(\boldsymbol{\omega}_j^\top \boldsymbol{x} + \boldsymbol{\omega}_j^\top \boldsymbol{r} + \xi_j) - f(\boldsymbol{\omega}_j^\top \boldsymbol{x} + \xi_j)| \\ &\leq L_f^\mu \mathop{\mathbb{E}}_{\boldsymbol{\omega}_j}|\boldsymbol{\omega}_j^\top \boldsymbol{r}| = L_f^\mu \eta C_\Lambda.\end{aligned} \tag{3.22}$$

Now we describe how the sum $\frac{1}{m}\sum_{j=1}^{m} H_j^f(\eta;\boldsymbol{x})$ concentrates around its mean $\mathbb{E}\, H_j^f$ with Hoeffding's inequality (note that $0 \leq H_j^f(\eta;\boldsymbol{x}) \leq 2\|f\|_\infty \leq 2$), and use $\mathbb{E}\, H_j^f \leq L_f^\mu \eta C_\Lambda$ to get a probabilistic bound

$$\mathbb{P}\left[\tfrac{1}{m}\textstyle\sum_j H_j^f(\eta;\boldsymbol{x}) \geq L_f^\mu \eta C_\Lambda + \epsilon_1\right] \leq \mathbb{P}\left[\frac{1}{m}\sum_j H_j^f(\eta;\boldsymbol{x}) - \mathbb{E}\, H_j^f \geq \epsilon_1\right] \leq \exp\big(-\tfrac{m\epsilon_1^2}{2}\big).$$

We take a union bound of this result over the $\mathcal{C}_\eta(\Sigma) = |\Sigma_\eta|$ centers $\boldsymbol{x} \in \Sigma_\eta$ to obtain

$$\mathbb{P}\left[\bar{\mathcal{E}}_1\right] \leq \mathcal{C}_\eta(\Sigma)\exp\left(-m\epsilon_1^2/2\right) = \exp\left(\mathcal{H}_\eta(\Sigma) - m\epsilon_1^2/2\right).$$

Moreover, if $m \geq 4\mathcal{H}_\eta(\Sigma)\epsilon_1^{-2}$, then we get $\mathbb{P}\left[\bar{\mathcal{E}}_1\right] \leq e^{-m\epsilon_1^2/4}$. An identical development for $\mathcal{E}_2$ yields $\mathbb{P}\left[\bar{\mathcal{E}}_2\right] \leq e^{-m\epsilon_2^2/4}$ if $m \geq 4\mathcal{H}_\eta(\Sigma)\epsilon_2^{-2}$. For $\mathcal{E}_3$, an union bound of Prop. 3.12 on all pairs in $\Sigma_\eta \times \Sigma_\eta$ gives

$$\mathbb{P}\left[\bar{\mathcal{E}}_3\right] \leq 2\tbinom{\mathcal{C}_\eta(\Sigma)}{2} e^{-m\epsilon_3^2/2} \leq \mathcal{C}_\eta(\Sigma)^2 e^{-m\epsilon_3^2/2} = e^{2\mathcal{H}_\eta(\Sigma) - m\epsilon_3^2/2}.$$

Moreover, if $m \geq 8\mathcal{H}_\eta(\Sigma)\epsilon_3^{-2}$, we get $\mathbb{P}\left[\bar{\mathcal{E}}_3\right] \leq e^{-m\epsilon_3^2/4}$. By union bound, and provided

$$m \geq 4\mathcal{H}_\eta(\Sigma)\cdot\max\left(\epsilon_1^{-2}, \epsilon_2^{-2}, 2\epsilon_3^{-2}\right),$$

the probability of failure of the deterministic bound above is lower than

$$\mathbb{P}\left[\bar{\mathcal{E}}_1 \cup \bar{\mathcal{E}}_2 \cup \bar{\mathcal{E}}_3\right] \leq \mathbb{P}\left[\bar{\mathcal{E}}_1\right] + \mathbb{P}\left[\bar{\mathcal{E}}_2\right] + \mathbb{P}\left[\bar{\mathcal{E}}_3\right] = e^{-m\epsilon_1^2/4} + e^{-m\epsilon_2^2/4} + e^{-m\epsilon_3^2/4}.$$

Finally, the desired result (less generic but more meaningful) is found by imposing equal contributions $\epsilon/4$ by each error term in (3.21), *i.e.*, $\epsilon_1 = \epsilon_2 = \epsilon_3 = \epsilon/4$ and $\eta = \epsilon/(4C_\Lambda[L_f^\mu + L_g^\mu + 2\min(L_f^\mu, L_g^\mu)])$. □

Prop. 3.15 shows that we can control (*e.g.*, by increasing m) the kernel approximation error uniformly, provided we control the smoothness of the "initial" kernel $\kappa^\Delta = \mathcal{F}^{-1}\Lambda$ (through C_Λ) and the mean smoothness of the maps f and g. Improvements are possible, for example, by more carefully setting the values of $\{\epsilon_1, \epsilon_2, \epsilon_3\}$ and η. If Σ is *structured* (*e.g.*, if it consists of sparse vectors or low-rank matrices) and Λ is Gaussian, the value of C_Λ in $\mathbb{E}_{\boldsymbol{\omega}}\,|\boldsymbol{\omega}^\top \boldsymbol{r}| \leq \eta C_\Lambda$ for $\boldsymbol{r} \in (\Sigma - \Sigma) \cap \eta\mathbb{B}_2^d$ (which controls the bounds on δ_1,

δ_2 and δ_4) can be related to the Gaussian mean width of $\Sigma - \Sigma$ [CRPW12].

Example 3.16. Consider once again the case of a Gaussian kernel with unit bandwidth (*i.e.*, $C_\Lambda = 1$), with a signal space Σ made of bounded signals (inside the unit Euclidean ball $\mathbb{B}_2^d$) lying in a union of S subspaces of $\mathbb{R}^d$ with dimension s. In this case, according to the entropy of this signal model (see Ex. 3.4), the kernel approximation error $|\widehat{\kappa}_{f,g}(\boldsymbol{x},\boldsymbol{y}) - \kappa_{f,g}(\boldsymbol{x},\boldsymbol{y})|$ is uniformly bounded over Σ, with high probability, provided that the number of features satisfies $m \geq C\epsilon^{-2}[s\log\left(\frac{4}{\epsilon}(L_f^\mu + L_g^\mu + 2\min(L_f^\mu, L_g^\mu))\right) + \log S]$. For instance, for bounded s-sparse signals, we need $m \geq Cs \cdot \epsilon^{-2}\log\left(\frac{4ed}{cs\epsilon}(L_f^\mu + L_g^\mu + 2\min(L_f^\mu, L_g^\mu))\right)$.

We conclude this section by showing that Prop. 3.15 allows characterizing the proximity of two approximated kernels $\widehat{\kappa}_{f,g}$ and $\widehat{\kappa}_{f',g}$ (given three functions $f, f', g \in \mathrm{PF}$) when they are related by identical expectations $\mathbb{E}\widehat{\kappa}_{f,g} = \mathbb{E}\widehat{\kappa}_{f',g} = \kappa_0$. While this result could be achieved by a simple use of the triangular inequality—from $|\widehat{\kappa}_{f,g} - \widehat{\kappa}_{f',g}| \leq |\widehat{\kappa}_{f,g} - \kappa_0| + |\widehat{\kappa}_{f',g} - \kappa_0|$ and using the same proposition to bound the last two terms—the following corollary provides a more direct bound, possibly tighter.

Corollary 3.17 (Proximity of approximated RPF kernels)**.** *Given $\epsilon > 0$, a compact set Σ, two 2π-periodic functions f, f' such that their difference $f - f' \in \mathrm{PF}$, as well as a third periodic function $g \in \mathrm{PF}$, such that there exist finite mean smoothness constants $L_{f-f'}^\mu$ and L_g^μ, and Λ such that $C_\Lambda < \infty$, if $\kappa_{f,g}(\cdot,\cdot) = \kappa_{f',g}(\cdot,\cdot)$ and if the feature dimension is larger than*

$$m \geq 128 \cdot \tfrac{1}{\epsilon^2} \cdot \mathcal{H}_{\epsilon/c}(\Sigma), \tag{3.23}$$

with constant $c = 4C_\Lambda(L_{f-f'}^\mu + L_g^\mu + 2\min(L_{f-f'}^\mu, L_g^\mu))$, then

$$\left|\widehat{\kappa}_{f,g}(\boldsymbol{x},\boldsymbol{y}) - \widehat{\kappa}_{f',g}(\boldsymbol{x},\boldsymbol{y})\right| \leq \epsilon, \quad \forall \boldsymbol{x},\boldsymbol{y} \in \Sigma, \tag{3.24}$$

with probability exceeding $1 - 3\exp(-\frac{m\epsilon^2}{64})$.

Proof. We simply observe that, by linearity of the kernels with respect to their supporting functions, for any $\boldsymbol{x},\boldsymbol{y} \in \Sigma$, we can write $\widehat{\kappa}_{f,g}(\boldsymbol{x},\boldsymbol{y}) - \widehat{\kappa}_{f',g}(\boldsymbol{x},\boldsymbol{y}) = \widehat{\kappa}_{\tilde{f},g}(\boldsymbol{x},\boldsymbol{y})$ with $\tilde{f} := f - f'$. The proof then follows by applying Prop. 3.15 to the RPFs supported by $\tilde{f}, g \in \mathrm{PF}$, with the vanishing kernel $\mathbb{E}\,\widehat{\kappa}_{\tilde{f},g}(\boldsymbol{x},\boldsymbol{y}) = \kappa_{f,g}(\boldsymbol{x},\boldsymbol{y}) - \kappa_{f',g}(\boldsymbol{x},\boldsymbol{y}) = 0$. □

In Sec. 3.6.3, we will use this corollary in combination with Prop. 3.15 to compare the performance of a machine learning algorithm (the kernel

support vector machine, SVM) on a given classification task when learning and inference are using identical approximated kernels (*i.e.*, when the learning is performed using the RFF) or only kernels that are asymptotically equal (when the learning stage uses the expected kernel).

3.5 Semi-quantized random Fourier features

In this section, we explore one practical application of our general results from Sec. 3.4 by instantiating them on the semi-quantized scenario motivated in the Introduction (see Fig. 3.1). More precisely, we consider the asymmetric RPF setting $\widehat{\kappa}_{f,g}(\boldsymbol{x},\boldsymbol{y}) = \langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_g(\boldsymbol{y})\rangle$ in the particular case where: *(i)* one of the signals $\boldsymbol{x}$ is available through its one-bit universal features $\boldsymbol{z}_q(\boldsymbol{x}) \in \{-\frac{1}{\sqrt{m}}, +\frac{1}{\sqrt{m}}\}^m$ (that is, the first periodic map f is the square wave q, alternating between ± 1 with period 2π, see Fig. 3.2); *(ii)* the other signal $\boldsymbol{y}$ is available through its classical (full-precision) random Fourier features $\boldsymbol{z}_{\cos}(\boldsymbol{y})$ (that is, the second map g is a cosine).

Concretely, we start by highlighting a striking general result: when the classical RFF (*i.e.*, for which $g(\cdot) = \exp(\mathrm{i}\cdot)$ or $\cos(\cdot)$) are combined with *any* mean smooth function $f \in \mathrm{PF}$, then the asymmetric inner product $\widehat{\kappa}_{f,g}(\boldsymbol{x},\boldsymbol{y})$ exactly recovers the initial kernel κ that would be approached by symmetric usual RFF $\widehat{\kappa}_{g,g}(\boldsymbol{x},\boldsymbol{y})$. Then, to combine this fact with the binary square wave $f = q$, we prove that q is mean smooth (Def. 3.13). This finally allows us to obtain a probabilistic uniform bound on the kernel approximation error for the semi-quantized scenario pair, demonstrating in the process how to deal with the scaling issues that appear in such schemes by using the normalization (3.13).

Expected kernel with a single-frequency nonlinearity: Let us begin by noting an interesting consequence of Prop. 3.8. From RPF $\boldsymbol{z}_f(\boldsymbol{x})$ captured on $\boldsymbol{x}$ with *any* nonlinearity $f \in \mathrm{PF}$ whose fundamental period is exactly 2π, one can recover in expectation, for a given vector $\boldsymbol{y}$, the evaluation the shift-invariant kernel $\kappa(\boldsymbol{x},\boldsymbol{y}) = \kappa^{\Delta}(\boldsymbol{x}-\boldsymbol{y})$ associated with the sampling of the projections $\boldsymbol{\omega}_j \sim \Lambda = \mathcal{F}\kappa^{\Delta}$.

Indeed, using Prop. 3.8 in the complex field, and setting $g(\cdot) = \exp(\mathrm{i}\cdot)$ for the RPF of $\boldsymbol{y}$—which in this case is the RFF (Def. 3.1)—ensures that $\kappa_{f,\exp(\mathrm{i}\cdot)}(\boldsymbol{x},\boldsymbol{y}) = F_1\kappa(\boldsymbol{x},\boldsymbol{y})$. Intuitively, the dithering averages out all the high-frequency components in f, leaving only its fundamental frequency. When dealing with real-valued quantities $\kappa, f \in \mathbb{R}$, we can use the real

RFF (where $g(\cdot) = \Re \exp(\mathrm{i}\cdot) = \cos(\cdot)$) instead, and using the normalized kernel (3.13) with (3.11) gives

$$\begin{aligned}\dot{\kappa}_{f,\cos}(\boldsymbol{x},\boldsymbol{y}) &= \tfrac{1}{\Re F_1}\kappa_{f,\cos}(\boldsymbol{x},\boldsymbol{y}) \\ &= \tfrac{1}{\Re F_1}\,\mathbb{E}\,\langle f(\boldsymbol{\Omega}^\top \boldsymbol{x}+\boldsymbol{\xi}), \cos(\boldsymbol{\Omega}^\top \boldsymbol{y}+\boldsymbol{\xi})\rangle = \kappa(\boldsymbol{x},\boldsymbol{y}),\end{aligned} \tag{3.25}$$

since $\langle f, g\rangle = \sum_k F_k G_k^* = \Re F_1$. We thus recover, through $\dot{\kappa}_{f,\cos}$ the initial kernel κ, thanks to *a rescaling* by $1/\langle f, g\rangle = 1/\Re\{F_1\}$ which must be taken into account for a fair comparison.

Remark 3.18. In theory, we can thus recover, from $\boldsymbol{z}_f(\boldsymbol{x})$, the kernel κ at many different scales by "probing" it with $\boldsymbol{z}_{\cos(k\,\cdot)}(\boldsymbol{y})$ for any scale k such that $F_k \neq 0$. However, we observe in practice that the kernel approximation error quickly increases with k. This can be understood in the light of Prop. 3.15, since one easily show that $L_{\cos(k\,\cdot)} = |k|$ and[3] $\frac{2}{\pi}|k| \leq L^\mu_{\cos(k\,\cdot)} \leq |k|$. Moreover, if $\|f\|^2 = \sum_k |F_k|^2$ is bounded, each rescaling factor $(\Re\{F_k\})^{-1}$ grows as k increases; for instance, $(\Re\{F_k\})^{-1} \propto |k|$ for $f = q$.

The asymmetric scheme in (3.25) is interesting because it allows the same level of control over the approximated kernel as the usual RFF (which is an improvement compared to the scale mixture of RFF kernels imposed by Prop. 3.7) while still enjoying the freedom to use any type of features $\boldsymbol{z}_f(\boldsymbol{x})$ for one of the signals being compared—a particularly appealing choice being $f = q$, the one-bit universal quantization. However, in order to use Prop. 3.15 to obtain uniform error bounds, we still need to prove the mean Lipschitz smoothness of this (discontinuous) map.

Mean Lipschitz smoothness of universal quantization: We now show that the one-bit universal quantization function q (*i.e.*, the square wave) has the mean Lipschitz property—although it is discontinuous. The same strategy could be used to prove the mean Lipschitz smoothness of any function in PF with a finite number of discontinuities per period (as done, for example, with the modulo map in Section 3.7).

Proposition 3.19. *The one-bit universal quantization function q, defined in* (3.8),

[3] First, $L^\mu_{\cos(k\cdot)} \leq L^\mu_{\exp(ik\cdot)} = |k|$ since $|\cos(\alpha) - \cos(\beta)| \leq |e^{\mathrm{i}\alpha} - e^{\mathrm{i}\beta}|$ for all $\alpha, \beta \in \mathbb{R}$. Second, from $|\cos(k(t+r)) - \cos(kt)| = 2|\sin(k(t+\frac{r}{2}))\sin\frac{kr}{2}|$ for all $t, r \in \mathbb{R}$, we get, by fixing $r = \delta$ in (3.18), $L^\mu_f \geq \sup_{\delta>0} \frac{2}{\delta}|\sin\frac{k\delta}{2}|\,\mathbb{E}_{t\sim\mathcal{U}([0,2\pi])}|\sin(k(t+\frac{\delta}{2}))| = \frac{2}{\pi}|k|$.

has the mean Lipschitz smoothness property (Def. 3.13) with constant

$$L_q^\mu = \tfrac{4}{\pi}\|q\|_\infty = \tfrac{4}{\pi}. \tag{3.26}$$

Proof. By definition of the mean smoothness property, we must find L_q^μ such that

$$\tfrac{1}{2\pi}\int_0^{2\pi} \max_{r\in[-\delta,\delta]}\{|q(t+r)-q(t)|\}\,\mathrm{d}t \le L_q^\mu \cdot \delta.$$

We start by characterizing the integrand $I_\delta(t) := \max_{|r|\le\delta}\{|q(t+r)-q(t)|\}$. Since $q(t)$ is constant (in particular, $q(t) = \pm\|q\|_\infty$) everywhere except on discontinuities at $t \in \frac{\pi}{2} + \pi\mathbb{Z}$ where its height changes by an absolute step of $2\|q\|_\infty$, we have for $k \in \mathbb{Z}$ that (see Fig. 3.2)

$$I_\delta(t) = \begin{cases} 0 & \frac{\pi}{2} + k\pi + \delta < t < \frac{\pi}{2} + (k+1)\pi - \delta \\ 2\|q\|_\infty & \frac{\pi}{2} + k\pi - \delta \le t \le \frac{\pi}{2} + k\pi + \delta. \end{cases}$$

Integrating this over one period gives $\int_0^{2\pi} I_\delta(t)\mathrm{d}t = \min(4\delta, 2\pi)\cdot 2\|q\|_\infty \le 8\|q\|_\infty\cdot\delta$, *i.e.*, $L_q^\mu = \frac{4}{\pi}\|q\|_\infty$. □

Combining quantized and cosine features: We are interested in approximating a specific kernel $\kappa(\boldsymbol{x},\boldsymbol{y})$ by the asymmetric features product, *i.e.*, $\langle \boldsymbol{z}_q(\boldsymbol{x}), \boldsymbol{z}_{\cos}(\boldsymbol{y})\rangle$. This product gives on average $\kappa_{q,\cos} = \frac{2}{\pi}\kappa$ (recall from (3.8) that $Q_k = \frac{2}{k\pi}(-1)^{(k-1)/2}$ for k odd and 0 otherwise), and the re-scaled approximation $\widetilde{\kappa}_{q,\cos}$ defined in (3.13) in this case is given by

$$\widetilde{\kappa}_{q,\cos}(\boldsymbol{x},\boldsymbol{y}) := \tfrac{\pi}{2}\cdot\widehat{\kappa}_{q,\cos}(\boldsymbol{x},\boldsymbol{y}) = \tfrac{\pi}{2}\langle \boldsymbol{z}_q(\boldsymbol{x}), \boldsymbol{z}_{\cos}(\boldsymbol{y})\rangle \approx \kappa(\boldsymbol{x},\boldsymbol{y}). \tag{3.27}$$

We bound the error of approximating the kernel over an infinite compact set Σ thanks to Prop. 3.15.

Corollary 3.20 (Uniform kernel approximation error from quantized-complex asymmetric features)**.** *Given $\epsilon > 0$, a compact set Σ, and the frequency distribution Λ such that $C_\Lambda < \infty$, provided that*

$$m \ge 32\pi^2\cdot\tfrac{1}{\epsilon^2}\cdot\mathcal{H}_{\epsilon/((8+6\pi)C_\Lambda)}(\Sigma), \tag{3.28}$$

the following kernel approximation bound holds uniformly:

$$\left|\widetilde{\kappa}_{q,\cos}(\boldsymbol{x},\boldsymbol{y}) - \kappa(\boldsymbol{x},\boldsymbol{y})\right| \le \epsilon, \quad \forall \boldsymbol{x},\boldsymbol{y}\in\Sigma, \tag{3.29}$$

with probability exceeding $1 - 3\exp(-\frac{m\epsilon^2}{16\pi^2})$

Proof. Apply Prop. 3.15 with $f = q$, $g = \cos$, using that $L_q^\mu = \frac{4}{\pi}$ (from Prop. 3.19) and $L_{\cos}^\mu = 1$: for any given $\epsilon' > 0$, if $m \geq 128 \cdot \frac{1}{\epsilon'^2} \cdot \mathcal{H}_{\epsilon'/c}(\Sigma)$ with $c = (12 + \frac{16}{\pi})C_\Lambda$,

$$\mathbb{P}\left[\exists\, \boldsymbol{x}, \boldsymbol{y} \in \Sigma \,:\, \left|\widehat{\kappa}_{q,\cos}(\boldsymbol{x}, \boldsymbol{y}) - \tfrac{2}{\pi}\kappa(\boldsymbol{x}, \boldsymbol{y})\right| > \epsilon'\right] \leq 3\exp\left(-\tfrac{m\epsilon'^2}{64}\right).$$

To take into account the scaling of the kernel, set $\epsilon = \frac{\epsilon'}{\Re Q_1} = \frac{\pi}{2}\epsilon'$. □

Example 3.21. Consider a final time our example of a union of S s-dimensional subspaces (see Ex. 3.4) combined with the Gaussian kernel with unit bandwidth (and $C_\Lambda = 1$). In this case, the kernel approximation error, given by $|\widetilde{\kappa}_{q,\cos}(\boldsymbol{x}, \boldsymbol{y}) - \kappa(\boldsymbol{x}, \boldsymbol{y})|$, is uniformly bounded over Σ, with high probability, provided that the number of features satisfies $m \geq C\epsilon^{-2}(s\log(\frac{8+6\pi}{\epsilon}) + \log S)$, which reduces to $m \geq C\epsilon^{-2}s\log(\frac{(8+6\pi)ed}{s\epsilon})$ for bounded s-sparse signals.

Corollary 3.20 provides a theoretical guarantee for the semi-quantized scheme presented in the Introduction. In the next section, we further validate this approach from numerical simulations.

3.6 Experiments

In all our experiments, we are interested in approximating a kernel $\kappa(\boldsymbol{x}, \boldsymbol{y})$, associated with the RFF sampled with Λ^m, by the inner product of random periodic features. We focus on (combinations of) the two types of features discussed in the previous section: the "real" random Fourier features $\boldsymbol{z}_{\cos}(\boldsymbol{x}) = \frac{1}{\sqrt{m}}\cos(\boldsymbol{\Omega}^\top \boldsymbol{x} + \boldsymbol{\xi}) \in \mathbb{R}^m$, and the universal features $\boldsymbol{z}_q(\boldsymbol{x}) = \frac{1}{\sqrt{m}}q(\boldsymbol{\Omega}^\top \boldsymbol{x} + \boldsymbol{\xi}) \in \{-\frac{1}{\sqrt{m}}, +\frac{1}{\sqrt{m}}\}^m$, where m is the number of features (or dimension), and where we generate $\boldsymbol{\Omega} \sim \Lambda^m$ and $\boldsymbol{\xi} \sim \mathcal{U}^m([0, 2\pi))$. Recalling the rescaling (3.13) for fair comparisons of the approximated kernels with κ, we thus consider three possible combinations: the classical (real) random Fourier features (with $\|f\|^2 = \|\cos(\cdot)\|^2 = \frac{1}{2}$),

$$\widetilde{\kappa}_{\cos,\cos}(\boldsymbol{x}, \boldsymbol{y}) = 2\langle \boldsymbol{z}_{\cos}(\boldsymbol{x}), \boldsymbol{z}_{\cos}(\boldsymbol{y})\rangle \approx \kappa(\boldsymbol{x}, \boldsymbol{y}),$$

our asymmetric "semi-quantized" scheme (with $\langle f, g\rangle = 2/\pi$),

$$\widetilde{\kappa}_{q,\cos}(\boldsymbol{x}, \boldsymbol{y}) = \tfrac{\pi}{2}\langle \boldsymbol{z}_q(\boldsymbol{x}), \boldsymbol{z}_{\cos}(\boldsymbol{y})\rangle \approx \kappa(\boldsymbol{x}, \boldsymbol{y}),$$

and the fully quantized inner product from [BR13] (with $\|q\|^2 = \sum_k |Q_k|^2 = 1$),

$$\widetilde{\kappa}_{q,q}(\boldsymbol{x},\boldsymbol{y}) = \langle \boldsymbol{z}_q(\boldsymbol{x}), \boldsymbol{z}_q(\boldsymbol{y})\rangle \approx \kappa_{q,q}(\boldsymbol{x},\boldsymbol{y}) = \textstyle\sum_{k\in\mathbb{Z}} |Q_k|^2 \kappa(k\boldsymbol{x}, k\boldsymbol{y}) \neq \kappa(\boldsymbol{x},\boldsymbol{y}).$$

3.6.1 Qualitative analysis of the expected kernel

As a first experiment, we visually demonstrate that our asymmetric product $\widetilde{\kappa}_{q,\cos}$ indeed approaches a target kernel κ. As target, we use the Gaussian kernel $\kappa(\boldsymbol{x},\boldsymbol{y}) = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_2^2}{2\sigma^2})$ (for which Λ is the Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \sigma^{-2}\boldsymbol{I}_d)$), as well as the Laplace kernel $\kappa(\boldsymbol{x},\boldsymbol{y}) = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_1}{\tau})$ (where Λ is the Cauchy distribution (C.9)), both in dimension $d = 5$.

We evaluate the three inner products $\widetilde{\kappa}_{\cos,\cos}$, $\widetilde{\kappa}_{q,\cos}$ and $\widetilde{\kappa}_{q,q}$ on $n = 2000$ pairs of vectors $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, that are generated as follows. We first sample $\boldsymbol{x}_i \in \mathbb{R}^5$ according to a standard normal distribution, then pick $\boldsymbol{y}_i = \boldsymbol{x}_i + \lambda_i \boldsymbol{u}_i$, where $\boldsymbol{u}_i$ is a randomly chosen unit vector (*i.e.*, normalized such that $\|\boldsymbol{u}_i\|_p = 1$ with $p = 2$ for the Gaussian kernel and $p = 1$ for the Laplace one), and $\lambda_i = \frac{(i-1)\lambda_{\max}}{(n-1)}$ is a controlled distance which is incremented for each pair, linearly increasing from 0 to $\lambda_{\max} = 5$. This ensures that we test the kernel approximations uniformly in the desired range of distances $\|\boldsymbol{x} - \boldsymbol{y}\|_p$.

We then generate one realization of $\boldsymbol{\Omega}, \boldsymbol{\xi}$ (with $m = 200$ for the Gaussian kernel, and $m = 2000$ for the Laplace kernel, values which were arbitrarily chosen to get pleasing visualizations), which we use to compute the real RFF $\left\{\left(\boldsymbol{z}_{\cos}(\boldsymbol{x}_i), \boldsymbol{z}_{\cos}(\boldsymbol{y}_i)\right)\right\}_{i=1}^n$ as well as the universal quantization features $\left\{\left(\boldsymbol{z}_q(\boldsymbol{x}_i), \boldsymbol{z}_q(\boldsymbol{y}_i)\right)\right\}_{i=1}^n$, from which we get n evaluations of the classical RFF inner product $\widetilde{\kappa}_{\cos,\cos}(\boldsymbol{x}_i,\boldsymbol{y}_i) = 2\langle \boldsymbol{z}_{\cos}(\boldsymbol{x}_i), \boldsymbol{z}_{\cos}(\boldsymbol{y}_i)\rangle$, the asymmetric product $\widetilde{\kappa}_{q,\cos}(\boldsymbol{x}_i,\boldsymbol{y}_i) = \frac{\pi}{2}\langle \boldsymbol{z}_q(\boldsymbol{x}_i), \boldsymbol{z}_{\cos}(\boldsymbol{y}_i)\rangle$, and the fully quantized product $\widetilde{\kappa}_{q,q}(\boldsymbol{x}_i,\boldsymbol{y}_i) = \langle \boldsymbol{z}_q(\boldsymbol{x}_i), \boldsymbol{z}_q(\boldsymbol{y}_i)\rangle$.

Those evaluations are shown as black dots in Fig. 3.3 for the Gaussian and Laplace kernels in the top and bottom rows, respectively. As predicted by the theory, both the RFF product $\widetilde{\kappa}_{\cos,\cos}$ and our semi-quantized product $\widetilde{\kappa}_{q,\cos}$ concentrate around the target kernel κ (in red). As expected from [BRM17], in the fully quantized case the product $\widetilde{\kappa}_{q,q}$ rather concentrates around a different "distorted" kernel, $\kappa_{q,q}$. Note that we increased the feature space dimension m tenfold for the Cauchy kernel, which reduced the variance of the approximation. However, it is difficult to notice a substantial difference of approximation quality between the plain RFF $\widetilde{\kappa}_{\cos,\cos}$ and the semi-quantized asymmetric scheme $\widetilde{\kappa}_{q,\cos}$. We thus per-

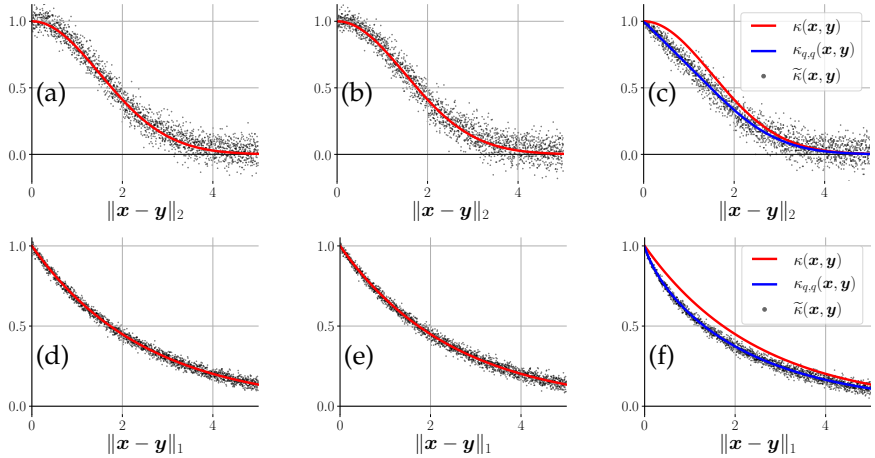


Fig. 3.3 Comparison between the target kernel $\kappa(\boldsymbol{x},\boldsymbol{y})$ (red curves), and the approximations (the black scatter plots each evaluated over $n = 200$ pairs $\{(\boldsymbol{x}_i,\boldsymbol{y}_i)\}_{i=1}^n$) using, for (a,d), the plain random Fourier features $\widetilde{\kappa}_{\cos,\cos}(\boldsymbol{x}_i,\boldsymbol{y}_i) = 2\langle \boldsymbol{z}_{\cos}(\boldsymbol{x}_i), \boldsymbol{z}_{\cos}(\boldsymbol{y}_i)\rangle$, for (b,e), our asymmetric cosine-quantized pair $\widetilde{\kappa}_{q,\cos}(\boldsymbol{x}_i,\boldsymbol{y}_i) = \frac{\pi}{2}\langle \boldsymbol{z}_q(\boldsymbol{x}_i), \boldsymbol{z}_{\cos}(\boldsymbol{y}_i)\rangle$, and for (c,f), only quantized features $\widetilde{\kappa}_{q,q}(\boldsymbol{x}_i,\boldsymbol{y}_i) = \langle \boldsymbol{z}_q(\boldsymbol{x}_i), \boldsymbol{z}_q(\boldsymbol{y}_i)\rangle$. In the last case, the "distorted" expected kernel $\mathbb{E}\,\widetilde{\kappa}_{q,q} = \kappa_{q,q}$ is shown in blue. For (a-c; top row), the comparison is made for the target Gaussian kernel $\kappa(\boldsymbol{x},\boldsymbol{y}) = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_2^2}{2\sigma^2})$ with scale $\sigma = 1.5$, and the approximated kernels use $m = 200$ random features evaluated. For (d-f; bottom row), the target kernel is the Laplace kernel $\kappa(\boldsymbol{x},\boldsymbol{y}) = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_1}{\tau})$ with scale $\tau = 1.5$, and its different approximations are set with $m = 2000$.

form a more quantitative exploration of the error $|\widetilde{\kappa}_{q,\cos} - \kappa|$ in the next experiment.

3.6.2 Quantitative analysis of the approximation error

To perform a more quantitative analysis of the kernel approximation from Cor. 3.20, we perform another set of experiments that highlight the evolution of the worst-case error (associated with the hybrid estimation (3.27)),

$$\epsilon_{q,\cos}(\Sigma) := \sup_{\boldsymbol{x},\boldsymbol{y}\in\Sigma} |\widetilde{\kappa}_{q,\cos}(\boldsymbol{x},\boldsymbol{y}) - \kappa(\boldsymbol{x},\boldsymbol{y})|,$$

as a function of m. In this synthetic experiment, we work with a finite set of signals $\Sigma = \{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^n$ obtained from a Gaussian distribution $\boldsymbol{x}_i \sim_{\text{i.i.d.}}$

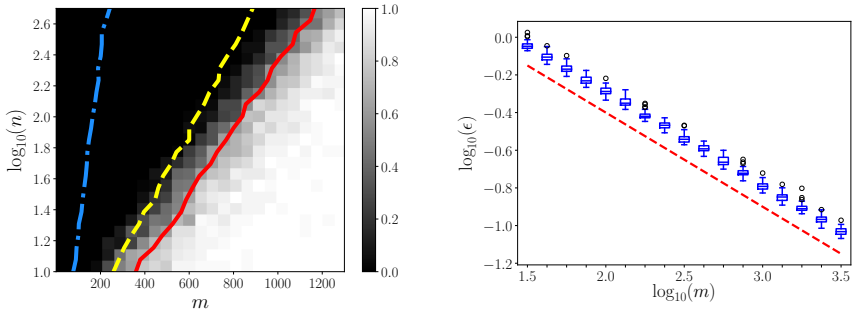
$\mathcal{N}(\mathbf{0}, \widetilde{\sigma}^2 \boldsymbol{I}_d)$ in with $\widetilde{\sigma} = 10$. We target a Gaussian kernel κ of bandwidth $\sigma = 0.25$, and evaluate the absolute approximation error $\epsilon_{q,\cos}(\Sigma)$ over all vector pairs of Σ. We record the largest error encountered this way, and repeat this process for several feature dimensions m.

First, we let $n = |\Sigma|$, the number of signals, vary between 10 and 500 (by sampling 21 equally-spaced values for $\log_{10}(n)$), and generate a new dataset in dimension $d = 32$ each time. For each value of m (varying uniformly between 100 and 1300), we repeat 50 independent draws of $\boldsymbol{\Omega}$ and $\boldsymbol{\xi}$ and report Fig. 3.4a the number of times that $\epsilon_{q,\cos}(\Sigma) \leq \bar{\epsilon}$ for a fixed threshold $\bar{\epsilon} = 0.15$ (*i.e.*, we report the empirical "success rate" of the embedding). As expected, the feature space dimension m needed to succeed (highlighted in red for 50% success rate) scales as $O(\log n)$. We also show in dashed yellow the same transition for the worst-case error $\epsilon_{\cos,\cos}(\Sigma)$ (evaluating $|\widetilde{\kappa}_{\cos,\cos} - \kappa|$ over all vector pairs of Σ) committed by the plain RFF, which shows the price to pay for quantization. Roughly speaking, the same success rate is achieved for $\widetilde{\kappa}_{q,\cos}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as for $\widetilde{\kappa}_{\cos,\cos}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ provided we take $\sim 33\%$ more random features, which still corresponds to a bitrate reduction for the features of the fist signal $\boldsymbol{x}_i$. Finally, for the sake of comparison we also show in blue the same success rate but when measuring the proximity error between the two approximations (semi-quantized and usual RFF), *i.e.*, $|\widetilde{\kappa}_{q,\cos} - \kappa_{\cos,\cos}|$, which relates to our bound in Cor. 3.17.

Second, we fix one single dataset Σ of $n = 200$ signals in $\mathbb{R}^5$, but record the precise value of the worst-case error $\epsilon := \epsilon_{q,\cos}(\Sigma)$ for each of the 50 draw of $\boldsymbol{\Omega}, \boldsymbol{\xi}$ at different values of m (this time varying along a logarithmic scale). We display the various errors ϵ obtained as box-plots in Fig. 3.4b. As can be seen by comparison with the $-1/2$ slope in red, the error is controlled with high probability (discarding the outliers from the box-plots) provided that $m = O(\epsilon^{-2})$, as expected from Prop. 3.15.

3.6.3 Application: semi-quantized support vector machines

As a last experiment, we demonstrate how the asymmetric features can be used in practice, for the particular case of Support Vector Machine classification [BGV92, SSB+02] (see Subsection 2.1.1), where the goal is to assign a class label $y' \in \mathcal{Z}$ to new query vectors $\boldsymbol{x}' \in \Sigma$ from labeled training data $\mathcal{T} := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$. In the binary classification case (labels $y_i \in \{\pm 1\}$), given a kernel $\kappa^\sharp : \Sigma \times \Sigma \to \mathbb{R}$, the learned SVM classifier θ predicts the



(a) Empirical succes rate *w.r.t.* m and n. **(b)** Largest detected error *w.r.t.* m.

Fig. 3.4 (a) Empirical success rate (from 100% success in white to 0% in black) of the kernel approximation (defining success as $\epsilon_{q,\cos}(\Sigma) < \bar{\epsilon} = 0.15$), as a function of m (log scale) for varying dataset size $n = |\Sigma|$. The transition to 50% success or more is highlighted in solid red; the same curve is shown for the success rate of the classical RFF (when $\epsilon_{\cos,\cos}(\Sigma) < \bar{\epsilon}$) in dashed yellow. The blue line represents the success rate related to the proximity between those two kernel approximations, *i.e.*, when $\sup_{x,y\in\Sigma} |\kappa_{q,\cos}(x,y) - \kappa_{\cos,\cos}(x,y)| < \bar{\epsilon}$. (b) Largest kernel approximation error $\epsilon := \epsilon_{q,\cos}(\Sigma)$ as a function of m for 50 draws of $\mathbf{\Omega}$ and $\boldsymbol{\xi}$ (the blue box-plots). The dashed red line shows the slope $\log_{10}(\epsilon) \sim -\frac{1}{2}\log_{10}(m)$, for reference.

class of an incoming vector $\boldsymbol{x}'$ as

$$\theta(\boldsymbol{x}') = \operatorname{sign}\big(\textstyle\sum_{i\in\mathcal{S}^*} \alpha_i y_i\, \kappa^\sharp(\boldsymbol{x}', \boldsymbol{x}_i) + b\big), \tag{3.30}$$

where $\mathcal{S}^* \subset [n]$ is the index set of support vectors $\boldsymbol{x}_i$, $0 < \alpha_i \leq R$ are the related weights, and b is a bias (or "intercept") term. The quantities $\{\mathcal{S}^*, \alpha_i, b\}$ are the parameters to be learned during the training stage, while the kernel $\kappa^\sharp$ and regularization strength $R > 0$ (where a smaller R corresponds to more regularization) are hyper-parameters to be set beforehand. In the multi-class case (where $y_i \in [N]$ for N classes), we use the "one-versus-rest[4]" strategy where one binary classifier is trained to recognize each class.

In this experiment, given a simple classification task described below, we propose to train the SVM with a given kernel $\kappa^\sharp_{\mathrm{L}}$, and to test the classification of new samples with another kernel $\kappa^\sharp_{\mathrm{T}}$ that approximates $\kappa^\sharp_{\mathrm{L}}$, hence assessing how the classifier θ is impacted by this modification. We consider two options. In the first we train a kernel SVM on the raw data $\mathcal{T}$ with a "true" kernel $\kappa^\sharp_{\mathrm{L}} = \kappa$ and use the approximated kernels provided by random periodic features (setting $\kappa^\sharp_{\mathrm{T}}$ to the kernels $\widetilde{\kappa}_{\cos,\cos}$, $\widetilde{\kappa}_{q,\cos}$ and $\widetilde{\kappa}_{q,q}$ defined from (3.13)) *only at the inference stage*. In this mode, which is the viewpoint we adopted in most of this work (*e.g.*, in Prop. 3.15), we thus interpret the RPF inner products as a means to approximate as well as possible the given kernel κ.

In a second case, we directly train a linear SVM on the cosine random Fourier features of the training set $\mathcal{T}' := \{(\boldsymbol{z}_{\cos}(\boldsymbol{x}_i), y_i)\}_{i=1}^n$, which amounts to using $\kappa^\sharp_{\mathrm{L}} = \widetilde{\kappa}_{\cos,\cos}$ in (3.30) as the reference kernel during training. At the testing stage, we still set $\kappa^\sharp_{\mathrm{T}}$ to $\widetilde{\kappa}_{\cos,\cos}$, $\widetilde{\kappa}_{q,\cos}$ and $\widetilde{\kappa}_{q,q}$. In this scenario, the random periodic features are rather (implicitly) used to define a specific kernel $\widetilde{\kappa}_{\cos,\cos}$ that generalizes as well as possible without caring about the approximation $\widetilde{\kappa}_{\cos,\cos} \approx \kappa$; this view is more faithful to recent research on the generalization capabilities of learning from RFF [ZMDR18, YLM+12, RR17, GLK+20]. Our other RPF products used at the test ($\widetilde{\kappa}_{q,\cos}$ and $\widetilde{\kappa}_{q,q}$) are then to be understood as approximations to $\kappa^\sharp_{\mathrm{L}} = \widetilde{\kappa}_{\cos,\cos}$ rather than to the original κ, as explained in Cor. 3.17, and as measured by the blue curve in Fig. 3.4a.

[4] also known as "one-versus-all".

Table 3.1 Scenarios and color coding for Fig. 3.5.

Color	RPF for		Kernel at the test: $\kappa_{\mathrm{T}}^{\sharp}(\boldsymbol{x}', \boldsymbol{x}_i)$
	Query $\boldsymbol{x}'$	SVs $\{\boldsymbol{x}_i\}_{i \in \mathcal{S}^*}$	
black	$\boldsymbol{z}_{\cos}(\boldsymbol{x}')$	$\boldsymbol{z}_{\cos}(\boldsymbol{x}_i)$	$\widetilde{\kappa}_{\cos,\cos}(\boldsymbol{x}', \boldsymbol{x}_i) = 2\langle \boldsymbol{z}_{\cos}(\boldsymbol{x}'), \boldsymbol{z}_{\cos}(\boldsymbol{x}_i) \rangle \approx \kappa(\boldsymbol{x}', \boldsymbol{x}_i)$
green	$\boldsymbol{z}_q(\boldsymbol{x}')$	$\boldsymbol{z}_{\cos}(\boldsymbol{x}_i)$	$\widetilde{\kappa}_{q,\cos}(\boldsymbol{x}', \boldsymbol{x}_i) = \frac{\pi}{2} \langle \boldsymbol{z}_q(\boldsymbol{x}'), \boldsymbol{z}_{\cos}(\boldsymbol{x}_i) \rangle \approx \kappa(\boldsymbol{x}', \boldsymbol{x}_i)$
blue	$\boldsymbol{z}_{\cos}(\boldsymbol{x}')$	$\boldsymbol{z}_q(\boldsymbol{x}_i)$	$\widetilde{\kappa}_{\cos,q}(\boldsymbol{x}', \boldsymbol{x}_i) = \frac{\pi}{2} \langle \boldsymbol{z}_q(\boldsymbol{x}'), \boldsymbol{z}_{\cos}(\boldsymbol{x}_i) \rangle \approx \kappa(\boldsymbol{x}', \boldsymbol{x}_i)$
red	$\boldsymbol{z}_q(\boldsymbol{x}')$	$\boldsymbol{z}_q(\boldsymbol{x}_i)$	$\widetilde{\kappa}_{q,q}(\boldsymbol{x}', \boldsymbol{x}_i) = \langle \boldsymbol{z}_q(\boldsymbol{x}'), \boldsymbol{z}_{\cos}(\boldsymbol{x}_i) \rangle$

Synthetic data

Specifically, for both contexts, we generate a synthetic dataset of 10 000 samples in $\mathbb{R}^2$ by generating a mixture of 4 Gaussians for each of the $N = 5$ different classes, separated into $n = 8000$ training and 2000 testing samples (see Fig. 3.5, left). Regarding the true kernel, we set it to a Gaussian kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa^{\Delta}(\boldsymbol{x} - \boldsymbol{x}') = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{2\sigma^2})$ with bandwidth $\sigma = 2$, and fixed the regularization R either to 5.0 (mild regularization) or 0.25 (strong regularization). For various feature space dimensions m, we generate the projections $\boldsymbol{\Omega} \sim \Lambda^m$ (with $\Lambda = \mathcal{F}\kappa^{\Delta}$, see Sec. 3.2) and dithering $\boldsymbol{\xi} \sim \mathcal{U}^m([0, 2\pi))$, and thus train (with Scikit-learn [PVG+11]) both an SVM classifier from the raw data with the Gaussian kernel, and another linear SVM from the associated cosine random Fourier features. We then evaluate these classifiers (in "inference mode") on the separate test set, using the different random features inner products and report the median accuracy (out of 25 draws) as a function of m in Fig. 3.5, right. In the four plots of this figure, we use a specific color coding of the RPF for both the incoming query vector $\boldsymbol{x}'$ and the learned support vectors (SVs) $\{\boldsymbol{x}_i\}_{i \in \mathcal{S}^*}$, as summarized in Table 3.1 for convenience. Note that, according to this table, the green and the blue curves are associated with the scenarios represented in the Introduction in Fig. 3.1a and Fig. 3.1b, respectively, and the red curves relate to the (symmetric) quantized approach of [BJKS15].

When approximating the "exact" SVM classifier (where $\kappa_{\mathrm{L}}^{\sharp} = \kappa$; top row in Fig. 3.5), a substantial number of features is required to reach the same performance ($m \approx 300$ to obtain accuracy $> 97.5\%$). As could be intuitively expected, the drop of accuracy is larger when more quantization of the features is being performed (the price to pay is particularly high when the support vectors are quantized, in blue and red). This difference is probably related to the fact that the SVM decision function is relatively sensitive to the position of the support vectors (in the feature space), because they directly lie on the decision boundary. Notice also the importance of regu-

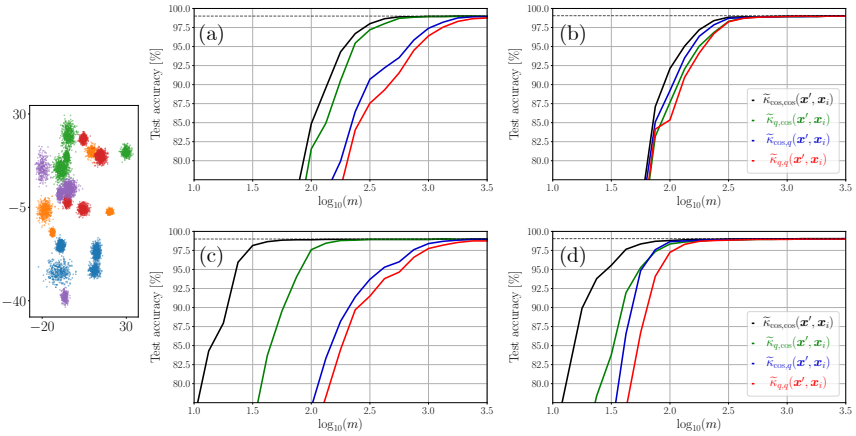


Fig. 3.5 Left: Considered dataset (each arbitrary color corresponds to one of the 5 classes). Right: Test accuracy of an SVM classifier learned with a Gaussian kernel on the raw training set $\mathcal{T}$ (top (a-b)) or on their cosine RFF (bottom (c-d)), for regularization parameters $R = 5$ (weak regularization; left (a-c)) and $R = 0.25$ (strong regularization; right (b-d)), and evaluated with several RPF combinations as a function of their dimension m. Classification performance is measured on the test set according to the scenarios and curve colors described in Table 3.1. The curves are the median out of 25 independent draws of $\boldsymbol{\Omega}$ and $\boldsymbol{\xi}$. The dashed line indicates the test accuracy of the exact SVM classifier (no random features are used).

larization: although changing R does not incur a noticeable change on the exact SVM accuracy (the horizontal dashed lines), it appears in Fig. 3.5b that stronger regularization improves the RPF-based classifiers.

When the SVM is directly trained on the random Fourier features (*i.e.*, $\kappa_{\mathrm{L}}^{\sharp} = \tilde{\kappa}_{\cos,\cos}$; bottom row in Fig. 3.5) the accuracy of $\tilde{\kappa}_{\cos,\cos}(\boldsymbol{x}', \boldsymbol{x}_i)$ (black) is strongly boosted (reaching 97.5% accuracy or higher with less than $m = 50$ features). The role of regularization is here exacerbated: reducing R hurts the performance of the plain RFF classifier, but is necessary to maintain good accuracy with the semi-quantized schemes for a reasonable feature dimension m. It appears that proper regularization is needed to account for the quantization noise.

Finally, note that in all cases, the fully quantized scheme (in red) is not much worse than our semi-quantized solution with dataset quantization (the kernel mismatch shown Fig. 3.3 is apparently not too harmful in this case), but still suffers from strictly more classification errors. Therefore, it

can be replaced by one of the asymmetric schemes for a negligible cost, it should be done. However, because we focus on approximating an imposed kernel without considerations for the underlying machine learning model, we did not compare to the fully-quantized case where the quantized features are already used during the training [BJKS15], *i.e.*, where the distorted kernel $\kappa_{q,q} = \mathbb{E}\widetilde{\kappa}_{q,q}$ is directly embraced to train the SVM rather than being used as an approximation for κ.

Real data: remote classification of hyperspectral pixels

In a last experiment, to prove the concept of using semi-quantized features in a concrete and practical setting, we consider the problem of hyperspectral pixel classification, *i.e.*, determine the class of a spatial pixel given its electromagnetic spectral response across d wavelengths. Kernel SVMs have been a quite popular solution to this challenge: our approach in particular is inspired by [GCCJ99, MB04] as well as [HSS15] for the use of RFF, but many more references can be found in the extensive review [MIO11].

To have a concrete and quantifiable measure of the computational gains allowed by the quantization of RFF, we focus on the quantized query context (illustrated Fig. 3.1a). More precisely, we consider the scenario of an aircraft (or satellite) equipped with a hyperspectral sensor that must send its readings $\boldsymbol{x}'$ for remote classification of the pixels it observes. We assume this task is entrusted to a kernel SVM, involving a weighted sum of $\kappa_{\mathrm{T}}^{\sharp}(\boldsymbol{x}', \boldsymbol{x}_i)$ terms. Since the communication link between the satellite and the remote server is presumably costly, it is important that the number of bits used to encode this query, noted b, is as small as possible.

We compare three strategies. First, the baseline strategy is to send the "raw" measurements $\boldsymbol{x}' \in \mathbb{R}^d$, which requires $b = Bd$ bits, where B designates the bit-depth of full-precision readings (we consider $B = 64$ bits in our experiments). In this strategy, the kernel in the learning and testing stages is the "true" Gaussian kernel, *i.e.*, $\kappa_{\mathrm{L}}^{\sharp} = \kappa_{\mathrm{T}}^{\sharp} = \kappa$. Second, the usual RFF strategy is to send the full-precision RFF $\boldsymbol{z}_{\cos}(\boldsymbol{x}') \in \mathbb{R}^m$, which requires $b = Bm$ bits. Following the observations from the previous experiments, we both learn and test on the RFF kernel, *i.e.*, $\kappa_{\mathrm{L}}^{\sharp} = \kappa_{\mathrm{T}}^{\sharp} = \widetilde{\kappa}_{\cos,\cos}$. Finally, the quantized RFF query strategy is to send the quantized RFF $\boldsymbol{z}_q(\boldsymbol{x}') \in \{-\frac{1}{\sqrt{m}}, +\frac{1}{\sqrt{m}}\}^m$, which takes up $b = m$ bits[5]. Although the kernel

[5]Strictly speaking, we would actually need to transmit $\frac{1}{2}(\sqrt{m}\boldsymbol{z}_q(\boldsymbol{x}') + 1) \in \{0,1\}^m$ and to remotely recover $\boldsymbol{z}_q(\boldsymbol{x}')$ from this binary stream, assuming m is known to the receiver.

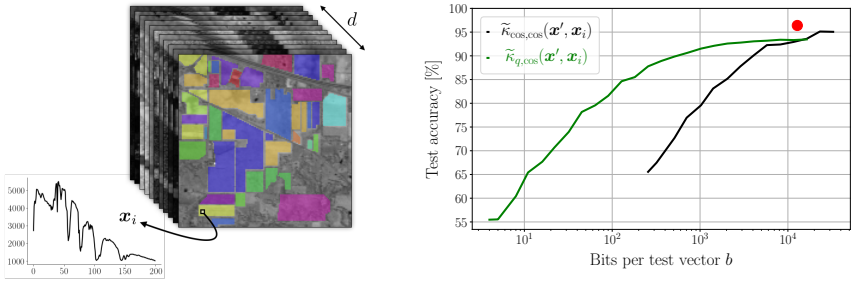
at test time is now the hybrid product $\kappa_{\mathrm{T}}^{\sharp} = \widetilde{\kappa}_{q,\cos}$, we still learn using the usual RFF kernel $\kappa_{\mathrm{L}}^{\sharp} = \widetilde{\kappa}_{\cos,\cos}$. Note that to simplify the comparison, we thus consider only a naive encoding of the query, neglecting the use of *e.g.*, entropy coding strategies.

We use the standard Indian Pines dataset [BBL15], a hyperspectral volume which contains 10 249 labeled pixels[6], measured across $d = 200$ wavelengths[7], separated into $N = 16$ classes (see Fig. 3.6a). We first separated 20% of those pixels into testing set, which left $n = 8204$ pixels for training. In order to select the hyper-parameters (kernel bandwidth σ and regularization strength R), following their sensitivity observed in the previous experiment, we performed a separate cross-validation (with 5 folds from the training set) for each of the three individual strategies. We then evaluated the test set accuracy reached by each strategy, while letting m vary for the two RFF-based strategies, and report the results Fig. 3.6b. In this figure, the baseline strategy is represented by the red dot, and the usual (resp. quantized) RFF strategies are represented by the black (resp. green) solid curves, which are obtained by varying m.

The baseline (red) achieves the best accuracy overall, but at the price of a quite substantial bandwidth usage. When using full-precision random features (black), the accuracy is only slightly reduced, but only if a relatively large number of random features m is used, which does bring substantial bitrate reduction. Indeed, to reduce the bandwidth b by, say, an order of magnitude, the full-precision RFF strategy must sacrifice more than 10% accuracy, which is probably not acceptable in practice. On the other hand, with the quantized query RFF strategy, we are able to achieve this same bitrate reduction by an order of magnitude at the cost of only about 4%, which sounds more reasonable. Overall, (keeping in mind that more involved compression methods could be applied to transmit the raw measurement $\boldsymbol{x}'$ in our scenario above, and hence still apply the first classification strategy after decompression) the quantized RFF strategy performs better whenever the bitrate b is significantly smaller than the baseline bitrate, hence showing the potential of the approach.

[6]The size of the full volume is 145×145 which gives 21025 pixels in total, but many of them are unlabeled, which we discard for this experiment.

[7]The initial volume contains 220 wavelengths, but following the workflow commonly adopted with this dataset, we removed the water absorption bands (*i.e.*, the spectral indices [104-108], [150-163], and 220 from the initial dataset).



(a) Indian Pines dataset. **(b)** Test accuracy *w.r.t.* bit-rate b.

Fig. 3.6 (a) Schematic representation of the Indian Pines hyperspectral volume, containing 10 249 labeled pixels $x_i \in \mathbb{R}^d$, which feature $d = 200$ wavelengths. (b) Test accuracy as a function of the number of bits b to be transmitted to the remote server, for each of the three considered strategies: the baseline strategy (red dot) sends the raw test vector x'; the RFF strategy (solid black) sends the full-precision RFF $z_{\cos}(x')$ of varying size m; and the quantized RFF strategy (solid green) sends the one-bit equivalent $z_q(x')$. The curves are the median out of 30 independent draws of Ω and ξ.

3.7 Asymmetric RPF beyond one-bit quantization*

So far, we applied the asymmetric RPF strategy on only one pair of periodic maps, namely $f = q$ and $g = \cos$ (see Section 3.5). In this section, to validate the generality of our results, we further investigate other maps, generalizing the (q,cos) scenario with respect to two aspects. First, we consider a different discontinuous periodicity, namely the *modulo map* $f = \mathrm{mod}$, for which we show the mean Lipschitz smoothness. This defines "modulo random Fourier features". Moreover, we investigate the *complex extension* of those two maps ($f = q$ and $f = \mathrm{mod}$), and demonstrate empirically on a short experiment how the performance of those different schemes compare.

3.7.1 Modulo random Fourier features

The modulo periodic map, denoted by $\mathrm{mod}(\cdot)$, is a linearly increasing function with "wraps around" when some threshold is reached. More precisely, we consider the *normalized modulo* operation $\mathrm{mod} \in \mathrm{PF}$, defined by

$$\mathrm{mod}(t) := \mathrm{mod}_{2\pi}(t) \quad \text{with} \quad \mathrm{mod}_T(t) := 2\left(\tfrac{t}{T} - \left\lfloor \tfrac{t}{T} \right\rfloor\right) - 1, \tag{3.31}$$

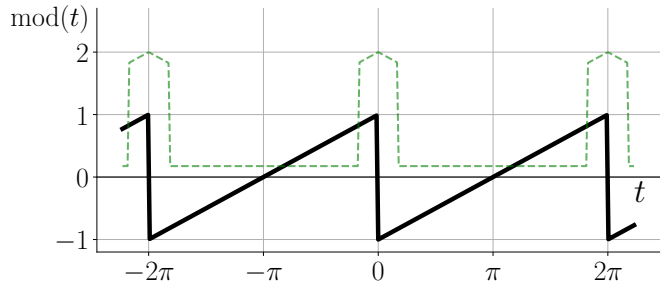


Fig. 3.7 The normalized periodic map $\mathrm{mod}(t)$ defined in (3.31) (black, plain), and the associated integrand $I_\delta(t)$ (where we fixed $\delta = 0.5$), as developed in the proof of Prop. 3.22 (green, dashed).

the "normalized" modulo T operation. This map, also known as the "sawtooth wave", is illustrated Fig. 3.7.

For now, the main purpose of this map is to check that our generic results apply to other (discontinuous) maps besides the one-bit universal quantization q. However, as will be further explored in the next chapter, there is a more pragmatic interest: motivated by the recent theory of *modulo sampling* [BKR17] and its extension to compressive modulo sampling [SH19], we could imagine efficient hardware implementation of the *modulo random Fourier features map*

$$\boldsymbol{z}_{\mathrm{mod}}(\boldsymbol{x}) := \mathrm{mod}(\boldsymbol{\Omega}^\top \boldsymbol{x} + \boldsymbol{\xi} - \tfrac{\pi}{2}), \tag{3.32}$$

where the purpose of the $\frac{\pi}{2}$ phase shift is to "synchronize" $f = \mathrm{mod}$ with the map $g = \cos$ (otherwise, we would have some form of "destructive interference")[8]. As explained in [BKR17,SH19], the hardware implementation of (3.32) is based on self-reset ADCs [RJ03].

To apply our theoretical results to the modulo map, we need to prove its mean Lipschitz smoothness.

Proposition 3.22. *The modulo function* mod*, defined in* (3.31)*, has the mean Lipschitz smoothness property (Def. 3.13) with constant*

$$L^\mu_{\mathrm{mod}} = \tfrac{3}{\pi}. \tag{3.33}$$

[8] Alternatively, we could have set $g = -\sin$.

Proof. By definition, we seek a constant L^{μ}_{mod} such that for all $\delta \in (0, \pi]$,

$$\frac{1}{2\pi}\int_0^{2\pi} \max_{r\in[-\delta,\delta]}\{|\mathrm{mod}(t+r) - \mathrm{mod}(t)|\}\,\mathrm{d}t \leq L^{\mu}_{q}\cdot\delta.$$

We first characterize $I_\delta(t) := \max_{|r|\leq\delta}\{|\mathrm{mod}(t+r) - \mathrm{mod}(t)|\}$. Since $\mathrm{mod}(t)$ is linearly increasing (with slope $\frac{1}{\pi}$) everywhere except on discontinuities at $t \in 2\pi\mathbb{Z}$, we can show that (see Fig. 3.7)

$$I_\delta(t) = \begin{cases} 2 - \frac{|t-2\pi k|}{\pi} & \text{when } |t-2\pi k| \leq \delta \text{ for some } k \in \mathbb{Z}, \\ \frac{\delta}{\pi} & \text{elsewhere.} \end{cases}$$

Integrating this over one period (summing the area) gives

$$\int_0^{2\pi} I_\delta(t)\mathrm{d}t = (2\pi - 2\delta)\cdot\tfrac{\delta}{\pi} + 2\delta\cdot\tfrac{1}{2}(2 + (2 - \tfrac{\delta}{2\pi})),$$

which gives the bound $L^{\mu}_{\mathrm{mod}} \leq \frac{2+4}{2\pi} = \frac{3}{\pi}$. Moreover, taking $\lim_{\delta\to 0}$, we can show that his bound is tight (*i.e.*, $L^{\mu}_{\mathrm{mod}} = \frac{3}{\pi}$ and there is no smaller constant such that mean smoothness is verified). □

We could thus say that the sawtooth wave is somehow "smoother" (according to mean Lipschitz smoothness) than the square wave, which is maybe what one would intuitively expect since the sawtooth wave exhibits less discontinuities (which should heavily penalize the "smoothness").

Recalling from Prop. 3.15 that a lower mean Lipschitz smoothness constant allows to decrease the feature dimension m (to achieve a given error), one might wonder if this argument makes the modulo feature map $\mathrm{mod}(t)$ "more accurate" than the universal quantization $q(t)$ when approximating a kernel through the asymmetric RPF scheme.

However, the mean smoothness constant is only half of the story. As explained in Section 3.5, to approximate a given kernel $\kappa(\boldsymbol{x}, \boldsymbol{y})$, the asymmetric RPF product $\langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_{\cos}(\boldsymbol{y})\rangle$ must further be re-scaled by $\Re\{F_1\}$ the first FS coefficient of f, the resulting kernel approximation being

$$\widetilde{\kappa}_{f,\cos}(\boldsymbol{x}, \boldsymbol{y}) := \tfrac{1}{\Re F_1}\langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_{\cos}(\boldsymbol{y})\rangle.$$

The "quality" of a periodic function f in this scheme is thus determined by two quantities: its mean smoothness (a lower smoothness constant is better) *and* its first Fourier series coefficient (here, a larger real part of the first FS coefficient is better). More precisely, for a fixed target error ϵ and a probability of failure δ, our results imply that, in order to achieve $|\widetilde{\kappa}_{f,\cos}(\boldsymbol{x}, \boldsymbol{y}) -$

$\kappa(\boldsymbol{x},\boldsymbol{y})\big| \leq \epsilon$ uniformly over Σ with probability at least $1-\delta$, one must have a feature dimension of at least

$$m \geq 64 \cdot \tfrac{1}{|\Re F_1|^2\epsilon^2} \cdot \max\left(2\mathcal{H}_{\epsilon/c'}(\Sigma), \log\left(\tfrac{3}{\delta}\right)\right),$$

with, a (conservative) constant[9] of $c' = 4C_\Lambda|\Re F_1|^{-1} \cdot (\frac{6}{\pi} + L_f^\mu)$.

Coming back to the comparison of $f = q$ with $f = \mathrm{mod}$ for example, in the first case we have $c'_q = 10C_\Lambda$, while in the latter[10] we have $c'_{\mathrm{mod}} = 18C_\Lambda$; in other words, the radius in the covering number is smaller for the modulo features. This estimate implies that (according to our theory) the modulo feature map should require more random features to reach a given approximation error, although it is a "smoother" periodic map; this is indeed what we will observe later in this section (Fig. 3.8).

3.7.2 Complex extensions

As mentioned at the beginning of Section 3.5, it is also possible to recover the "original" kernel κ when $g(t) = e^{\mathrm{i}t}$. In this setting, to be closer in spirit to the "complex" version of the random Fourier features, we can moreover take complex-valued RPF, by *complex extension* of the real-valued ones. Intuitively, the idea of the complex extension is to generalize the relationship between $\cos(t)$ and $\exp(\mathrm{i}t)$. To be specific, if $f : \mathbb{R} \to \mathbb{R}$ is a real-valued periodic map, we define its complex extension $f_{\mathbb{C}} : \mathbb{R} \mapsto \mathbb{C}$ as

$$f_{\mathbb{C}}(t) := f(t) + \mathrm{i}f(t - \tfrac{\pi}{2}). \tag{3.34}$$

In this case, the asymmetric RPF estimation strategy is, using that the first FS of $f_{\mathbb{C}}$ is $F_{\mathbb{C},1} = 2F_1$ (see Lemma A.2),

$$\widetilde{\kappa}_{f_{\mathbb{C}},\exp(\mathrm{i}\cdot)}(\boldsymbol{x},\boldsymbol{y}) := \tfrac{1}{2F_1}\langle \boldsymbol{z}_{f_{\mathbb{C}}}(\boldsymbol{x}), \boldsymbol{z}_{\exp(\mathrm{i}\cdot)}(\boldsymbol{y})\rangle \quad \simeq \quad \kappa(\boldsymbol{x},\boldsymbol{y}).$$

Intuitively, the advantage of this complex extension is that, for a given dimension m (thus, for a given amount of random projections in $\boldsymbol{\Omega}$), we obtain more "information" (in the form of real and imaginary components). This potentially allows to decrease the kernel estimation error without increasing the amount of random projections to perform.

To formalize this idea, we would like to obtain bounds on approxima-

[9] Compared to the bound $L^\mu_{\cos} \leq 1$ used above, here we rely on the tighter value $L^\mu_{\cos} = \frac{2}{\pi}$.

[10] Recall that we phase-shift $f = \mathrm{mod}$ by $\frac{\pi}{2}$ to synchronize with $g = \cos$. We use that in this case, its first FS coefficient is $|\Re\{F_1\}| = \frac{1}{\pi}$; see Appendix A.

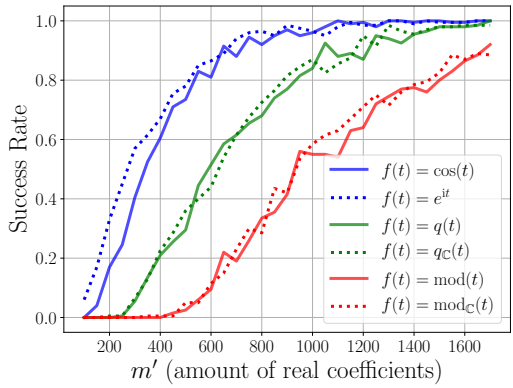


Fig. 3.8 Success rate (average number of times $\epsilon(\Sigma) < \bar{\epsilon}$, out of 100 independent draws of $\boldsymbol{\Omega}$ and $\boldsymbol{\xi}$), as a function of the amount of real coefficients m', for several random periodic features combinations: either usual symmetric RFF (blue), semi-quantized (green) or semi-modulo (red). Each combination uses real-valued features (plain) or complex-valued features (dashed).

tion error through Prop. 3.15. For this, we could for example use $L^{\mu}_{f_{\mathbb{C}}} \leq 2L^{\mu}_{f}$. However, since the involved quantities are not necessarily tight in general (the bounds are not necessarily achieved), in this work we content ourselves with a simple numerical simulation instead.

3.7.3 A small experiment

To compare the different asymmetric RPF schemes, we perform a small version of the experiment from Subsection 3.6.2: for varying embedding dimensions m, we compute the kernel approximation error on all pairs in a finite set Σ (generated by picking $n = 100$ samples from two Gaussian modes in dimension $d = 32$) through the asymmetric RPF schemes described above[11] and report whether or not all the estimations had error smaller than some fixed threshold $\bar{\epsilon} = 0.1$. This is repeated for several independent trials, and we compute the average number of "successes" (*i.e.*, cases where the kernel estimation error $\epsilon_{f,g}(\Sigma) := \sup_{\boldsymbol{x},\boldsymbol{y}\in\Sigma} |\widetilde{\kappa}_{f,g}(\boldsymbol{x},\boldsymbol{y}) - \kappa(\boldsymbol{x},\boldsymbol{y})|$ is smaller than $\bar{\epsilon}$).

[11] That is, for the pairs (f,g) given by $(\cos,\cos)$, $(\exp(it),\exp(it))$, $(q,\cos)$, $(q_{\mathbb{C}},\exp(it))$, $(\mathrm{mod},\cos)$ and $(\mathrm{mod}_{\mathbb{C}},\exp(it))$, with phase synchronization in the case of $(\mathrm{mod},\cos)$. For the pair $(\exp(it),\exp(it))$ we do not use dithering since it is not required.

To facilitate the comparison between the real and complex-valued maps, we report the success rate as a function of the amount of "real coefficients" to be stored (*i.e.*, $m' = m$ for $f(t) \in \mathbb{R}$ and $m' = 2m$ for $f_{\mathbb{C}}(t) \in \mathbb{C}$). The results are shown Fig. 3.8.

First, we can observe that the real-valued and complex-valued features perform similarly at equal number of real coefficients m'. That is, by using the complex extension, we can achieve the same kernel approximation error while reducing by two the amount of random projections m to perform (in Chapter 4, we'll thus focus on the complex-valued RPF). Second, as anticipated above from our theory, the modulo RFF show a worse performance in approximating the kernel.

3.8 Conclusion

This chapter introduced the framework of *asymmetric random periodic features*, where random projections are passed through two different periodic maps, and whose inner products are used to approximate a kernel. We provided an expression of this kernel, with a uniform error bound holding on infinite compact signal sets, provided the periodic maps satisfy a property we called the *mean smoothness*. The mean smoothness holds for some discontinuous maps such as the one-bit universal quantization (a square wave). We studied (theoretically and empirically) semi-quantized kernel approximations, and showed how the impact of quantization can be controlled. As a side note, with those developments we also generalized the local geometry-preserving embeddings from [BRM17], and corrected an error in their main result in the process (this is the content of Appendix C).

As highlighted by the applied experiments in Sec. 3.6, these theoretical guarantees do not necessarily ensure an accurate control over the generalization performance in a machine learning context. Indeed, it seems crucial to incorporate the random periodic features directly into the training stage, and to anticipate that some features might be quantized later (for example, when picking the regularization strength).

In the two applicative scenarii we proposed in the Introduction (Fig. 3.1), the ultimate objective is to improve the bitrate-kernel accuracy trade-off. Allowing a finer (but still coarse) quantization of the RFF (*i.e.*, coding each entry on $b > 1$ bits instead of 1 as we proposed, for example with the b-bit universal quantization [BRM17]) is a promising idea to reach a better trade-off. Since the mean smoothness of these finer quantization functions can also be verified, our results would carry over easily to this multi-bit

quantization of the RFF.

Further perspectives are discussed at the end of this thesis, in Chapter 7. In the next chapter, we rely on the main results from this one to obtain formal guarantees in the context of a compressive learning which relies on asymmetric periodic functions, *e.g.*, the (complex extended) universal quantizer and the complex exponential.

4

Quantized Sketching with Guarantees

In this chapter we study a generalization of the compressive learning framework, called *asymmetric compressive learning*, which allows us (among others) to incorporate *harsh quantization of the sketch contributions*, (almost) without sacrificing learning performance.

Usual "symmetric" compressive learning (CL) breaks training down into two steps: a *sketching phase*, that summarizes the dataset to a lightweight sketch vector, constructed by passing the data through a well-chosen *feature map* Φ, and averaging those contributions; and a *learning phase*, which extract the desired model parameters by solving an optimization problem, that also involves the feature map Φ (see Section 2.5). We call this setting "symmetric" because both phases use the *same* feature map Φ.

However, the sketching and learning phase intuitively "desire" different properties from this map. For example, the sketching phase might benefit from quantization of the features (which we succinctly call *quantized sketching*), while the gradient-based algorithms of the learning phase require differentiability of this map. Motivated by this observation, the general *asymmetric compressive learning framework* proposed in this chapter is a relaxation where the feature map is allowed to be *different for each phase* (*i.e.*, Ψ during sketching and Φ during learning, with $\Psi \neq \Phi$).

The main achievement of this chapter is to extend the theoretical *statistical learning guarantees* of (symmetric) CL from [GBKT17] to this new asymmetric setting. Extensive numerical experiments are provided as well to validate this approach. As a particular case, we obtain learning guarantees for the quantized sketch contributions by building on the asymmetric random periodic features guarantees from Chapter 3. In fact, the asymmetric CL scheme from this chapter can be seen as the "averaged equivalent" of the asymmetric RPF scheme from the that chapter, as we use the same basic principles to embed probability distributions instead of individual data vectors.

This chapter is almost entirely based on a couple of publications. We proposed a first short description of the asymmetric CL scheme in "Quantized Compressive K-Means" [SJ18b] (Signal Processing Letters, 2018), in which we focused on the k-means problem, and mainly backed our approach by numerical experiments. The in-depth formal theoretical guarantees (as well as the GMM experiments) are from "Asymmetric compressive learning guarantees with applications to quantized sketches" [SJ21], which is currently under review in IEEE Transaction on Signal Processing. A few extra experiments were added, indicated by a star *.

4.1 Introduction

4.1.1 Motivation

Context: symmetric compressive learning In order to highlight the specificity of *asymmetric* compressive learning, let us here briefly recall the basic principles of (symmetric) compressive learning. CL was proposed to learn from datasets $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^n$ of massive scale (in particular, with a large number of examples n, typically at least several millions) while keeping computational resources (*i.e.*, memory, training time) under control [GBKT17, GCK+20]. To do so, CL first compresses the data as a single m-dimensional (possibly complex-valued) vector $\boldsymbol{z}_{\Phi,\mathcal{X}} = \frac{1}{n}\sum_{i=1}^n \Phi(\boldsymbol{x}_i)$, called *sketch*, by simple averaging of a *feature map* $\Phi : \mathbb{R}^d \to \mathbb{C}^m$.

Importantly, the choice of the feature map Φ determines the range of machine learning tasks that one is able to solve from the sketch $\boldsymbol{z}_{\Phi,\mathcal{X}}$[1]. For example, it is possible to solve k-means [KTTG17] or Gaussian mixture modeling [KBGP18] within this framework, when Φ is chosen to be ran-

[1]Note that Φ should be nonlinear, otherwise the sketch cannot capture anything beside the data mean.

dom Fourier features (RFF) [RR08] (see Subsection 2.1.4). As a reminder, those features are defined as the complex exponential of random projections of the data, *i.e.*,

$$\Phi_{\text{RFF}}(\boldsymbol{x}) := \tfrac{1}{\sqrt{m}} \exp(\mathrm{i}(\boldsymbol{\Omega}^\top \boldsymbol{x} + \boldsymbol{\xi})), \tag{4.1}$$

where the exponential is applied component-wise, and $\boldsymbol{\Omega} \in \mathbb{R}^{d\times m}$ has randomly drawn columns $\boldsymbol{\omega}_j \sim_{\text{i.i.d.}} \Lambda$ for some probability distribution Λ.The random *dither* $\boldsymbol{\xi}$, which has uniform entries $\xi_j \sim_{\text{i.i.d.}} \mathcal{U}([0, 2\pi))$, will be useful when we consider quantization (as could be guessed from our developments in Chapter 3).

After this "sketching phase", the "learning phase" of CL extracts the desired machine learning model parameters $\boldsymbol{\theta} \in \Theta$ from the sketch. This is achieved by solving an optimization problem of the form

$$\min_{\boldsymbol{\theta}\in\Theta} \mathcal{C}_\Phi(\boldsymbol{\theta}; \boldsymbol{z}_{\Phi,\mathcal{X}}).$$

Since the cost $\mathcal{C}_\Phi$ involves only the sketch of size $m \ll nd$ (*i.e.*, much smaller than the volume of $\mathcal{X}$), this procedure is typically much more efficient from a computational point of view than the classical approach of learning directly from the entire dataset $\mathcal{X}$, especially for large n.

Intuitively, given a map Φ (*e.g.*, Φ_{RFF}), the cost $\mathcal{C}_\Phi(\boldsymbol{\theta}; \boldsymbol{z})$ captures the mismatch between the vector $\boldsymbol{z}$ and another sketch, obtained using Φ, associated with the candidate model $\boldsymbol{\theta}$. To solve k-means for instance (where one seeks a set of K centroids $\boldsymbol{\theta} = \{\boldsymbol{c}_k\}_{k=1}^K \subset \mathbb{R}^d$ that best cluster the data), this cost is given by $\mathcal{C}_\Phi(\boldsymbol{\theta}; \boldsymbol{z}) = \|\boldsymbol{z} - \frac{1}{K}\sum_k \Phi(\boldsymbol{c}_k)\|_2$, *i.e.*, the Euclidean distance between the sketch of the data and the "sketch of the centroids" [KTTG17].

Numerical experiments demonstrated the power of CL, sometimes solving k-means clustering [KTTG17] and Gaussian mixture modeling [KBGP18] with training time and memory consumption reduced by several orders of magnitude compared to classical approaches. Moreover, formal *statistical learning guarantees* (bounds on the so-called excess risk) were derived [GBKT17]: the risk is controlled whenever a form of Lower Restricted Isometry Property (LRIP) holds, which translates the compatibility between the sketch map Φ and the target learning task. Proving this LRIP is however quite technical; *e.g.*, see [GBKT20] for the LRIP between Φ_{RFF} and k-means and GMM. This makes the design of a compressive learning schemes a bit rigid, as any deviation from the beaten path breaks the strong, but tediously constructed, theoretical guarantees. The ambition of this chap-

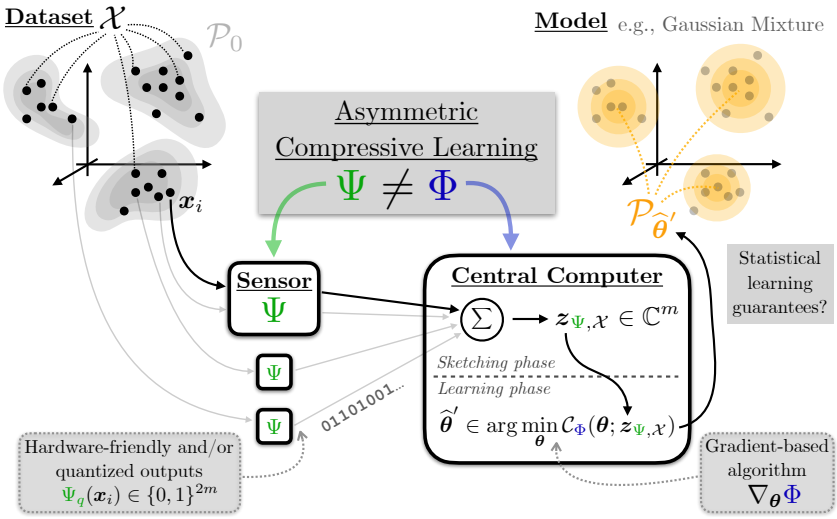


Fig. 4.1 Our Asymmetric Compressive Learning (ACL) scheme: a dataset $\mathcal{X}$ of n examples x_i (sampled *i.i.d.* from $\mathcal{P}_0$) is first compressed as a lightweight vector—the *sketch*—by averaging data features $\Psi(x_i)$. This operation can be performed in parallel by a sensor network, which benefits greatly from hardware-friendliness and quantization. A model $\widehat{\theta}'$ is then learned from the sketch $z_{\Psi,\mathcal{X}}$ by solving a CL optimization procedure that uses a different, differentiable map $\Phi \neq \Psi$. Our goal is to prove statistical learning guarantees (*w.r.t.* $\mathcal{P}_0$) for the model $\widehat{\theta}'$.

ter is to somehow relax this constraint on a specific aspect: allowing the designer to tweak the sketching phase (*e.g.*, to further improve its computational efficiency) without having to re-prove the LRIP-based guarantees from scratch.

The general asymmetric CL scheme More precisely, we study the scenario where the feature map for the sketching phase (now noted Ψ) is allowed to differ from the one used for the learning phase (noted Φ), *i.e.*, we consider the *asymmetric* CL (ACL) strategy modeled by $\min_\theta \mathcal{C}_\Phi(\theta; z_{\Psi,\mathcal{X}})$, with $\Psi \neq \Phi$. This scheme is interesting from a practical point of view because both phases, occurring in different contexts, may require very different properties from their respective feature maps.

This is best explained by a concrete example, illustrated Fig. 4.1: consider a sensor network, where each node collects a few data samples x_i, and

sends their contributions $\Psi(\boldsymbol{x}_i)$ to a centralized server which aggregates them to construct the sketch $\boldsymbol{z}_{\Psi,\mathcal{X}} = \frac{1}{n}\sum_{i=1}^{n}\Psi(\boldsymbol{x}_i)$. For efficient transmission of those numerous messages, *quantization* of the contributions $\Psi(\boldsymbol{x}_i)$ is critical. Moreover, to ensure low power consumption of the sensor nodes, a compact *hardware implementation* of Ψ is highly desirable.

The learning phase however, being performed locally (*i.e.*, on a single workstation) and in software, does not benefit as much from these aspects. Instead, the cost $\mathcal{C}_\Phi(\boldsymbol{\theta}; \boldsymbol{z}_{\Psi,\mathcal{X}})$ must have *meaningful minimizers* (*i.e.*, as ensured by statistical guarantees), and must be efficient to optimize; *e.g.*, gradient-based algorithms [KBGP18, KTTG17] require *differentiability* of Φ. All these objectives do not necessarily align, and some are even incompatible (for example, differentiability cannot be reconciled with the discontinuity induced by quantization), hence the interest of allowing different sketching and learning maps $\Psi \neq \Phi$.

Quantized sketching as our key motivating use-case We are particularly interested in the *quantized sketching* scenario, *i.e.*, where Ψ produces binary contributions. Consider for example as reference map the random Fourier features $\Phi = \Phi_{\text{RFF}}$ (4.1). Computing Φ_{RFF} for each sample $\boldsymbol{x}_i$ is typically done as follows: one must *(i)* record (in full-precision) the random projections $\boldsymbol{\Omega}^\top \boldsymbol{x}_i \in \mathbb{R}^m$, *(ii)* evaluate the complex exponentiation operation $\exp(\mathrm{i}\cdot)$, which is typically expensively done in software (*e.g.*, through Taylor series approximations[2]) and *(iii)* store (at least temporarily), in full-precision, the resulting contributions $\Phi_{\text{RFF}}(\boldsymbol{x}_i) = \frac{1}{\sqrt{m}}\exp(\mathrm{i}\boldsymbol{\Omega}^\top \boldsymbol{x}_i) \in \mathbb{C}^m$.

A resource-preserving (*e.g.*, computational or energy efficient) CL sensor should directly and solely acquire $\Phi_{\text{RFF}}(\boldsymbol{x}_i)$. While the random projections *(i)* can be computed relatively cheaply (*e.g.*, using compressive-sensing-based techniques [CRT06], such as fast structured random projections [CGK18] or, possibly relying on optical random processes [SCC+16]), the evaluation of the complex exponential *(ii)* is complicated to implement in hardware—although this is the costliest step in fast (software) computations of the sketch [CGK18]. To add insult to injury, the resulting full-precision values contributions in *(iii)* seem quite wasteful given their incidental contribution to the overall sketch, an average of $n \gg 1$ such contributions. This problem is even more striking when we assume the sensors implementing the sketch feature map must then send their high-bitrate contributions $\Phi_{\text{RFF}}(\boldsymbol{x}_i)$ over a communication network (see Fig. 4.1).

[2] A good engineer would tell you to set $\sin(\theta) \simeq \theta$, but that does not quite do the trick here.

We propose a quantized sketch procedure that circumvents these limitations, conceptually much simpler to integrate directly in hardware (*e.g.*, using voltage controlled oscillators [YKJC08])[3]. To achieve this, we replace the costly $\exp(\mathrm{i}\cdot)$ signature function by *1-bit universal quantization* [Bou12] $q(\cdot) = \mathrm{sign}\,(\cos(\cdot)) = 2(\lfloor \frac{\cdot}{2\pi} \rfloor \mod 2) - 1$; this corresponds to taking the least significant bit (LSB) of a uniform quantizer with quantizer stepsize π. Actually, motivated by the observations from Section 3.7 we rather consider the *complex extension* $q_{\mathbb{C}}(\cdot) = q(\cdot) + \mathrm{i}q(\cdot - \frac{\pi}{2})$ of this function instead (depicted Fig. 4.3). Since in this chapter we always consider $q_{\mathbb{C}}$, we redefine our notations and simply denote the complex-valued universal quantization function by q, dropping the $_{\mathbb{C}}$ subscript for convenience.

This thus amounts to replacing the RFF sketch contributions $\Phi_{\mathrm{RFF}}(\boldsymbol{x}_i) \in \mathbb{C}^m$ by quantized ones $\Psi_q(\boldsymbol{x}_i) \in \frac{1}{\sqrt{m}}\{\pm 1 \pm \mathrm{i}\}^m$, obtained by taking the sign of usual RFF:

$$\Psi_q(\boldsymbol{x}) := \tfrac{1}{\sqrt{m}}\,\mathrm{sign}\,\Big(\exp(\mathrm{i}(\boldsymbol{\Omega}^\top \boldsymbol{x} + \boldsymbol{\xi}))\Big) = \tfrac{1}{\sqrt{m}}\, q(\boldsymbol{\Omega}^\top \boldsymbol{x} + \boldsymbol{\xi}), \tag{4.2}$$

where the sign of a complex number $z \in \mathbb{C}$ is applied component-wise to its real and imaginary part, *i.e.*, $\mathrm{sign}\,(z) = \mathrm{sign}\,(\Re(z)) + \mathrm{i}\,\mathrm{sign}\,(\Im(z)) \in \{\pm 1 \pm \mathrm{i}\}$. The embedding Ψ_q is known as (the complex extension of) the one-bit universal embedding studied, among others, in [Bou12, BR13, BRM17] (see Subsec. 2.2.3).

Remark: As will become clearer below, our results allow to replace, in the sensor implementing Ψ, the complex exponential $\exp(\mathrm{i}t)$ by a *completely generic* periodic map $f(t)$ (of which $q(t) = \mathrm{sign}\,(\exp(\mathrm{i}t))$ defined above is a particular case). Remark that our results thus apply in particular to the case where, while the sensor can ensure the periodicity of the embedded map f, the precise shape of f cannot be fully controlled, such that we can't really rely the sensor to accurately implement any *specific* map—*e.g.*, due to imperfections and non-linearities.

4.1.2 Chapter contributions

To set the stage, we remind the essential statistical and compressive learning elements we'll need in Section 4.2 (which can be freely skipped if those concepts are fresh in the reader's memory). We detail the Asymmetric

[3] Even when the sketch is performed in software, the quantized sketch contributions are still potentially quite cheaper to compute, albeit with a more modest gain compared to the potential of hardware implementations (*e.g.*, speed-up factor of about 2 on my machine, with a naive Python implementation).

Compressive Learning (ACL) strategy in Section 4.3, for which we provide an intuitive justification.

In Section 4.4 we build solid theoretical guarantees for the ACL scheme. We first prove a "general" statistical guarantee for ACL in Subsection 4.4.1. To do so, we introduce a Limited Projected Distortion (LPD) property (capturing, roughly speaking, the similarity between Φ and Ψ), which we combine with the existing LRIP (which holds for Φ). This result is "general" in the sense that it makes no assumption on the task to solve or the maps Φ and Ψ (beyond the LRIP and LPD).

We then work towards concrete applications of this generic result in Subsection 4.4.2, dedicated to proving that the LPD actually holds for specific choices of Φ and Ψ. To achieve this, we first introduce—at the cost of an additional (but mild) assumption—a sufficient condition for the LPD, called "signal-level LPD" (sLPD). Next, we prove the sLPD on one particular combination of feature maps, where the "learning phase feature map" are random Fourier features $\Phi = \Phi_{\text{RFF}}$, and the "sketching phase feature map " are *random periodic features* $\Psi = \Psi_f$ studied in Chapter 3. This allows to obtain at last, as a corollary of all the results above, formal statistical learning guarantees for the ACL scheme for the pair $(\Phi_{\text{RFF}}, \Psi_q)$ considered in the motivation above, which ensures the theoretical soundness of that approach. To demonstrate the broader applicability of our results, we also apply them to the pair $(\Phi_{\text{RFF}}, \Psi_{\text{mod}})$, where Ψ_{mod} represents the—hardware-friendly—modulo measurement map studied (among others) in [SH19].

Moving on, we provide in Section 4.5 extensive empirical validations for the ACL strategy, confirming that, as our theory supports, it applies various tasks (both k-means and GMM) and feature maps (such as Ψ_{mod} the modulo measurements map). We highlight the practical advantage of our quantized compressive learning strategy on a large-scale audio classification task. Finally, we conclude in Section 4.6.

4.2 Preliminaries

We here recall here the elements and notations from statistical and compressive learning that are crucial to follow this chapter. To the attentive reader, this section should not bring anything new to the table and can be safely skipped, but since many different ingredients will be handled here, we prefer to make sure that they are all fresh in memory. For the sake of clarity, the main concepts and distributions introduced below and by our further developments are summarized in Table 4.1.

Concepts	Description
$\mathcal{M} = \mathcal{M}_+^1(\mathbb{R}^d)$	Probability measure set.
$\mathcal{P}, \mathcal{Q} \in \mathcal{M}$	Arbitrary distributions in $\mathcal{M}$.
$\mathcal{P}_0 \in \mathcal{M}$	Data distribution.
$\mathcal{P}_{\boldsymbol{\theta}} \in \mathcal{M},\ \boldsymbol{\theta} \in \Theta$	Parametric distribution (see Table 4.2).
$\mathcal{G} := \{\mathcal{P}_{\boldsymbol{\theta}} \,\vert\, \boldsymbol{\theta} \in \Theta\} \subset \mathcal{M}$	Model set.
$\widehat{\mathcal{G}} \subset \mathcal{M}$	Empirical set (see Subsec. 4.4.1).
$\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$	Dataset with $\boldsymbol{x}_i \sim_{\text{i.i.d.}} \mathcal{P}_0$.
$\widehat{\mathcal{P}}_{\mathcal{X}} := \frac{1}{n}\sum_{i=1}^n \delta_{\boldsymbol{x}_i} \in \widehat{\mathcal{G}}$	Empirical distribution of $\mathcal{X}$.

Table 4.1 Main concepts and distributions used in this chapter.

4.2.1 Chapter-specific notations and definitions

We consider learning examples living in the ambient space $\mathbb{R}^d$. The set of probability measures over $\mathbb{R}^d$, *i.e.*, $\mathcal{M}_+^1(\mathbb{R}^d)$ (see Section 2.4), is here simply noted $\mathcal{M}$, and $\delta_{\boldsymbol{c}} \in \mathcal{M}$ is the Dirac delta measure located at $\boldsymbol{c} \in \mathbb{R}^d$. To characterize the intrinsic dimension of a compact set $\Sigma \subset \mathbb{R}^d$, we use the Kolmogrov ν-entropy [KT61]: for any radius $\nu > 0$ it is given by $\mathcal{H}_\nu(\Sigma) := \log \mathcal{C}_\nu(\Sigma) < \infty$, where $\mathcal{C}_\nu(\Sigma)$ is the covering number of Σ by Euclidean balls of radius ν, *i.e.*,

$$\mathcal{C}_\nu(\Sigma) := \min\{|\mathcal{S}| : \mathcal{S} \subset \Sigma \subset \mathcal{S} + \nu\mathbb{B}_2^d\}, \tag{4.3}$$

where the cardinality of $\mathcal{S}$ is written $|\mathcal{S}|$, the d-dimensional unit ball *w.r.t.* the ℓ_p-norm is $\mathbb{B}_p^d$, and the Minkowski sum of two sets $\mathcal{A}$ and $\mathcal{B}$ is $\mathcal{A} + \mathcal{B} = \{a + b : a \in \mathcal{A}, b \in \mathcal{B}\}$. We refer the reader to Subsection 3.2.1 for examples.

When dealing with *generic periodic functions* $f : \mathbb{R} \to \mathbb{C}$, we assume without loss of generality that f is normalized such that it is centered and with period given by 2π; it can thus be decomposed as Fourier series $f(t) = \sum_{k\in\mathbb{Z}} F_k e^{\mathrm{i}kt}$ where $F_k := \frac{1}{2\pi}\int_0^{2\pi} f(t)e^{-\mathrm{i}kt}\mathrm{d}t$ and $F_0 = 0$. For such functions, we introduced in the previous chapter the *mean Lipschitz smoothness property* (Definition 3.13), to characterize the "smoothness on average", that we recall here for convenience.

Definition 4.1. A 2π-periodic function $f : \mathbb{R} \to \mathbb{C}$ is *mean Lipschitz smooth* with mean Lipschitz constant L_f^μ if for all radii $\delta \in (0, \pi]$ the maximum deviation of f in the interval $[-\delta, \delta]$ is, on average, bounded by $L_f^\mu \delta$, *i.e.*,

$$\tfrac{1}{2\pi}\textstyle\int_0^{2\pi} \sup_{r\in[-\delta,\delta]}\{|f(t+r) - f(t)|\}\,\mathrm{d}t \leq L_f^\mu \cdot \delta. \tag{4.4}$$

The advantage of this particular "smoothness" criterion is that is allows to handle *discontinuous functions*; *e.g.*, the maps depicted Fig. 4.3 are mean smooth (see Appendix A).

4.2.2 Statistical Learning

In the *statistical learning* (SL) framework [Vap99, SSBD14] (see detailed description in Section 2.1), one assumes that the signals of interest are generated by a *data distribution* $\mathcal{P}_0 \in \mathcal{M}$. The goal is then to fit some machine learning model, parametrized by a vector $\boldsymbol{\theta} \in \Theta$, to that distribution. More precisely, one seeks the model parameters $\boldsymbol{\theta}^*$ that minimize the *risk* objective $\mathcal{R}(\boldsymbol{\theta}; \mathcal{P}_0) := \mathbb{E}_{\boldsymbol{x} \sim \mathcal{P}_0} \ell(\boldsymbol{x}, \boldsymbol{\theta})$, *i.e.*, the expectation of a loss $\ell : \mathbb{R}^d \times \Theta \to \mathbb{R}$ with respect to the data distribution:

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta} \in \Theta} \mathcal{R}(\boldsymbol{\theta}; \mathcal{P}_0) = \arg\min_{\boldsymbol{\theta} \in \Theta} \mathbb{E}_{\boldsymbol{x} \sim \mathcal{P}_0} \ell(\boldsymbol{x}, \boldsymbol{\theta}). \tag{4.5}$$

In practice, the true data distribution $\mathcal{P}_0$ is unknown, but a dataset $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ of n samples $\boldsymbol{x}_i \sim_{\text{i.i.d.}} \mathcal{P}_0$ is available. The "ideal" risk minimization (4.5) is thus replaced by *empirical risk minimization* (ERM), which uses the empirical distribution $\widehat{\mathcal{P}}_{\mathcal{X}} := \frac{1}{n} \sum_{i=1}^n \delta_{\boldsymbol{x}_i} \in \mathcal{M}$ instead of the true data distribution:

$$\widetilde{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta} \in \Theta} \mathcal{R}(\boldsymbol{\theta}; \widehat{\mathcal{P}}_{\mathcal{X}}) = \arg\min_{\boldsymbol{\theta} \in \Theta} \sum_{\boldsymbol{x}_i \in \mathcal{X}} \ell(\boldsymbol{x}_i, \boldsymbol{\theta}). \tag{4.6}$$

In Section 2.1, we explained how many common machine learning task can be cast into the SL framework. We recall the two tasks we will focus on in this chapter, *i.e.*, k-means and Gaussian mixture modeling. As summarized in Table 4.2, *k-means clustering* seeks K centroids $\boldsymbol{c}_k \in \mathbb{R}^d$ which minimize the sum of squared errors (SSE) over the dataset (the "error" is the distance between each sample $\boldsymbol{x}_i$ and the centroid closest to it):

$$\text{SSE}(\boldsymbol{\theta}; \mathcal{X}) := \sum_{i=1}^n \min_{1 \leq k \leq K} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2 \tag{4.7}$$

On the other hand, *Gaussian mixture modeling* (GMM) seeks a weighted mixture of K Gaussians $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Gamma}_k)$ (*i.e.*, weights $w_k \geq 0$ that sum to one, centers $\boldsymbol{\mu}_k \in \mathbb{R}^d$, and positive definite covariance matrices $\boldsymbol{\Gamma}_k \in \mathbb{R}^{d \times d}$) that *maximizes* the log-likelihood (LL) of the dataset $\mathcal{X}$:

$$\text{LL}(\boldsymbol{\theta}; \mathcal{X}) := \sum_{i=1}^n \log\left(\sum_{k=1}^K w_k p_{\mathcal{N}}(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Gamma}_k)\right), \tag{4.8}$$

where $p_{\mathcal{N}}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Gamma})$ is the probability density function of the Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Gamma})$ evaluated at $\boldsymbol{x}$.

The central goal of SL is to control the *excess risk* $\mathcal{R}(\widetilde{\boldsymbol{\theta}}; \mathcal{P}_0) - \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0)$ (also known as generalization error for prediction tasks), in the form of *statistical guarantees*: for some $\delta \in (0, 1)$ and $\eta > 0$, the ERM solution $\widetilde{\boldsymbol{\theta}}$ satisfies

$$\mathbb{P}[\mathcal{R}(\widetilde{\boldsymbol{\theta}}; \mathcal{P}_0) - \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0) \leq \eta] \geq 1 - \delta. \tag{4.9}$$

In words, this guarantee ensures that, with probability larger than $1 - \delta$ over the sampling of $\mathcal{X}$, the estimate of the ERM is not worse than the optimal solution $\boldsymbol{\theta}^*$ (on the true data distribution $\mathcal{P}_0$) by a margin smaller than η.

4.2.3 Compressive Learning with guarantees

Recall that CL actually introduces a general *sketch operator* $\mathcal{A}_\Phi$, which acts on the space probability distributions $\mathcal{M}$. This operator "compresses" any input distribution $\mathcal{P} \in \mathcal{M}$ by computing m of its *generalized moments*, as defined by the associated feature map Φ.

Definition 4.2 (Sketch)**.** Given a feature map $\Phi : \mathbb{R}^d \to \mathbb{C}^m$, the associated *sketch operator* $\mathcal{A}_\Phi : \mathcal{M} \to \mathbb{C}^m$ is

$$\mathcal{A}_\Phi(\mathcal{P}) := \mathbb{E}_{\boldsymbol{x} \sim \mathcal{P}} \Phi(\boldsymbol{x}) = \int \Phi(\boldsymbol{x}) \, \mathrm{d}\mathcal{P}(\boldsymbol{x}) \in \mathbb{C}^m. \tag{4.10}$$

In particular, the *sketch of a dataset* $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$, noted $\boldsymbol{z}_{\Phi,\mathcal{X}}$, is actually the sketch of its empirical distribution,

$$\boldsymbol{z}_{\Phi,\mathcal{X}} := \mathcal{A}_\Phi(\widehat{\mathcal{P}}_{\mathcal{X}}) = \tfrac{1}{n} \textstyle\sum_{i=1}^n \Phi(\boldsymbol{x}_i) \in \mathbb{C}^m. \tag{4.11}$$

We provide an informal outline (not fully rigorous but sufficient for our purposes; see [GBKT17] for details) of how to "learn" (find good parameters $\boldsymbol{\theta}$) from the sketch with guarantees. One first associates to each parameter vector $\boldsymbol{\theta} \in \Theta$ a distribution $\mathcal{P}_{\boldsymbol{\theta}} \in \mathcal{M}$ (this map $\boldsymbol{\theta} \mapsto \mathcal{P}_{\boldsymbol{\theta}}$ is not necessarily injective [SGD19]), which respects a *risk consistency* property:

$$\mathcal{R}(\boldsymbol{\theta}; \mathcal{P}_{\boldsymbol{\theta}}) \leq \mathcal{R}(\boldsymbol{\theta}'; \mathcal{P}_{\boldsymbol{\theta}}), \quad \forall \boldsymbol{\theta}' \in \Theta. \tag{4.12}$$

Table 4.2 gives examples of this map for k-means[4] and GMM. When

[4]To emphasize the connection between the SL and CL formulations of k-means, we assign in Table 4.2 equal weights $\frac{1}{K}$ to all the Dirac deltas of the centroids $\delta_{\boldsymbol{c}_k}$; however the com-

	k-means clustering [KTTG17]	**Gaussian Mixture Modeling** [KBGP18]
$\boldsymbol{\theta}$	centroids $\{\boldsymbol{c}_k\}_{k=1}^K$	params. $\{w_k, \boldsymbol{\mu}_k, \boldsymbol{\Gamma}_k\}_{k=1}^K$
Θ	$\boldsymbol{c}_k \in \mathbb{R}^d$	$w_k \geq 0, \sum_k w_k = 1, \boldsymbol{\mu}_k \in \mathbb{R}^d$ $\boldsymbol{\Gamma}_k \in \mathbb{R}^{d\times d}, \boldsymbol{\Gamma}_k^\top = \boldsymbol{\Gamma}_k \succeq 0$
$\ell(\boldsymbol{x}, \boldsymbol{\theta})$	$\min_k \|\boldsymbol{x} - \boldsymbol{c}_k\|_2^2$	$-\log \sum_k w_k p_{\mathcal{N}}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Gamma}_k)$
$\mathcal{P}_{\boldsymbol{\theta}}$	$\sum_{k=1}^K \frac{1}{K}\delta_{\boldsymbol{c}_k}$	$\sum_k w_k \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Gamma}_k)$
$\mathcal{C}_\Phi(\boldsymbol{\theta}; \boldsymbol{z})$	$\|\boldsymbol{z} - \frac{1}{K}\sum_k \Phi(\boldsymbol{c}_k)\|_2$	$\|\boldsymbol{z} - \sum_k w_k \mathcal{A}_\Phi(\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Gamma}_k))\|_2$

Table 4.2 Description of two SL tasks and their equivalent in the CL framework. K-means seeks the K centroids $\boldsymbol{c}_k$ that minimize the sum of squared errors $\mathrm{SSE}(\boldsymbol{\theta}; \mathcal{X})$; in CL the parameters are mapped to a sum of K Dirac deltas. GMM seeks a weighted mixture of K Gaussians $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ that maximize the log-likelihood of the data $\mathrm{LL}(\boldsymbol{\theta}; \mathcal{X})$; in this case CL simply maps the parameters to the GMM distribution itself.

the parameter vector $\boldsymbol{\theta}$ varies, the resulting distributions constitute a *model set* $\mathcal{G} := \{\mathcal{P}_{\boldsymbol{\theta}} \,|\, \boldsymbol{\theta} \in \Theta\} \subset \mathcal{M}$. Learning then amounts to finding the parametrized distribution $\mathcal{P}_{\boldsymbol{\theta}}$ from $\mathcal{G}$ whose sketch—with respect to Φ—best fits the dataset sketch $\boldsymbol{z}_{\Phi,\mathcal{X}}$, as defined by the cost $\mathcal{C}_\Phi$:

$$\widehat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}\in\Theta} \mathcal{C}_\Phi(\boldsymbol{\theta}; \boldsymbol{z}_{\Phi,\mathcal{X}}) := \|\boldsymbol{z}_{\Phi,\mathcal{X}} - \mathcal{A}_\Phi(\mathcal{P}_{\boldsymbol{\theta}})\|_2. \tag{4.13}$$

Guarantees of the form (4.9) can be proven for this *sketch matching principle*. The idea is to show (4.13) is a surrogate approximating (4.6). Intuitively, it is possible to solve a task from the sketch if it somehow "encodes" that tasks risk objective. To assess how well the risk is encoded, [GBKT17] defines a seminorm $\|\cdot\|_{\mathcal{R}}$ to measure the difference between two distributions $\mathcal{P}, \mathcal{Q} \in \mathcal{M}$ with respect to the task-specific risk $\mathcal{R}$,

$$\|\mathcal{P} - \mathcal{Q}\|_{\mathcal{R}} := \sup_{\boldsymbol{\theta}\in\Theta} |\mathcal{R}(\boldsymbol{\theta}; \mathcal{P}) - \mathcal{R}(\boldsymbol{\theta}; \mathcal{Q})|. \tag{4.14}$$

Equipped with this metric, we say that the sketch operator $\mathcal{A}_\Phi$ "encodes" the risk $\mathcal{R}$ if the sketch distance $\|\mathcal{A}_\Phi(\mathcal{P}) - \mathcal{A}_\Phi(\mathcal{Q})\|_2$ bounds $\|\mathcal{P} - \mathcal{Q}\|_{\mathcal{R}}$ for all distributions in $\mathcal{G}$; the Lower Restricted Isometry Property (LRIP) formalizes this notion (this specific LRIP generalizes its well-known equiv-

plete formulation of CL k-means actually considers those weights as free parameters to be optimized [KTTG17].

alent from compressive sensing literature [CT05, FR17]).

Definition 4.3 (LRIP)**.** The sketch operator $\mathcal{A}_\Phi$ has the LRIP with constant γ on the model set $\mathcal{G}$, noted LRIP$(\gamma; \mathcal{G})$, if

$$\forall \mathcal{P}, \mathcal{Q} \in \mathcal{G}, \quad \|\mathcal{P} - \mathcal{Q}\|_{\mathcal{R}} \leq \gamma \|\mathcal{A}_\Phi(\mathcal{P}) - \mathcal{A}_\Phi(\mathcal{Q})\|_2. \tag{4.15}$$

One key theorem of CL[5], proved in [GBKT17], is that the LRIP implies statistical guarantees for the sketch matching (4.13).

Theorem 4.4 (LRIP implies excess risk control)**.** *Assume that $\mathcal{A}_\Phi$ has the* LRIP$(\gamma; \mathcal{G})$*. The excess risk of the solution $\widehat{\boldsymbol{\theta}}$ to* (4.13) *satisfies $\mathcal{R}(\widehat{\boldsymbol{\theta}}; \mathcal{P}_0) - \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0) \leq \eta$, where*

$$\eta = 2D(\mathcal{P}_0, \mathcal{G}) + 4\gamma \|\mathcal{A}_\Phi(\mathcal{P}_0) - \mathcal{A}_\Phi(\widehat{\mathcal{P}}_{\mathcal{X}})\|_2, \tag{4.16}$$

with $D(\mathcal{P}, \mathcal{G})$ a "distance" from $\mathcal{P}$ to the model set $\mathcal{G}$,

$$D(\mathcal{P}, \mathcal{G}) := \inf_{\mathcal{Q} \in \mathcal{G}} \{\|\mathcal{P} - \mathcal{Q}\|_{\mathcal{R}} + 2\gamma \|\mathcal{A}_\Phi(\mathcal{P}) - \mathcal{A}_\Phi(\mathcal{Q})\|_2\}.$$

The first term in (4.16) is a modeling bias term, the second one captures a sampling error, which decreases with n.

Theorem 4.4 guarantees that the excess risk is under control (bounded by η) provided that the related LRIP holds; it remains thus to prove the latter. This endeavor is highly specific to the considered model $\mathcal{G}$ (*i.e.*, the learning task) and feature map Φ. One usually proves that the LRIP holds *with high probability* $1 - \delta$ on the random draw of Φ, where the failure probability δ depends on the desired LRIP constant γ, the complexity of the model set $\mathcal{G}$, and the number of "measurements" m. These proofs are rather technical, see [GBKT20] for the case of compressive k-means and Gaussian mixture modeling from RFF sketches.

Remark: While we focus on theoretical guarantees for to the sketch matching program (4.13), it is worth remembering that this optimization problem is usually nonconvex. In practice, heuristics, such as compressive learning orthogonal matching pursuit (`CLOMP`) [KBGP18, KTTG17], thus *approximately* solve (4.13). Although they showed empirical success, one

[5]For the sake of presentation, Thm. 4.4 is taken from [GBKT17, Sec. 2.4] which presents a simplified but sub-optimal version of the "true" CL guarantees, established in [GBKT17, Sec. 2.5]. However, the extension we prove in this chapter can be carried over seamlessly to the main CL guarantees, as the improvements from [GBKT17, Sec. 2.5] are independent of our developments.

should keep in mind that those heuristics do not necessarily find the global solution $\widehat{\boldsymbol{\theta}}$, and that there might still be a performance gap between experimental results and the theoretical statistical learning guarantees (which apply to the global solution $\widehat{\boldsymbol{\theta}}$). This chapter mainly focuses on theoretical guarantees, except for the numerical validation (Sec. 4.5).

4.3 Asymmetric Compressive Learning

To summarize, the typical CL scenario consists of two phases: the sketching phase which maps $\mathcal{X}$ to $\boldsymbol{z}_{\Phi,\mathcal{X}}$ by averaging some feature map Φ over the data samples, followed by the learning phase where $\boldsymbol{\theta}$ is extracted by solving $\widehat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}\in\Theta} \mathcal{C}_\Phi(\boldsymbol{\theta}; \boldsymbol{z}_{\Phi,\mathcal{X}})$, *i.e.*, the sketch matching principle described in (4.13). The success of this scheme can be guaranteed by establishing the LRIP of the sketch operator $\mathcal{A}_\Phi$ associated to the "reference" feature map $\Phi : \mathbb{R}^d \to \mathbb{C}^m$ (*e.g.*, the random Fourier features, Φ_{RFF}), which must hold over the relevant task's model set $\mathcal{G}$.

4.3.1 Asymmetric sketch matching principle

In this chapter, we extend this scheme by allowing the sketching phase to use a different—or "distorted"—feature map $\Psi \neq \Phi$ (*e.g.*, the binarized RFF, Ψ_q from (4.2)). This amounts to studying this *asymmetric compressive learning* scenario (ACL for short): given a *reference feature map* Φ and a different *distorted feature map* Ψ, and having observed the "distorted sketch" $\boldsymbol{z}_{\Psi,\mathcal{X}} = \mathcal{A}_\Psi(\widehat{\mathcal{P}}_\mathcal{X}) = \frac{1}{n}\sum_{i=1}^n \Psi(\boldsymbol{x}_i)$, we select the parameters $\widehat{\boldsymbol{\theta}}'$ that solve the "asymmetric sketch matching" problem, *i.e.*,

$$\widehat{\boldsymbol{\theta}}' \in \arg\min_{\boldsymbol{\theta}\in\Theta} \mathcal{C}_\Phi(\boldsymbol{\theta}; \boldsymbol{z}_{\Psi,\mathcal{X}}) = \arg\min_{\boldsymbol{\theta}\in\Theta} \|\boldsymbol{z}_{\Psi,\mathcal{X}} - \mathcal{A}_\Phi(\mathcal{P}_{\boldsymbol{\theta}})\|_2. \tag{4.17}$$

To be perfectly clear, the "asymmetry" here refers to the fact that *only* the sketching map is distorted, since we still use the reference map Φ to learn from $\boldsymbol{z}_{\Psi,\mathcal{X}}$, as precised by the subscript in the cost $\mathcal{C}_\Phi$ (the only difference with the symmetric sketch matching from (4.13) is the sketch map).

This (perhaps surprising) strategy is inspired by a well-known equivalent in classical signal estimation: provided a signal follows a low-complexity model (such as a sparse or a low-rank description), one can treat its nonlinearly distorted measurements as noisy linear observations, *e.g.*, ignoring quantization, with provable reconstruction guarantees if the involved nonlinearity respects a few mild conditions [PV16]. Our work adopts a simi-

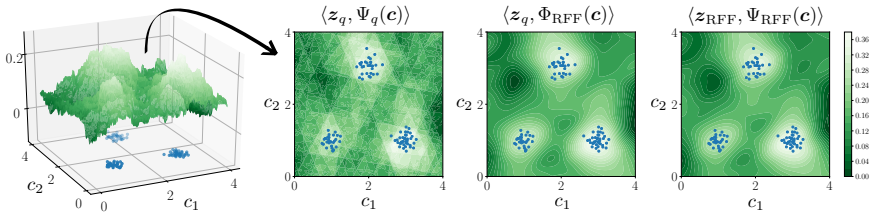


Fig. 4.2 Criterion $\langle z, \Phi(c) \rangle$ used to (greedily) select the first centroid $c = [c_1, c_2]$ in compressive k-means—on a dataset (in blue) made out of $K = 3$ clusters in dimension $d = 2$—when guided by several CL cost functions. For conciseness, we write the quantized sketch $z_q = z_{\Psi_q, \mathcal{X}}$ and the full-precision sketch $z_{\text{RFF}} = z_{\Phi_{\text{RFF}}, \mathcal{X}}$. *Surface plot and left contour plot*: when the (symmetric) fully quantized cost is used $\mathcal{C}_{\Psi_q}(\theta; z_q)$; *middle*: asymmetric cost $\mathcal{C}_{\Phi_{\text{RFF}}}(\theta; z_q)$ (our proposed approach); *right*: symmetric un-quantized cost $\mathcal{C}_{\Phi_{\text{RFF}}}(\theta; z_{\text{RFF}})$, for reference.

lar approach to get risk control guarantees for (4.17), of the same form as Thm. 4.4. This enables to quantify when (and under which conditions on Ψ) the asymmetric sketch matching scheme (4.17) succeeds.

What not to do: a naive approach to quantized sketching At this point, the interest of minimizing the asymmetric cost $\mathcal{C}_\Phi(\theta; z_{\Psi,\mathcal{X}}) = \|z_{\Psi,\mathcal{X}} - \mathcal{A}_\Phi(\mathcal{P}_\theta)\|_2$, instead of the (more intuitive) symmetric "distorted" cost

$$\mathcal{C}_\Psi(\theta; z_{\Psi,\mathcal{X}}) = \|z_{\Psi,\mathcal{X}} - \mathcal{A}_\Psi(\mathcal{P}_\theta)\|_2,$$

might not yet be fully clear to the reader. Well, we mentioned at the beginning of this section the advantage of being able to differentiate the cost function (*i.e.*, compute the gradient $\nabla_\theta \mathcal{C}_\Phi(\theta; z_{\Psi,\mathcal{X}})$), but this is a technical issue rather than a fundamental one. In particular, one could imagine other heuristics to still (approximately) solve $\mathcal{C}_\Psi(\theta; z_{\Psi,\mathcal{X}})$, *e.g.*, using various strategies to carefully "smoothe" Ψ when computing the gradient—in fact, this was exactly the subject of my Master's thesis [Sch17].

However, the disadvantages of optimizing $\mathcal{C}_\Psi(\theta; z_{\Psi,\mathcal{X}})$ go beyond the lack of a gradient: the optimization landscape as a whole is simply much harder to navigate in this case. This is illustrated on Fig. 4.2, which shows, on the left, the landscape of (a criterion based on) this fully quantized cost function; it is highly irregular, exhibiting many spurious optima. Empirically, optimizing this cost directly (using heuristics from [Sch17]) requires,

to reach target performance, a sketch size about 20 to 50 times as large as in the un-quantized case.

As we will see, the asymmetric cost function (on the middle of Fig. 4.2) behaves much better in this regard. To explain this nicer landscape, we will show that, in the context of random Fourier features sketching $\Phi = \Phi_{\text{RFF}}$, the asymmetric cost approximates *exactly the same cost* as the un-quantized scenario (on the right of Fig. 4.2). In fact, this analysis will not be restricted to quantized sketching $\Psi = \Psi_q$ defined in (4.2), but holds for more general random periodic features (RPF) $\Psi = \Psi_f$. This intuitive justification will be further elaborated on in Section 4.4 to establish formal ACL guarantees.

4.3.2 ACL from RPF and quantized sketches

Before working on the guarantees of the ACL scheme, let's build some understanding on why this scheme is interesting, by focusing on the case of *RPF sketches*. Indeed, when the reference map Φ is the random Fourier features map Φ_{RFF}, our analysis allows us in particular to replace the complex exponential in (4.1) by any (properly normalized) 2π-periodic function $f : \mathbb{R} \to \mathbb{C}$, *i.e.*, to consider as distorted feature map Ψ the *random periodic features* (RPF), extensively discussed in Chapter 3, defined as

$$\Psi_f(\boldsymbol{x}) := \tfrac{1}{\sqrt{m}} f\left(\boldsymbol{\Omega}^\top \boldsymbol{x} + \boldsymbol{\xi}\right). \tag{4.18}$$

Remark 4.5. Actually (as could be guessed from similar observations in the previous chapter), the RPF sketch cannot be used "as such" in the ACL optimization, but must be scaled by F_1, the first Fourier Series (FS) coefficient of f. We thus consider the *renormalized features* $\overline{\Psi}_f := \frac{1}{F_1}\Psi_f$ during the ACL optimization procedure. This renormalization is not restrictive, it can for example be performed after the sketch has been computed, since $\boldsymbol{z}_{\overline{\Psi}_f,\mathcal{X}} = \frac{1}{F_1}\boldsymbol{z}_{\Psi_f,\mathcal{X}}$.

Motivation of the ACL cost with RPF sketches In this case, the ACL scheme is motivated by the following observation: the asymmetric cost approaches—in expectation over the uniform dither $\boldsymbol{\xi} \sim \mathcal{U}^m([0, 2\pi))$—the symmetric cost. Intuitively, the dithering $\boldsymbol{\xi}$ allows us to "separate" $\boldsymbol{z}_{\Psi_f,\mathcal{X}}$ into two terms: one associated with the low frequencies of f, that contributes to the target objective $\mathcal{C}_{\Phi_{\text{RFF}}}(\boldsymbol{\theta}; \boldsymbol{z}_{\Phi_{\text{RFF}},\mathcal{X}})$, and one "high-frequency" term that is constant for the relevant optimization problem. This is formally proven in the following proposition.

Proposition 4.6. *Given a* fixed *dataset $\mathcal{X}$ and parameter vector $\boldsymbol{\theta}$, a random Fourier features map Φ_{RFF} and an associated normalized random periodic features map $\overline{\Psi}_f$ (for some 2π-periodic function f). The (squared) ACL cost approximates, on average on the draw of the dithering $\boldsymbol{\xi}$, the (squared) symmetric cost up to a constant:*

$$\mathbb{E}_{\boldsymbol{\xi}}\left[\mathcal{C}^2_{\Phi_{\mathrm{RFF}}}(\boldsymbol{\theta}; \boldsymbol{z}_{\overline{\Psi}_f,\mathcal{X}}) - \mathcal{C}^2_{\Phi_{\mathrm{RFF}}}(\boldsymbol{\theta}; \boldsymbol{z}_{\Phi_{\mathrm{RFF}},\mathcal{X}})\right] = c_{f,\mathcal{X}}, \tag{4.19}$$

with $c_{f,\mathcal{X}} = \|\boldsymbol{z}_{\overline{\Psi}_f,\mathcal{X}}\|_2^2 - \|\boldsymbol{z}_{\Phi_{\mathrm{RFF}},\mathcal{X}}\|_2^2$ a constant shift depending only on f and $\mathcal{X}$, that does not impact the optimization procedure.

Proof. We give here only a sketch of the proof to provide the intuitive justification of ACL (adapted from [SJ18b]). Rewrite $\Delta\mathcal{C} := \mathcal{C}^2_{\Phi_{\mathrm{RFF}}}(\boldsymbol{\theta}; \boldsymbol{z}_{\overline{\Psi}_f,\mathcal{X}}) - \mathcal{C}^2_{\Phi_{\mathrm{RFF}}}(\boldsymbol{\theta}; \boldsymbol{z}_{\Phi_{\mathrm{RFF}},\mathcal{X}})$ as

$$\begin{aligned}
\Delta\mathcal{C} &= \|\boldsymbol{z}_{\overline{\Psi}_f,\mathcal{X}} - \mathcal{A}_{\Phi_{\mathrm{RFF}}}(\mathcal{P}_{\boldsymbol{\theta}})\|_2^2 - \|\boldsymbol{z}_{\Phi_{\mathrm{RFF}},\mathcal{X}} - \mathcal{A}_{\Phi_{\mathrm{RFF}}}(\mathcal{P}_{\boldsymbol{\theta}})\|_2^2 \\
&= \|\boldsymbol{z}_{\overline{\Psi}_f,\mathcal{X}}\|_2^2 - \|\boldsymbol{z}_{\Phi_{\mathrm{RFF}},\mathcal{X}}\|_2^2 - 2\Re\langle \boldsymbol{z}_{\overline{\Psi}_f,\mathcal{X}} - \boldsymbol{z}_{\Phi_{\mathrm{RFF}},\mathcal{X}}, \mathcal{A}_{\Phi_{\mathrm{RFF}}}(\mathcal{P}_{\boldsymbol{\theta}})\rangle \\
&= c_{f,\mathcal{X}} + \sum_{j=1}^{m} Z_j
\end{aligned}$$

with random variables defined as

$$Z_j := -2\Re\big[\big(\tfrac{1}{n}\textstyle\sum_{\boldsymbol{x}_i\in\mathcal{X}}(\overline{\Psi}_{f,j}(\boldsymbol{x}_i) - \Phi_{\mathrm{RFF},j}(\boldsymbol{x}_i))\big)\mathcal{A}^*_{\Phi_{\mathrm{RFF}},j}(\mathcal{P}_{\boldsymbol{\theta}})\big].$$

We can then show that each random variables Z_j has zero mean. The full argument is not developed here but is completely analogous to the one of Proposition 3.8, which relies on Fourier series expansions and the orthogonality of the complex exponentials $\mathbb{E}_{\xi}\, e^{\mathrm{i}k\xi} = \delta_{k,0}$. □

Specific RPF sketches of interest Besides usual RFF (a special case of RPF), we consider two specific instances of RPF sketches, which are illustrated by Figure 4.3.

Of course, the first one is the running example we used since Section 4.1, namely the *quantized RFF* Ψ_q , defined by (4.2), for which

$$f(t) = q(t) := \operatorname{sign}(e^{\mathrm{i}t}) \quad \in \{\pm 1 \pm \mathrm{i}\}, \tag{4.20}$$

with sign acting independently on the real and imaginary component—*i.e.*, the real and imaginary components of q are phase shifted "square waves".

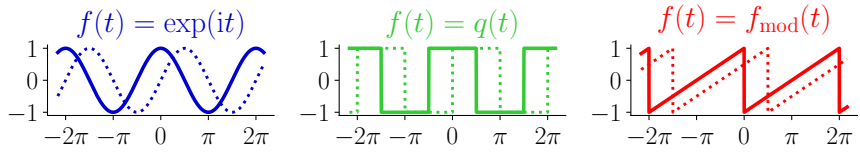


Fig. 4.3 The 2π-periodic functions $f : \mathbb{R} \to \mathbb{C}$ of particular interest ($\Re f$ in plain, $\Im f$ in dashed), related to RPF $\Psi_f(\boldsymbol{x}) = \frac{1}{\sqrt{m}} f(\boldsymbol{\Omega}^\top \boldsymbol{x} + \boldsymbol{\xi})$. Left, the complex exponential $\exp(\mathrm{i}\cdot)$, related to the usual random Fourier features Φ_{RFF}; middle, the one-bit universal quantization function q, related to quantized RFF Ψ_q; and right, the (complex and normalized) modulo function mod, related to the modulo features Ψ_{mod}.

This feature map is (the complex extension[6] of) the so-called "one-bit universal quantization" introduced in [Bou12], and further studied in [BRM17]. The two main advantages which motivated us to consider Ψ_q are that (i) it produces *quantized* sketch contributions[7] $\Psi_q(\boldsymbol{x}_i) \in \{\frac{\pm 1 \pm \mathrm{i}}{\sqrt{m}}\}^m$, which heavily reduces the cost of their (potential) transmission or storage (*e.g.*, $\Psi_q(\boldsymbol{x}_i)$ can be trivially encoded using only $2m$ bits); and (ii) it is amenable to *plausible hardware implementations* (even if q is implemented in software, it is still cheaper to evaluate than $\exp(\mathrm{i}\cdot)$).

Another particular case of the RPF with promising applications is the (complex) *modulo RFF* Ψ_{mod} defined as

$$f(t) = \mathrm{mod}(t) := \mathrm{mod}_{2\pi}(t) + \mathrm{i} \cdot \mathrm{mod}_{2\pi}(t - \tfrac{\pi}{2}) \in \mathbb{C}, \tag{4.21}$$

where $\mathrm{mod}_T(t) := 2\left(\frac{t}{T} - \left\lfloor \frac{t}{T} \right\rfloor\right) - 1$ is the "normalized" modulo T operation—*i.e.*, the real and imaginary components of $\mathrm{mod}()$ are phase shifted "sawtooth waves" (Fig. 4.3). These modulo features take their roots from the recent theory of *modulo sampling* of signals [BKR17], and—much closer to our definition of Ψ_{mod}—its extension to compressive modulo measurements of structured signals [SH19].

As explained in [BKR17, SH19], the advantage of this scheme is the existence of dedicated modulo sensors (*e.g.*, using self-reset analog-to-digital converters [RJ03]) which again paves the way for *efficient hardware computation* of the sketch contributions $\Psi_{\mathrm{mod}}(\boldsymbol{x}_i)$.

[6] While we focus on *complex-valued* maps (Ψ_q, Ψ_{mod}), as motivated by Section 3.7, their *real-valued* counterpart ($\Re\Psi_q$, $\Re\Psi_{\mathrm{mod}}$), closer to their initial formulations in the literature, could very well also be considered.

[7] The sketch itself (*i.e.*, after averaging) is not necessarily quantized.

Finally, we recall that as remarked above, the RPF are also of interest if one seeks to implement f in hardware sensors but can only ensure its periodicity, without accurate control of its precise shape f due to imperfections.

What now? Let's take a step back. We formally defined, in Subsection 4.3.1, the generic ACL strategy. To keep things concrete, in Subsection 4.3.2, we took some care to explain in detail the application of this strategy to the case of RPF sketches, providing an intuitive justification on why it is expected to work (Prop. 4.6), and further detailing the practical relevance of RPF sketches.

However, Prop. 4.6, while a promising start to gain intuition, does not *guarantee* anything regarding the excess risk achieved by ACL (*e.g.*, for finite values of the sketch size m). In the next section, we provide formal guarantees for the ACL scheme. In particular, in Subsection 4.4.1 we provide a guarantee for the generic ACL scheme by building upon the existing LRIP; this ensures that our result can be applied to other and future sketch constructions, provided these satisfy the LRIP. Then, in Subsection 4.4.2 we turn our attention back to RPF sketchs in particular, for which we provide explicit instances of this guarantee.

4.4 Excess risk guarantees

4.4.1 A generic guarantee for ACL

Our goal is to prove guarantees in the spirit of Thm. 4.4 for the ACL scheme. To derive unifying guarantees for each combination of model set $\mathcal{G}$ (task), reference map Φ and distortion Ψ, we build upon the existing LRIP (characterizing the compatibility between $\mathcal{G}$ and Φ), which we combine with another property, the *Limited Projected Distortion* (LPD) property (extending its definition from [XJ20]). The LPD better characterizes the closeness between the distorted and reference sketches, $\boldsymbol{z}_{\Psi,\mathcal{X}}$ and $\boldsymbol{z}_{\Phi,\mathcal{X}}$, than the too restrictive Euclidean distance $\|\boldsymbol{z}_{\Psi,\mathcal{X}} - \boldsymbol{z}_{\Phi,\mathcal{X}}\|_2$: for example, with the quantized RFF sketch, this distance does not vanish as m grows to infinity.

The LPD relies on an additional assumption characterizing over which datasets $\mathcal{X}$ it should hold.

Assumption 4.7. There exists a set $\widehat{\mathcal{G}} \subseteq \mathcal{M}$, coined *empirical set*, such that, for any considered dataset $\mathcal{X}$, the empirical distribution $\widehat{\mathcal{P}}_{\mathcal{X}}$ belongs to $\widehat{\mathcal{G}}$.

Note that Assumption 4.7 is quite permissive. We allow in particular to pick $\widehat{\mathcal{G}} = \mathcal{M}$, which means it holds trivially. Moreover, even when $\widehat{\mathcal{G}} \subsetneq \mathcal{M}$, it is quite mild; it is for example fulfilled under the natural assumption that the data samples $\boldsymbol{x}_i$ belong to a bounded domain Σ (more on this later). That being said, we can now formalize the LPD.

Definition 4.8 (LPD)**.** Given the empirical set $\widehat{\mathcal{G}} \subseteq \mathcal{M}$, an error $\epsilon > 0$, a reference sketch operator $\mathcal{A}_\Phi$, and a (task-dependent) model set $\mathcal{G} \subset \mathcal{M}$, we say that the distorted sketch operator $\mathcal{A}_\Psi$ satisfies the LPD of error ϵ on $\widehat{\mathcal{G}}$ with respect to $\mathcal{A}_\Phi$ and $\mathcal{G}$, or shortly LPD$(\epsilon; \widehat{\mathcal{G}}, \mathcal{G}, \mathcal{A}_\Phi)$, if

$$\forall \mathcal{P} \in \widehat{\mathcal{G}}, \mathcal{Q} \in \mathcal{G}, \quad |\langle \mathcal{A}_\Psi(\mathcal{P}) - \mathcal{A}_\Phi(\mathcal{P}), \mathcal{A}_\Phi(\mathcal{Q}) \rangle| \leq \epsilon. \tag{4.22}$$

In words, the LPD thus ensures that, for any considered dataset $\mathcal{X}$, the difference between its distorted sketch $\boldsymbol{z}_{\Psi,\mathcal{X}}$ and its reference sketch $\boldsymbol{z}_{\Phi,\mathcal{X}}$ is sufficiently small when projected on any possible "reference sketch" of the model set, *i.e.*, projected on any $\mathcal{A}_\Phi(\mathcal{P}_{\boldsymbol{\theta}})$ for all parameters $\boldsymbol{\theta} \in \Theta$. As made clear below, $\{\mathcal{A}_\Phi(\mathcal{P}_{\boldsymbol{\theta}}) : \mathcal{P}_{\boldsymbol{\theta}} \in \mathcal{G}\}$ contains actually the "directions" that matter for solving (4.17).

Our first main result (Prop. 4.10) states that if the reference sketching operator $\mathcal{A}_\Phi$ satisfies the LRIP, and the distorted sketching operator $\mathcal{A}_\Psi$ the LPD, then the excess risk of the ACL solution (4.17) can be controlled. To prove this, we adapt the proof of Thm 4.4 found in [GBKT17,KG18], and we leverage the LPD to show that distorting the sketch does not modify the cost function too much—as first shown in Lemma 4.9.

Lemma 4.9. *Assume $\mathcal{A}_\Psi$ satisfies the* LPD$(\epsilon; \widehat{\mathcal{G}}, \mathcal{G}, \mathcal{A}_\Phi)$*. For any $\widehat{\mathcal{P}}_{\mathcal{X}} \in \widehat{\mathcal{G}}$, the asymmetric sketch matching solution $\widehat{\boldsymbol{\theta}}'$ to* (4.17) *is sub-optimal with respect to the symmetric matching solution $\widehat{\boldsymbol{\theta}}$ to* (4.13) *by at most*

$$\mathcal{C}_\Phi(\widehat{\boldsymbol{\theta}}'; \boldsymbol{z}_{\Phi,\mathcal{X}}) - \mathcal{C}_\Phi(\widehat{\boldsymbol{\theta}}; \boldsymbol{z}_{\Phi,\mathcal{X}}) \leq 2\sqrt{\epsilon}. \tag{4.23}$$

Proof. For conciseness, we drop the $\mathcal{X}$ subscript and denote $\boldsymbol{a} := \mathcal{A}_\Phi(\mathcal{P}_{\widehat{\boldsymbol{\theta}}})$, $\boldsymbol{a}' := \mathcal{A}_\Phi(\mathcal{P}_{\widehat{\boldsymbol{\theta}}'})$. The LPD implies both

$$\begin{aligned} \|\boldsymbol{z}_\Psi - \boldsymbol{a}\|_2^2 - \|\boldsymbol{z}_\Phi - \boldsymbol{a}\|_2^2 &\leq 2\epsilon + \|\boldsymbol{z}_\Psi\|_2^2 - \|\boldsymbol{z}_\Phi\|_2^2, \\ \|\boldsymbol{z}_\Phi - \boldsymbol{a}'\|_2^2 - \|\boldsymbol{z}_\Psi - \boldsymbol{a}'\|_2^2 &\leq 2\epsilon + \|\boldsymbol{z}_\Phi\|_2^2 - \|\boldsymbol{z}_\Psi\|_2^2. \end{aligned}$$

By optimality of (4.17), $\|\boldsymbol{z}_\Psi - \boldsymbol{a}'\|_2^2 - \|\boldsymbol{z}_\Psi - \boldsymbol{a}\|_2^2 \leq 0$, and adding the three inequalities together gives $\|\boldsymbol{z}_\Phi - \boldsymbol{a}'\|_2^2 \leq \|\boldsymbol{z}_\Phi - \boldsymbol{a}\|_2^2 + 4\epsilon$; a square root

completes the proof. □

Proposition 4.10 (Asymmetric sketch matching risk control)**.** *Assume* $\mathcal{A}_\Phi$ *satisfies the* LRIP$(\gamma; \mathcal{G})$ *and* $\mathcal{A}_\Psi$ *the* LPD$(\epsilon; \widehat{\mathcal{G}}, \mathcal{G}, \mathcal{A}_\Phi)$*. The solution* $\widehat{\boldsymbol{\theta}}'$ *to the asymmetric problem* (4.17) *satisfies* $\mathcal{R}(\widehat{\boldsymbol{\theta}}'; \mathcal{P}_0) - \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0) \leq \eta'$*, where*

$$\eta' = 2D(\mathcal{P}_0, \mathcal{G}) + 4\gamma \|\mathcal{A}_\Phi(\mathcal{P}_0) - \mathcal{A}_\Phi(\widehat{\mathcal{P}}_\mathcal{X})\|_2 + 4\gamma\sqrt{\epsilon}, \tag{4.24}$$

with $D(\mathcal{P}, \mathcal{G})$ *as in Thm. 4.4*

Proof. We use the same notations as in the previous proof. For some arbitrary $\mathcal{Q} \in \mathcal{G}$, since $\|\cdot\|_\mathcal{R}$ is a seminorm [GBKT17],

$$\|\mathcal{P}_{\widehat{\boldsymbol{\theta}}'} - \mathcal{P}_0\|_\mathcal{R} \leq \|\mathcal{P}_{\widehat{\boldsymbol{\theta}}'} - \mathcal{Q}\|_\mathcal{R} + \|\mathcal{Q} - \mathcal{P}_0\|_\mathcal{R}.$$

Since $\mathcal{P}_{\widehat{\boldsymbol{\theta}}'}, \mathcal{Q} \in \mathcal{G}$, the LRIP and the triangle inequality give

$$\begin{aligned}\|\mathcal{P}_{\widehat{\boldsymbol{\theta}}'} - \mathcal{Q}\|_\mathcal{R} &\leq \gamma \|\mathcal{A}_\Phi(\mathcal{P}_{\widehat{\boldsymbol{\theta}}'}) - \mathcal{A}_\Phi(\mathcal{Q})\|_2 \\ &\leq \gamma \|\boldsymbol{a}' - \boldsymbol{z}_\Phi\|_2 + \gamma \|\boldsymbol{z}_\Phi - \mathcal{A}_\Phi(\mathcal{Q})\|_2.\end{aligned}$$

Using Lemma 4.9 then the optimality of (4.13), we get

$$\begin{aligned}\tfrac{\|\mathcal{P}_{\widehat{\boldsymbol{\theta}}'} - \mathcal{Q}\|_\mathcal{R}}{\gamma} &\leq 2\sqrt{\epsilon} + \|\boldsymbol{a} - \boldsymbol{z}_\Phi\|_2 + \|\boldsymbol{z}_\Phi - \mathcal{A}_\Phi(\mathcal{Q})\|_2 \\ &\leq 2\sqrt{\epsilon} + 2\|\boldsymbol{z}_\Phi - \mathcal{A}_\Phi(\mathcal{Q})\|_2.\end{aligned}$$

We develop the second term with the triangle inequality,

$$\|\boldsymbol{z}_\Phi - \mathcal{A}_\Phi(\mathcal{Q})\|_2 \leq \|\boldsymbol{z}_\Phi - \mathcal{A}_\Phi(\mathcal{P}_0)\|_2 + \|\mathcal{A}_\Phi(\mathcal{P}_0) - \mathcal{A}_\Phi(\mathcal{Q})\|_2.$$

Gathering the results, and taking the infimum with respect to $\mathcal{Q} \in \mathcal{G}$, we obtain that $\|\mathcal{P}_{\widehat{\boldsymbol{\theta}}'} - \mathcal{P}_0\|_\mathcal{R} \leq \frac{\eta'}{2}$, with

$$\tfrac{\eta'}{2} = D(\mathcal{P}_0, \mathcal{G}) + 2\gamma \|\boldsymbol{z}_\Phi - \mathcal{A}_\Phi(\mathcal{P}_0)\|_2 + 2\gamma\sqrt{\epsilon}.$$

Finally, we combine this with the risk metric definition (4.14), and since the

map $\boldsymbol{\theta} \mapsto \mathcal{P}_{\boldsymbol{\theta}}$ satisfies the risk consistency property (4.12), we get

$$\begin{aligned}
&\mathcal{R}(\widehat{\boldsymbol{\theta}}'; \mathcal{P}_0) - \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0) \\
&= \mathcal{R}(\widehat{\boldsymbol{\theta}}'; \mathcal{P}_0) - \mathcal{R}(\widehat{\boldsymbol{\theta}}'; \mathcal{P}_{\widehat{\boldsymbol{\theta}}'}) + \mathcal{R}(\widehat{\boldsymbol{\theta}}'; \mathcal{P}_{\widehat{\boldsymbol{\theta}}'}) - \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_{\widehat{\boldsymbol{\theta}}'}) \\
&\quad + \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_{\widehat{\boldsymbol{\theta}}'}) - \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0) \\
&\leq \tfrac{\eta'}{2} + 0 + \tfrac{\eta'}{2} = \eta'.
\end{aligned}$$

□

For comparison, the excess risk guarantee of Prop. 4.10 for the asymmetric sketch matching solution (4.17) is thus exactly the same as the guarantee in Thm. 4.4 for the symmetric solution (4.13), up to the additive term $4\sqrt{\epsilon}$ in the excess risk bound (*i.e.*, $\eta' = \eta + 4\sqrt{\epsilon}$), which expresses the "mismatch" between Ψ and Φ. Whenever the LPD holds with a reasonably small error ϵ, we can thus expect that the asymmetric scheme (learning from the distorted sketch $\boldsymbol{z}_{\Psi,\mathcal{X}}$) will perform almost as well as the symmetric one (learning from the reference sketch $\boldsymbol{z}_{\Phi,\mathcal{X}}$).

Of course, it remains to show that the LPD actually holds in practice to complete this guarantee, which we tackle next.

4.4.2 Proving the LPD for quantized CL

In this subsection, we first provide one possible strategy to prove the LPD. Under an additional assumption (which we first introduce and show to be met in practice for k-means and GMM), it is sufficient to prove a somewhat simpler "signal-level" version of the LPD instead (which we call signal-LPD, or *sLPD* for short). We then apply this strategy to the specific case where $\Phi = \Phi_{\text{RFF}}$ are the random Fourier features and $\Psi = \Psi_f$ are generic random periodic features (4.18), *i.e.*, the same RFF but where $t \mapsto \exp(\mathrm{i}t)$ is replaced by a generic periodic function $f(t)$. This finally allows us to obtain, as a particular case, statistical learning guarantees for the quantized CL scheme introduced in [SJ18b], *i.e.*, where the distorted map is the binarized RFF Ψ_q (4.2). To demonstrate the generic nature of our result, we also apply it to the modulo features Ψ_{mod}.

Assumption on the data domain

In practical machine learning applications, the data vectors $\boldsymbol{x}_i$ do not take *any* possible value in $\mathbb{R}^d$—one can usually assume *a priori* that they belong

to a compact set $\Sigma \subset \mathbb{R}$. We extend this assumption to the *probability distributions* involved in the compressive learning problem (4.17), *i.e.*, to the empirical distributions $\widehat{\mathcal{P}}_{\mathcal{X}} \in \widehat{\mathcal{G}}$ (see Assumption 4.7) and the parametric distributions $\mathcal{P}_{\boldsymbol{\theta}} \in \mathcal{G}$.

Assumption 4.11. For a compact set $\Sigma \subset \mathbb{R}^d$, the model set $\mathcal{G}$ and the empirical set $\widehat{\mathcal{G}}$ are subsets of $\mathcal{M}_{\Sigma,\zeta} \subset \mathcal{M}$, the set of probability measures that are *mostly* supported on Σ, *i.e.*, for some $0 \leq \zeta < 1$, $\mathcal{G}, \widehat{\mathcal{G}} \subset \mathcal{M}_{\Sigma,\zeta}$ with

$$\mathcal{M}_{\Sigma,\zeta} := \{\mathcal{P} \in \mathcal{M} : \mathcal{P}(\Sigma) = \textstyle\int_{\Sigma} \mathrm{d}\mathcal{P} \geq 1 - \zeta\}. \tag{4.25}$$

In particular, for $\zeta = 0$ all $\mathcal{P} \in \mathcal{M}_{\Sigma,0}$ are "almost surely" supported on Σ, *i.e.*, $\operatorname{supp}(\mathcal{P}) \subset \Sigma$.

Assumption 4.11 holds in practice. Consider the common case where there are known lower and upper bounds $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^d$ for the values that the learning samples $\boldsymbol{x}_i$ can take (*e.g.*, due to physical constraints). This means that all the learning examples lie in a "box" $\Sigma_{\boldsymbol{l},\boldsymbol{u}} \subset \mathbb{R}^d$:

$$\boldsymbol{x}_i \in \Sigma_{\boldsymbol{l},\boldsymbol{u}} := \{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}.$$

Since all the examples of any considered dataset $\mathcal{X}$ necessarily lie in that box, this directly implies that all the related empirical distributions $\widehat{\mathcal{P}}_{\mathcal{X}} \in \widehat{\mathcal{G}}$ satisfy $\widehat{\mathcal{P}}_{\mathcal{X}} \in \mathcal{M}_{\Sigma_{\boldsymbol{l},\boldsymbol{u}},0}$.

The inclusion of the model set $\mathcal{G}$ can be reached from additional constraints on Θ, imposed during the optimization procedure. In the simplest case, the *k-means* task (Table 4.2, left), the optimal centroids obviously lie inside the data-enclosing box, hence it makes sense (as done in [KTTG17]) to restrict the problem to $\boldsymbol{c}_k \in \Sigma_{\boldsymbol{l},\boldsymbol{u}}$. These constraints can be encoded in Θ, which in turn imply that $\mathcal{G} \subset \mathcal{M}_{\Sigma_{\boldsymbol{l},\boldsymbol{u}},0}$. If the data-enclosing box $\Sigma_{\boldsymbol{l},\boldsymbol{u}}$ is known, Assumption 4.11 thus holds for k-means, with $\Sigma = \Sigma_{\boldsymbol{l},\boldsymbol{u}}$ and $\zeta = 0$.

For the *Gaussian mixture modeling* task (Table 4.2, right), one can similarly enforce that the Gaussian centers lie in the box $\boldsymbol{\mu}_k \in \Sigma_{\boldsymbol{l},\boldsymbol{u}}$. Moreover, given that the data lie in a bounded domain, it is also reasonable to bound the variance of the Gaussian modes (the typical spread of a Gaussian mode should not be much larger than the box). This can be done by bounding the eigenvalues of the covariance matrices $\boldsymbol{\Gamma}_k$, *i.e.*, $\lambda_{\max}(\boldsymbol{\Gamma}_k) \leq S$ for a bound $S > 0$ to be set according to the size of the box. Assuming diagonal covariances, as commonly done CL for GMM [KBGP18], the following lemma shows that $\mathcal{G} \subset \mathcal{M}_{\Sigma^{(\rho)},\zeta}$ for some slightly extended set $\Sigma^{(\rho)} \supset \Sigma_{\boldsymbol{l},\boldsymbol{u}}$ and small $\zeta > 0$.

Lemma 4.12 (Assumption 4.11 for GMM with box contraints)**.** *Let $\mathcal{G}$ be the model set of a GMM task with diagonal covariances $\mathbf{\Gamma}_k$ and the box constraints $\boldsymbol{\mu}_k \in \Sigma_{\boldsymbol{l},\boldsymbol{u}}$ and $\lambda_{\max}(\mathbf{\Gamma}_k) \leq S$. Given $\rho > d$, we define the bounds of an "extended" box $\widetilde{\boldsymbol{l}} = \boldsymbol{l} - \rho S\mathbf{1}$ and $\widetilde{\boldsymbol{u}} = \boldsymbol{u} + \rho S\mathbf{1}$, with $\mathbf{1} = (1, 1, ..., 1) \in \mathbb{R}^d$. Then, for $\Sigma^{(\rho)} = \Sigma_{\widetilde{\boldsymbol{l}},\widetilde{\boldsymbol{u}}}$, we have*

$$\mathcal{G} \subset \mathcal{M}_{\Sigma^{(\rho)},\zeta}, \quad \textit{with } \zeta \lesssim e^{-\rho^2}.$$

Proof. From (4.25), we can set $\zeta = \sup\{\int_{\Sigma^{(\rho)c}} \mathrm{d}\mathcal{P} : \mathcal{P} \in \mathcal{G}\}$ to ensure $\mathcal{G} \subset \mathcal{M}_{\Sigma^{(\rho)},\zeta}$, *i.e.*, the maximal failure probability is reached when the largest amount of the GMMs probability mass lies outside $\Sigma^{(\rho)}$. For diagonal covariances, this is reached if we have both $\mathbf{\Gamma}_k = S\boldsymbol{I}_d$, *i.e.*, the Gaussian is maximally spread in each dimension, and, by symmetry, each Gaussian mode is located at a corner of the box $\Sigma_{\boldsymbol{l},\boldsymbol{u}}$, *e.g.*, with $\boldsymbol{\mu}_k = \boldsymbol{u}$ for all k. Denoting by $\mathcal{P}^* \subset \mathcal{G}$ this GMM configuration, we easily show that, for $\Sigma^{(\rho)} = \Sigma_{\widetilde{\boldsymbol{l}},\widetilde{\boldsymbol{u}}}$, *i.e.*, $\Sigma^{(\rho)} = \Sigma_{\boldsymbol{l},\boldsymbol{u}} + \rho S\mathbb{B}_\infty^d$, $\int_{\Sigma^{(\rho)}} \mathrm{d}\mathcal{P}^*$ is bounded by $\phi^d(\rho)$, with ϕ the cumulative density function of a one-dimensional standard normal random variable. Using well-known Gaussian tail bounds [Ver18], we get $\zeta = 1 - \phi^d(\rho) \leq d \cdot \phi(\rho) \leq \frac{d}{\rho\sqrt{2\pi}} e^{-\rho^2/2}$, which decays exponentially fast in ρ; in particular $\zeta \lesssim e^{-\rho^2}$ when $\rho > d$. □

To wrap up, Assumption 4.11 thus seems fairly reasonable for the GMM task as well. There is however still the issue of which value of ρ one should pick in Lemma 4.12; we'll come back to this nontrivial question at the end of this section.

Reducing the LPD to the signal-level LPD

Under Assumption 4.11, to have the LPD, it is sufficient to prove a simpler "signal-LPD" (sLPD) over Σ, defined as

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \Sigma, \ |\langle \Psi(\boldsymbol{x}) - \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y})\rangle| \leq \epsilon_s. \tag{4.26}$$

This is formalized by the following lemma.

Lemma 4.13 (sLPD implies LPD)**.** *For a compact set Σ and $0 \leq \zeta < 1$, assume $\mathcal{G}, \widehat{\mathcal{G}} \subset \mathcal{M}_{\Sigma,\zeta}$ and the sLPD (4.26) holds on Σ with error ϵ_s. Moreover, assume that the feature maps are bounded, $\sup_{\boldsymbol{x}\in\mathbb{R}^d} \|\Phi(\boldsymbol{x})\|_2 \leq C_\Phi$ (similarly for Ψ). Then, $\mathcal{A}_\Psi$ has the $\mathrm{LPD}(\epsilon; \widehat{\mathcal{G}}, \mathcal{G}, \mathcal{A}_\Phi)$ with error $\epsilon = \epsilon_s + c\zeta$, where $c \leq 3C_\Phi(C_\Phi + C_\Psi)$.*

Proof. Define the "difference kernel"

$$\widetilde{\kappa}(\boldsymbol{x}, \boldsymbol{y}) := \langle \Psi(\boldsymbol{x}) - \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}) \rangle,$$

which is bounded by $|\widetilde{\kappa}(\boldsymbol{x}, \boldsymbol{y})| \leq C_\Phi(C_\Phi + C_\Psi) =: C_{\widetilde{\kappa}}$ using Cauchy-Schwarz. Given any $\mathcal{P}, \mathcal{Q} \in \mathcal{M}_{\Sigma,\zeta}$, we have

$$\begin{aligned}
|\langle \mathcal{A}_\Psi(\mathcal{P}) - \mathcal{A}_\Phi(\mathcal{P}), \mathcal{A}_\Phi(\mathcal{Q}) \rangle| &= \left| \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{P}} \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{Q}} \widetilde{\kappa}(\boldsymbol{x}, \boldsymbol{y}) \right| \\
&\leq \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}} \mathbb{E}_{\boldsymbol{y}\sim\mathcal{Q}} \left| \widetilde{\kappa}(\boldsymbol{x}, \boldsymbol{y}) \right| \\
&= I(\Sigma, \Sigma) + I(\Sigma^c, \Sigma) + I(\Sigma, \Sigma^c) + I(\Sigma^c, \Sigma^c) \\
&\leq \epsilon_s + C_{\widetilde{\kappa}}(2\zeta + \zeta^2) \\
&\leq \epsilon_s + 3C_{\widetilde{\kappa}}\zeta,
\end{aligned}$$

with $I(U, V) := \int_U \int_V |\widetilde{\kappa}(\boldsymbol{x}, \boldsymbol{y})| \ \mathrm{d}\mathcal{P}(\boldsymbol{x}) \, \mathrm{d}\mathcal{Q}(\boldsymbol{y})$, $U, V \subset \mathbb{R}^d$. □

Combining with Prop. 4.10, under Assumption 4.11, the sLPD (4.26) with error ϵ_s thus implies the excess risk is controlled, with an additive increase given by $4\sqrt{\epsilon_s + c\zeta}$ compared to Thm. 4.4.

Proving the signal-LPD for random periodic features

The tools developed up to now were purposefully as general as possible. We now focus on proving the LPD for the case where the reference feature map are random Fourier features (4.1) Φ_{RFF}, and the distorted feature map are random periodic features (4.18) $\Psi_f(\boldsymbol{x}) := \frac{1}{\sqrt{m}} f(\boldsymbol{\Omega}^\top \boldsymbol{x} + \boldsymbol{\xi})$. Recalling that in Prop. 4.6 the RPF feature map must be scaled by F_1, the first Fourier Series (FS) coefficient of f, we actually consider the *renormalized features* $\overline{\Psi}_f := \frac{1}{F_1}\Psi_f$ during the ACL optimization procedure.

We can then prove our second main result: the LPD (4.22) holds—with high probability on the draw of $\boldsymbol{\Omega}$ and $\boldsymbol{\xi}$—for the (normalized) RPF sketch operator. The proof is based on Lemma 4.13 to reduce the LPD to the sLPD, and the latter is shown using Corollary 3.17 from Chapter 3.

Proposition 4.14 (LPD for normalized RPF)**.** *Let us consider a compact set* Σ *with Kolmogorov* ν*-entropy* $\mathcal{H}_\nu(\Sigma) < \infty$*, and*

- *the random Fourier features* Φ_{RFF} *defined in* (4.1) *associated with the distribution* Λ *(which generates the* m *columns of* $\boldsymbol{\Omega}$*), with* smoothness constant

$$C_\Lambda := \max_{\boldsymbol{a}\in\mathbb{R}^d, \|\boldsymbol{a}\|_2=1} \mathbb{E}_{\boldsymbol{\omega}\sim\Lambda} |\boldsymbol{\omega}^\top \boldsymbol{a}| < \infty, \tag{4.27}$$

- *the related RPF* $\overline{\Psi}_f := \frac{1}{F_1}\Psi_f$*, with* Ψ_f *defined in* (4.18)*, and* f *a periodic, mean Lipschitz smooth function with constant* $L^\mu_f < \infty$ *(see Def. 3.13), FS coefficients* $\{F_k\}_{k\in\mathbb{N}}$*, and* $C_f := (1 + \|f\|_\infty/|F_1|) < \infty$*.*

Given some $0 \leq \zeta < 1$ *and* $\epsilon_0 > 0$*, assume that:*

(i) $\mathcal{G}, \widehat{\mathcal{G}} \subset \mathcal{M}_{\Sigma,\zeta}$ *(Assumption 4.11);*

(ii) the sketch dimension m *satisfies*

$$m \geq 128 \cdot \epsilon_0^{-2} \cdot \mathcal{H}_{\epsilon_0/c_f}(\Sigma)$$

with constant $c_f := 4C_\Lambda(4 + L^\mu_f/|F_1|)$*.*

Then, with probability exceeding $1 - 3\exp(-m\epsilon_0^2/64)$ *on the draw of* $\boldsymbol{\Omega}$ *and* $\boldsymbol{\xi}$*, the normalized RPF sketch operator* $\mathcal{A}_{\overline{\Psi}_f}$ *has the LPD over* $\widehat{\mathcal{G}}$ *with respect to* $\mathcal{A}_{\Phi_{\mathrm{RFF}}}$*, with error*

$$\epsilon = C_f(\epsilon_0 + 3\zeta).$$

Proof. Let us define the periodic function $\widehat{f}(t) = \frac{1}{D}\left(\frac{1}{F_1}f(t) - \exp(\mathrm{i}t)\right)$, with $D := \max(1, \|\frac{1}{F_1}f(\cdot) - \exp(\mathrm{i}\cdot)\|_\infty)$. Since $\|\exp(\mathrm{i}\cdot)\|_\infty \leq 1$ and $\|\widehat{f}\|_\infty \leq 1$, Corollary 3.17 ensures that, for any $\epsilon_0 > 0$, if $m \geq 128 \cdot \epsilon_0^{-2} \cdot \mathcal{H}_{\epsilon_0/c}(\Sigma)$, then, with probability at least $1 - 3\exp(-m\epsilon_0^2/64)$, we have for all $\boldsymbol{x}, \boldsymbol{y} \in \Sigma$

$$\tfrac{1}{m}\left|\langle \widehat{f}(\boldsymbol{\Omega}^\top \boldsymbol{x} + \boldsymbol{\xi}), \exp(\mathrm{i}(\boldsymbol{\Omega}^\top \boldsymbol{y} + \boldsymbol{\xi}))\rangle\right| \leq \epsilon_0, \tag{4.28}$$

where the constant c in the metric entropy radius is

$$c = 4C_\Lambda(L^\mu_{\widehat{f}} + L^\mu_{\exp(\mathrm{i}\cdot)} + 2\min(L^\mu_{\widehat{f}}, L^\mu_{\exp(\mathrm{i}\cdot)})).$$

By definition of $\widehat{f}$, Φ_{RFF}, and $\overline{\Psi}_f$, the bound (4.28) is equivalent to the sLPD property (4.26) with error $\epsilon_s = D\epsilon_0$.

The mean Lipschitz constant of the relevant functions reads $L^\mu_{\exp(\mathrm{i}\cdot)} \leq 1$ and $L^\mu_{\widehat{f}} \leq D^{-1}(\frac{1}{|F_1|}L^\mu_f + L^\mu_{\exp(\mathrm{i}\cdot)})$, and we further simplify the bound using $\min(L^\mu_{\widehat{f}}, L^\mu_{\exp(\mathrm{i}\cdot)}) \leq L^\mu_{\exp(\mathrm{i}\cdot)}$. Since $D \geq 1$, we thus get that c is upper bounded by

$$4C_\Lambda\left(3 + D^{-1}(1 + \tfrac{1}{|F_1|}L^\mu_f)\right) \leq c_f := 4C_\Lambda(4 + \tfrac{1}{|F_1|}L^\mu_f).$$

We can now turn the sLPD (4.28) into the LPD (with a proper rescaling of the error) by applying Lemma 4.13. We note that $C_{\Phi_{\mathrm{RFF}}} \leq \|\exp(\mathrm{i}\cdot)\|_\infty =$

1 and $C_{\overline{\Psi}_f} \leq \|f/F_1\|_\infty = \|f\|_\infty/|F_1|$, which implies that $3C_{\Phi_{\mathrm{RFF}}}(C_{\Phi_{\mathrm{RFF}}} + C_{\overline{\Psi}_f}) \leq 3(1 + \|f\|_\infty/|F_1|) = 3C_f$. Finally, since $D \leq (1 + \|f\|_\infty/F_1) = C_f$, Lemma 4.13 shows that the desired LPD holds with error $C_f(\epsilon_0 + 3\zeta)$.

□

ACL guarantees for quantized or modulo contributions

To formulate our final guarantees and use Lemma 4.12, we need this bound on the Kolmogorov entropy of $\Sigma_{\widetilde{l},\widetilde{u}} \supset \Sigma_{l,u}$.

Lemma 4.15. *In the notations of Lemma 4.12, we have*

$$\mathcal{H}_\nu(\Sigma_{\widetilde{l},\widetilde{u}}) \leq d \log\left(1 + \tfrac{\sqrt{d}(2\rho S + \|u-l\|_\infty)}{\nu}\right). \tag{4.29}$$

Proof. Since $\Sigma_{\widetilde{l},\widetilde{u}} \subset c + (\rho S + r)\mathbb{B}_\infty^d$, with $2c = l + u$ and $2r = \|u - l\|_\infty$, the entropy of $\Sigma_{\widetilde{l},\widetilde{u}}$ is bounded by the one of $\mathcal{B} := (\rho S + r)\mathbb{B}_\infty^d$. From Lemma 4.10 in [Pis99], given $\nu' > 0$, there exists a ν'-covering $\mathcal{S}$ of $\mathcal{B}$ in the ℓ_∞-metric—*i.e.*, for all $x \in \mathcal{B}$, there is one $q \in \mathcal{S}$ such that $\|x - q\|_\infty \leq \nu'$—where $|\mathcal{S}| \leq (1 + \frac{2(\rho S + r)}{\nu'})^d$. Since $\|x - q\|_\infty \geq \|x - q\|_2/\sqrt{d}$, $\mathcal{S}$ is also a (ℓ_2) covering of $\mathcal{B}$ with radius $\nu'\sqrt{d}$. Taking $\nu' = \nu/\sqrt{d}$ shows that the Kolmogorov ν entropy of $\mathcal{B}$, and thus that of $\Sigma_{\widetilde{l},\widetilde{u}}$, is bounded as in (4.29).

□

We can finally combine all our results together to obtain statistical learning guarantees (excess risk bounds) for the ACL problem with RPF sketches (and in particular, for quantized or modulo RFF), when solving the tasks of k-means and GMM specifically, under the assumption that the data is constrained in the box $\Sigma_{l,u}$.

Corollary 4.16. *In the notations of Prop. 4.14, consider $\widehat{\theta}'$ the solution to the ACL problem (4.17), where the sketch is obtained by the normalized random periodic features $\overline{\Psi}_f$, and the reference sketch map is Φ_{RFF}.*

We assume that, (i) *all the data samples lie in a box $\Sigma_{l,u}$, and the known upper and lower bounds l and u are used to restrict the optimization procedure (for GMM, a constant S is also used to upper bound the variance of the modes), and* (ii)*, the LRIP holds for Φ_{RFF} on the chosen task with constant γ.*

Then, given $\epsilon_0 > 0$, we have the following guarantees with probability at least $1 - 3\exp(-m\epsilon_0^2/64)$:

- *[for **k-means**] If the sketch size is at least*

$$m \geq 128 \cdot \epsilon_0^{-2} \cdot d \log\left(1 + \frac{c_f \sqrt{d} \|\boldsymbol{u} - \boldsymbol{l}\|_\infty}{\epsilon_0}\right),$$

then, for $C_{\mathrm{km}} := 4\gamma\sqrt{C_f}$*, the excess risk is bounded by*

$$\mathcal{R}(\widehat{\boldsymbol{\theta}}'; \mathcal{P}_0) - \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0) \leq \eta + C_{\mathrm{km}}\sqrt{\epsilon_0},$$

with η *the excess risk in symmetric CL* (4.16).

- *[for **GMM**] For some* $\rho > d$*, if the sketch size is at least,*

$$m \geq 128 \cdot \epsilon_0^{-2} \cdot d \log\left(1 + \frac{c_f \sqrt{d}(2\rho S + \|\boldsymbol{u} - \boldsymbol{l}\|_\infty)}{\epsilon_0}\right),$$

then the excess risk is bounded by

$$\mathcal{R}(\widehat{\boldsymbol{\theta}}'; \mathcal{P}_0) - \mathcal{R}(\boldsymbol{\theta}^*; \mathcal{P}_0) \leq \eta + C_{\mathrm{km}}\sqrt{\epsilon_0} + C_{\mathrm{gmm}} \cdot e^{-\rho^2/4},$$

with $C_{\mathrm{gmm}} := 4\gamma\left(\frac{9C_f^2}{2\pi}\right)^{\frac{1}{4}}$*, and* η *and* C_{km} *as above.*

Proof. The proof consists in applying Prop. 4.10 and Prop. 4.14, combined with Lem. 4.12 and Lem. 4.15, the entropy of $\Sigma_{\boldsymbol{l},\boldsymbol{u}}$ being found by setting $\rho = 0$ in (4.29). □

A few remarks can be made about this corollary. First, we rely on the fact that Φ_{RFF} satisfies the LRIP; actually, ensuring that this holds (with high probability on the draw of $\boldsymbol{\Omega}$) imposes additional constraints on m. They depend on the considered task and the complexity of the related model set $\mathcal{G}$; for example, for a GMM with K modes in $\mathbb{R}^d$, we should have $m = \Omega(Kd)$, up to some additional factors and restrictions on $\mathcal{G}$ (see [GCK+20, Sec. 5.5] and [GBKT20]). Second, the choice of the parameter ρ necessitates solving a trade-off: increasing ρ decreases the excess risk bound (excess risk proportional to $e^{-\rho^2/4}$), but at the cost of logarithmically increasing the required sketch size m.

Finally, Cor. 4.16 allows us to determine which between quantized or modulo RPF requires more measurements. Indeed, referring to Appendix A for the relevant constants, we compute that

$$C_q = (1 + \|q\|_\infty/|Q_1|) = 1 + \frac{\pi}{2\sqrt{2}} < C_{\mathrm{mod}} = 1 + \frac{\sqrt{5}\pi}{4},$$

as well as

$$c_q = 4C_\Lambda(4 + L_q^\mu/|Q_1|) = 24C_\Lambda < c_{\text{mod}} = (24 + 2\sqrt{2})C_\Lambda.$$

Therefore, both for k-means and GMM, the sample complexities and the excess risk bounds of Cor. 4.16 shows that, while requiring a higher number of measurements, ACL with modulo features gets a higher bound on the excess risk compared to that of a quantized sketch. Thus, the sketch size must be further increased (to allow a smaller ϵ_0) when using modulo sketches in order to meet quantized sketch performance[8]. We observe this effect experimentally in the following section.

4.5 Experiments

In this section we validate the ACL approach through various practical numerical experiments. In a series of synthetic data experiments, we first cover compressive k-means with quantized contributions Ψ_q (which we call `QCKM`), for which we closely analyze the required sketch size compared to the full-precision `CKM`. We then extend our results to the ACL scheme with the modulo feature map Ψ_{mod}, for the GMM task, and explore the impact of the dataset size. We then turn to large-scale real-life datasets, and apply ACL to spectral clustering of handwritten digits, sensor readings, and audio feature extraction for event recognition.

Remember that the cost $\mathcal{C}_\Phi(\boldsymbol{\theta}; \boldsymbol{z})$ is not convex, and that the symmetric and asymmetric compressive learning problems, described respectively by (4.13) and (4.17), cannot be solved exactly in practice. To approximate those solutions, we thus mainly use the `CLOMP` greedy algorithm [KBGP18]; the exception is the last experiment which uses the Gaussian splitting algorithm (algorithm 2 in [KBGP18]) for better performance when the number of Gaussians K is large.

Remark: Since the cost $\mathcal{C}_\Phi(\boldsymbol{\theta}; \boldsymbol{z})$ behaves in the same manner with respect to $\boldsymbol{\theta}$ in the ACL case ($\boldsymbol{z} = \boldsymbol{z}_{\Psi,\mathcal{X}}$) as in the usual symmectric CL case ($\boldsymbol{z} = \boldsymbol{z}_{\Phi,\mathcal{X}}$), we can use the exact same algorithms in both scenarii.

Implementation details and metrics We first consider a controlled environment, where we generate a synthetic dataset $\mathcal{X}$ according to a known "ground-truth" Gaussian mixture model $\mathcal{P}_0$. To evaluate the quality of a

[8] This could be intuitively expected given our "signal-level" observations in Section 3.7.

CL solution $\widehat{\boldsymbol{\theta}}$, we use the *empirical excess risk*,

$$\Delta\mathcal{R}(\widehat{\boldsymbol{\theta}}) := \mathcal{R}(\widehat{\boldsymbol{\theta}}; \widehat{\mathcal{P}}_{\mathcal{X}}) - \mathcal{R}(\widetilde{\boldsymbol{\theta}}; \widehat{\mathcal{P}}_{\mathcal{X}}), \tag{4.30}$$

where the empirical risk minimizer $\widetilde{\boldsymbol{\theta}}$ is estimated by keeping the best out of several independent trials of traditional ML algorithms operating on the full dataset: the `k-means++` algorithm [Llo82, AV07] for k-means clustering, and Expectation-Maximization [Moo96] for GMM. Referring to Table 4.2, for k-means the empirical excess risk corresponds to the excess SSE (4.7), $\Delta\mathcal{R}(\widehat{\boldsymbol{\theta}}) = \mathrm{SSE}(\widehat{\boldsymbol{\theta}}; \mathcal{X}) - \mathrm{SSE}(\widetilde{\boldsymbol{\theta}}; \mathcal{X})$, while for GMM, $\Delta\mathcal{R}(\widehat{\boldsymbol{\theta}}) = \mathrm{LL}(\widetilde{\boldsymbol{\theta}}; \mathcal{X}) - \mathrm{LL}(\widehat{\boldsymbol{\theta}}; \mathcal{X})$ is the excess negative log-likelihood (4.8). Because the performance depend on the random draw of $\boldsymbol{\Omega}$ and $\boldsymbol{\xi}$, we perform several independent trials of (A)CL and report the median performance.

The numerical value of the excess risk, while relevant to our theoretical guarantees, is not always easy to interpret. Therefore, another metric we use to assess the quality of solutions $\widehat{\boldsymbol{\theta}}$ is the *success rate*: the average number of "successes" obtained over all trials. For our purposes, we arbitrarily define the "success" of solution $\widehat{\boldsymbol{\theta}}$ as follows: when we solve k-means, $\widehat{\boldsymbol{\theta}}$ succeeds if $\mathrm{SSE}(\widehat{\boldsymbol{\theta}}; \mathcal{X}) \leq 1.2 \times \mathrm{SSE}(\widetilde{\boldsymbol{\theta}}; \mathcal{X})$; when we solve GMM, $\widehat{\boldsymbol{\theta}}$ succeeds if $\mathrm{LL}(\widehat{\boldsymbol{\theta}}; \mathcal{X}) \geq \frac{\mathrm{LL}(\widetilde{\boldsymbol{\theta}}; \mathcal{X})}{1.2}$ (where we ensure $\mathrm{LL}(\widetilde{\boldsymbol{\theta}}; \mathcal{X}) > 0$).

4.5.1 Synthetic data: Quantized Compressive K-Means

We first empirically verify that quantized compressive k-means (`QCKM`) requires only $m = \mathcal{O}(dK)$ measurements to find good centroids, with a hidden multiplicative constant only slightly higher (15 to 25%) than for (full-precision) compressive k-means (`CKM`)—remembering that `QCKM` receives m-bit sketch contributions whereas `CKM` uses full-precision contributions.

To do so, we compute *phase transition diagrams* (Fig. 4.4) to highlight the relationship between the required amount of measurements m, and the ambient space dimension d or the number of clusters K. These diagrams show how the empirical success rate (as defined above, averaged over 100 trials) of `QCKM` evolves with m, as d or K varies.

First, we draw $n = 10^4$ samples uniformly from $\mathcal{P}_0$ a mixture of $K = 2$ isotropic Gaussians in *varying dimension* d, with means $\pm(1, \cdots, 1)^\top \in \mathbb{R}^d$ and covariance matrix $\frac{d}{20}\boldsymbol{I}_d$. The phase transition diagram is reported Fig. 4.4a, along with lines showing the transition to more than 50% success rate of `QCKM` (red solid) and, for comparison, of `CKM` (yellow dotted). This transition happens (except for a deviation at small dimensions) at a

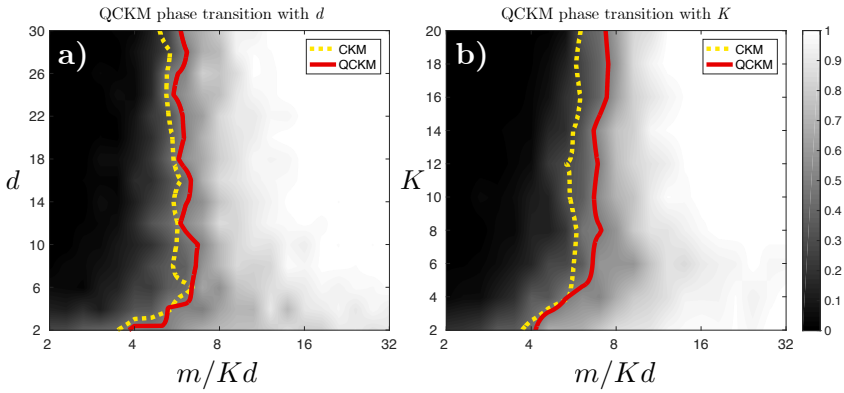


Fig. 4.4 Empirical success rate—from 0% (black) to 100% (white)—evolution of QCKM with m/nK, and **(a)** n, or **(b)** K. The red solid line shows the transition to a success rate above 50% for QCKM; for comparison, the yellow dotted line shows the same transition for CKM.

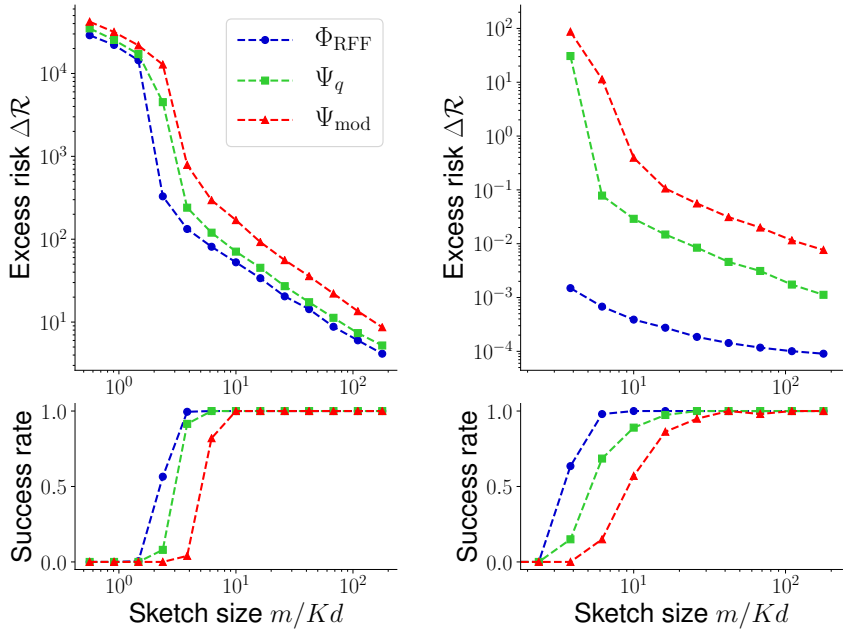
constant value of m/dK: as CKM, QCKM requires m to be proportional to d. In this experiment, QCKM requires about 1.13 more measurements than CKMs (complex and full precision) measurements.

Fig. 4.4b is the phase transition for *varying numbers of centroids* K while fixing $d = 5$. Samples are drawn from K Gaussians with means chosen randomly in $\{\pm 1\}^d$, other parameters being identical to the previous experiment. Successful estimation occurs when m scales linearly with K, with a factor of about 1.23 between QCKM and CKM sample complexities. These experiments suggest that CKMs empirical rule $m = \mathcal{O}(dK)$ holds for QCKM, with a slightly higher multiplicative constant.

4.5.2 Synthetic data: quantized versus modulo sketch contributions

For this next experiment, $\mathcal{P}_0$ is a mixture of $K = 10$ Gaussian modes in dimension $d = 5$, from which we draw a dataset $\mathcal{X}$ of $n = 10^5$ samples. We then sketch this dataset, using the standard random Fourier features Φ_{RFF}, but also the quantized RFF Ψ_q and the modulo feature map Ψ_{mod}, and solve both k-means and GMM from those sketches[9]. We draw a varying amount m of random frequencies $\boldsymbol{\omega}_j \sim_{\text{i.i.d.}} \Lambda$ from Λ given by the "Folded Gaus-

[9] The fully symmetric CL case, where Φ_{RFF} is used for sketching, does not require the dither $\boldsymbol{\xi}$, so we impose $\boldsymbol{\xi} = \mathbf{0}$ in that case. In the asymmetric case, recall we moreover perform a normalization $\boldsymbol{z}_{\Psi_f,\mathcal{X}} = \frac{1}{F_1}\boldsymbol{z}_{\Psi_f,\mathcal{X}}$ before learning.

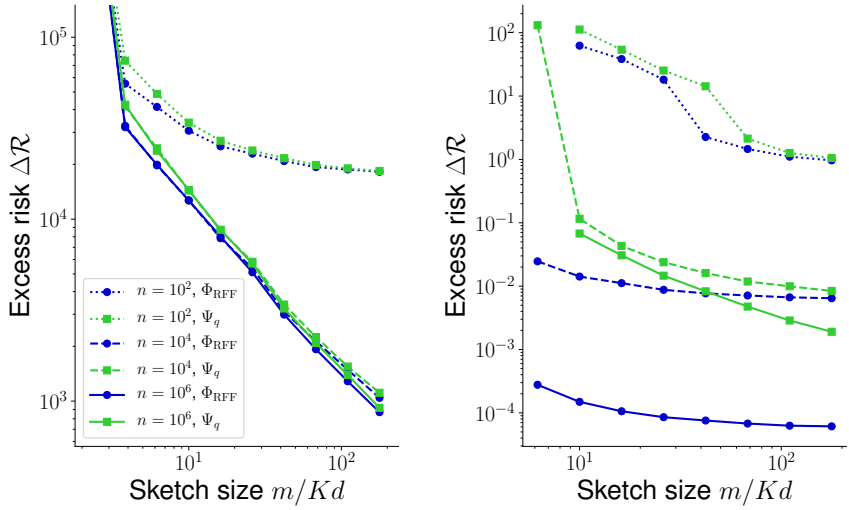


(a) k-means **(b)** GMM

Fig. 4.5 *Top*: empirical excess risk (4.30) for the k-means (*left*) and GMM (*right*) tasks, as a function of sketch size, obtained by the following (A)CL strategies: usual symmetric CL with $\Psi = \Phi = \Phi_{\text{RFF}}$ (blue circles), asymmetric CL with quantized sketch contributions $\Psi = \Psi_q$ (green squares), and modulo sketch contributions $\Psi = \Psi_{\text{mod}}$ (red triangles). Each data point is the median out of 25 or more independent trials. *Bottom:* the associated success rate.

sian" heuristic described in [KBGP18], with scale $\sigma^2 = \frac{1}{10d}$ for k-means and $\sigma^2 = \frac{1}{100d}$ for GMM. Compared to a Gaussian distribution, this folded variant improves the sampling of low frequencies.

The results are shown Fig. 4.5. From Fig. 4.5a, as we already observed in the previous experiment, in the case of k-means one can use quantized sketch contributions with only a minor performance decrease (or, equivalently, a slight increase of the sketch size reaches the same performance). Moreover, ACL with modulo sketch contributions is also successful, but the sketch size (to reach a given performance level) must be larger than in the quantized case. This is indeed what would be expected from our



(a) k-means **(b)** GMM

Fig. 4.6 Empirical excess on a synthetic dataset of total size $\widetilde{n} = 10^7$ of solutions obtained through ACL where one sketches a subset of the dataset with varying sizes n (dotted, dashed, plain lines for $n = 10^2, 10^4, 10^6$ respectively), using the full-precision sketch (blue circles) or the quantized sketch (green squares), as a function of the sketch size m.

theoretical results, as explained at the end of Sec. 4.4.2.

From Fig. 4.5b, we can observe that quantized or modulo ACL is also applicable to the task of GMM. As suggested by Cor. 4.16, the required sketch size (to reach equivalent performance) increases more for this task than for k-means.

4.5.3 Synthetic data: influence of the dataset size

Next, we study the role of the dataset size in the ACL scheme (focusing on quantized ACL). Unless explicitly mentioned below, all parameters are identical to the previous experiment. We first generate a "full-size" dataset $\widetilde{\mathcal{X}}$ (with size $\widetilde{n} = 10^7$) from the previous GMM $\mathcal{P}_0$. For each trial, we use a smaller dataset $\mathcal{X}$ for compressive learning obtained by picking uniformly at random a subset of n samples (without replacement) in $\widetilde{\mathcal{X}}$. Here, the empirical excess risk $\Delta\mathcal{R}$ is evaluated using the full dataset $\widetilde{\mathcal{X}}$ (the ERM minimizer $\widetilde{\boldsymbol{\theta}}$ being also learned on this full dataset).

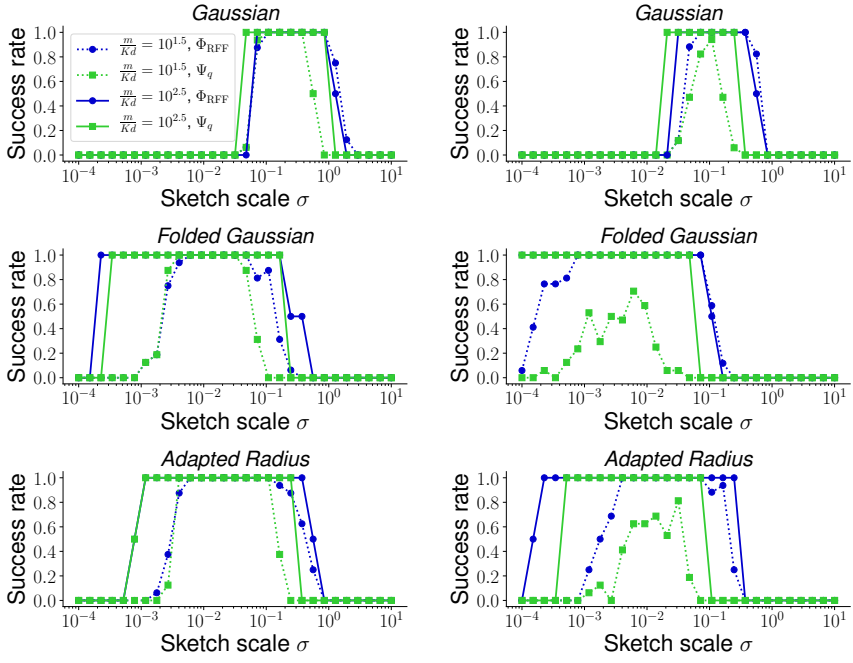
The results are shown Fig. 4.6. As can be observed from Fig. 4.6a, having a larger amount of samples n improves the performance (at constant sketch size m), as could be expected. In terms of excess risk guarantees, this can be related to the sampling term $\gamma\|\mathcal{A}_\Phi(\mathcal{P}_0) - \mathcal{A}_\Phi(\widehat{\mathcal{P}}_\mathcal{X})\|_2$ in (4.24). When m increases, the excess risk quickly saturates on smaller datasets, but in the two larger-size datasets ($n = 10^4$ and 10^6) this is not the case. An interesting phenomenon can be observed by looking (very) closely at those last two curves: there is a "crossing" between $m/Kd = 10$ and $m/Kd = 100$. When the sketch size is small, the curves are grouped by sketching feature map Ψ (*i.e.*, by color); the dominant effect on the excess risk is whether the quantized sketch is used or not, regardless of the dataset size, which in our theoretical results can be associated with the LPD error term ϵ. But as the sketch size increases, the curves are grouped by dataset size instead (*i.e.*, the plain and dashed curves go together); the dominant effect is now the sampling error. This can be explained by our theory through the fact that the LPD error ϵ decreases with[10] m. Similar conclusions can be drawn for GMM modeling (Fig. 4.6b), albeit with a more significant impact related to the dataset size, which makes the crossing described before easier to observe.

4.5.4 Synthetic data: sensitivity of the sketch parameters*

As a last synthetic experiment, we visualize how quantizing the sketch contributions impacts the choice of the sketching parameters, *i.e.*, the sampling pattern and scale σ generating the random projection $\mathbf{\Omega}$. Here, the dataset generated by sampling $n = 10^6$ points from $\mathcal{P}_0$, a mixture of $K = 5$ Gaussians in dimension $d = 10$. We generate the random projection $\mathbf{\Omega}$ (used to define both Ψ and Φ) according to the various heuristics introduced in [KBGP18], namely the Gaussian (G), Folded Gaussian (FG) and Adapted Radius (AR) sampling strategies (we refer the reader to Section 2.5 for their mathematical descriptions). Moreover, we let the sketch scale parameter σ (*i.e.*, the scale of the associated kernel; decreasing σ increases the "resolution" of the sketch operator) vary, and check how the asymmetric (quantized) and symmetric CL schemes compare.

The results are shown Fig. 4.7. As could be expected, the range of acceptable sketch scales σ increase when the sketch size m increases. However, the crucial observation here is that in some cases, the choice of sketch

[10] Note that the LRIP constant γ also should decrease with m, but this same constant appears in all the terms of the excess risk, so this effect should impact all the curves in the same way.



(a) k-means **(b)** GMM

Fig. 4.7 Success rate of k-means (left) and GMM (right) as a function of the sketch scale σ for the three considered drawing strategies: Gaussian (top row), Folded Gaussian (middle row), Adapted Radius (bottom row). The sketch size is either medium $m = 10^{1.5}Kd$ (dotted) or large $m = 10^{2.5}Kd$ (plain).

scale σ can be much more difficult when the contributions are quantized[11]; this is especially striking for the GMM task. The main takeaway is that, to use quantized sketch contributions, special care should be put in the choice of the scale parameter (which is already a difficult problem we'll come back to in a later chapter); otherwise a price in terms of increased sketch size could have to be paid.

4.5.5 Real data: spectral clustering of MNIST digits

We now validate the ACL scheme on "real" (*i.e.*, non-Gaussian) data. We first consider k-means in the context of a spectral clustering [VL07] pipeline of the MNIST dataset (70000 28×28 pixel images of handwritten digits [LCB]). This experiment aims at detecting, in an unsupervised setting, the 10 clusters corresponding to the digits $0-9$ from their representation in a 10-dimensional feature space[12]. We run the compressive clustering algorithms with $m = 1000$ frequencies. To avoid bad local minima, several replicates of `k-means` are usually run and the solution that achieves the best SSE is then selected. We thus also perform several replicates of the compressive clustering variants, but since computing the SSE requires access to whole dataset (which is not supposed available to the compressive algorithms), we select the solution of that minimizes the related sketch matching objective; see [KTTG17].

We use two performance metrics to assess the clustering algorithms: the SSE (4.7), *i.e.*, the k-means objective function, and the Adjusted Rand Index (ARI) [VEB10] that compares the clusters (and not the centroids) produced by the different algorithms with the ground truth digits. A higher ARI means the *clusters* are closer to the ground truth, with ARI $= 1$ if the partitions are identical and ARI $= 0$ (on average) if the clusters are assigned at random.

Fig. 4.8 reports the mean and standard deviation (excluding a few clear outliers for `CKM` and `QCKM`, occuring about 5% of the time on average) obtained for both performance metrics. Globally, `QCKM` performs similarly to `CKM`, retaining its advantages over `k-means`. First, the compressive learning algorithms are more stable: their performance exhibit small variance, in contrast with `k-means` that hence benefits the most from several replicates. In addition, while for several replicates `k-means` outperforms the compres-

[11] This could be interpreted as a consequence of the fact that quantized CL requires a larger sketch size; in any case, this observation is not especially surprising, but still worth mentioning explicitly.

[12] We thank the authors of [KTTG17] for having shared this spectral clustering dataset.

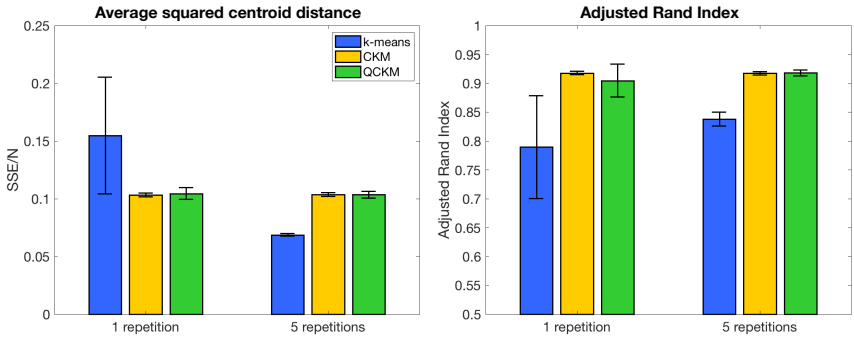


Fig. 4.8 Mean with standard deviation over 100 experiments of the performance (SSE/n and ARI on left and right, respectively) of the different compared clustering algorithms—`k-means` in blue, full-precision compressive k-means (`CKM`) in yellow, quantized compressive learning (`QCKM`) in green—both for 1 and 5 repetitions of each learning algorithm.

sive approaches in terms of SSE, the solutions of `(Q)CKM` are closer (as quantified by the ARI score) to the ground truth labels: this suggests that the sketch matching objectives (4.13) or (4.17) are better suited than the SSE for (at least) this task. Note that `QCKM` performance have moderately higher variance than those of `CKM`: this is probably due to the increased measurement rate of `QCKM` required to reach similar performance (as suggested by the first experiments) while here both algorithms ran with $m = 1000$.

4.5.6 Real data: density fitting on Intel Lab sensor reading*

We now consider the GMM task. As illustration of our motivating example in Fig. 4.1, we consider a scenario where quantized sketch contributions could be especially beneficial: a sensor network where the sensor nodes transmit the sketch contributions to a central compute unit. We use the *Intel Lab* dataset [BHG+04] gathering $n = 2.3 \times 10^6$ readings from a network of 54 sensors. We focus on $d = 5$ attributes (time of the day, temperature, humidity, light and voltage) which were normalized between 0 and 1 after clipping outliers. In our scenario, each sensor frequently—twice per minute—obtains measurements $\boldsymbol{x}_i \in \mathbb{R}^5$, and computes the related sketch contribution ($\Phi_{\text{RFF}}(\boldsymbol{x}_i)$ or $\Psi_q(\boldsymbol{x}_i)$) locally. Those contributions are then sent wirelessly to a central server which averages them all to construct the overall sketch: each sensor thus transmits messages of either $b = 128m$ bits for the full-precision case $\Phi_{\text{RFF}}(\boldsymbol{x}_i)$ (assuming 64-bit floating-point precision),

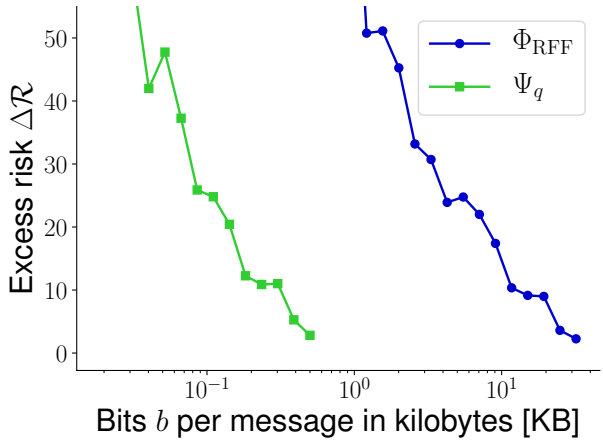


Fig. 4.9 Excess risk (log-likelihood deficit compared to ground-truth) versus number of bits per contribution, for full-precision (blue) or quantized (green) sketch contribution, on the Intel Lab dataset.

or $b = 2m$ bits for the quantized contributions $\Psi_q(\boldsymbol{x}_i)$. The central server then learns a GMM from the resulting sketch, *e.g.*, to discover fundamental operating modes.

The resulting excess risk as a function of the required bitrate is shown Fig. 4.9. We can see that the same performance can be reached with a much smaller bandwidth usage thanks to quantized CL. Note that we did not target a particular downstream application, but assumed one is interested in a general modeling of the distribution of the sensor's readings. In the next experiment we tackle a more task-oriented application.

4.5.7 Real data: feature extraction for ESC-50 audio clips classification

As large-scale proof-of-concept, we tackle an audio event classification task, where (A)CL is used to alleviate the computational cost of learning a GMM in a feature extraction phase. Note that our goal is not to propose a particularly competitive audio classification scheme, but to compare the ACL strategy to symmetric CL on large-scale, realistic data.

Our scheme follows the "alpha features" strategy described in [KR16]. We use the ESC-50 dataset [Pic15], which contains $J = 2500$ audio clips lasting 5s, each associated to one of $C = 50$ classes (*e.g.*, animals, water sounds, urban noises). We assume that the J audio clips $\boldsymbol{s}^{(j)}$ are distributed

across a sensor network, which perform local preprocessing as follows. For each audio clip $\boldsymbol{s}^{(j)}$, we extract[13] Mel Frequency Cepstral Coefficients (MFCC), using $d = 10$ frequency bands, 30 ms-long time intervals, and 15 ms-long hops. We then take the (distorted) features $\Psi(\boldsymbol{x}_i^{(j)})$ (with Ψ to be specified) of each of the $N = \lceil \frac{5\,\mathrm{s}}{15\,\mathrm{ms}} \rceil = 334$ resulting MFCC vectors $\boldsymbol{x}_i^{(j)} \in \mathbb{R}^d$. Gathering features from the J audio clips, the $n = JN$ resulting contributions $\Psi(\boldsymbol{x}_i^{(j)})$, each encoded by b bits (more on this below), are aggregated, by a central server, into one sketch vector $\boldsymbol{z}_{\mathcal{X},\Psi}$. A GMM of $K = 32$ modes, describing the distribution of all clips in the frequency domain, is then extracted from this sketch by (A)CL. Each audioclip $\boldsymbol{s}^{(j)}$ can then by summarized by its "alpha features" $\boldsymbol{\alpha}^{(j)} \in \mathbb{R}^K$, defined by the average soft assignement of that clip's MFCC vectors to each of the K Gaussian modes, *i.e.*,

$$\alpha_k^{(j)} := \frac{1}{N} \sum_{i=1}^{N} \frac{w_k p_{\mathcal{N}}(\boldsymbol{x}_i^{(j)}; \boldsymbol{\mu}_k, \boldsymbol{\Gamma}_k)}{\sum_{k=1}^{K} w_k p_{\mathcal{N}}(\boldsymbol{x}_i^{(j)}; \boldsymbol{\mu}_k, \boldsymbol{\Gamma}_k)}.$$

Finally, a SVM (see Section 2.1.1) is learned on the alpha features to classify the J audio samples; see [KR16] for additional details.

We train this classification scheme and evaluate it on a separate test set (20% of the full dataset), for various values of the sketch dimension m. Assuming a scenario where minimizing the transmission cost is crucial, we compare the performance at given values of the number of bits b sent per "message", *i.e.*, per featurized MFCC vector $\Psi(\boldsymbol{x}_i)$. For usual full-precision RFF $\Psi = \Phi_{\mathrm{RFF}}$, we assume the real and imaginary numbers are encoded by 64 bits, *i.e.*, $b = 128m$. For the quantized RFF $\Psi = \Psi_q$ we have, by construction, $b = 2m$. As baseline, we also report the accuracy when Expectation-Maximization (EM) is performed on the whole dataset to learn the GMM ("un-compressed learning").

The results are shown Fig. 4.10. In this specific scenario, the ACL scheme with quantized contributions is particularly advantageous over the full-precision symmetric CL scheme, as *the same performance can be reached with a bitrate reduction by a factor of at least* 30. Moreover, when the sketch size is large enough, both compressive learning approaches are competitive with the EM baseline, which requires several passes over the entire database $\mathcal{X}$. Of course, this doesn't mean CL is necessarily the best candidate for the scenario described here, as other approaches using *e.g.*, distributed learn-

[13] The MFCC extraction used the `librosa` [MLM+20] package. The subsequent SVM model is trained with `scikit-learn` [PVG+11].

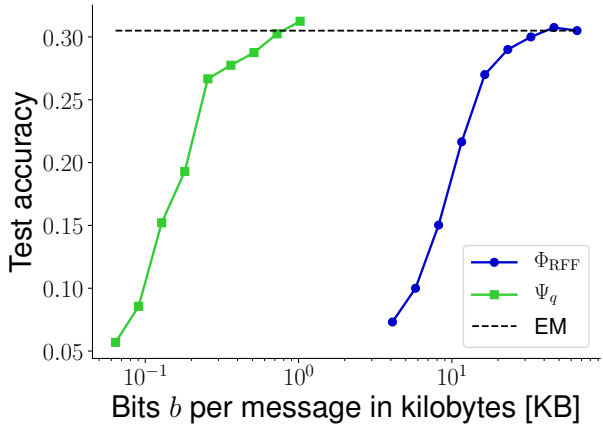


Fig. 4.10 Test accuracy of the GMM-based audio classification procedure described in [KR16], as versus number of bits per contribution, for full-precision (blue) or quantized (green) sketch contribution, and with plain Expectation-Maximization (dashed black) on the ESC-50 dataset.

ing could be considered; recall that we focus here on the comparison between symmetric and asymmetric CL.

4.6 Conclusion

Motivated by the benefits promised by quantization of the sketch contributions, in this chapter we defined the asymmetric compressive learning (ACL) scheme and formally established excess risk bounds for it. To achieve this, we introduced a specific LPD property—telling us "how far" a distorted feature map is from an undistorted one—that combines with the classical LRIP of compressive learning to explain the ACL performance. Our second key contribution was to apply this result (*i.e.*, proving the LPD property) to the specific case of random periodic features, which covers quantized (and modulo) sketch contributions. We then further validated this strategy with a series of numerical simulations.

However, our contribution focused on deriving an excess risk bound without particular care for its tightness. In particular, using Lemma 4.13 to prove the LPD requires to have a sketch size m scaling with the complexity (Kolmogorov entropy) of a signal set Σ, which is not a required ingredient in previous (symmetric) CL guarantees [GBKT17]; this could be subopti-

mal if Σ is large (in the Kolmogorov entropy sense); ideally, our results would depend on the complexity of the model set $\mathcal{G}$ rather than the signal space Σ.

Moreover, just as the existing CL guarantees, the excess risk bound is not readily exploitable in practice. For instance, the LRIP constant and the Kolmogorov entropy are hard to pin down accurately, and involve solving non-trivial trade-offs to be interpreted properly (*e.g.*, between the sketch size m, the probability of failures, the error contributions ϵ). As a last caveat, let us recall that the current compressive learning algorithms used in practice are heuristics that do not have convergence guarantees. Future work is thus needed to bridge the gap between the theoretical guarantees and empirical performance of compressive learning—symmetric or not.

Some additional perspectives/extensions are discussed in Chapter 7.

PART II
Private Sketches

5

Private Sketching

In an increasingly large amount of applications, data samples contain sensitive personal information (*e.g.*, when working with medical records, online surveys, or measurements coming from personal devices) and data providers ask that individuals' contributions to the dataset remain private. Learning from such data collections while protecting the privacy of individual contributors, *i.e.*, *privacy-preserving machine learning*, has thus become a crucial challenge [FWCY10, ARC19].

Intuitively, the goals of privacy-preserving machine learning and compressive learning (which represents the entire data distribution by a sketch vector of small size) seem to align: to extract the general statistics from the dataset, while somehow "forgetting" the individual data points as much as possible. This synergy lies at the core of this chapter, which presents a privacy-protecting sketching mechanism, able to at the same time compress the dataset and provide formal privacy-preserving guarantees for its contributors.

Specifically, we focus on Differential Privacy (DP), a particularly strong and popular mathematical formalism to study the privacy of algorithms, as explained in Section 5.1. We combine differential privacy with compressive learning into a private sketching mechanism which is described and theoretically studied in Section 5.2, and empirically validated against state-of-the-art approaches in Section 5.3. Further advantages and challenges of

private sketching are discussed in Section 5.4, before concluding in Section 5.5.

This work is the result of a collaboration between myself and (at the time) PhD students Antoine Chatalic (IRISA, Université de Rennes) and Florimond Houssiau (Imperial College London), and our respective supervisors: Laurent, Rémi Gribonval and Yves-Alexandre de Montjoye. We published several works on this line of work: preliminary results (the main private sketching mechanism and experiments on private k-means) where presented at ICASSP [SCH+19a] and (with some new variations) at the SPARS workshop [SCH+19b]. We then further extended the analysis of this scheme (among others, focusing on the sharpness of the guarantees) in a journal paper which was accepted at Information and Inference [CSH+21]. Most of this chapter is heavily inspired by those publications (and a few passages from their summary in [GCK+20]), with a specific focus on the aspects I contributed to.

Remark 5.1. In particular, the experimental results on private k-means (*i.e.*, Figures 5.2, 5.3 and 5.5) are to be credited to Antoine Chatalic. Some other contributions that were mainly his work (such as sharpness guarantees, the subsampling mechanism, the noise-to-signal ratio as proxy for utility, and private compressive PCA) were left out of the current chapter.

5.1 Preliminaries: notions of differential privacy

5.1.1 Motivation

The amount of data collected has increased exponentially over the last two decades. In parallel, data has become more fine-grained, from medical records to GPS traces with a temporal resolution on the scale of seconds. While this increased availability and precision of data have resulted in tremendous advances, they raise serious privacy concerns. Such datasets often contain highly detailed summaries of our lives, and are notoriously hard to anonymize (*data (pseudo-)anonymization* is removal of personally identifiable information from the dataset, which is the basis of GDPR regulations for example [Eur]). Indeed, individuals have been shown to be easily re-identifiable in large-scale behavioral datasets, such as mobile phone metadata [dMHVB13], credit card data [dMRS+15], web browsing behavior [BZH06] and movie watching history [NS08]. One can then argue that the dataset itself should thus not be disclosed. But publishing seemingly harmless quantities computed from it—*e.g.*, a machine learning model or

aggregate statistics—can still compromise the privacy of these users, even when these quantities result from aggregation over millions of data providers [DN03]. This motivates the need for defining (and enforcing) a strong mathematical quantification of privacy.

Differential privacy (DP) [Dwo08a] was introduced by Dwork et al. as a precise mathematical property of algorithms that protect the privacy of users in a dataset. As will become clearer below, it requires for a randomized algorithm's outputs to be distributed approximately identically whether any one individual is in the dataset or not (*i.e.*, the output depends negligibly on the presence or absence of any individual). The discrepancy between distributions is controlled by a parameter ϵ known as the *privacy budget*. DP is a particularly strong requirement, and is considered by many as the gold standard definition for privacy loss in aggregated data releases. It has been studied extensively in research and industry [EPK14,T+17], and in machine learning and signal processing in particular [SC13]. It been deployed by companies with large user bases, such as Google to measure changes in mobility patterns caused by confinement measures [ABC+20], LinkedIn to answer analytics queries [KT18] or Apple to estimate Emoji usage [TVV+17].

Alternative privacy definitions Although we focus on DP, many alternative definitions of privacy have been proposed in the literature [WE18]. Traditional statistical disclosure metrics, such as $k-$anonymity [Swe02], define anonymity as a property of the data, *e.g.*, requiring that each user is indistinguishable from $k-1$ others. However, as argued above, anonymizing large-scale high-dimensional data was shown to be hard, due to the high uniqueness of users in such datasets. Researchers have also proposed to make privacy a property of the algorithm, enforcing for instance that the mutual information leakage is bounded [DJW14]. Differential privacy is the most popular of such definitions, as it considers a worst-case adversary, and is hence "future-proof": no future release of auxiliary information can break the privacy guarantees. Connections between differential privacy and other information-theoretic definitions have also been investigated [WYZ16].

5.1.2 Differential Privacy

Randomness is an old tool for introducing uncertainty ("privacy by plausible deniability") when using sensitive information, e.g. implemented as

randomized response surveys [War65]. Differential privacy [Dwo08b] also relies on this tool, and provides a formal definition of the privacy guarantees offered by a *randomized* data release mechanism (*e.g.*, a machine learning algorithm) $\mathcal{M}$. Intuitively, a mechanism $\mathcal{M}$ provides differential privacy if its output does not depend significantly on the presence of any one user in the database, hence hiding this presence from an adversary.

More specifically, DP states that the distribution of a differentially private algorithm's output is similar for any two *neighboring* datasets—typically, if they differ by the addition or removal of any one record. The guarantees of DP are characterized by a privacy "budget" $\epsilon > 0$ which bounds the information disclosure from the dataset. Differential privacy is robust to many forms of attack, such as when the adversary can access side information that nullifies privacy guarantees based on anonymization or mutual information measures (*e.g.*, when the adversary can control some of the data vectors $\boldsymbol{x}_i$, or can access additional databases that are correlated with the primary database).

Definition of (pure) Differential Privacy Denote by $\mathbb{D} = \{\mathcal{X} \subset \mathbb{R}^d, |\mathcal{X}| < \infty\}$ the set of all possible datasets $\mathcal{X}$, equipped with a *neighboring relation* $\sim$. In this chapter[1], we consider that two datasets are neighbors if they differ by the addition or deletion by a single record, *i.e.*,

$$\mathcal{X} \sim \mathcal{X}' \quad \Leftrightarrow \quad \exists j \text{ s.t. } \mathcal{X}' = \mathcal{X} \cup \{\boldsymbol{x}_j\} \text{ or } \mathcal{X} = \mathcal{X}' \cup \{\boldsymbol{x}_j\}.$$

Denote by $\mathcal{S}$ the output space of the considered algorithm $\mathcal{M} : \mathbb{D} \to \mathcal{S}$.

Definition 5.2 (Differential Privacy)**.** Given a neighboring relation $\sim$ between datasets in $\mathbb{D}$, a randomized algorithm $\mathcal{M}$ is said to achieve differential privacy with privacy parameter ϵ (noted $\epsilon-$DP) if for any set of possible outcomes $S \subset \mathcal{S}$:

$$\forall \mathcal{X} \sim \mathcal{X}' \in \mathbb{D}, \ \mathbb{P}[\mathcal{M}(\mathcal{X}) \in S] \leq e^{\epsilon} \mathbb{P}[\mathcal{M}(\mathcal{X}') \in S]. \tag{5.1}$$

In particular, this implies that if the random variable $\mathcal{M}(\mathcal{X})$ (resp. $\mathcal{M}(\mathcal{X}')$) has a probability density function $p_{\mathcal{M}(\mathcal{X})}(\boldsymbol{s})$ (resp. $p_{\mathcal{M}(\mathcal{X}')}(\boldsymbol{s})$), then for all

[1]Other neighboring relations exist in the DP litterature. In [CSH+21], we also considered the "replacement" neighboring relation (which is called "Bounded" DP, as opposed to "Unbounded" DP considered in this chapter).

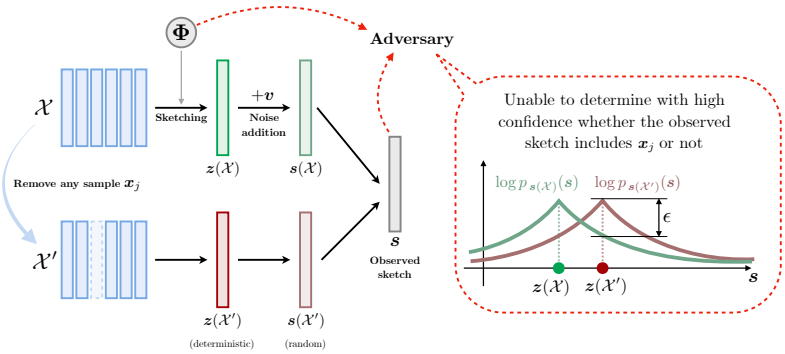


Fig. 5.1 Differential privacy for our private sketching mechanism (*i.e.*, $\mathcal{M}(\cdot) = s(\cdot)$ as described in (5.7)). Two neighboring datasets $\mathcal{X} \sim \mathcal{X}'$ (*e.g.*, with and without the inclusion of one individual record x_j) would produce distinct (deterministic) sketches $z(\mathcal{X})$ and $z(\mathcal{X}')$. To guarantee differential privacy, we thus add carefully calibrated noise on the sketch, which then gives two random quantities $s(\mathcal{X})$ and $s(\mathcal{X}')$. Differential privacy ensures that an adversary (that potentially has knowledge of the feature map Φ) cannot distinguishing between those $s(\mathcal{X})$ and $s(\mathcal{X}')$, in the sense that their likelihood ratio $\frac{p_{s(\mathcal{X})}(s)}{p_{s(\mathcal{X}')}(s)}$ is bounded by $\exp(\epsilon)$, *i.e.*, $\log(p_{s(\mathcal{X})}(s)) - \log(p_{s(\mathcal{X}')}(s)) \leq \epsilon$.

output values $s \in \mathcal{S}$

$$\forall \mathcal{X} \sim \mathcal{X}' \in \mathbb{D}, \ \exp(-\epsilon) \leq \frac{p_{\mathcal{M}(\mathcal{X})}(s)}{p_{\mathcal{M}(\mathcal{X}')}(s)} \leq \exp(\epsilon). \tag{5.2}$$

Intuitively, differential privacy ensures that if two datasets are neighbours, then their (random) output is almost indistinguishable, in the sense that any outcome has almost the same probability to be observed—up to an multiplicative term $\exp(\epsilon)$. The parameter $\epsilon > 0$, called the *privacy budget* or *privacy loss*, characterizes the strength of the privacy guarantee. As ϵ becomes smaller, then the output distribution for any two neighboring datasets are required to be closer together, and the privacy guarantee is thus stronger. Conversely, when ϵ increases, the privacy guarantee becomes weaker (when $\epsilon \to \infty$, the algorithm is nonprivate). This is represented Fig. 5.1.

Remark 5.3 ("Bayesian" interpretation of DP). The condition (5.2) can be interpreted as bounding a "likelihood ratio" [Poo13], a familiar quantity in

signal processing. Consider two hypotheses: one that the dataset equals $\mathcal{X}$ (*e.g.*, contains $\boldsymbol{x}_j$), and the other that the dataset equals $\mathcal{X}'$ (*e.g.*, doesn't contain $\boldsymbol{x}_j$). Then $p_{\mathcal{M}(\mathcal{X})}(\boldsymbol{s})$ would be the likelihood of observing an output (*e.g.*, a private sketch) $\boldsymbol{s}$ under the first hypothesis, while $p_{\mathcal{M}(\mathcal{X}')}(\boldsymbol{s})$ would be the same for the second hypothesis. Say an adversary wanted to detect whether or not the dataset contains $\boldsymbol{x}_j$. By appropriately thresholding the likelihood ratio $p_{\mathcal{M}(\mathcal{X})}(\boldsymbol{s})/p_{\mathcal{M}(\mathcal{X}')}(\boldsymbol{s})$, one can obtain hypothesis tests that are optimal from various perspectives (*e.g.*, Bayes, minimax, Neyman-Pearson) [Poo13, Ch. 2]. Thus, when (5.2) holds with small ϵ, it is fundamentally difficult for an adversary to determine whether $\boldsymbol{x}_j$ or $\boldsymbol{x}'_j$ was present in the dataset. Even if the adversary had non-trivial prior knowledge of the true hypothesis (as in so-called "linkage attacks," which make use of a second public dataset to which the target user contributed), (5.2) implies that—for any method—the probability of recovering the true hypothesis from the output of $\mathcal{M}$ is only slightly higher than that which is achievable *without* observing that output; see [WZ10]

Privacy-Utility tradeoff A common way to preserve privacy is to have a trusted dataholder (or "curator") corrupt the response to each query of the dataset [FWCY10] in a controlled manner (*e.g.*, by noise addition, as we will see in Subsection 5.1.3). A query may ask for something as simple as counting the number of times a given event occurred, or it may ask for more sophisticated information that requires the dataholder to run an inference algorithm. As the distortion becomes more significant, the privacy guarantee gets stronger (*i.e.*, ϵ decreases), but the quality of the response to the query (called the *utility*) degrades. Privacy-preserving mechanisms are thus characterized by a *privacy-utility tradeoff*, which describes how the utility evolves with respect to the privacy strength ϵ [FWCY10, SC13].

Main properties of DP Differential privacy has several desirable properties. First, *composition* guarantees that accessing the same dataset with I different mechanisms uses a total budget $\epsilon_{total} = \sum_{i=1}^{I} \epsilon_i$ that grows as the sum of each sub-budget ϵ_i used.

Theorem 5.4 (Sequential composition)**.** *If $\mathcal{M}_1$ and $\mathcal{M}_2$ are two mechanisms that are respectively ϵ_1-DP and ϵ_2-DP, the composed mechanism $\mathcal{M}_C(\mathcal{X}) := (\mathcal{M}_1(\mathcal{X}), \mathcal{M}_2(\mathcal{X}))$ is $(\epsilon_1 + \epsilon_2)$-DP.*

Remark 5.5. Composition is a "desirable" property in the sense that it makes the analysis of compositions of mechanisms straightforward. However, on

a pragmatic perspective, it also sets strong limits on the number of accurate analyses that can be performed on a dataset. Indeed, if analysts want to run I differentially private tasks over the data—or, typically, an iterative algorithm that requires I iterations—and the total acceptable privacy budget is set to ϵ_{total}, each task i has, on average, access to only $\epsilon_i = \frac{\epsilon_{total}}{I}$. Since typical DP mechanisms add noise of variance scaling with $\frac{1}{\epsilon_i^2}$ (see Subsection 5.1.3), this often yields inaccurate results when I is high.

Second, *post-processing* ensures that once some quantities have been computed by a differentially private algorithm, no further operation on these quantities can weaken the privacy guarantees (some form of "data processing inequality"). The latter will be particularly relevant for private sketching, as it implies that all information extracted from a differentially private sketch are differentially private.

Theorem 5.6 (Post-processing)**.** *Let $\mathcal{M}$ an ϵ-DP mechanism, and $\mathcal{F} : \mathcal{S} \to \mathcal{Y}$ any algorithm. Then $\mathcal{F} \circ \mathcal{M}$ is ϵ-DP.*

Approximate Differential Privacy Differential privacy is a very strong guarantee, and for many real-world tasks it can lead to severe degradations of the algorithms performance (utility) for small privacy budgets. For this reason, many relaxations[2] of DP have been introduced, the most prominent of which is *approximate differential privacy* (ADP) , also commonly called (ϵ, δ)-DP [DMNS]. This definition introduces a tolerance $\delta > 0$ to violations of the ϵ bound on output probabilities (also called catastrophic events).

Definition 5.7 (Approximate Differential Privacy)**.** Given a neighboring relation $\sim$ between datasets in $\mathbb{D}$, a randomized algorithm $\mathcal{M}$ is said to achieve approximate differential privacy with privacy parameters ϵ, δ (noted (ϵ, δ)-DP) if for any set of possible outcomes $S \subset \mathcal{S}$:

$$\forall \mathcal{X} \sim \mathcal{X}' \in \mathbb{D}, \ \mathbb{P}[\mathcal{M}(\mathcal{X}) \in S] \leq e^{\epsilon} \mathbb{P}[\mathcal{M}(\mathcal{X}') \in S] + \delta. \tag{5.3}$$

The composition and post-processing properties of ϵ-DP carry over to (ϵ, δ)-DP; in particular, the composition of (ϵ_1, δ_1)- and (ϵ_2, δ_2)-DP mechanism is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP, although tighter composition theorems exist [KOV17]. The post-processing theorem holds as well for (ϵ, δ)-DP.

[2] As a side note, over 200 different variants of differential privacy have been proposed [DP19]; we focus here on the—by far—most common one.

5.1.3 DP building blocks

We now review the standard tools to achieve differential privacy.

The Laplace mechanism The most common and simple method to compute a function f over a dataset with ϵ-DP is the so-called Laplace mechanism [DMNS]. For a target function $f : \mathbb{D} \to \mathbb{R}^m$ of a dataset, this mechanism adds centered Laplace noise with scale proportional to the L^1-*sensitivity* $\Delta_1(f)$ of f, which measures the maximum variation of a f between two neighboring datasets.

Definition 5.8 (L^1-sensitivity)**.** The L^1-*sensitivity* of a function $f : \mathbb{D} \to \mathbb{R}^m$ for a neighborhood relation $\sim$ is defined as

$$\Delta_1(f) := \sup_{\mathcal{X} \sim \mathcal{X}'} \|f(\mathcal{X}) - f(\mathcal{X}')\|_1. \tag{5.4}$$

This definition extends to complex-valued functions by the canonical isomorphism between $\mathbb{C}^m$ and $\mathbb{R}^{2m}$.

Definition 5.9 (Laplace Mechanism)**.** The *Laplace mechanism* to estimate privately a function $f : \mathbb{D} \to \mathbb{R}^m$ is defined as the addition of centered Laplace noise[3] $\boldsymbol{\nu}$. This mechanism achieves ϵ-DP provided the Laplace noise distribution has scale parameter $\beta = \frac{\Delta_1(f)}{\epsilon}$, *i.e.*,

$$\mathcal{M}_f^{\mathcal{L}}(\mathcal{X}) = f(\mathcal{X}) + \boldsymbol{\nu} \quad \text{where} \quad \nu_j \sim_{\text{i.i.d.}} \mathcal{L}(\beta),\ j = 1, ..., m.$$

We can extend the Laplace mechanism to complex-valued functions: the *complex Laplace mechanism* to compute a function $f : \mathbb{D} \to \mathbb{C}^k$ is defined as the addition of *complex* centered Laplace noise[4] with scale parameter $\beta = \frac{\Delta_1(f)}{\epsilon}$ (where $\Delta_1(f)$ is the complex-valued sensitivity.)

$$\mathcal{M}_f^{\mathcal{L}_{\mathbb{C}}}(\mathcal{X}) = f(\mathcal{X}) + \boldsymbol{\nu} \quad \text{where} \quad \nu_j \sim_{\text{i.i.d.}} \mathcal{L}_{\mathbb{C}}(\beta) = \mathcal{L}(\beta) + \mathrm{i}\mathcal{L}(\beta).$$

For completeness, let us shortly mention that besides the Laplace mechanism, another popular mechanism to achieve ϵ-DP is the *exponential mechanism* [MT07], that randomly selects one output at random among *a finite*

[3] A random variable ν that follows the centered Laplace distribution $\nu \sim \mathcal{L}(\beta)$ with scale β has probability density function $p_\nu(\nu) = \frac{1}{2\beta} \exp(-\frac{|\nu|}{\beta})$. Its variance is $\sigma_\nu^2 = 2\beta^2$.

[4] We say that a complex random variable follows the complex Laplace distribution with scale β, $\nu \sim \mathcal{L}_{\mathbb{C}}(\beta)$, if its real and imaginary part follow independently a real Laplace distribution of parameter β. ν admits a probability density function $p_\nu(\nu) \propto \exp(-\frac{(|\Re\nu| + |\Im\nu|)}{\beta})$, and has variance $\sigma_\nu^2 = \mathbb{E}\,|\nu|^2 = 4\beta^2$.

number of candidates; this mechanism is thus better suited to tasks where the output is not a numerical value. To ensure that the selected candidate is a good one, the sampling is biased towards candidates that have a higher "score" (to be defined by the application); while to ensure privacy, the importance of this bias decreases when ϵ does.

The Gaussian mechanism The Laplace mechanism is quite straightforward, but for many functions f, the L^1-sensitivity can be large, leading to a strong degradation of the utility. This motivates the relaxation to the more permissive requirement of (ϵ, δ)-DP.

Similarly to the Laplace mechanism, the *Gaussian mechanism* is a common and simple method to achieve $(\epsilon(\delta), \delta)$-DP, for a curve of values $\epsilon(\delta)$ parametrized by δ, by simple noise addition. It scales with the L^2-sensitivity, which is analogous to the L^1-sensitivity, but with the advantage that it is often smaller.

Definition 5.10 (L^2-sensitivity)**.** The *L^2-sensitivity* of a function $f : \mathbb{D} \to \mathbb{R}^m$ for a neighborhood relation $\sim$ is defined as

$$\Delta_2(f) := \sup_{\mathcal{X} \sim \mathcal{X}'} \|f(\mathcal{X}) - f(\mathcal{X}')\|_2. \tag{5.5}$$

This definition extends to complex-valued functions by the canonical isomorphism between $\mathbb{C}^m$ and $\mathbb{R}^{2m}$.

Definition 5.11 (Gaussian Mechanism)**.** The *Gaussian mechanism* to estimate privately a function $f : \mathbb{D} \to \mathbb{R}^m$ is defined as the addition of Gaussian noise $\boldsymbol{\nu}$ with variance σ^2, *i.e.*,

$$\mathcal{M}_f^{\mathcal{G}}(\mathcal{X}) = f(\mathcal{X}) + \boldsymbol{\nu} \quad \text{where} \quad \nu_j \sim_{\text{i.i.d.}} \mathcal{N}(0, \sigma^2),\ j = 1, ..., m.$$

Similarly, we define the *complex Laplace mechanism* to compute a function $f : \mathbb{D} \to \mathbb{C}^m$ as the addition of *complex* Gaussian noise[5] $\boldsymbol{\nu}$ with (per-component) variance σ^2

$$\mathcal{M}_f^{\mathcal{G}_{\mathbb{C}}}(\mathcal{X}) = f(\mathcal{X}) + \boldsymbol{\nu} \quad \text{where} \quad \nu_j \sim_{\text{i.i.d.}} \mathcal{N}_{\mathbb{C}}(0, \sigma^2) = \mathcal{N}(0, \sigma^2) + \mathrm{i}\mathcal{N}(0, \sigma^2).$$

This mechanism was originally shown [DR, Appendix A] to guarantee (ϵ, δ)-DP provided $\sigma \geq \sqrt{2\log(\frac{1.25}{\delta})}\,\frac{\Delta_2(f)}{\epsilon}$. This bound is commonly used

[5] That is, $\nu \sim \mathcal{N}_{\mathbb{C}}(0, \sigma^2)$ if its real and imaginary part follow independently a real normal distribution of variance σ^2 each. The overall variance of this complex random variable is $\sigma_v^2 = \mathbb{E}\,|\nu|^2 = 2\sigma^2$.

but not sharp, especially in the high privacy regime (i.e. small ϵ), and restricted to $\epsilon < 1$. Balle et al. recently proved [BW18] a tight bound on the noise level needed to guarantee a choice of (ϵ, δ).

Theorem 5.12 (Theorem 9, [BW18])**.** *For any $\epsilon, \delta > 0$, the smallest noise level σ such that the Gaussian mechanism satisfies (ϵ, δ)-DP is given by*

$$\sigma = \alpha(\epsilon, \delta)\frac{\Delta_2(f)}{\sqrt{2\epsilon}},$$

where $\alpha(\epsilon, \delta)$ is a numerical algorithmic procedure given in [BW18].

The noise level σ thus scales with the $\Delta_2(f)$ sensitivity. Note that the term $\alpha(\epsilon, \delta)$ depends on ϵ, hence it is incorrect to say that σ scales in $\epsilon^{-1/2}$. In particular, when $\epsilon \to 0$, the noise level converges to a finite constant [BW18, Section 2.1].

In our private sketches, we rely on the classical Laplace and Gaussian mechanisms. Before describing our approach however, we complete our tour of differential privacy by reviewing related works.

5.1.4 Related Work

We focus on the two learning tasks considered in this chapter: Gaussian modeling (GMM) and k-means clustering. The latter has already received a lot of attention in the differential privacy literature, while the former has been less studied.

As we saw, addition of noise is the most common way to achieve differential privacy, whether it is on the intermediate steps of an iterative algorithm or directly on the output. Private variants of standard iterative methods include DPLloyd for k-means [BDMN05], and variants with improved convergence guarantees [LS19]. The popular k-means++ seeding method has also been generalized to a private framework [NCBN16]. For Gaussian modeling, DP-GMM [WWP+16] and DP-EM [PFCW16] have been proposed. Note that for iterative algorithms, the privacy budget needs to be split between iterations, de facto limiting the total number of iterates I, which becomes a hyper-parameter (see Remark 5.5). Our approach does not suffer from this drawback since the sketch is released at once. Moreover, the same sketch can be used to run the learning algorithm multiple times with *e.g.*, different initializations.

Releasing a private synopsis of the data (similarly to our sketch) rather than directly a noisy solution has already been studied as well. For exam-

ple, EUGkM [QYL13,SCL+16] suggests to use noisy histograms for clustering (but this method is by nature limited to small dimensions), and private coresets have been investigated by Feldman et al. [FFKN09, FXZR17].

The exponential mechanism is another standard noise-additive approach for privacy. A random perturbation is drawn according to a distribution calibrated using a user-defined quality measure, and added to the output. It has been used with genetic algorithms for k-means [ZXY+13]. Such algorithms depend strongly on the quality measure of the output, which must be chosen carefully. Our sketch-based approach is in contrast more generic: the same sketch allows to solve different tasks such as clustering and GMM fitting, and it can easily be extended to new sketches in the future. Alternatively, our mechanism can be seen as a straightforward instantiation of the exponential mechanism, where the output (the sketch) is carefully designed so that is makes sense to simply use the L^1 or L^2 norms as quality measures.

Our sketching mechanism makes use of random projections, which have proven to be very useful to solve efficiently large-scale problems, and induce as well a controlled loss of information which can be leveraged to derive privacy guarantees. Balcan et al. investigated the large-scale high-dimensional clustering setting with an approach based on Johnson-Lindenstrauss dimensionality reduction [BDL+17]. Many other embeddings based on random projections have been proposed, see e.g. [KKMM13]. Linear compression of the number of samples (rather than reducing the dimension) has been considered [ZLW09] but is less scalable. Note however that as explained in the next section, the features resulting from the random projection undergo in our setting a *nonlinear transformation*, in the spirit of random features [RR08], and are averaged; they thus differ a lot from what is done in these works, although they share this common idea.

Private empirical risk minimization [CMS11, WYX17] has emerged as a generic way to design private learning algorithms, but it relies on specific assumptions (e.g. convexity, which does not hold for GMM modeling and k-means) on the loss function which defines the learning task, and still relies on multiple passes over the whole dataset.

Closer to our work, Balog et al. recently proposed to release kernel mean embeddings [BTS17], either as sets of synthetic data points in the input space or using feature maps, similarly to our method. However, to the best of our knowledge, the impact of privacy on the quality of learning in such methods has not been studied in the literature.

Some private sketching mechanism that precisely fall into our paradigm

were very recently proposed, independently from (and after) our own developments. A private version of the RACE sketch (where Φ are Locality Sensitive Hashing maps, which can be loosely assimilated to the quantized sketch we considered in Chapter 4) is presented in [CS20a]. In [HAP20], the authors consider the same RFF-based private sketch as we did in this work, but used it to train a generative network (using the technique described in Section 6.2, also independently developed) to generate synthetic data that can then be fed to standard machine learning methods.

5.2 Private Sketching Mechanism

Recall that in compressive learning [GBKT17] the sketch of a dataset $\mathcal{X}$ is defined as the average of a *feature map* Φ:

$$\boldsymbol{z}(\mathcal{X}) = \frac{1}{n}\sum_{i=1}^{n}\Phi(\boldsymbol{x}_i) =: \frac{\Sigma_\Phi(\mathcal{X})}{|\mathcal{X}|}, \tag{5.6}$$

where we define the "sum-of-features" function $\Sigma_\Phi : \mathbb{D} \to \mathbb{C}^m : \mathcal{X} \mapsto \sum_{i=1}^{n}\Phi(x_i)$ for later use. For example, most of CL literature, as well as this chapter, focuses on the random Fourier features map

$$\Phi(\boldsymbol{x}) = \Phi_{\text{RFF}}(\boldsymbol{x}) := \exp(\mathrm{i}\boldsymbol{\Omega}^\top\boldsymbol{x}),$$

which we consider here *w.l.o.g.* without $\sqrt{m}$ normalization for simplicity.

We first remark that sketching, as proposed in (5.6), is not sufficient per se to ensure the differential privacy of user contributions, despite the fact that the sketch itself (which is just at most $m \ll nd$ complex numbers) cannot contain much information about each of the n samples $\boldsymbol{x}_i \in \mathbb{R}^d$. In particular, although the vectors $(\boldsymbol{\omega}_j)_{j=1}^m$ are randomly generated, the sketching mechanism induced by a given set of such vectors is *deterministic*. Moreover, we assume that the realization $\Phi(\cdot)$ of the random feature map is publicly known (as the data analyst must know Φ to learn from the sketch), in contrast to other cryptography-based approaches like [TVBM19, RB13].

5.2.1 Generic approach

To ensure differential privacy, we thus construct a *noisy sketch* $\boldsymbol{s}(\mathcal{X})$, by perturbing the "clean sketch" $\boldsymbol{z}(\mathcal{X})$ based on the Laplacian (resp. Gaussian) mechanism, that guarantees ϵ-differential privacy (resp. (ϵ, δ)-differential privacy). First, note that from a learning perspective, instead of releasing

the clean sketch $\boldsymbol{z} \in \mathbb{C}^m$ directly, we can very well release the numerator and denominator separately, *i.e.*, release $(\Sigma_\Phi(\mathcal{X}), |\mathcal{X}|) \in \mathbb{C}^m \times \mathbb{N}$ and the fraction in 5.6 would then be computed by the data user before learning. This decomposition of the fraction simplifies the analysis of the privacy-preserving layer.

More specifically, our mechanism adds noise to the numerator and denominator of (5.6) separately, *i.e.*, it releases $(\Sigma_\Phi(\mathcal{X}) + \boldsymbol{\nu}, |\mathcal{X}| + \zeta)$ where both $\boldsymbol{\nu}$ and ζ are random. Both quantities are thus made private provided that the noise levels are properly chosen, as discussed in a moment. After reconstruction of the fraction, the obtained "private sketch" is thus

$$\boldsymbol{s}(\mathcal{X}) := \frac{\Sigma_\Phi(\mathcal{X}) + \boldsymbol{\nu}}{|\mathcal{X}| + \zeta} = \frac{\sum_{i=1}^n \Phi(x_i) + \boldsymbol{\nu}}{n + \zeta}. \tag{5.7}$$

If the mechanism that releases $(\Sigma_\Phi(\mathcal{X}) + \boldsymbol{\nu}, |\mathcal{X}| + \zeta)$ is differentially private, then the private sketch (5.7) is private as well (by composition), and so are all quantities that are learned from this sketch (by the post-processing property). Note that this decomposition also allows to further average several private sketches after computation in a distributed setting. As another remark, since DP is robust to postprocessing, one could for instance replace $|\mathcal{X}| + \zeta$ by $\max(|\mathcal{X}| + \zeta, 1)$ to avoid dividing by a null or negative quantity. The noise $\boldsymbol{\nu}$ added to $\Sigma_\Phi(\mathcal{X})$ can be either Laplacian or Gaussian depending on the desired privacy guarantee, as established in the following subsection.

5.2.2 Privacy of the mechanism

Pure ϵ-DP sketch It can be shown using the composition theorem that adding independent Laplace noises to the numerator and denominator of equation (5.7) yields differential privacy.

Proposition 5.13. *For any privacy parameter $\epsilon > 0$ and any privacy budget allocation $\epsilon_1, \epsilon_2 > 0$ such that $\epsilon_1 + \epsilon_2 = \epsilon$, if*

$$\Delta_1(\Phi) := \sup_{\boldsymbol{x} \in \mathbb{R}^d} \|\Re\Phi(\boldsymbol{x})\|_1 + \|\Im\Phi(\boldsymbol{x})\|_1$$

is finite, then the private sketching mechanism (5.7) *instantiated with noise*

$$\nu_j \sim_{\text{i.i.d.}} \mathcal{L}_{\mathbb{C}}(\beta_1 := \tfrac{\Delta_1(\Phi)}{\epsilon_1}), \quad \textit{and} \quad \zeta \sim \mathcal{L}(\beta_2 := \tfrac{1}{\epsilon_2}),$$

is ϵ-differentially private.

Proof. By relying on the composition theorem (Thm. 5.4), it suffices to show that $\mathcal{M}_1(\mathcal{X}) := \Sigma_\Phi(\mathcal{X}) + \nu$ and $\mathcal{M}_2(\mathcal{X}) := |\mathcal{X}| + \zeta$ are ϵ_1- and ϵ_2-DP, respectively. Since $\mathcal{M}_1$ and $\mathcal{M}_2$ are instances of the Laplace mechanism (Def. 5.9), it remains to prove that the L^1-sensitivities of $\Sigma_\Phi(\mathcal{X})$ and $|\mathcal{X}|$ are given by $\Delta_1(\Phi)$ and 1, respectively. The latter is trivial (removing/adding one sample changes the size of the dataset by one), so it remains to show

$$\begin{aligned}\Delta_1(\Sigma_\Phi) &= \sup_{\mathcal{X}\sim\mathcal{X}'} \|\Re\Sigma_\Phi(\mathcal{X}) - \Re\Sigma_\Phi(\mathcal{X}')\|_1 + \|\Im\Sigma_\Phi(\mathcal{X}) - \Im\Sigma_\Phi(\mathcal{X}')\|_1 \\ &= \sup_{\mathcal{X}\sim\mathcal{X}'} \|\Re(\textstyle\sum_{\boldsymbol{x}_i\in\mathcal{X}} \Phi(\boldsymbol{x}_i) - \sum_{\boldsymbol{x}'_i\in\mathcal{X}} \Phi(\boldsymbol{x}'_i))\|_1 + \|\Im(...)\|_1 \\ &= \sup_{\boldsymbol{x}} \|\Re\Phi(\boldsymbol{x})\|_1 + \|\Im\Phi(\boldsymbol{x})\|_1 =: \Delta_1(\Phi),\end{aligned}$$

where $\boldsymbol{x}$ is the sample that is added/removed to obtain $\mathcal{X}'$ from $\mathcal{X}$. □

To use this result in practice, we need to estimate the value of $\Delta_1(\Phi)$, which we do now for random Fourier features Φ_{RFF}.

Lemma 5.14.

$$\Delta_1(\Phi_{\text{RFF}}) = \sup_{\boldsymbol{x}\in\mathbb{R}^d} \|\Re\Phi_{\text{RFF}}(\boldsymbol{x})\|_1 + \|\Im\Phi_{\text{RFF}}(\boldsymbol{x})\|_1 \leq \sqrt{2}m.$$

Proof. We can decompose the objective as

$$\Delta_1(\Phi_{\text{RFF}}) = \sup_{\boldsymbol{x}} \textstyle\sum_{j=1}^m |\cos(\boldsymbol{\omega}_j^\top \boldsymbol{x})| + |\sin(\boldsymbol{\omega}_j^\top \boldsymbol{x})|.$$

Since $\sup_t |\cos(t)| + |\sin(t)| = \sqrt{2}$, a naive bound gives $\Delta_1(\Phi_{\text{RFF}}) \leq \sqrt{2}m$. □

In fact, a more sophisticated argument allows to show that in practice this bound is sharp (*i.e.*, we have a strict equality $\Delta_1(\Phi_{\text{RFF}}) = \sqrt{2}m$) [CSH+21].

Approximate (ϵ, δ)-DP sketch We can also apply the Gaussian mechanism on the numerator[6] to obtain an Approximate DP sketch. As we will see, the advantage is that the L^2-sensitivity (and thus the required noise level) of the RFF sketch scales as $\mathcal{O}(\sqrt{m})$, while we saw above that its L^1-sensitivity scales as $\mathcal{O}(m)$. In practical applications where m is of the order of hundreds or thousands, the ADP relaxation is thus especially helpful.

[6]Note that we still add Laplacian noise on the dataset size $|\mathcal{X}|$; if Gaussian noise was added we would have to split not only ϵ but also δ between the sum of features and the dataset size. As there is no difference between $\Delta_1(|\cdot|)$ and $\Delta_2(|\cdot|)$, allocating a part of δ to the denominator would not bring any substantial gain compared to putting all this budget on the numerator.

Proposition 5.15. *For any privacy parameters $\epsilon > 0$ and $\delta > 0$, for any privacy budget allocation $\epsilon_1, \epsilon_2 > 0$ such that $\epsilon_1 + \epsilon_2 = \epsilon$, if*

$$\Delta_2(\Phi) := \sup_{\boldsymbol{x} \in \mathbb{R}^d} \sqrt{\|\Re\Phi(\boldsymbol{x})\|_2^2 + \|\Im\Phi(\boldsymbol{x})\|_2^2}$$

is finite, then the private sketching mechanism (5.7) *instantiated with noise*

$$\nu_j \sim_{\text{i.i.d.}} \mathcal{N}_{\mathbb{C}}(0, \sigma^2), \quad \textit{and} \quad \zeta \sim \mathcal{L}(\beta_2 := \tfrac{1}{\epsilon_2}),$$

where $\sigma = \alpha(\epsilon_1, \delta)\frac{\Delta_2(\Phi)}{\sqrt{2\epsilon_1}}$ (see Thm. 5.12), is (ϵ, δ)-differentially private.

Proof. Similarly to the pure DP case, we rely on the composition theorem[7]. The only novel part is that we need to show that $\mathcal{M}_1(\mathcal{X}) := \Sigma_\Phi(\mathcal{X}) + \nu$ is (ϵ_1, δ)-DP, or equivalently, since we use the Gaussian mechanism, we need to show that the L^2-sensitivity of $\Sigma_\Phi(\mathcal{X})$ is $\Delta_2(\Phi)$. Recalling that, after isomorphism of $\mathbb{C}$ to $\mathbb{R}^{2m}$, this sensitivity needs to bound

$$\sup_{\mathcal{X} \sim \mathcal{X}'} \sqrt{\|\Re\Sigma_\Phi(\mathcal{X}) - \Re\Sigma_\Phi(\mathcal{X}')\|_2^2 + \|\Im\Sigma_\Phi(\mathcal{X}) - \Im\Sigma_\Phi(\mathcal{X}')\|_2^2},$$

we then proceed as in Prop. 5.13. □

We can again easily compute $\Delta_2(\Phi)$ for random Fourier features sketch.

Lemma 5.16.

$$\Delta_2(\Phi_{\text{RFF}}) = \sup_{\boldsymbol{x} \in \mathbb{R}^d} \sqrt{\|\Re\Phi_{\text{RFF}}(\boldsymbol{x})\|_2^2 + \|\Im\Phi_{\text{RFF}}(\boldsymbol{x})\|_2^2} = \sqrt{m}.$$

Proof. We can decompose the objective as

$$\Delta_2^2(\Phi_{\text{RFF}}) = \sup_{\boldsymbol{x}} \textstyle\sum_{j=1}^m \cos^2(\boldsymbol{\omega}_j^\top \boldsymbol{x}) + \sin^2(\boldsymbol{\omega}_j^\top \boldsymbol{x}) = m.$$

□

Remark 5.17. For conciseness, we presented here only the two most important use-cases of our private sketching mechanism. In the extended paper on this work [CSH+21], we derived guarantees that also hold for: the quantized sketch contributions (discussed in Chapter 4), for the "replace" neighboring relation $\sim$ (BDP), for the quadratic random features used to

[7] In particular, the composition of an (ϵ_1, δ)-DP mechanism with an ϵ_2-DP mechanism, the latter being an $(\epsilon_2, 0)$-DP mechanism, is $(\epsilon_1 + \epsilon_2, \delta)$-DP.

solve compressive PCA (see Section 2.5), as well when combined with a subsampling mechanism that allows to reduce the computational burden, (sometimes) without loss of privacy and utility. Moreover, whenever possible those guarantees were proven to be sharp (*i.e.*, the sensitivity upper bound is not loose). We refer the reader to [CSH+21, Table 1 and 2] for a complete summary of the obtained guarantees.

5.2.3 Utility

Having established the differential privacy properties of our noisy sketching mechanisms, we are now naturally interested about its impact on the *utility* of the sketch for subsequent learning. The straightforward and pragmatic approach to study the privacy-utility tradeoffs achieved by our private sketching, which is pursued in the next section, is simply by means of numerical experiments.

However, there is a strong interest in being able to predict this utility *a priori*. This would allow, in particular, to tune the different hyperparameters of the mechanism (*e.g.*, how to split the privacy budget $\epsilon_1 + \epsilon_2$) following a principled approach. Such a principle is crucial in practical privacy-preserving machine learning, as, given a fixed target privacy level, many choices of parameters are indeed possible, that can each yield a different utility value. Our goal is to pick the best choice of parameters (or at least a promising one), *without accessing the data*, because any parameter tuning that relies on probing the dataset requires to allocate a significant part of the privacy budget ϵ for parameter tuning, which further decreases the overall utility.

In the extended version of this work [CSH+21], such a predictor of the utility is proposed and studied, which we only briefly mention here. Specifically, this predictor is a proxy comprising of a noise-to-signal ratio (NSR) and the sketch size m. Given some reference sketch $\boldsymbol{z}$, that will typically be the clean empirical sketch of $\mathcal{X}$, $\boldsymbol{z}(\mathcal{X})$, or the "true" sketch $\mathcal{A}(\mathcal{P}_0)$ of the assumed underlying distribution $\mathcal{P}_0$, the noise-to-signal ratio is defined as

$$\mathrm{NSR} := \frac{\|\boldsymbol{s}(\mathcal{X}) - \boldsymbol{z}\|_2^2}{\|\boldsymbol{z}\|_2^2}.$$

It can be observed [CSH+21] that, at least for the k-means task, compressive learning from the noisy sketch $\boldsymbol{s}(\mathcal{X})$ is successful as soon as

1. The sketch size is large enough for the given task, $m \geq m_{\min}$. This

requirement is not related to privacy, but the usual condition on the number of measurements (see Section 2.5); for k-means for instance, $m_{\min} \simeq 2kd$.

2. The NSR does not exceed a critical threshold which grows with m, *i.e.*, $\mathrm{NSR} \leq \mathrm{NSR}_{\max}(m)$. For k-means, we have for instance $\mathrm{NSR}_{\max}(m) \simeq \frac{m}{10^3 kd}$.

One possible approach to tune the hyperparameters[8] of the private sketching mechanism (without access to the data), is thus to (i) obtain an analytical expression of the relationship between NSR and the different hyper-parameters, (ii) fix a sketch size m, for example slightly larger than $m_{\min}$, and (iii) to select the hyper-parameters that minimize the NSR expression. This approach is further detailed in [CSH+21].

5.3 Experiments

We now evaluate experimentally the privacy-utility trade-off for the two standard tasks in compressive learning, namely k-means and Gaussian mixture modeling, and compare to existing literature. We use the acronyms CKM (here implemented in Julia) and CGMM (using `pycle`, see App. B) for our compressive methods. Experiments on PCA can be found in [Cha20].

5.3.1 K-means clustering

Synthetic data We first consider synthetic data drawn according to a mixture of k multivariate normal distributions $\mathcal{P}_0 = \frac{1}{k}\sum_{i=1}^{k} \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{I}_d)$, where $(\boldsymbol{\mu}_i)_{1\leq i\leq k} \sim \mathcal{N}(0, (sk^{1/d})^2 \boldsymbol{I}_d)$. The parameter s controls the separation between Gaussians, which we set to $s = 2.5$ to get separated clusters. The RFF frequencies are drawn according to a Gaussian distribution with variance $\sigma^2 = 10(sk^{1/d})^2$. Figure 5.2 shows the tradeoff obtained between utility and privacy on such a synthetic dataset, using $d = 10$, $k = 10$ (both for generating the data and k-means), and $n = 10^7$. Utility is measured using the SSE (2.23). We include results for different sketch sizes, with (r=1) and without (r=m) subsampling[9], using both standard and quantized Fourier

[8] We come back in more detail to the difficulty of chosing those parameters in Section 5.4.

[9] This approach, explained in [CSH+21], consists in computing only $r \leq m$ features per sketch contribution (or equivalently, to "mask" $m - r$ of the sketch contributions, selected at random, of each data contributor). Note that, after proper re-scaling, this does not yield any benefit from the point of view of the privacy-utility curve (as can be seen from the figure); it's purpose is to potentially speed-up the sketch computation. We do not focus on this here.

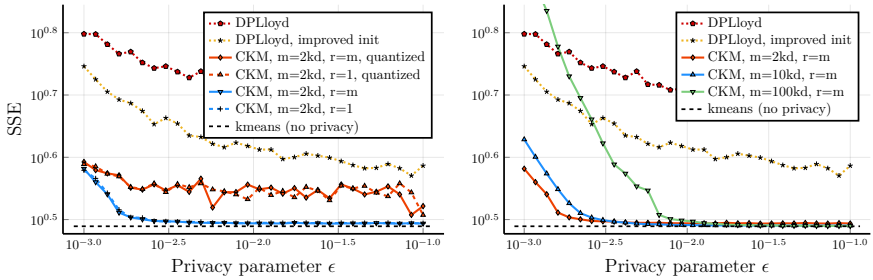


Fig. 5.2 Privacy-utility tradeoff on synthetic data for ϵ-DP. Improved initialization for DPLloyd refers to the approach proposed in [SCL+16]. Parameters: $k = d = 10$, $n = 10^7$. Medians over 100 trials. The r parameter, of little importance in this report, is the subsampling parameter explained in footnote 9.

features (see Chapter 4). We also compare to DPLloyd [BDMN05], and a variant of DPLloyd with improved initialization [SCL+16]. These iterative methods suffer from their fixed number of iterations, even for large values of ϵ, whereas our approach does not add any noise in this setting, and thus yields better results for well-chosen sketch sizes. We observe that quantization degrades slightly the performance (this is actually remedied at larger sketch sizes, as explained in Chapter 4), but subsampling with only one frequency per sample has no significant effect on the results in this setting. On the right of Fig 5.2, one can see that using a large sketch size yieds worse results in the high-privacy regime, but slightly better asympotes when ϵ is small, as one captures more information (in the nonprivate setting, a larger sketch size is always better).

Real data We provide clustering results for two real datasets, Gowalla[10] which consists in $n = 6,442,892$ spatial locations in dimension $d = 2$, and FMA[11] [DBVB16], a dataset for music analysis. In the latter case, we only consider the MFCC attributes, yielding $n = 106,574$ features in dimension $d = 20$. We compare our results with EUGkM [SCL+16] on Gowalla, as this methods relies on histograms and is only appropriate in small dimension, and with PrivGene [ZXY+13] but only for FMA as this method does not scale well.

We use for CKM (unquantized) RFF to construct a sketch size of $m =$

[10] https://snap.stanford.edu/data/loc-gowalla.html
[11] https://github.com/mdeff/fma

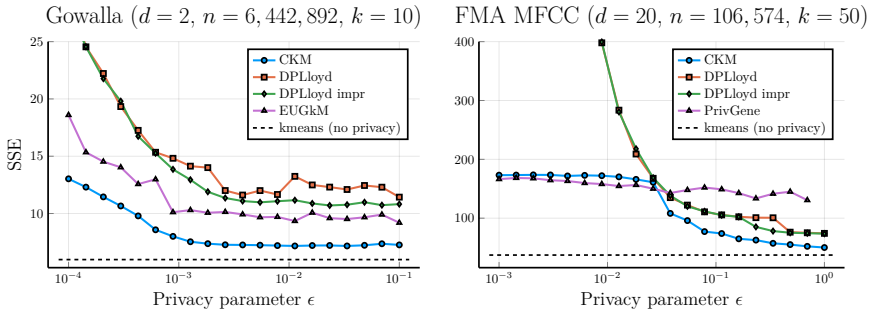


Fig. 5.3 Privacy-utility tradeoff on Gowalla and FMA (MFCC features only) datasets. Medians over 200 trials, except for privgene which is too slow.

$4kd$, where the random frequencies are drawn w.r.t. a Gaussian distributions with covariances σ^2 where $\sigma^2 = 1/310$ for Gowalla and $\sigma^2 = 1/2500$ for FMA. The results are shown Fig. 5.3. Our approach achieves performance that are always better than DPLloyd and EUGkM. PrivGene yields slightly better results for small values of ϵ, but produces very degraded centroids even for higher values of ϵ, and moreover scales poorly with the number of samples n.

5.3.2 Gaussian Mixtures Modeling*

To solve the Gaussian Mixture Modeling task from the noisy sketch, we used `CL-OMPR` adapted for mixtures of Gaussians estimation [KBGP18], denoted as CGMM below. Note that currently CGMM is limited to mixtures of Gaussians that have diagonal covariance (although non-diagonal covariances could be considered in future works). In the existing literature, differentially private estimation of GMM (with unknown, anisotropic covariances, unlike *e.g.*, in [NRS]) has been marginally studied. The main—and, to the best of our knowledge, only—existing technique is to adapt the classical expectation-maximization (EM) algorithm by adding noise on all computed quantities (the weight, mean and covariance of every Gaussian) at each iteration. We implemented both the "DP-EM" [PFCW16] and the "DP-GMM" methods [WWP+16] that follow this idea. However, we did not include the poor performance that were obtained by DP-GMM, which required systematically higher noise variance at each iteration ; moreover, its privacy guarantee relies on a separation assumption that is difficult to

control in practice. Several variants of DP-EM have been proposed corresponding to different allocations of the budget for each iteration; here we focused on the allocation that showed the best experimental performance, referred to as "zCDP-GGG" in the original paper. For a fair comparison we selected the approximate differential privacy setting, as it is the one considered in DP-EM. Since we use the improved Gaussian mechanism [BW18] in our noisy sketch mechanism, we also implemented this improvement in the different calls to the Gaussian mechanism made at each iteration by DP-EM.

Synthetic data We first generated samples from an artificially generated Gaussian mixture model $\mathcal{P}_0 = \sum_{i=1}^{K} \alpha_i \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$. The mixture weights are sampled from a uniform distribution $\alpha_i \sim \mathcal{U}([0.01, 1])$ and then normalized to sum to one. The means of the Gaussians are generated from $\boldsymbol{\mu}_i \sim \mathcal{N}(\mathbf{0}, 25/d\boldsymbol{I}_d)$, and the covariances are diagonal matrices $\Sigma_i = \mathrm{diag}(\boldsymbol{\sigma}_i)$ generated as $(\boldsymbol{\sigma}_i)_j \sim \mathcal{U}([s_i/2, 3s_i/2])$ for every i, where s_i is a scale parameter different for each Gaussian, generated as $s_i \sim \mathcal{U}([0.05/d^{1/2}, 0.95/d^{1/2}])$. We instantiated this mixture for $K = 6$ and $d = 8$, then sampled $n = 10^5$ samples from it to generate the dataset. We fixed $\delta = 10^{-7}$ for this experiment, and fixed the per-iteration budget $\delta_i = 10^{-9}$ in the zCDP allocation for DP-EM, with $I = 5$ or $I = 10$ total iterations.

Real data Following DP-EM, we ran experiments on the `ds1.10` Lifesci dataset[12], a dataset containing the $d = 10$ principal component analysis features from $n = 26,733$ biological and chemical experiments, with $K = 3$. For this experiment, we fixed $\delta = 10^{-6}$ (thus ensuring $\delta \ll 1/n$, a requirement to get meaningful privacy guarantees), and we set $\delta_i = 10^{-8}$ for the DP-EM iterations.

Both datasets have been normalized to fit in the unit ℓ_2-ball[13]. In both cases, we sampled the random frequencies of CGMMs sketch according to the "folded Gaussian" heuristic [KBGP18] with scale $\sigma = 10^{-3}$. The obtained privacy-utility curves are shown in Fig. 5.4. First, note that our performance for DP-EM don't match the results from [PFCW16]: below some critical value of ϵ, our implementation systematically fails to find any sensible solution. We observed that this happens when one of the weights

[12] Previously available at `http://komarix.org/ac/ds/`, however the website seems down. I did not find it elsewhere online since.

[13] This is required by DP-EM.

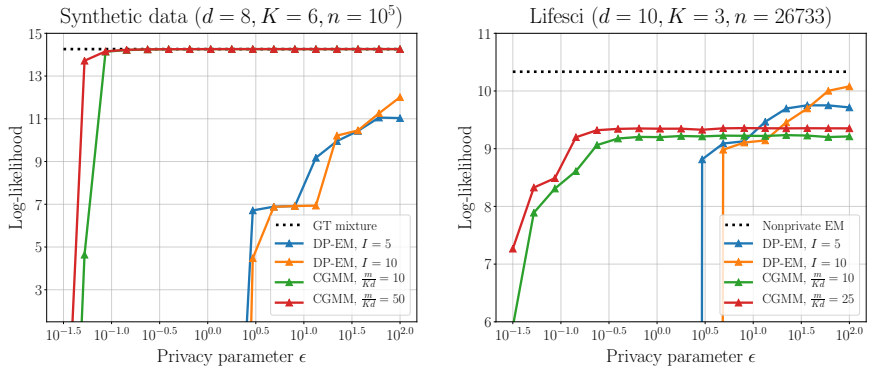


Fig. 5.4 Privacy-utility tradeoff for differentially private GMM estimation on synthetic (left) and real (right) data. Medians over 50 trials.

becomes negligible $\alpha_i \simeq 0$, often due to noise addition on it. The computed sensitivities of $\boldsymbol{\mu}_i$ and Σ_i are proportional to α_i^{-1}, so the estimated parameters of the $i-$th Gaussian will be very noisy in consequence. Because those noisy parameters most often lie far away from any actual data samples, EM is very likely to assign $\alpha_i \simeq 0$ during the next iteration (even before noise addition), causing even more noise to be added on the already noisy parameters, and the $i-$th Gaussian becomes lost in this vicious circle. It is unclear if this behavior is due to a misinterpretation in our implementation, or there is some unreported post-processing in the experiments from [PFCW16] to avoid this bad behavior. It is important to keep in mind that the DP-EM performance should thus be taken with a grain of salt.

On both datasets, CGMM performs much better in the high-privacy setting (i.e. low ϵ), reaching a plateau rapidly (even compared to the curves from [PFCW16] that we were not able to reproduce). Compared to the clustering results above (where $\delta = 0$), increasing m always led to improved results for the values we tried (this is further discussed below), saturating at $m = 25kd$, which is the reason why we included only this curve. On the synthetic dataset, which features diagonal-covariance Gaussians, CGMM reaches the same log-likelihood as the ground truth density, while on the lifesci dataset, CGMM saturates below the log-likelihood reached by the nonprivate EM algorithm. This is probably due to the diagonal covariance limitation of the CGMM implementation. Note that when the privacy constraints are stringent, it is better to limit the number of iterations of DP-EM.

5.4 Discussion

In addition to being a promising privacy-inducing mechanism (*i.e.*, in terms of privacy-utility curves), our framework has several key interesting features compared to other methods from the literature; we discuss them here, together with main limitations and perspectives.

5.4.1 Desirable properties of private sketching

Efficient and distributed private learning Firstly, the computational advantages of non-private sketching [GBKT17] remain valid after our addition of a privacy layer. In particular, learning from the sketch can be done with time and space complexities which do not depend on the number of samples n in the dataset. Moreover, the sketching process is embarassingly parallel due to the averaging operation. Sketches coming from several separate data holders can thus be aggregated again after sketching, providing *distributed* differential privacy without any additional protocol or trusted central party (which is in general not easy to achieve).

Private sketches as versatile, re-usable data summaries Another advantage is that the sketch, acting as a surrogate for the whole dataset, contains more information than just the output of one specialized algorithm, and can thus be used multiple times. This can be leveraged to solve different learning tasks from a same sketch without breaking privacy, assuming that those tasks can be solved using the same sketching operator [GBKT17]. This is what we already observed for random Fourier features, which can be used for both k-means clustering and fitting a Gaussian mixture model, two different but related estimation problems.

This potential versatility of the sketch also allows to run the learning algorithm with different initializations and parameters, producing multiple solutions; the distance to the empirical sketch can be used as a metric to pick the best of these solutions. This is in contrast with usual (e.g. iterative) differentially private methods that can be highly sensitive to the choice of such parameters (which have to be selected *a priori*, as accessing the data for parameter tuning breaks the privacy guarantee). Of course the devil is in the details, and further research is needed to investigate to what extent it is possible to choose parameters such as the sketch dimension or the "scale parameter" of random Fourier features (see below) so as to combine privacy, utility and versatility. Preliminary investigations indicate that one

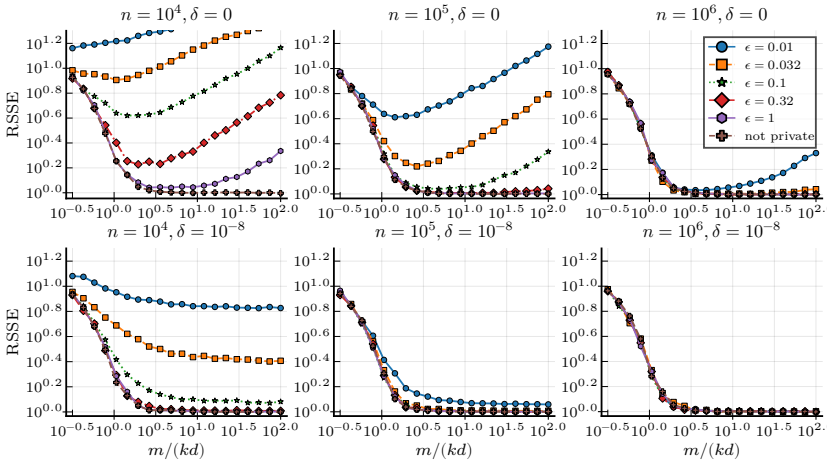


Fig. 5.5 Performance of differentially private compressive k-means as a function of m for $\delta = 0$ (top) and $\delta = 10^{-8}$ (bottom), $n = 10^4, 10^5, 10^6$ and different values of ϵ. Medians over 200 trials. Synthetic data, $k = 4, d = 8$.

can find sketch sizes resulting in good utility for both compressive gaussian mixture modeling and compressive k-means with $\delta = 10^{-8}$ and ϵ above or equal to $10^{-1.5}$

5.4.2 Open challenges: *a priori* sketch design

Although the sketch serves as a general-purpose synopsis of the dataset, at least *some* a priori knowledge about the data distribution and/or the target task is required when designing the sketch feature map Φ and other hyper-parameters of our method, as already mentioned in Subsec. 5.2.3.

Sketch size Because the noise level depends on the sketch size m, the design of a sketching procedure becomes delicate since overestimating m decreases the performance, unlike in the non-private case where increasing m usually only helps. As an illustration of this fact, consider the numerical experiment represented Fig. 5.5 (top row), where we estimate the relative SSE (RSSE)[14] achieved by compressive k-means (CKM) from the ϵ-DP sketch as a function of its size m.

As expected, in the non-private setting the SSE decreases monotonically

[14]The relative SSE is the ratio between the SSE obtained by the considered method, and that of Lloyd's standard kmeans algorithm, which is not private nor compressive.

with m. However, when $\epsilon < \infty$ and n is moderate, increasing m (and thus the noise, which is proportional to m according to Lemma 5.14) results in a worse SSE at some point. This phenomenon is more pronounced when the privacy constraints are higher, i.e. a smaller ϵ induces a smaller range of "optimal" values for the sketch size. There is thus a trade-off to make between revealing enough information for CKM to succeed (m large enough) and not revealing too much information, such that the noise needed to ensure the privacy guarantee is not too penalizing, this trade-off being more difficult in the high privacy regime. As explained in [CSH+21], this can be understood with the NSR criterion, which can thus guide the choice of the sketch size.

As shown on Fig. 5.5 (bottom), relaxing the privacy constraint to allow $\delta > 0$ mitigates the impact of m on the noise to add (recall from theorem 5.16 that the noise is then proportional to $\sqrt{m}$ instead of m), even for smaller values of n. This relaxation has the clear advantage of improving the utility for similar values of n and ϵ even for small δ, and also facilitates the choice of m, as good utilities can be reached on a wider range of sketch sizes.

Sketch scale Another crucial point is the choice of the frequencies distribution (*i.e.*, the distribution that generates $\mathbf{\Omega}$). Even when the general shape of the frequency distribution is selected and only a single scale parameter σ has to be pinned down (σ essentially controls the scale at which we can detect individual clusters), estimating an appropriate value for it is not straightforward. This might be a limitation to using sketching in practice but, on the other hand, any heuristic that could be developed in the future to estimate σ should be easy to make private as it releases a single scalar value.

5.5 Conclusion

We have proposed and analyzed a sketching mechanism that is able to guarantee the differential privacy of the data contributors. Intuitively, our approach takes advantage of the synergy between compressive learning (which aims at solving a given task from only a small sketch vector as opposed to the entire data distribution) and the need in privacy-preserving machine learning to reveal as little information as possible while still being able to solve a given task.

Numerical experiments suggest that, for k-means and GMM, this approach competes favorably with state-of-the-art alternatives. We expect that compressive learning will be extended to more learning tasks in future works (some possibilities will be explored in the next chapter); the private sketching framework presented here would be directly transferable to those new algorithms, although the sketch sensitivity would have to be re-computed for novel feature functions Φ. The true potential of private sketching will depend on how well the general field of compressive learning will be able to answer this challenge in the coming years. Those perspectives are further discussed in Chapter 7.

PART III
Towards General Compressive Learning

6

Extensions to novel tasks

So far, compressive learning is restricted to a rather limited set of tasks; *e.g.*, we mainly focused on k-means and GMM in Chapters 4 and 5, although a few other tasks (PCA, ICA, ...) were discussed in Section 2.5. There is thus a strong interest in being able to extend this framework to novel tasks. This chapter presents two proof-of-concepts of such extensions—namely, for classification and generative network fitting—as well as a generic compressive learning "debugging" methodology, potentially useful to the research of any compressive learning method.

Remark 6.1. Note that the extensions presented here are *tentative* (not mature), and have in particular been mainly validated only on toy example datasets. After some hesitation, those contributions were still added to this thesis manuscript for the two following reasons: *(i)* they can serve as *baseline* and/or *starting point* for more advanced CL methods, and *(ii)* they serve as support to *discuss some of their limitations*, which prevented us from going further in their analysis. Both of those elements seem useful to make progress in the field.

More specifically, Section 6.1 presents a *compressive classification* baseline, built on the idea of constructing *one sketch per class*. It also quickly introduces a sketch feature map Φ based on random neural networks, which is used to build a basic sketched image recognition scheme. This work was presented at the iTWIST'18 workshop [SJ18a].

In Section 6.2, we discuss the possibility to *compressively learn generative networks*, by drawing a connection with a strategy known as MMD-GAN [LSZ15]. It was presented at the ESANN'20 conference [SJ20b].

Finally, in Section 6.3 we present a generic methodology to "debug" compressive learning methods. Indeed, one difficulty in researching CL is that, when a given scheme does not show empirical success, it is hard to tell if this failure is to be blamed on the sketching phase (*i.e.*, the choice of the random features which are averaged), learning phase (*i.e.*, the heuristic algorithm that extracts parameters from this sketch), or if the problem is simply not solvable in a compressive manner. The purpose of this pragmatic method is to differentiate between those three situations, an insight which should be useful for the design of any compressive learning scheme (and in particular, when extending CL to new tasks). This work was presented at the iTWIST'20 workshop [SJ20c].

6.1 Compressive Classification

So far, CL mainly focused on *unsupervised* ML tasks (cfr. Subsection 2.1.5), where learning examples don't belong to a (known) class. We propose a simple baseline to extend CL to *supervised* ML tasks by proposing (in Subsec. 6.1.2) and experimentally validating (in Subsec. 6.1.3) a first simple *compressive classification method* using only a sketch of the labeled dataset (Fig. 6.1). We also introduce a sketch feature function leveraging a random convolutional neural network to better capture information in images. While not as accurate as ML methods learning from the full dataset, this compressive classification scheme still attains nontrivial accuracy levels considering its *unlearned nature*. Our method also enjoys a nice geometric interpretation, *i.e.*, Maximum A Posteriori classification performed in the Reproducible Kernel Hilbert Space associated with the sketch.

6.1.1 Preliminaries

(Unsupervised) Compressive Learning For convenience, we recall here the notations and concepts from CL that will be relevant for this section. Unsupervised ML usually amount to estimate parameters of a true data distribution $\mathcal{P}_0$, from a dataset $\mathcal{X} = \{\boldsymbol{x}_i \sim_{\text{i.i.d.}} \mathcal{P}_0\}_{i=1}^n \subset \mathbb{R}^d$ of examples—associated to an *empirical distribution* $\widehat{\mathcal{P}}_{\mathcal{X}} = \frac{1}{n}\sum_{\boldsymbol{x}_i \in \mathcal{X}} \delta_{\boldsymbol{x}_i}$, with $\delta_{\boldsymbol{u}}$ the Dirac measure at $\boldsymbol{u}$. In CL, a *dataset sketch* $\boldsymbol{z}_{\mathcal{X}}$ serves as a proxy for the true *distribution sketch* $\mathcal{A}(\mathcal{P}_0)$, where the map $\mathcal{A}$ is a linear embedding of the proba-

bility distribution $\mathcal{P}_0$ into the lower-dimensional space $\mathbb{C}^m$:

$$\mathcal{A}(\mathcal{P}_0) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}_0}\,\Phi(\boldsymbol{x}) \quad \simeq \quad \boldsymbol{z}_{\mathcal{X}} := \mathcal{A}(\widehat{\mathcal{P}}_{\mathcal{X}}) = \tfrac{1}{n}\textstyle\sum_{\boldsymbol{x}_i\in\mathcal{X}}\Phi(\boldsymbol{x}_i), \qquad (6.1)$$

where $\Phi : \mathbb{R}^d \to \mathbb{C}^m$ is a (typically) random nonlinear feature map. This map defines a positive definite kernel $\kappa(\boldsymbol{u},\boldsymbol{v}) := \mathbb{E}\langle\Phi(\boldsymbol{u}),\Phi(\boldsymbol{v})\rangle$, and κ in turn provides a Reproducible Kernel Hilbert Space (RKHS) $\mathcal{H}_\kappa$ to embed distributions (see Subsec. 2.1.3). As can be understood in light of Section 2.4, $\mathcal{A}$ indirectly maps $\mathcal{P}$ to its *mean map* $\kappa(\cdot,\mathcal{P}) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\,\kappa(\cdot,\boldsymbol{x}) \in \mathcal{H}_\kappa$ [Aro50, SGSS07, S+10]. Typical CL methods [KTTG17, KBGP18] use Random Fourier Features [RR08] as map Φ, defined by

$$\Phi_{\text{RFF}}(\boldsymbol{x}) = \left[\exp(\mathrm{i}\,\boldsymbol{\omega}_j^\top\boldsymbol{x})\right]_{j=1}^{m} \quad \text{with} \quad \boldsymbol{\omega}_j \sim_{\text{i.i.d.}} \Lambda. \qquad (6.2)$$

The induced kernel κ is then shift-invariant and the Fourier transform of the distribution Λ, *i.e.*, $\kappa(\boldsymbol{x},\boldsymbol{x}') = \kappa^\Delta(\boldsymbol{x}-\boldsymbol{x}') := (\mathcal{F}\Lambda)(\boldsymbol{x}-\boldsymbol{x}')$ [Rud62]. CL is promising because the sketch $\boldsymbol{z}_{\mathcal{X}}$ (ideally) retains sufficient information to solve the task at hand whenever its size m exceeds some value independent on the number of examples n, yielding algorithms that scale well when n increases.

Random Convolutional Neural Networks (CNN) Shift-invariant kernels are not that relevant when dealing with images: for example, they are sensitive (more precisely, "not invariant") to simple image perturbations such as translations, rotations, or re-scaling. Recent studies have shown that the last layer of a *randomly weighted* (convolutional) neural network `CNN` (combining convolutions with random weights, nonlinear activations, and pooling operations) captures surprisingly meaningful image features [CS09, GSB16, RT19]. We thus propose to use the feature map $\Phi_{\texttt{CNN}}(\boldsymbol{x}) = \texttt{CNN}(\boldsymbol{x}) \in \mathbb{R}^m$ as sketch map Φ for images. The associated kernel κ is (for a fully connected network) an *arc-cosine kernel*, that surpasses shift-invariant kernels for solving image classification tasks with kernel methods [CS09].

6.1.2 Compressive learning classification

Our scheme (Fig. 6.1) consists of two sequential stages.

Observation phase *Supervised classification* infers a mathematical model from a labeled dataset $\mathcal{X} := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ where each signal $\boldsymbol{x}_i \in \mathbb{R}^d$ be-

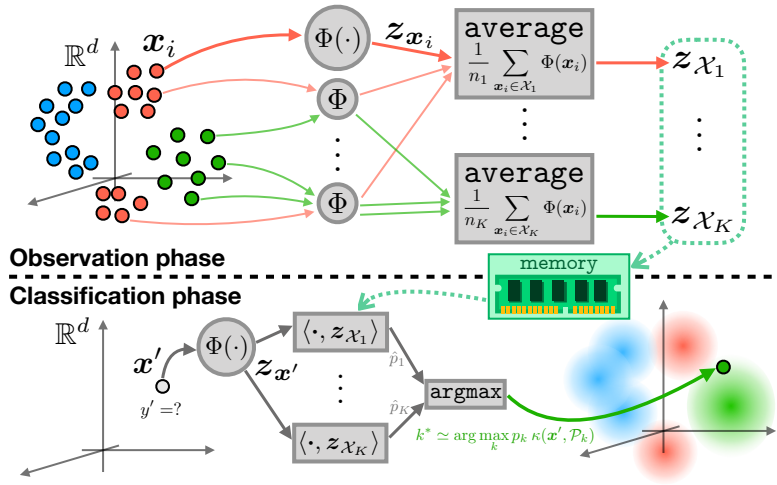


Fig. 6.1 **Observation phase:** we summarize the dataset $\mathcal{X}$ as the K class sketches $\boldsymbol{z}_{\mathcal{X}_k}$: the class average of non-linear maps $\boldsymbol{z}_{\boldsymbol{x}_i} = \Phi(\boldsymbol{x}_i)$ of the examples $\boldsymbol{x}_i$. **Classification phase:** a new sample $\boldsymbol{x}'$ gets assigned to the class label k^* that maximizes the correlation between its sketch $\boldsymbol{z}_{\boldsymbol{x}'}$ and the stored class sketches; this can be interpreted as MAP classification in a RKHS $\mathcal{H}_\kappa$.

longs to a class $\mathcal{C}_k$ (out of K different classes), as designated by its *class label* $y_i \in [K]$. Denoting $p_k := \mathbb{P}(\boldsymbol{x} \in \mathcal{C}_k) = \mathbb{P}(y = k)$, the signals are assumed drawn from an unknown distribution $\mathcal{P}$:

$$\boldsymbol{x}_i \sim_{\text{i.i.d.}} \mathcal{P} = \textstyle\sum_{k=1}^{K} p_k\, p(\boldsymbol{x}|\, \boldsymbol{x} \in \mathcal{C}_k) = \sum_k p_k \mathcal{P}_k(\boldsymbol{x}), \tag{6.3}$$

where $\mathcal{P}_k(\boldsymbol{x}) := p(\boldsymbol{x}|\, \boldsymbol{x} \in \mathcal{C}_k)$ is the probability density of $\boldsymbol{x}$ conditioned on the class k. As illustrated in Fig. 6.1(top), our *supervised compressive learning* framework considers that $\mathcal{X}$ is not explicitly available but compressed as a collection of K *class sketches* $\boldsymbol{z}_{\mathcal{X}_k}$ defined as:

$$\boldsymbol{z}_{\mathcal{X}_k} = \mathcal{A}(\widehat{\mathcal{P}}_{\mathcal{X}_k}) = \tfrac{1}{n_k}\textstyle\sum_{\boldsymbol{x}_i\in\mathcal{X}_k} \Phi(\boldsymbol{x}_i) \quad \text{where} \quad \mathcal{X}_k := \{\boldsymbol{x}_i \,|\, \boldsymbol{x}_i \in \mathcal{C}_k\}. \tag{6.4}$$

We can also require approximated *a priori* class probabilities $\hat{p}_k$, *e.g.*, $\hat{p}_k = \frac{n_k}{n}$ if we count the class occurrences $n_k = |\mathcal{X}_k|$, or setting an uniform prior $\hat{p}_k = \frac{1}{K}$ otherwise.

Classification phase Under (6.3), the optimal classifier (*i.e.*, in the sense that it minimizes the error probability) for a test example $\boldsymbol{x}'$ is the Maxi-

mum A Posteriori (MAP) estimator $k^{\texttt{MAP}} := \arg\max_k p_k \mathcal{P}_k(\boldsymbol{x}')$; in practice, $\mathcal{P}_k$ is generally hard to estimate. In our CL framework, we classify $\boldsymbol{x}'$ from $\boldsymbol{z}_{\mathcal{X}_k}$ and $\hat{p}_k$ only (Fig. 3.1, bottom): we acquire its sketch $\boldsymbol{z}_{\boldsymbol{x}'} = \Phi(\boldsymbol{x}')$ and maximize the correlation with the class sketch weighted by $\hat{p}_k$, *i.e.*, we assign to $\boldsymbol{x}'$ the label

$$k^* := \arg\max_k \ \hat{p}_k \langle \boldsymbol{z}_{\boldsymbol{x}'}, \boldsymbol{z}_{\mathcal{X}_k} \rangle \tag{6.5}$$

Note that this "Compressive Classifier" (CC) does not require parameter tuning (the choice of Φ being rather a *hyper*-parameter). Interestingly, under a few approximations, this procedure can be seen as a MAP estimator in the RKHS $\mathcal{H}_\kappa$. Indeed, we first note that if m is large, the law of large numbers (LLN) provides the *kernel approximation*:

$$\langle \Phi(\boldsymbol{u}), \Phi(\boldsymbol{v}) \rangle \simeq \kappa(\boldsymbol{u}, \boldsymbol{v}), \quad \forall \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d. \tag{6.6}$$

Assuming n_k is also large, another use of the LLN gives the *mean map approximation*: we have both $\hat{p}_k \simeq p_k$ and (using the kernel approximation)

$$\begin{aligned} &\langle \boldsymbol{z}_{\boldsymbol{u}}, \boldsymbol{z}_{\mathcal{X}_k} \rangle = \tfrac{1}{n_k} \textstyle\sum_{\boldsymbol{x}_i \in \mathcal{X}_k} \langle \Phi(\boldsymbol{u}), \Phi(\boldsymbol{x}_i) \rangle \simeq \tfrac{1}{n_k} \textstyle\sum_{\boldsymbol{x}_i \in \mathcal{X}_k} \kappa(\boldsymbol{u}, \boldsymbol{x}_i) \\ &\simeq \ \mathbb{E}_{\boldsymbol{x} \sim \mathcal{P}_k} \kappa(\boldsymbol{u}, \boldsymbol{x}) \ =: \ \kappa(\boldsymbol{u}, \mathcal{P}_k) \quad \forall \boldsymbol{u} \in \mathbb{R}^d. \end{aligned} \tag{6.7}$$

Consequently, plugging the kernel approximation (6.6) and mean map approximation (6.7) into (6.5), we have that

$$k^* \ \simeq \ \arg\max_k \ p_k \, \kappa(\boldsymbol{x}', \mathcal{P}_k). \tag{6.8}$$

In other words, we replace $\mathcal{P}_k$ in the MAP estimator by its Mean Map $\kappa(\cdot, \mathcal{P}_k)$—its embedding in $\mathcal{H}_\kappa$—such that CC computes a *MAP estimation inside the RKHS* $\mathcal{H}_\kappa$. In all generality $\kappa(\cdot, \mathcal{P}_k)$ is not a probability density function, but can be interpreted as a smoothing of $\mathcal{P}_k$ by convolution with $\kappa^\Delta(\boldsymbol{u}) := \kappa(\boldsymbol{u}, 0)$ if κ is a properly scaled shift-invariant kernel. Alternatively, (6.8) can be seen as a Parzen-windows classifier—a nonparametric Support Vector Machine (without weights learning)—evaluated compressively thanks to the sketch [DH+73, SSB+02].

6.1.3 Experimental proof of concept

Synthetic datasets We build two datasets that are not linearly separable (Fig. 6.2 left), and sketch them using $\Phi = \Phi_{\text{RFF}}$ with $\Lambda \sim \mathcal{N}(\boldsymbol{0}, \sigma^{-2} \boldsymbol{I}_d)$, which corresponds to a Gaussian kernel $\kappa(\boldsymbol{u}, \boldsymbol{v}) = \exp(-\frac{\|\boldsymbol{u}-\boldsymbol{v}\|^2}{2\sigma^2})$. As shown

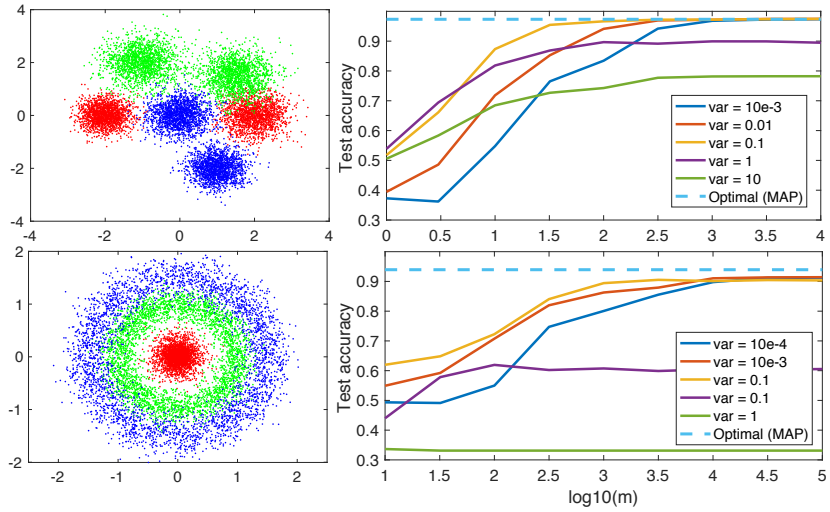


Fig. 6.2 **Left**: synthetic 2-d datasets of $n = 10^4$ examples from $K = 3$ equiprobable classes, separated into $2/3$ for "training" (observation phase) and $1/3$ for testing (classification phase). **Right**: testing accuracy (average over 10 trials) of our compressive classification method for different values of σ (noted `var`) and increasing m (solid), compared to MAP classification (dashed).

Fig. 6.2 (right), the test accuracy of CC improves with m until reaching—when the kernel approximation is good enough—a constant floor depending on the compatibility between κ and $\mathcal{P}$. Accuracy is almost optimal when κ is close to the constituents of $\mathcal{P}$ (*e.g.*, 1st dataset, $\sigma = 0.1$), but degrades when the kernel scale and/or shape mismatches the data clusters (*e.g.*, 1st dataset, $\sigma = 10$; or 2nd dataset). CC thus reaches good accuracy provided m is large enough (note that in our examples, this value might be quite high) and κ is well adapted to the task.

Standard datasets We also test CC on some well-known "real-life" datasets from the UCI ML Repository [AN07]. Table 6.1 compares the error rates of CC and kernel SVM, a fully learned approach (see Section 2.1). Although worse than SVM, CC is surprisingly accurate considering its compressive nature, low computational cost (especially when $m = 50$), and the fact that κ is a standard, non-tuned kernel.

	n	d	K	SVM	$m = 50$	$m = 1000$
Iris	150	4	3	2.00 **4.00**	6.51 ± 1.81 $\mathbf{8.22 \pm 3.25}$	5.51 ± 1.23 $\mathbf{6.18 \pm 2.40}$
Wine	178	13	3	0.84 **1.69**	4.56 ± 2.34 $\mathbf{13.75 \pm 4.09}$	2.43 ± 0.72 $\mathbf{8.19 \pm 1.29}$
Breast cancer	569	30	2	3.67 **2.13**	7.00 ± 1.40 $\mathbf{9.22 \pm 2.33}$	3.93 ± 0.39 $\mathbf{6.23 \pm 0.69}$
Adult (3 attr.)	30718	3	2	21.03 **21.06**	23.88 ± 4.37 $\mathbf{36.09 \pm 6.67}$	23.11 ± 1.05 $\mathbf{35.04 \pm 1.63}$

Table 6.1 Standard datasets: train set (gray) and test set (black and bold) average error rates $\pm$ standard deviation (in %, 100 repetitions), for SVM and CC with $m \in \{50, 1000\}$, and with $\sigma = 2$ arbitrarily set (data re-scaled inside $[-1, +1]^n$).

Image classification We also considerimage classification datasets: handwritten digit recognition (MNIST) and vehicle/animal recognition (CIFAR-10). Here, we use $\Phi = \Phi_{\texttt{CNN}}$ (the default architecture provided by [VLB18] with random initialization), which yielded better accuracy than $\Phi_{\texttt{RFF}}$ (not reported here). We compare this CC scheme with the same CNN architecture equipped with an additional classification layer, with all weights learned in one pass over $\mathcal{X}$ (for a somewhat fair comparison). Again CC is outperformed by the learned approach, but still achieves reasonable, non-trivial accuracy (although the error rate is still relatively high on CIFAR-10).

	n	d	CNN	m = 250	m = 5000
MNIST	60000	$28 \times 28 \times 1$	1.60 ± 0.12	17.73 ± 1.43	16.60 ± 1.54
	10000		$\mathbf{1.63 \pm 0.11}$	$\mathbf{16.83 \pm 1.39}$	$\mathbf{15.80 \pm 1.61}$
CIFAR10	50000	$32 \times 32 \times 3$	39.08 ± 1.48	71.76 ± 1.85	72.83 ± 2.00
	10000		$\mathbf{40.28 \pm 1.36}$	$\mathbf{71.12 \pm 1.72}$	$\mathbf{72.02 \pm 1.85}$

Table 6.2 Image datasets: train (gray) and test (black, bold) average error rates $\pm$ standard deviation (in %, 10 repetitions), for CNN and CC with $m \in \{250, 5000\}$.

6.1.4 Discussion

Our *compressive classification* method is quite simple, as it requires no training, but only to compute the class sketches (accumulated random nonlinear features $\Phi(\cdot)$ of the learning examples). This classifier is also relatively cheap to evaluate, and has an interesting interpretation: a MAP estimator inside the RKHS $\mathcal{H}_\kappa$ associated with the kernel κ defined by Φ. Our preliminary experimental results are an encouraging proof of concept, but indicate room for improvement if the mapping Φ (and the associated kernel κ) are adapted to the true data distribution; for example, image classification accuracy improves when Φ is a random CNN (which defines a shift-variant κ).

However, this scheme presents crucial limitations, the main one being that the choice of the feature map Φ (or equivalently, of the associated kernel κ) is quite difficult. Intuitively, κ should be such that the mean maps $\kappa(\cdot, \mathcal{P}_k) \in \mathcal{H}_\kappa$ of different classes k are "well separated" (ideally, as much separated as the initial, unknown densities $\mathcal{P}_k$). It is not at all obvious that such a kernel even exists (and if this is not the case, this would severely limit the performance of even the best possible CC scheme), let alone that an efficient feature map Φ (*e.g.*, of low dimension m) can be tuned easily in practice. Moreover, a last drawback is that the idea to have one sketch per class can be costly as the number of classes grow, and is not generalizable to the regression problem.

6.2 Compressive Learning of Generative Networks

Generative networks implicitly approximate complex densities from their sampling with impressive accuracy. However, because of the enormous scale of modern datasets, this training process is often computationally expensive. In this section, we cast generative network training into the framework of compressive learning, which has the potential to reduce the computational burden on large-scale datasets. In particular, we propose a cost function to guide the network's parameters, which approximates the Maximum Mean Discrepancy metric, but requires *only* this sketch, which makes it time- and memory-efficient to optimize.

6.2.1 Motivation

Generative networks (GNs) received a significant amount of interest for their ability to embed data-driven priors in general applications, *e.g.*, for solving

inverse problems such as super-resolution, deconvolution, inpainting, or compressive sensing to name a few [B+17, M+17, RC+17, L+18]. As further explained in Subsection 6.2.2, GNs are deep neural networks (DNNs) trained to generate samples that "mimic" those available in a given dataset. By minimizing some well-crafted cost-function during the training stage, these networks implicitly learn the probability distribution synthesizing this dataset (more precisely, one minimizes a *divergence* between the empirical probability distributions of the training and generated data; see Section 2.4). Passing randomly generated low-dimensional inputs through to the GN then generates new high-dimensional samples.

In generative *adversarial* networks (GANs) this cost is dictated by a second "discriminator" network that classifies real (training) and fake (generated) examples. The generative and the discriminator networks are learned simultaneously in a two-player zero-sum game [G+14]. If the discriminator is perfectly trained, the cost function boils down to the Jensen-Shannon divergence between the training data and generated data distributions. While GANs are the golden standard, achieving the state-of-the-art for a wide variety of tasks, they are notoriously hard to learn due to the need to balance carefully the training of the two networks.

In contrast, MMD-GNs minimize the simpler Maximum Mean Discrepancy (MMD) cost function [LSZ15, D+15], *i.e.*, a "kernelized" distance measuring the similarity of generated and real samples. Although training MMD-GNs is conceptually simpler than GANs—we can resort to straightforward gradient descent-based solvers (*e.g.*, stochastic gradient descent)—its computational complexity scales poorly with large-scale datasets: each iteration necessitates numerous (typically of the order of thousands) accesses to the whole dataset. This severely limits the practical use of MMD-GNs, as argued in [A+17].

This work proposes and assesses the potential of sketching to "compressively learn" such deep generative networks (MMD-GNs) with greatly reduced computational cost (see Fig. 6.3). By defining a cost function and practical learning scheme, our approach serves as a prototype for compressively learning general generative models from sketches. The effectiveness of this scheme is tested on toy examples.

6.2.2 Preliminaries

Given some space $\Sigma \subset \mathbb{R}^d$, we assimilate any dataset $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n \subset \Sigma$ with n samples to a discrete probability measure $\widehat{\mathcal{P}}_{\mathcal{X}}$, *i.e.*, an empirical

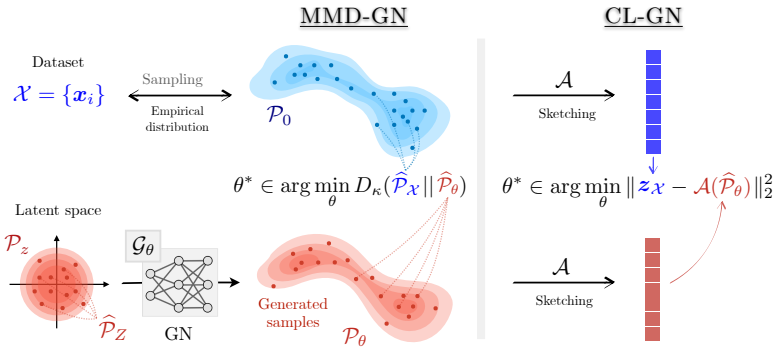


Fig. 6.3 General overview of our approach. The moment matching of MMD-GNs is replaced by sketching both $\mathcal{X}$ and the sampling $\widehat{\mathcal{P}}_\theta$. This compressive learning approach of GNs (that we call CL-GN) is allowed by relating the RFF frequency distribution Λ to the MMD kernel κ.

estimate for the *probability distribution* $\mathcal{P}_0$ generating $\mathcal{X}$. Said differently, $\boldsymbol{x}_i \sim_{\text{i.i.d.}} \mathcal{P}_0$ and $\widehat{\mathcal{P}}_\mathcal{X} := \frac{1}{|\mathcal{X}|} \sum_{i=1}^n \delta_{\boldsymbol{x}_i}$, where $\delta_{\boldsymbol{c}}$ is the Dirac measure at $\boldsymbol{c} \in \Sigma$.

Compressive Learning As usual, we find it useful to recall the concepts from CL we will need. In this section, we focus on the sketch operator induced by random Fourier features $\Phi(\boldsymbol{x}) := \frac{1}{\sqrt{m}} \exp(\mathrm{i}\boldsymbol{\Omega}^\top \boldsymbol{x})$, *i.e.*,

$$\mathcal{A}(\mathcal{P}) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\, \Phi(\boldsymbol{x}) = \frac{1}{\sqrt{m}} \left[\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}} \exp\left(\mathrm{i}\boldsymbol{\omega}_j^\top \boldsymbol{x}\right) \right]_{j=1}^m,$$

which embeds the probability measure $\mathcal{P}$ into the low-dimensional domain $\mathbb{C}^m$. The matrix $\boldsymbol{\Omega} := (\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_m) \in \mathbb{R}^{d\times m}$ is randomly generated by drawing m frequencies $\boldsymbol{\omega}_j \sim_{\text{i.i.d.}} \Lambda$, where $\Lambda = \mathcal{F}\kappa^\Delta$ is the Fourier transform of some shift-invariant *kernel* $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa^\Delta(\boldsymbol{x} - \boldsymbol{x}')$. For large values of n, we expect that *sketch of the dataset* $\boldsymbol{z}_\mathcal{X}$ approaches that of the data distribution

$$\mathcal{A}(\mathcal{P}_0) \;\approx\; \boldsymbol{z}_\mathcal{X} := \mathcal{A}(\widehat{\mathcal{P}}_\mathcal{X}) = \frac{1}{n} \sum_{i=1}^n \Phi(\boldsymbol{x}_i) \in \mathbb{C}^m. \tag{6.9}$$

Usually, CL aims at learning, from only the sketch $\boldsymbol{z}_\mathcal{X}$, an approximation $\mathcal{P}_{\boldsymbol{\theta}}$ for the density $\mathcal{P}_0$, parametrized by $\boldsymbol{\theta} \in \Theta$. For example, $\boldsymbol{\theta}$ collects the position of the K centroids for compressive K-means, and the weights, centers and covariances of different Gaussians for compressive Gaussian mixtures fitting. This is achieved by solving the following density fitting

("sketch matching") problem:

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{z}_{\mathcal{X}} - \mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}})\|_2^2 \quad \text{s.t.} \quad \boldsymbol{\theta} \in \Theta. \tag{6.10}$$

For large values of m, the cost in (6.10) estimates a metric D_κ between $\widehat{\mathcal{P}}_{\mathcal{X}}$ and $\mathcal{P}_{\boldsymbol{\theta}}$, called the Maximum Mean Discrepancy (MMD) [GBR+12], that is kernelized by κ, *i.e.*, writing $\kappa(\mathcal{P}, \mathcal{Q}) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P},\boldsymbol{y}\sim\mathcal{Q}}\, \kappa(\boldsymbol{x}, \boldsymbol{y})$, the MMD reads

$$D_\kappa^2(\mathcal{P}, \mathcal{Q}) := \kappa(\mathcal{P}, \mathcal{P}) + \kappa(\mathcal{Q}, \mathcal{Q}) - 2\kappa(\mathcal{P}, \mathcal{Q}). \tag{6.11}$$

Using Bochner's theorem, we can indeed rewrite (6.11) as

$$\begin{aligned} \|\mathcal{A}(\widehat{\mathcal{P}}_{\mathcal{X}}) - \mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}})\|_2^2 &= \tfrac{1}{m}\textstyle\sum_{j=1}^m \left| \mathbb{E}_{\boldsymbol{x}\sim\widehat{\mathcal{P}}_{\mathcal{X}}}\, e^{\mathrm{i}\boldsymbol{\omega}_j^\top \boldsymbol{x}} - \mathbb{E}_{\boldsymbol{y}\sim\mathcal{P}_{\boldsymbol{\theta}}}\, e^{\mathrm{i}\boldsymbol{\omega}_j^\top \boldsymbol{y}} \right|^2 \\ &\simeq \mathbb{E}_{\boldsymbol{\omega}\sim\Lambda} \left| \mathbb{E}_{\boldsymbol{x}\sim\widehat{\mathcal{P}}_{\mathcal{X}}}\, e^{\mathrm{i}\boldsymbol{\omega}^\top \boldsymbol{x}} - \mathbb{E}_{\boldsymbol{y}\sim\mathcal{P}_{\boldsymbol{\theta}}}\, e^{\mathrm{i}\boldsymbol{\omega}^\top \boldsymbol{y}} \right|^2 = D_\kappa^2(\widehat{\mathcal{P}}_{\mathcal{X}}, \mathcal{P}_{\boldsymbol{\theta}}). \end{aligned} \tag{6.12}$$

Provided Λ is supported on $\mathbb{R}^d$, $D_\kappa(\mathcal{P}, \mathcal{Q}) = 0$ if and only if $\mathcal{P} = \mathcal{Q}$ [S+10]. Thus, minimizing (6.10) accurately estimates $\widehat{\mathcal{P}}_{\mathcal{X}}$ from $\mathcal{P}_{\boldsymbol{\theta}^*}$ if m is large compared to the complexity of the model; *e.g.*, in compressive K-means, CL requires experimentally $m = \mathcal{O}(Kd)$ to learn the centroids of K clusters in $\mathbb{R}^d$.

The non-convex sketch matching problem (6.10) is generally solved with greedy heuristics (*e.g.*, CL-OMPR [KBGP18]). As they require a closed-form expression of $\mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}})$ and the Jacobian $\nabla_{\boldsymbol{\theta}}\mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}})$, CL has so far be limited to cases where $\mathcal{P}_{\boldsymbol{\theta}}$ is explicitly available and easy to manipulate.

Generative networks In order to generate realistic data samples, a generative network $\mathcal{G}_{\boldsymbol{\theta}^*} : \Sigma_z \to \Sigma$ (which follows some DNN architecture) with weights $\boldsymbol{\theta}^* \in \mathbb{R}^{d_\theta}$ is trained as follows. Given some weights configuration $\boldsymbol{\theta} \in \Theta$, we compute the *synthetic empirical distribution*

$$\widehat{\mathcal{P}}_{\boldsymbol{\theta}} := \mathcal{G}_{\boldsymbol{\theta}}(\widehat{\mathcal{P}}_Z) = \tfrac{1}{n'}\textstyle\sum_{i=1}^{n'} \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{z}_i),$$

obtained by n' inputs $Z = \{\boldsymbol{z}_i\}_{i=1}^{n'}$ randomly drawn in a low-dimensional *latent space* $\Sigma_z \subset \mathbb{R}^p$ from a "simple" distribution $\mathcal{P}_z$, *e.g.*, $\boldsymbol{z}_i \sim_{\text{i.i.d.}} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_p)$. By design, $\widehat{\mathcal{P}}_{\boldsymbol{\theta}}$ is related to sampling the *pushforward* distribution of $\mathcal{P}_z$ by $\mathcal{G}_{\boldsymbol{\theta}}$. The parameter $\boldsymbol{\theta}^*$ is then fit to the dataset, in order to ensure that $\widehat{\mathcal{P}}_{\boldsymbol{\theta}^*} \approx \widehat{\mathcal{P}}_{\mathcal{X}}$. While several divergences have been proposed to quantify this objective, we focus here on minimizing the MMD metric $D_\kappa^2(\widehat{\mathcal{P}}_{\mathcal{X}}, \widehat{\mathcal{P}}_{\boldsymbol{\theta}})$ [LSZ15,

D$^+$15]. Using (6.11) and discarding constant terms, we get the MMD-GN fitting problem:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{\boldsymbol{z}_i,\boldsymbol{z}_j \in Z} \kappa(\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{z}_i), \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{z}_j)) - 2\sum_{\boldsymbol{x}_i \in \mathcal{X}, \boldsymbol{z}_j \in Z} \kappa(\boldsymbol{x}_i, \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{z}_j)). \tag{6.13}$$

Li et al. called this approach generative *moment matching* [Hal05] networks, as minimizing (6.11) amounts to matching all the (infinite) moments of $\mathcal{P}$ and $\mathcal{Q}$ in the feature space associated to κ (see Fig. 6.3).

If κ is differentiable, gradient descent-based methods can be used to solve (6.13), using back-propagation to compute the gradients of $\mathcal{G}_{\boldsymbol{\theta}}$. However, for n true samples and n' generated samples (or a batch-size), each evaluation of $D^2_\kappa(\widehat{\mathcal{P}}_{\mathcal{X}}, \widehat{\mathcal{P}}_{\boldsymbol{\theta}})$ (and its gradient) requires $\mathcal{O}(nn' + n'^2)$ computations. Training MMD-GNs, while conceptually simpler than training GANs, is much slower due to all the pairwise evaluations of the kernel required at each iteration—especially for modern large-size datasets.

6.2.3 Compressive Learning of Generative Networks

In this work, given a dataset $\mathcal{X}$, we propose to learn a generative network $\mathcal{G}_{\boldsymbol{\theta}}$ using *only* the sketch $\boldsymbol{z}_{\mathcal{X}} = \mathcal{A}(\widehat{\mathcal{P}}_{\mathcal{X}})$ defined in (6.9) (see Fig. 6.3). For this, given n' samples $Z = \{\boldsymbol{z}_i \sim \mathcal{P}_z\}_{i=1}^{n'}$, we solve a *generative network sketch matching problem* that selects $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{z}_{\mathcal{X}})$ with

$$\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{z}_{\mathcal{X}}) := \left\| \mathcal{A}(\widehat{\mathcal{P}}_{\mathcal{X}}) - \mathcal{A}(\mathcal{G}_{\boldsymbol{\theta}}(\widehat{\mathcal{P}}_Z)) \right\|_2^2 = \left\| \boldsymbol{z}_{\mathcal{X}} - \tfrac{1}{n'} \sum_{i=1}^{n'} \Phi(\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{z}_i)) \right\|_2^2. \tag{6.14}$$

From (6.12), we reach $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{z}_{\mathcal{X}}) \simeq D^2_\kappa(\widehat{\mathcal{P}}_{\mathcal{X}}, \mathcal{G}_{\boldsymbol{\theta}}(\widehat{\mathcal{P}}_Z))$ for large values of m, as established from the link relating κ and Λ. Compared to the exact MMD in (6.13), $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{z}_{\mathcal{X}})$ is, however, much easier to optimize. Once the dataset sketch $\boldsymbol{z}_{\mathcal{X}}$ has been pre-computed (in one single pass over $\mathcal{X}$, possibly in parallel), we only need to compute $\mathcal{A}(\mathcal{G}_{\boldsymbol{\theta}}(\widehat{\mathcal{P}}_Z))$ (*i.e.*, by computing n' contributions $\boldsymbol{z}_i \to \Phi(\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{z}_i))$ by feed-forward, before averaging them) to compute the Euclidean distance between both quantities. In short, we access $\mathcal{X}$ only once then discard it, and evaluating the cost has complexity $\mathcal{O}(n')$, *i.e.*, potentially[1] much smaller than $\mathcal{O}(nn' + n'^2)$, the complexity of the exact MMD (6.13).

Equally importantly, the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{z}_{\mathcal{X}})$ is easily computed. With

[1] Depending on the sketch size m.

the residual $\boldsymbol{r} := \boldsymbol{z}_{\mathcal{X}} - \frac{1}{n'}\sum_{i=1}^{n'} \Phi(\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{z}_i))$ and $\boldsymbol{r}^H$ its conjugate transpose,

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{z}_{\mathcal{X}}) = -2 \cdot \tfrac{1}{n'}\textstyle\sum_{i=1}^{n'} \Re\big[\boldsymbol{r}^H \big(\frac{\partial \Phi(\boldsymbol{u})}{\partial \boldsymbol{u}}\big|_{\boldsymbol{u}=\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{z}_i)} \cdot \frac{\partial \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{z}_i)}{\partial \boldsymbol{\theta}}\big)\big]. \tag{6.15}$$

Above, $\frac{\partial \Phi(\boldsymbol{u})}{\partial \boldsymbol{u}} = \frac{\mathrm{i}}{\sqrt{m}} \mathrm{diag}\,(e^{\mathrm{i}\boldsymbol{\Omega}^\top \boldsymbol{u}})\,\boldsymbol{\Omega}$ is the $m \times d$ Jacobian matrix listing the partial derivatives of the m sketch entries with respect to the d dimension of $\boldsymbol{u} \in \Sigma$, which is evaluated at the generated samples $\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{z}_i)$. The last term $\frac{\partial \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{z}_i)}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{d \times d_{\boldsymbol{\theta}}}$ is computed by the back-propagation algorithm as it contains the derivative of the network output $\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{z}_i)$ (for $\boldsymbol{z}_i$ *fixed*) with respect to the parameters $\boldsymbol{\theta} \in \mathbb{R}^{d_{\boldsymbol{\theta}}}$. Algorithmically, the feature function Φ amounts to an extra layer on top of the GN, with fixed weights $\boldsymbol{\Omega}$ and activation $t \mapsto \exp(\mathrm{i}t)$. We can then plug those expressions in any gradient-based optimisation solver[2].

6.2.4 Experiments

For this preliminary work, we visually illustrate the soundness of minimizing (6.14) by considering three 2-D synthetic datasets made of $n = 10^5$ samples (see the top row of Fig. 6.4): *(i)* a 2-D spiral $\{(r_i, \phi_i)\}_{i=1}^n$, with $\phi_i \sim_{\text{i.i.d.}} \mathcal{U}([0, 2\pi))$ and $r_i \sim_{\text{i.i.d.}} \frac{\phi_i}{2\pi} + \mathcal{N}(0, \sigma_r^2)$, *(ii)* a Gaussian mixture models of 6 Gaussians, and *(iii)* samples in a circle, *i.e.*, $\phi_i \sim \mathcal{U}([0, 2\pi))$ and $r_i \sim R + \mathcal{N}(0, \sigma_r^2)$ for R and σ_r fixed. We learn a GN mapping $10-$dimensional random Gaussian vectors to $\mathbb{R}^2$, passing through seven fully connected hidden layers of 10 units each, activated by a Leaky ReLU function with slope 0.2. For this simple illustration, we sketch all datasets to a sketch of size $m = n/10 = 10^4$. We found experimentally from a few trials that setting Λ to a folded Gaussian distribution (see [KBGP18]) of scale $\sigma^2 = 10^{-3}$ is appropriate to draw the m frequencies $\{\boldsymbol{\omega}_j\}_{j=1}^m$. From those sketches, we then trained our generators according to (6.14), using the `keras` framework. We fixed the number of generated samples to $n' = 10^5$, which we split into mini-batches of $n_b = 1000$ samples when computing the gradient.

Fig. 6.4 compares the densities of generated samples and re-generated samples after the training (from the known densities) through their 2-D histograms. Note that while the datasets are simplistic, we restricted the training time to a few minutes and, except for the frequency distribution,

[2] To boost the evaluations of (6.15), we can split Z into several minibatches Z_b of size $n_b \ll n'$; (6.15) is then replaced by successive minibatch gradients evaluated on the batches Z_b. As reported for MMD-GNs [LSZ15, D+15], this only works for sufficiently large n_b, *e.g.*, $n_b = 1000$ in Sec. 6.2.4.

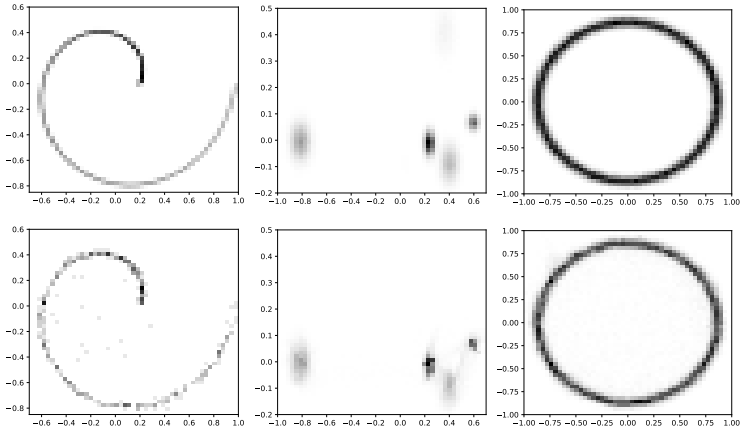


Fig. 6.4 Histograms (varying from white to black as the number of samples increases) of considered datasets (top) and 50000 generated samples, after training from the sketch (bottom).

no hyper-parameter tuning was performed. Despite a few outliers and missing probability masses, the visual proximity of the histograms proves the capacity of our method to learn complex 2-D distributions.

6.2.5 Discussion

We proposed and tested a method that incorporates compressive learning ideas into generative network training from the Maximum Mean Discrepancy metric. When dealing with large-scale datasets, our approach is *potentially* orders of magnitude faster than exact MMD-based learning.

However, to embrace higher-dimensional applications (*e.g.*, for image restorations or large scale inverse problems), future works will need to *(i)* devise efficient techniques to adjust the kernel κ (*i.e.*, the frequency distribution Λ for RFF sketches, but other feature maps could also be considered) to the dataset $\mathcal{X}$, and *(ii)* determine theoretically the required sketch size m in function of the dataset distribution $\mathcal{P}_0$. As already mentioned in the discussion of the previous section, it is not fully clear how to deal with the problem of sketching high-dimensional distributions such as collections of images. As for the required sketch size, this problem certainly relates to measuring the "complexity" of the true generating density $\mathcal{P}_0$, and to the general open question of why over-parametrized deep neural networks generalize so well.

Remark 6.2. In [HAP20] (which was uploaded as a preprint only a few months after we submitted this work to ESANN), the authors independently derived the same CL-GN approach, combined with a differential privacy-protecting layer (as in Chapter 5). Their experiments also tackle the generation of simple images (MNIST and fashion-MNIST), which seems to indicate that the approach presented in this section is indeed applicable to medium-dimensional data as well.

We conclude this section by an interesting remark on (6.14). While usual CL requires closed form expressions for $\mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}})$ and $\nabla_{\boldsymbol{\theta}}\mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}})$, our GN formalism actually estimates those quantities by Monte-Carlo sampling, *i.e.*, replacing $\mathcal{P}_{\boldsymbol{\theta}}$ by $\widehat{\mathcal{P}}_{\boldsymbol{\theta}}$. This idea could be applied to other CL tasks where $\mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}})$ and/or $\nabla_{\boldsymbol{\theta}}\mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}})$ cannot be computed in closed-form.

6.3 When Compressive Learning fails

Compressive learning usually (*e.g.*, compressive k-means or GMM) requires solving a *non-convex optimization* problem, hence in practice approximate heuristics (such as `CLOMPR`) are used. In this work we explore, by numerical simulations, properties of this non-convex optimization landscape and those heuristics.

6.3.1 Motivation

Context In general, the sketch of a distribution $\mathcal{P}$ is a linear operator, constructed as a set of generalized moments $\mathcal{A}(\mathcal{P}) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\,\Phi(\boldsymbol{x})$ where the feature map Φ is typically randomly generated; the sketch of a dataset is then the *empirical* average those features. In this work we focus, on the random Fourier features sketch, associated with the feature map $\Phi(\boldsymbol{x}) := e^{\mathrm{i}\boldsymbol{\Omega}^\top\boldsymbol{x}}$, *i.e.*, the sketch of the dataset $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ is

$$\boldsymbol{z} := \mathcal{A}_\Phi\left(\tfrac{1}{n}\textstyle\sum_{i=1}^n \delta_{\boldsymbol{x}_i}\right) = \tfrac{1}{n}\textstyle\sum_{i=1}^n e^{\mathrm{i}\boldsymbol{\Omega}^\top\boldsymbol{x}_i} \in \mathbb{C}^m, \tag{6.16}$$

where $\delta_{\boldsymbol{x}}$ is the Dirac measure at $\boldsymbol{x}$ and $\boldsymbol{\Omega} = (\boldsymbol{\omega}_j \in \mathbb{R}^d)_{j=1}^m$ is a set of m "frequencies" generated i.i.d. according to some law Λ. To avoid oversampling large frequencies (due to the curse of dimensionality), a good heuristic [KBGP18] is to set $\boldsymbol{\omega} = R\boldsymbol{\varphi}$, where $\boldsymbol{\varphi} \sim \mathcal{U}(S^{d-1})$ is a normed random direction and $R \sim p_R(R;\sigma)$ is the norm of $\boldsymbol{\omega}$. In [KBGP18], this latter distribution is either (FG) a folded Gaussian $p_R \propto e^{-(\sigma R)^2}$, or (AR)

the adapted radius distribution defined as $p_R \propto \left((\sigma R)^2 + \frac{(\sigma R)^4}{4}\right)^{\frac{1}{2}} e^{-(\sigma R)^2}$. In both cases σ is a *scale parameter*, which should be adjusted to the current dataset either by prior knowledge or from a fast heuristic (see Sec. 6.3.3).

Learning some target parameters $\boldsymbol{\theta}$ from the sketch is then formulated as an inverse problem, where the "signal" to recover is a density $\mathcal{P}_{\boldsymbol{\theta}}$ (see Section 2.5). More precisely, the model $\widehat{\boldsymbol{\theta}}$ estimated from $\boldsymbol{z}$ is found using the "sketch matching" cost function $\mathcal{C}$,

$$\widehat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}} \mathcal{C}(\boldsymbol{\theta}; \boldsymbol{z}) \text{ where } \mathcal{C}(\boldsymbol{\theta}; \boldsymbol{z}) := \|\boldsymbol{z} - \mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}})\|_2^2. \tag{6.17}$$

The cost $\mathcal{C}(\boldsymbol{\theta}; \boldsymbol{z})$ is non-convex, hence challenging to optimize exactly. In practice, a heuristic algorithm is used to solve it approximately, *i.e.*, a *decoder* $\Delta : \boldsymbol{z} \mapsto \boldsymbol{\theta}_\Delta \simeq \arg\min_{\boldsymbol{\theta}} \mathcal{C}(\boldsymbol{\theta}; \boldsymbol{z})$. For k-means and GMM, the standard decoder is `CLOMPR`. It starts from an empty solution and greedily adds new atoms $\boldsymbol{\theta}'_k$ (*i.e.*, a single Gaussian or centroid) to the current solution $\boldsymbol{\theta}$, where new atoms $\boldsymbol{\theta}'_k$ are found by maximizing the non-convex criterion $\langle \mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}'_k}), \boldsymbol{z} - \mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}}) \rangle$, starting from a random intial point. To increase the chances of success, T trials of `CLOMPR` can be run independently and the solution with the lowest cost is selected, an approach we call `CLOMPRxT`. This decoder showed convincing empirical success for both GMM [KBGP18] and k-means [KTTG17]. For k-means specifically, the `CLAMP` decoder (Compressive Learning with Approximate Message Parsing) was also been proposed [BCGS19], which allowed to recover the centroids with lower sketch sizes, but in this work we focus mainly on `CLOMPR`.

Problem statement Previous works thus observed that compressive learning (*e.g.*, using `CLOMPR`) performed well under the right circumstances (*e.g.*, when the sketch size m is large enough, and its frequency sampling pattern Λ is well-chosen). *But is it possible to improve the existing CL schemes further? And if so, how? To perform constructive research on these question, we must first understand when and why existing compressive learning schemes fail.* In this work, we provide some insights on this question, based on observations from numerical simulations.

6.3.2 Methodology: CL failure scenarii

In Fig. 6.5, we classify the different outcomes $(a) - (e)$ that are possible when one runs a compressive learning decoder Δ on a given sketch $\boldsymbol{z}$, arranged by "what could go wrong". If the sketch was poorly designed, then

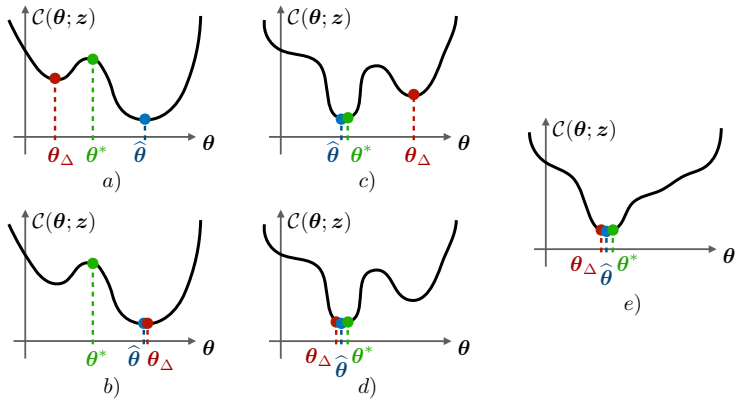


Fig. 6.5 Five scenarii in compressive learning: the cost function $\mathcal{C}$ (determined by the sketch $\boldsymbol{z}$) can be bad (when $\widehat{\boldsymbol{\theta}} \neq \boldsymbol{\theta}^*$, cases (a) and (b)), acceptable (when $\widehat{\boldsymbol{\theta}} \simeq \boldsymbol{\theta}^*$, but with a small basin of attraction, cases (c) and (d)), or good ($\widehat{\boldsymbol{\theta}} \simeq \boldsymbol{\theta}^*$ with a large basin of attraction, case (e)). Given this cost function, the decoder Δ can either fail to find the global optimum (when $\boldsymbol{\theta}_\Delta \neq \widehat{\boldsymbol{\theta}}$, cases (a) and (c)) or "succeed" (when $\boldsymbol{\theta}_\Delta \simeq \widehat{\boldsymbol{\theta}}$, the latter being possibly different from $\boldsymbol{\theta}^*$, cases (b), (d) and (e)).

$\mathcal{C}$ does not indicate (through its global minimum $\widehat{\boldsymbol{\theta}}$) the desired parameters $\boldsymbol{\theta}^*$, as shown in cases $(a) - (b)$. In [GBKT17], the authors derive theoretical guarantees on (an upper bound for) the probability that this failure occurs, as a function of the sketch size m and the compatibility between the sketching function and the target learning task (see also Chapter 4).

However, even if the sketch matching cost function aligns with the ideal parameters, it is possible that the decoder fails to solve the non-convex problem, as shown in case (c) (the existing decoders do not have any guarantee). Whether or not the decoder succeeds depends on the general shape of the cost $\mathcal{C}$ (besides the position of its global optimum). In `CLOMPR` for example, convergence to $\widehat{\boldsymbol{\theta}}$ is ensured if the random initialization point falls into its *basin of attraction*. It was shown (for the centroid selection in k-means) that the size of this basin of attraction increases with the sketch size m [TA19]. To understand how to improve current CL schemes, we would like to classify practical failures of decoders into either scenarii $(a) - (b)$ (where we know that performance improvement is possible—and, for (b), necessary—by changing the sketch function) or scenario (c) (where we know that improvement is possible by changing the decoder).

In what follows, we analyze how this classification is modified when two parameters of the sketch are affected, namely the sketch size m and scale parameter σ in the frequency sampling. Our general strategy consists in focusing on controlled toy problems, where the true parameters $\boldsymbol{\theta}^*$ are known, and to compare the value of the cost function at this solution $\mathcal{C}(\boldsymbol{\theta}^*; \boldsymbol{z})$ with the one obtained by the decoder $\mathcal{C}(\boldsymbol{\theta}_\Delta; \boldsymbol{z})$. If $\mathcal{C}(\boldsymbol{\theta}^*; \boldsymbol{z}) < \mathcal{C}(\boldsymbol{\theta}_\Delta; \boldsymbol{z})$, we know for sure that the decoder failed $\boldsymbol{\theta}_\Delta \neq \widehat{\boldsymbol{\theta}}$, which provides a useful clue in identifying the relevant scenario.

6.3.3 Experimental results

Influence of the sketch size In this section we focus on the k-means problem, where the goal is to position K centroids $\boldsymbol{c}_k$ such that the Sum of Squared Errors, $\mathrm{SSE}(\{\boldsymbol{c}_k\}_{k=1}^K) := \sum_i \min_k \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2$, is minimized. In Fig. 6.6 we plot the quality of compressively learned centroids (by `CLOMPR` with 10 trials) as a function of m; as expected from previous works, they are always perfectly recovered when $m \geq 10Kd$, which corresponds to case (e). We also compared the cost of those centroids $\mathcal{C}(\boldsymbol{\theta}_{\texttt{CLOMPRx10}})$ with the cost of the ground-truth centroids $\mathcal{C}(\boldsymbol{\theta}^*)$, and reported the average number of times that $\mathcal{C}(\boldsymbol{\theta}_{\texttt{CLOMPRx10}}) > \mathcal{C}(\boldsymbol{\theta}^*)$. This allows us to identify an additional regime: when $m \leq Kd$, we know that we are most probably in case $(a-b)$, *i.e.*, the cost is ill-defined (note that here the condition $m \leq Kd$ coincides with the over-parameterized regime where there are less measurements than degrees of freedom). In between ($1 \leq m/Kd \leq 10$), the situation is less clear: as the number of measurements increase, we transition from $(a-b)$ to (c) with gradually more (d), then finally to (e), but it's not clear at which point the $(a)-(b)$ situation switches to (c).

To investigate this transition further, we perform another experiment, where this time we keep $\boldsymbol{\Omega}$ fixed to analyze only the performance of the decoder for a fixed cost: this is shown Fig. 6.7. Moreover, we would like to approach the global minimizer $\widehat{\boldsymbol{\theta}}$ as much as possible. For this purpose, we propose a new genetic-algorithm-based decoder, that we name `GenetiCL`. Its principle is to maintain a population of "chromosome" candidate solutions $\boldsymbol{\theta}$ that explore the optimization landscape by random "mutations" (noise) and combinations by "crossover" (swapping centroids) based on a fitness score that we set to be $\|\boldsymbol{z} - \mathcal{A}(\mathcal{P}_{\boldsymbol{\theta}})\|_2^{-\gamma}$ for $\gamma > 0$. While orders of magnitude slower than `CLOMPR` (we strongly discourage using it for anything other than very specific research purposes), it finds more promising (local) minimizers to $\mathcal{C}$, see Fig. 6.7. For $m \geq 2Kd$, `GenetiCL` performs bet-

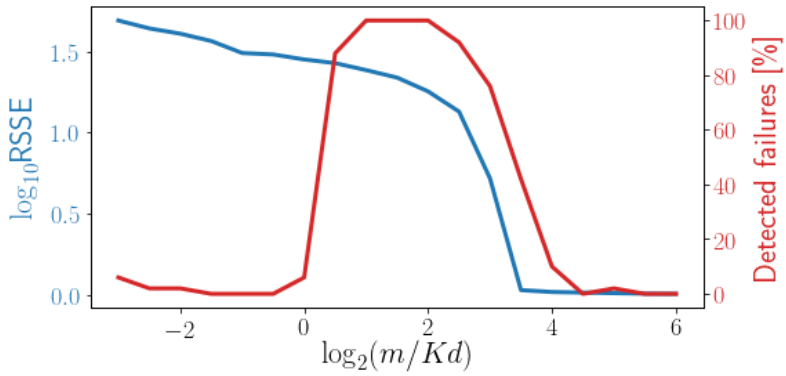


Fig. 6.6 **Blue**: relative SSE (with respect to `k-means`) of the compressively learned centroids $\boldsymbol{\theta}_{\texttt{CLOMPRx10}}$, as a function of the sketch size m (log scale, median over 50 draws of $\boldsymbol{\Omega}$). **Red**: number of times a failure of the type $\mathcal{C}(\boldsymbol{\theta}_{\texttt{CLOMPRx10}}; \boldsymbol{z}) > \mathcal{C}(\boldsymbol{\theta}^*; \boldsymbol{z})$ was detected. The data are $n = 5 \times 10^4$ points drawn from $K = 10$ Gaussians in $d = 10$.

ter, and we are for sure in case (c) for `CLOMPR`. In this region, we can hope that better decoders would succeed, although the existence of an efficient one (*i.e.*, not `GenetiCL`) is an open question. However, uncertainty remains in the zone $Kd \leq m \leq 2Kd$, as the ground truth cost is lower than what the decoders can find, hence it's not certain whether we are in case (c) or (a).

Influence of the scale and task We now analyse how the cost function $\mathcal{C}$ changes with the sketch scale σ, as well as with the considered task (solving k-means or GMM from a same sketch), which is shown Fig. 6.8. First, notice that there exist scales ($\log_{10}(\sigma) \in [0, 0.5]$ in our case) where k-means succeeds but GMM fails. Intuitively, the culprit is the sketch (case (b)), which captures a very coarse view of the data distribution; it cannot estimate the clusters shape precisely, but can still locate them. Conversely, there are also scales ($\log_{10}(\sigma) \in [-3, -1.5]$ in our case) where GMM succeeds but k-means fails. This is more surprising, as the successfully extracted GMM contains[3] the centroids (in the means of the Gaussians); we thus intuitively expect that the decoder failed (case (c)), which would mean that switching the task from k-means to GMM leads to a better cost function (going from (c) to (e)). Plotting the detected failures (as

[3]Since the Gaussian clusters are well-separated.

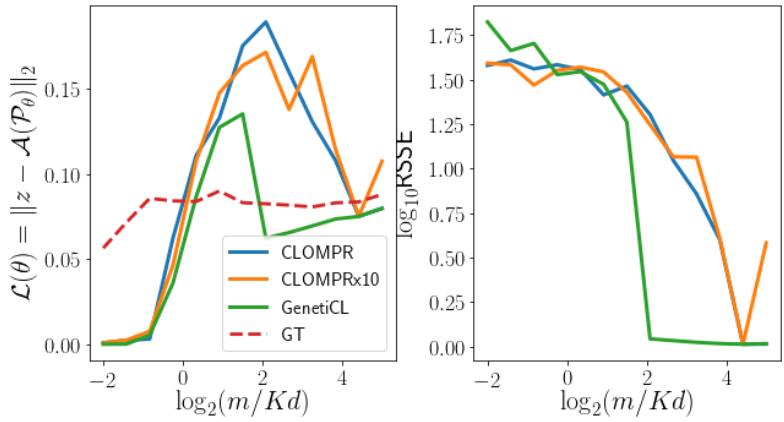


Fig. 6.7 **Left**: cost function $\mathcal{C}(\boldsymbol{\theta}; \boldsymbol{z})$ as a function of an increasing sketch size m (for a fixed draw of $\boldsymbol{\Omega}$), evaluated on the centroids $\boldsymbol{\theta}_\Delta$ obtained by different decoders ($\Delta \in \{\texttt{CLOMPR}, \texttt{CLOMPRx10}, \texttt{GenetiCL}\}$) as well as the on "ground-truth" centroids $\boldsymbol{\theta}^*$. **Right**: relative SSE corresponding to those centroids $\boldsymbol{\theta}_\Delta$. Data are $n = 10^5$ samples from $K = 10$ Gaussians in $d = 5$.

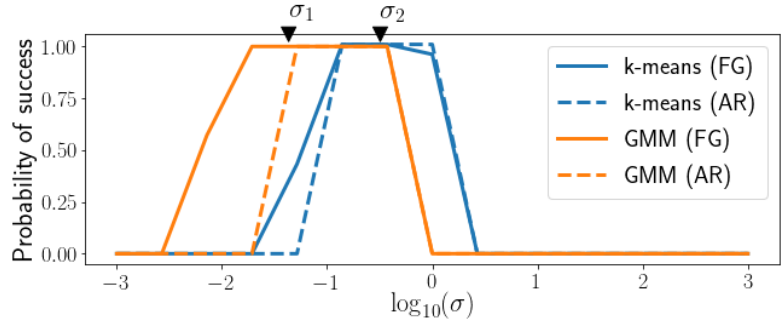


Fig. 6.8 Empirical success rate of `CLOMPRx3` when solving k-means (blue, defining success as RSSE $\leq$ 1.3) and GMM fitting (orange, defining success as $\mathcal{P}_{\boldsymbol{\theta}}(X) \geq (1.3)^{-1}\mathcal{P}_{\boldsymbol{\theta}^*}(X)$ with $\mathcal{P}(X)$ the likelihood of X given $\mathcal{P}$) from a sketch with $m = 20Kd$ frequencies drawn from the FG (plain) and AR (dashed) distributions with varying scale σ. The heuristic proposed in [KBGP18] for GMM (resp. in [BCGS19] for k-means) yields σ_1 (resp. σ_2), indicated by black triangles. Data are $n = 10^5$ samples from $K = 6$ Gaussians in $d = 5$.

in Fig. 6.6, not shown here for conciseness) supports these explanations: when $\log_{10}(\sigma) \in [-3, -1.5]$ we systematically detect failures of the k-means decoder (meaning that (c) is possible), but when $\log_{10}(\sigma) \in [0, 0.5]$ no such failures are detected from the GMM decoder (meaning we are for sure in $(a) - (b)$).

7

Conclusion

On the second of October 2017, sitting on the bus for Louvain-la-Neuve to start my first day as a PhD student, my head was full of wild ambitions about how I would revolutionize compressive learning, which would in turn revolutionize machine learning. While neither of those revolutions happened (of course[1]), it is time to take a step back and reflect on what was (actually) done, what could be improved, and why compressive learning still has not taken over the world (yet?).

7.1 Summary and perspectives of the contributions

Chapter 3: Asymmetric Random Periodic Features

Summary: This chapter was about *random periodic features* (RPF), a family of feature maps where signals $\boldsymbol{x} \in \Sigma$ are projected onto random vectors $\boldsymbol{\omega}_j \sim \Lambda$, offset by a random dither $\boldsymbol{\xi} \sim \mathcal{U}^m([0, 2\pi))$, and passed through a generic 2π-periodic function f, *i.e.*, $\boldsymbol{z}_f(\boldsymbol{x}) := \frac{1}{\sqrt{m}} f(\boldsymbol{\Omega}^\top \boldsymbol{x} + \boldsymbol{\xi})$. More specifically, we studied *asymmetric RPF*, where the periodic map $f \neq g$ can be different for the two signals $\boldsymbol{x}, \boldsymbol{x}' \in \Sigma$ that are being compared. We studied the geometry induced by the dot product $\langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_g(\boldsymbol{x}') \rangle \simeq \kappa_{f,g}(\boldsymbol{x}, \boldsymbol{x}')$ where $\kappa_{f,g}(\boldsymbol{x}, \boldsymbol{x}')$ is some approximated kernel.

[1] One thing that I quickly discovered was that, apparently, you cannot revolutionize a field on your own. A lot of research is required for that, and research, well, takes a lot of time.

We showed how $\kappa_{f,g}$ relates to κ (the Fourier transform of Λ and the shift-invariant kernel approximated by RFF, when f and g are a cosine or complex exponential), which is a scale mixture of κ weighted by the Fourier series of f and g. We then studied how the kernel approximation error $|\langle z_f(\boldsymbol{x}), z_g(\boldsymbol{x}')\rangle - \kappa_{f,g}(\boldsymbol{x}, \boldsymbol{x}')|$ evolves with respect to the embedding dimension m. In particular, we showed how to control the error uniformly (*i.e.*, for all $\boldsymbol{x}, \boldsymbol{x}' \in \Sigma$) over an infinite but compact set Σ. Special efforts were dedicated to obtain results that are valid even when f and/or g are *discontinuous*, as long as they are smooth "on average" (where the averaging is obtained through the dither $\boldsymbol{\xi}$), a notion we formalized as the *mean Lipschitz smoothness property*.

As concrete application of this strategy, we explored *semi-quantized kernel machines*, *i.e.*, kernel methods that are built on the random Fourier features approximation but where one of the signals being compared (*e.g.*, the test vector or the dataset vectors) is binarized to one-bit features. This potentially has a huge interest in terms of storage and transmission costs. We showed through numerical simulations that the kernel approximation suffers only slightly from the binarization of one of the two features, and highlighted the interest of the strategy in a learning context (*e.g.*, on hyperspectral classification data).

Perspectives: As already mentioned at the end of that chapter, the kernel approximation capabilities alone are not enough to use this scheme effectively in practice. It is indeed important to take the kernel approximation error into account while tuning the parameters $\boldsymbol{\theta}$ as well as the hyperparameters of the related kernel method (*e.g.*, the regularization strength and the kernel bandwidth). While one can of course still use a validation set (where it is possible to apply the asymmetric RPF at test time), to guide the choice of parameters, this method is somewhat indirect. A better solution would be to incorporate the asymmetry directly during the training stage (while searching for the optimal parameters $\boldsymbol{\theta}$). However, the best way to do this (ideally, with associated statistical learning guarantees) is still an open question (*e.g.*, would it be meaningful to use an asymmetric Gram matrix $K_{i,j} = \widetilde{\kappa}_{f,g}(\boldsymbol{x}_i, \boldsymbol{x}_j)$?).

It is also worth mentioning that the asymmetric features strategy opens many possibilities beyond those studied in Chapter 3, where one of the two periodic maps was systematically $g(\cdot) = \cos(\cdot)$ or $g(\cdot) = \exp(\mathrm{i}\cdot)$ in order to recover a specific kernel κ—that is assumed a priori to be the target. A systematic approach to the asymmetric RPF strategy could for instance

simultaneously design the frequency sampling pattern Λ and the two periodic maps f and g (instead of sequentially, as we did). This co-design would then be guided by specific constraints on the specific devices that must respectively evaluate f and g (*e.g.*, a client and server), and of course the generalization capabilities of the resulting kernel method (*i.e.*, based on $\kappa_{f,g}$, which would here not necessarily be given by $\kappa = \mathcal{F}^{-1}\Lambda$).

Chapter 4: Quantized Sketching with Guarantees

Summary: This chapter can somehow be seen as the "sketched" equivalent of the previous one. It introduced a generic relaxation of compressive learning, called *asymmetric compressive learning*, where the feature map during the sketching stage $\Psi \neq \Phi$ is allowed to differ from Φ, the one used during the learning stage. More formally, we compute the dataset sketch as $\boldsymbol{z}_{\Psi,\mathcal{X}}$ but learn from it by optimizing $\mathcal{C}_\Phi(\boldsymbol{\theta}; \boldsymbol{z}_{\Psi,\mathcal{X}})$ (where the Φ subscript denotes that the cost function depends on Φ). We identified a sufficient condition, the Limited Projected Distortion (LPD) property, which guarantees that asymmetric CL is not much worse than usual (symmetric) CL. Intuitively, the LPD ensures that the sketch computed from Ψ or from Φ is not too different along the directions that are relevant in evaluations the cost functions $\mathcal{C}_\Phi(\boldsymbol{\theta}; \boldsymbol{z}_{\Psi,\mathcal{X}})$ and $\mathcal{C}_\Phi(\boldsymbol{\theta}; \boldsymbol{z}_{\Phi,\mathcal{X}})$, respectively.

We then focused on *RPF sketches*, computed as the dataset average of random periodic features (see Chapter 3). In order to obtain guarantees for compressive learning (where $\Phi = \Phi_{\text{RFF}}$ is the usual random Fourier features map, commonly used in CL) from such sketches, we introduced a hack to only have to prove the LPD on the space of signals $\boldsymbol{x} \in \Sigma$ rather than the space of probability distributions $\mathcal{P} \in \mathcal{G}, \mathcal{Q} \in \widehat{\mathcal{G}}$. We were then able to leverage the results from the previous chapter in order to guarantee the LPD (and thus, the overall statistical learning guarantees) for CL with RPF sketches.

While developing those generic results, our main motivation was to study *quantized sketching*, where the sketch contributions $\Psi(\boldsymbol{x}_i)$ are binarized random Fourier features. This approach is interesting to further reduce the computational cost of sketching from large-scale datasets, *e.g.*, using dedicated hardware implementations. To confirm the power of this approach, we performed experiments in a wide variety of settings, solving both k-means and GMM from quantized sketches while incurring only a small loss of performance with respect to "full-precision sketches".

Perspectives: We already deplored at the end of that chapter the fact that our argument relied on the low-complexity nature of a *signal set* Σ (which, to be leveraged in the LPD, requires to enforce additional constraints on the learning procedure) instead of the low-complexity nature of the *model set* $\mathcal{G}$ (which is the only assumption needed in symmetric CL [GBKT17]). Although we argued that such constraints are reasonable in known practical applications of CL, a more elegant solution would thus be to prove the LPD (or another sufficient property) using only the latter assumption, which would make our results for asymmetric CL completely analogous to those of symmetric CL. However, our several attempts towards this direction remain unsuccessful so far. An impossibility result could maybe be a consolation price?

On a more cheerful note, several novel applications of our generic results on asymmetric CL seem to be worth exploring. First, the LPD could be used to formally analyze other types of "distortions" applied to the sketch feature map (beyond quantization). For example, our framework could be used to analyze the impact of the *subsampling mechanism* proposed in [CSH+21], where the sketching feature map Ψ is a masked version of the learning feature map Φ (although the random nature of the masking operation must be carefully handled, and the "signal-LPD hack" would here be severely sub-optimal). Second, in our results we focused on distortions of $\Phi = \Phi_{\text{RFF}}$ the RFF feature map (used for compressive k-means and GMM), but other feature maps Φ and related distortions (*e.g.*, quantization) could be worth studying in this setting, such as the random quadratic features (used for compressive PCA).

Chapter 5: Private Sketching

Summary: This chapter focused on the aspects I contributed to in a collaboration, which explored the addition of a (differential) privacy-protecting layer to the sketching phase. Indeed, since the contribution of a specific individual to the sketch gets "drowned" into the average operation of the sketch, one expects the sketch to be a "privacy-friendly" summary of the dataset. To formalize this intuitive claim, we considered the golden standard definition of privacy, *i.e.*, *differential privacy*, which ensures that the output of a (random) data-processing algorithm depends negligibly on the presence or absence of any individual in the dataset—as captured by a *privacy parameter* ϵ. The lower ϵ, the stronger the privacy guarantee, but also the higher the risk of degrading the *utility* of the algorithm's output.

We thus considered a *noise addition mechanism* on top of the sketching phase, which adds well-calibrated Laplacian (resp. Gaussian) noise to the sketch to ensure pure (resp. approximate) differential privacy of its contributors. To precisely calibrate this noise (which is moreover inversely proportional to the privacy parameter ϵ), we established the *sensitivity* of the sketching feature map. While we presented only a direct upper bound (for the L^1 sensitivity) in this thesis, a significant part of our efforts was dedicated to show that this bound is in fact sharp, *i.e.*, the sensitivity bound cannot be improved.

We compared the practical interest of our private sketching mechanism by comparing it in numerical experiments to state-of-the-art privacy-preserving machine learning algorithms on the k-means and Gaussian mixture modeling tasks. Our experiments suggest that our approach outperforms (*i.e.*, in terms of *privacy-utility tradeoff*) preceding approaches in several regimes, although with the caveat that the choice of the hyperparameters is tricky for *all* the involved methods.

On top of those promising results, private sketching presents several desirable features, which were also discussed. A first example is the ability to—almost trivially—handle the *distributed* privacy-preserving machine learning context. Another promising attribute is the possibility to re-use the private sketch *ad infimum* without weakening the privacy guarantee, which is ensured "once and for all". This stands in sharp contrast with the majority of existing approaches, which rely on a multitude of specific queries to the datasets (one for each iteration of some algorithm), each of those requiring some part of the precious privacy budget ϵ.

Perspectives: To achieve differential privacy, noise addition to the sketch seems to be the best available solution (we explored several other schemes without much success), for which we already characterized the optimal noise level in most practical use-cases. However, differential privacy can be a very strong requirement in practice. Intuitively, even without noise addition, almost no "information"[2] about a particular individual can be inferred from the sketch alone[3], an intuition which could be formalized by other notions of privacy (*e.g.*, based on mutual information).

Interestingly, if we consider the private sketching mechanism *solely* as a privacy-enhancing tool, we somehow gain the freedom to consider more

[2]This however requires additional assumptions on the side information available to the attacker.

[3]Especially when combined with the feature subsampling strategy from [CSH+21].

expensive methods to extract information from the sketch (as the computational cost aspect is now secondary). Based on this premise, we tentatively explored (not reported in this manuscript) a heuristic to compute arbitrary *moments* of the dataset, *i.e.*, any quantity of the form $F = \frac{1}{n}\sum_{\boldsymbol{x}_i \in \mathcal{X}} f(\boldsymbol{x}_i)$ for some generic function $f : \Sigma \to \mathbb{R}$, from a private sketch. The underlying scenario is that a data analyst, who already acquired a given private sketch, might try to squeeze out as much information from it as possible using this method. Preliminary results show that this approach can sometimes give accurate answers, but a more thorough analysis is still missing.

Chapter 6: Extensions to novel tasks

Summary: Three contributions were grouped into this chapter. First, we presented and interpreted a simple strategy to perform **compressive clasification**, *i.e.*, computing *one sketch* $\boldsymbol{z}_{\mathcal{X}_k}$ *per class* k (which amounts to sketching the different conditional distributions $\mathcal{P}(\boldsymbol{x}|k)$ independently). To classify new signals $\boldsymbol{x}'$, we pick the class whose sketch $\boldsymbol{z}_{\mathcal{X}_k}$ is the most correlated with its features $\Phi(\boldsymbol{x}')$. We argued that—under some law of large numbers approximations—this scheme can be interpreted as *maximum a posteriori* classification performed in the reproducible kernel Hilbert space associated with the sketch kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle$. We empirically validated this approach on toy example and classic datasets. We also suggested the idea of using randomly weighted (convolutional) neural networks as feature map that is better suited to image data, which indeed outperformed plain random Fourier features in our simulations.

As a second task extension, we described a strategy to **compressively learn generative networks** using only the dataset sketch (CL-GN). Our scheme is intimately related to the moment matching generative network approach (MMD-GN) [LSZ15,D+15], which it approximates in expectation (over the draw of the sketch feature map). Compared to MMD-GNs, *provided the required sketch size m is sufficiently small*, our approach should be significantly cheaper computation-wise. We performed a proof-of-concept validation of the approach on 2-d toy example datasets, and noted that more extensive experiments on this same scheme were later performed in [HAP20].

The last contribution of that chapter was the description and demonstration of generic techniques to investigate **why compressive learning might fail** in practice. Is it because the sketch $\boldsymbol{z}$ is poorly designed (*e.g.*, its size m or scale σ is not adapted to the data and problem) or because the

heuristic compressive learning algorithm (*i.e.*, the *decoder*, such as CLOMP) did not find the global optimum of its nonconvex cost function? The main technique we presented relies on comparing the value of the cost function at the obtained solution $\mathcal{C}(\boldsymbol{\theta}_\Delta; \boldsymbol{z})$ to the one at the ground-truth solution $\mathcal{C}(\boldsymbol{\theta}^*; \boldsymbol{z})$. Another one was to compare the decoder to one that is closer to brute-force optimization of the cost, which is more expensive but might be more successful in finding the global solution (here based on genetic optimization). We demonstrated how this approach allows to obtain new insights on the selection of the sketch parameters m and σ.

Perspectives: For **compressive classification**, we already highlighted that the scheme seems limited due to the difficulty of finding an appropriate feature map Φ (*i.e.*, an appropriate kernel). There are however a few steps that could be taken to shed some light on the question. For example, it might be judicious to first further investigate the un-sketched version of this classifier, namely the Parzen-windows classifier, to obtain a better idea of the possible performance. Results from kernel mean embeddings [MFSS17] should be particularly useful in this regard. Another modified scheme that might provide insights on the potential of compressive classification is a relaxation where we are allowed to *learn the matrix of projections* $\boldsymbol{\Omega}$ (and thus the kernel), *e.g.*, in a neural network fashion, which provides an upper bound on the achievable performance.

The idea to use randomly weighted neural networks as sketch feature map might be worth further investigations as well. There are however several points that make the analysis of this feature map significantly harder than the usual random Fourier features. From a pragmatic point of view, the space of design choices becomes much larger, as one has to pick a network architecture, as well as the random initialization distributions for all the weights in the network. Moreover, one loses the direct relation between the generation of the feature map and the approximated kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle$ (*i.e.*, the Fourier transform in the case of RFF), which might further complicate the theoretical analysis of the DNN-features approach. Recent results from neural tangent kernels might prove to be helpful in this regard [JGH18].

For **compressive learning of generative networks**, beyond the same difficulties in the design of Φ, the main open question is to determine the *sketch size* m which is required for the approach to work (if m must be too large to learn accurate GNs in practice, the computational advantages might be nullified, except for very-large-scale datasets). As generative net-

works are typically over-parameterized, this seems to be a challenging task as the learned model $\mathcal{P}_{\boldsymbol{\theta}}$ is not "sparse", which stands in sharp contrast with the current framework to obtain CL guarantees [GBKT17]. The *implicit regularization* phenomenon which is currently researched in deep neural networks literature can maybe play a role here.

Beside the techniques we presented to investigate **why compressive learning fails**, there is another idea that might be of interest, when no "failures" of the decoder are detected but the compressive learning algorithm still fails. This additional technique consists in numerically optimizing the cost $\mathcal{C}(\boldsymbol{\theta})$ by providing the ground-truth solution $\boldsymbol{\theta}^*$ as initialization point. If the optimization solver steers away from $\boldsymbol{\theta}^*$, we know that we are in a scenario where the cost function is poorly designed (cases $(a) - (b)$), *i.e.*, the sketch $\boldsymbol{z}$ should be improved.

We expect that the simple techniques we described to "debug" compressive learning decoders (comparison of the cost functions and empirical risk of $\boldsymbol{\theta}_\Delta$ and $\boldsymbol{\theta}^*$, using a computationally expensive but potentially more accurate decoder, initialization of the decoder at $\boldsymbol{\theta}^*$) might be of general interest to CL researchers, and might be especially useful in particular in the quest to extend CL to novel tasks.

7.2 Final thoughts: compressive learning, hype or hope?

Compressive learning is a promising tool to tackle the problem of learning from large-scale datasets with limited computational resources. This thesis broadened the scope of this field by introducing asymmetric compressive learning, privacy-aware compressive learning, and some leads on how to perform compressive classification and generative networks.

However, it must be acknowledged that CL is still rather limited in its applications. Of course, some progress has been made recently, both on the theoretical side [GBKT20, SGD19] as well as on the practical side, *e.g.*, tackling the tasks of PCA [Cha20] or ICA [SKD19]. But many of the most important machine learning techniques (*e.g.*, supervised learning) still cannot be convincingly solved by CL.

The main explanation, I believe, is that for the "core" CL methods—mixture model density fitting and k-means, the latter being a special case of the former—it so happens that *the stars are extremely well aligned*. In particular, the *random Fourier features* sketch $\mathcal{A}_{\Phi_{\text{RFF}}}$ is particularly appropriate to density fitting tasks, because (i) in density fitting tasks the data samples are assumed to live in Euclidean space $\mathbb{R}^d$, for which shift-invariant ker-

nels κ^Δ (such as the Gaussian kernel) are meaningful. In addition, (ii) the existence of a large body of work on kernel mean embeddings [MFSS17] provides numerous insights and theoretical tools to study properties of the sketch-induced distance $\|\mathcal{A}_{\Phi_{\text{RFF}}}(\mathcal{P}_{\boldsymbol{\theta}}) - \mathcal{A}_{\Phi_{\text{RFF}}}(\widehat{\mathcal{P}}_{\mathcal{X}})\|_2$, which is a particularly relevant quantity in density fitting tasks (see [GBKT20]).

Moreover, the *low-complexity prior* $\mathcal{P}_{\boldsymbol{\theta}}$ for mixture models (a linear combination of a few simple "atomic" probability distributions) is also particularly favorable. This is because (iii) it enjoys a strong connection with compressive sensing theory, from which it can thus draw many inspiration and insights (*e.g.*, the OMP-based recovery procedure), and finally (iv) when combined with the RFF, the atoms of main density fitting problems (Diracs or Gaussians) have simple closed-form expression for $\mathcal{A}_{\Phi_{\text{RFF}}}(\mathcal{P}_{\boldsymbol{\theta}})$ and $\nabla_{\boldsymbol{\theta}}\mathcal{A}_{\Phi_{\text{RFF}}}(\mathcal{P}_{\boldsymbol{\theta}})$, quantities required by the recovery procedure.

I would argue that the impressive success of compressive learning for density fitting (*i.e.*, GMM, k-means) is highly dependent on the combination of all the advantages (i)-(iv). As soon as either the sketch feature map Φ is changed (*e.g.*, to work on structured data, or to capture other types of information from the data than its distribution $\mathcal{P}_0$, as in semi-parametric CL [SGD19]) or if the low-complexity prior $\mathcal{P}_{\boldsymbol{\theta}}$ is changed (*e.g.*, to solve a different task), those advantages are lost. As a consequence, CL then becomes conceptually much more difficult.

This does not mean that all hope is lost for compressive learning. If sufficient patience and research efforts are devoted to it, I believe that in the long run CL can truly become a valuable tool in real-life applications of large-scale machine learning. However, it is probably worth keeping the vulnerabilities of the CL framework in mind (*e.g.*, to losing advantages (i)-(iv)) to guide further research efforts in the field.

Bibliography

[A+17] Martin Arjovsky et al. Wasserstein GAN. *arXiv preprint:1701.07875*, 2017.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[ABC+20] Ahmet Aktay, Shailesh Bavadekar, Gwen Cossoul, John Davis, Damien Desfontaines, Alex Fabrikant, Evgeniy Gabrilovich, Krishna Gadepalli, Bryant Gipson, Miguel Guevara, et al. Google covid-19 community mobility reports: Anonymization process description (version 1.0). *arXiv preprint arXiv:2004.04145*, 2020.

[ACH+13] Pankaj K Agarwal, Graham Cormode, Zengfeng Huang, Jeff M Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Transactions on Database Systems (TODS)*, 38(4):1–28, 2013.

[ADHP09] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

[ADR16] Ulaş Ayaz, Sjoerd Dirksen, and Holger Rauhut. Uniform recovery of fusion frame structured sparse signals. *Applied and Computational Harmonic Analysis*, 41(2):341–361, 2016.

[Agg07] Charu C Aggarwal. *Data streams: models and algorithms*, volume 31. Springer Science & Business Media, 2007.

[AJYM+15] Omar Y Al-Jarrah, Paul D Yoo, Sami Muhaidat, George K Karagiannidis, and Kamal Taha. Efficient machine learning for big data: A review. *Big Data Research*, 2(3):87–93, 2015.

[ALNT14] Mohammad Abu Alsheikh, Shaowei Lin, Dusit Niyato, and Hwee-Pink Tan. Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Communications Surveys & Tutorials*, 16(4):1996–2018, 2014.

[AN07] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[ARC19] Mohammad Al-Rubaie and J Morris Chang. Privacy-preserving machine learning: Threats and solutions. *IEEE Security & Privacy*, 17(2):49–58, 2019.

[Aro50] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[AS66] Syed Mumtaz Ali and Samuel D Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society: Series B (Methodological)*, 28(1):131–142, 1966.

[AS04] Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2004.

[AV07] David Arthur and Sergei Vassilvitskii. k-means++: The Advantages of Careful Seeding. *ACM-SIAM symposium on Discrete algorithms*, pages 1027–103, 2007.

[AW20] Nur Ahmed and Muntasir Wahed. The de-democratization of AI: Deep learning and the compute divide in artificial intelligence research. *arXiv preprint arXiv:2010.15581*, 2020.

[B+17] Ashish Bora et al. Compressed sensing using generative models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 537–546, 2017.

[Bac17] Francis Bach. On the equivalence between kernel quadrature rules and random feature expansions. *The Journal of Machine Learning Research*, 18(1):714–751, 2017.

[BBL15] Marion F. Baumgardner, Larry L. Biehl, and David A. Landgrebe. 220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3, Sep 2015. URL: https://purr.purdue.edu/publications/1947/1, doi:doi:/10.4231/R7RX991C.

[BCDH10] Richard G Baraniuk, Volkan Cevher, Marco F Duarte, and Chinmay Hegde. Model-based compressive sensing. *IEEE Transactions on information theory*, 56(4):1982–2001, 2010.

[BCGS19] Evan Byrne, Antoine Chatalic, Rémi Gribonval, and Philip Schniter. Sketched clustering via hybrid approximate message passing. *IEEE Transactions on Signal Processing*, 67(17):4556–4569, 2019.

[BD09] Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis*, 27(3):265–274, 2009.

[BDDW08] Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, 2008.

[BDEL03] Shai Ben-David, Nadav Eiron, and Philip M Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3):496–514, 2003.

[BDL+17] Maria-Florina Balcan, Travis Dick, Yingyu Liang, Wenlong Mou, and Hongyang Zhang. Differentially private clustering in high-dimensional euclidean spaces. In *International Conference on Machine Learning*, pages 322–331, 2017.

[BDMN05] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the SuLQ framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138. ACM, 2005.

[BDP+14] Anthony Bourrier, Mike E Davies, Tomer Peleg, Patrick Pérez, and Rémi Gribonval. Fundamental performance limits

for ideal decoders in high-dimensional linear inverse problems. *IEEE Transactions on Information Theory*, 60(12):7928–7946, 2014.

[BG18] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.

[BGC17] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press Massachusetts, USA:, 2017.

[BGLL+20] Karsten Borgwardt, Elisabetta Ghisu, Felipe Llinares-López, Leslie O'Bray, and Bastian Rieck. Graph kernels: State-of-the-art and future challenges. *arXiv preprint arXiv:2011.03854*, 2020.

[BGMMS21] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623, 2021.

[BGV92] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[BHG+04] Peter Bodik, Wei Hong, Carlos Guestrin, Sam Madden, Mark Paskin, and Romain Thibaux. Intel lab data, 2004. Available online at `http://db.csail.mit.edu/labdata/labdata.html`.

[BHK20] Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of data science*. Cambridge University Press, 2020.

[BJKS15] Petros T Boufounos, Laurent Jacques, Felix Krahmer, and Rayan Saab. Quantization and compressive sensing. In *Compressed sensing and its applications*, pages 193–237. Springer, 2015.

[BKR17] Ayush Bhandari, Felix Krahmer, and Ramesh Raskar. On unlimited sampling. In *2017 International Conference on Sampling Theory and Applications (SampTA)*, pages 31–35. IEEE, 2017.

[BLK17] Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017.

[Blo70] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[BM15] Petros T Boufounos and Hassan Mansour. Universal embeddings for kernel machine classification. In *2015 International Conference on Sampling Theory and Applications (SampTA)*, pages 307–311. IEEE, 2015.

[Bou12] Petros T Boufounos. Universal rate-efficient scalar quantization. *IEEE transactions on information theory*, 58(3):1861–1872, 2012.

[BR13] Petros T. Boufounos and Shantanu Rane. Efficient Coding of Signal Distances Using Universal Quantized Embeddings. In *2013 Data Compression Conference*, pages 251–260, March 2013. doi:10.1109/DCC.2013.33.

[BRM17] Petros T. Boufounos, Shantanu Rane, and Hassan Mansour. Representation and coding of signal geometry. *Information and Inference: A Journal of the IMA*, 6(4):349–388, 2017.

[BTS17] Matej Balog, Ilya Tolstikhin, and Bernhard Schölkopf. Differentially private database release via kernel mean embeddings. *arXiv:1710.01641 [stat]*, 2017. URL: http://arxiv.org/abs/1710.01641, arXiv:1710.01641.

[BW18] Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. 2018. URL: http://arxiv.org/abs/1805.06530, arXiv:1805.06530.

[BZD10] Christos Boutsidis, Anastasios Zouzias, and Petros Drineas. Random Projections for k-means Clustering. *CoRR*, abs/1011.4632, 2010. arXiv:1011.4632.

[BZH06] Michael Barbaro, Tom Zeller, and Saul Hansell. A face is exposed for aol searcher no. 4417749. *New York Times*, 9(2008):8For, 2006.

[CAS16] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[CDK17] Surajit Chaudhuri, Bolin Ding, and Srikanth Kandula. Approximate query processing: No silver bullet. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 511–519, 2017.

[CDS01] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[CGHJ12] Graham Cormode, Minos Garofalakis, Peter J Haas, and Chris Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1–3):1–294, 2012.

[CGK18] Antoine Chatalic, Rémi Gribonval, and Nicolas Keriven. Large-Scale High-Dimensional Clustering with Fast Sketching. *ICASSP 2018 - IEEE International Conference on Acoustics, Speech and Signal Processing*, Apr 2018.

[Cha20] Antoine Chatalic. *Efficient and Privacy-Preserving Compressive Learning. (Méthodes efficaces pour l'apprentissage compressif avec garanties de confidentialité)*. PhD thesis, University of Rennes 1, France, 2020. URL: `https://tel.archives-ouvertes.fr/tel-03103525`.

[CJS09] Robert Calderbank, Sina Jafarpour, and Robert Schapire. Compressed learning: Universal sparse dimensionality reduction and learning in the measurement domain. *preprint*, 2009.

[CM05] Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[CMS11] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. 12:1069–1109, 2011.

[CP11] Emmanuel J Candes and Yaniv Plan. Tight oracle inequalities for low-rank matrix recovery from a minimal number of noisy random measurements. *IEEE Transactions on Information Theory*, 57(4):2342–2359, 2011.

[CRPW12] Venkat Chandrasekaran, Benjamin Recht, Pablo A Parrilo, and Alan S Willsky. The convex geometry of linear inverse problems. *Foundations of Computational mathematics*, 12(6):805–849, 2012.

[CRT06] Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006. URL: `http://dx.doi.org/10.1002/cpa.20124`, `doi:10.1002/cpa.20124`.

[CRW17] Krzysztof Marcin Choromanski, Mark Rowland, and Adrian Weller. The unreasonable effectiveness of structured random orthogonal embeddings. In *NIPS*, 2017.

[CS09] Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, pages 342–350, 2009.

[CS20a] Benjamin Coleman and Anshumali Shrivastava. A one-pass private sketch for most machine learning tasks. *arXiv preprint arXiv:2006.09352*, 2020.

[CS20b] Benjamin Coleman and Anshumali Shrivastava. Sub-linear race sketches for approximate kernel density estimation on streaming data. In *Proceedings of The Web Conference 2020*, pages 1739–1749, 2020.

[CSH+21] Antoine Chatalic, Vincent Schellekens, Florimond Houssiau, Yves-Alexandre de Montjoye, Laurent Jacques, and Rémi Gribonval. Compressive learning with privacy guarantees. *Information and Inference*, 2021.

[Csi67] Imre Csiszár. Information-type measures of difference of probability distributions and indirect observation. *studia scientiarum Mathematicarum Hungarica*, 2:229–318, 1967.

[CT05] Emmanuel J Candes and Terence Tao. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215, 2005.

[D+15] Gintare Karolina Dziugaite et al. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint:1505.03906*, 2015.

[Dau88] Ingrid Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 41(7):909–996, 1988.

[DBVB16] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. FMA: A dataset for music analysis. In *ISMIR*, 2016. `arXiv:1612.01840`.

[DBWB10] Mark A Davenport, Petros T Boufounos, Michael B Wakin, and Richard G Baraniuk. Signal processing with compressive measurements. *IEEE Journal of Selected topics in Signal processing*, 4(2):445–460, 2010.

[DCL08] Wei Dong, Moses Charikar, and Kai Li. Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 123–130, 2008.

[DDT+08] Marco F Duarte, Mark A Davenport, Dharmpal Takhar, Jason N Laska, Ting Sun, Kevin F Kelly, and Richard G Baraniuk. Single-pixel imaging via compressive sampling. *IEEE signal processing magazine*, 25(2):83–91, 2008.

[DG10] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: a flexible data processing tool. *Communications of the ACM*, 53(1):72–77, 2010.

[DH+73] Richard O Duda, Peter E Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[Dir19] Sjoerd Dirksen. Quantized compressed sensing: a survey. In *Compressed Sensing and Its Applications*, pages 67–95. Springer, 2019.

[DJW14] John C Duchi, Michael I Jordan, and Martin J Wainwright. Privacy aware learning. *Journal of the ACM (JACM)*, 61(6):38, 2014.

[DKBM21] Mauricio Delbracio, Damien Kelly, Michael S Brown, and Peyman Milanfar. Mobile computational photography: A tour. *arXiv preprint arXiv:2102.09000*, 2021.

[dMHVB13] Yves-Alexandre de Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3:1376, 2013.

[DMNS] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. Theory of Cryptography Conf.*, pages 265–284.

[dMRS+15] Yves-Alexandre de Montjoye, Laura Radaelli, Vivek Kumar Singh, et al. Unique in the shopping mall: On the reidentifiability of credit card metadata. *Science*, 347(6221):536–539, 2015.

[DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210. ACM, 2003.

[Don06] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[DP19] Damien Desfontaines and Balázs Pejó. Sok: differential privacies. *arXiv preprint arXiv:1906.01337*, 2019.

[DR] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. 9(3-4):211–407.

[DS20] Sjoerd Dirksen and Alexander Stollenwerk. Binarized johnson-lindenstrauss embeddings. *arXiv preprint arXiv:2009.08320*, 2020.

[Dup18] Elsa Dupraz. K-means algorithm over compressed binary data. In *Data compression conference (DCC)*, 2018.

[Dwo08a] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[Dwo08b] Cynthia Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[Eur] European Parliament and Council of European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), Chapter I, Article 4. URL: `https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN`.

[EW15] Armin Eftekhari and Michael B Wakin. New analysis of manifold embeddings and signal recovery from compressive measurements. *Applied and Computational Harmonic Analysis*, 39(1):67–109, 2015.

[FFGM07] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Discrete Mathematics and Theoretical Computer Science*, pages 137–156. Discrete Mathematics and Theoretical Computer Science, 2007.

[FFKN09] Dan Feldman, Amos Fiat, Haim Kaplan, and Kobbi Nissim. Private coresets. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 361–370. ACM, 2009. URL: `http://doi.acm.org/10.1145/1536414.1536465`, `doi:10.1145/1536414.1536465`.

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[FM85] Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.

[FR17] Simon Foucart and Holger Rauhut. A mathematical introduction to compressive sensing. *Bull. Am. Math*, 54:151–165, 2017.

[FSGS08] Kenji Fukumizu, Bharath K Sriperumbudur, Arthur Gretton, and Bernhard Schölkopf. Characteristic kernels on groups and semigroups. In *NIPS*, pages 473–480, 2008.

[FWCY10] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4):14:1–14:53, June 2010. URL: `http://doi.acm.org/10.1145/1749603.1749605`, `doi:10.1145/1749603.1749605`.

[FXZR17] Dan Feldman, Chongyuan Xiang, Ruihao Zhu, and Daniela Rus. Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks. In *Information Processing in Sensor Networks (IPSN), 2017 16th ACM/IEEE International Conference on*, pages 3–16. IEEE, 2017.

[G+14] Ian Goodfellow et al. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[Gär03] Thomas Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

[GBKT17] Rémi Gribonval, Gilles Blanchard, Nicolas Keriven, and Yann Traonmilin. Compressive Statistical Learning with Random Feature Moments. *ArXiv e-prints*, June 2017. `arXiv:1706.07180`.

[GBKT20] Rémi Gribonval, Gilles Blanchard, Nicolas Keriven, and Yann Traonmilin. Statistical learning guarantees for compressive

clustering and compressive mixture modeling. *arXiv preprint arXiv:2004.08085*, 2020.

[GBR+12] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[GCCJ99] JA Gualtieri, Samir R Chettri, RF Cromp, and LF Johnson. Support vector machine classifiers as applied to AVIRIS data. In *Proc. Eighth JPL Airborne Geoscience Workshop*, 1999.

[GCK+20] Rémi Gribonval, Antoine Chatalic, Nicolas Keriven, Vincent Schellekens, Laurent Jacques, and Philip Schniter. Sketching datasets for large-scale learning (long version). *arXiv preprint arXiv:2008.01839*, 2020.

[GLK+20] Federica Gerace, Bruno Loureiro, Florent Krzakala, Marc Mézard, and Lenka Zdeborová. Generalisation error in learning with random features and the hidden manifold model. *arXiv preprint arXiv:2002.09339*, 2020.

[GLP+13] C Sinan Güntürk, Mark Lammers, Alexander M Powell, Rayan Saab, and Ö Yılmaz. Sobolev duals for random frames and $\sigma\delta$ quantization of compressed sensing measurements. *Foundations of Computational mathematics*, 13(1):1–36, 2013.

[GN98] Robert M. Gray and David L. Neuhoff. Quantization. *IEEE transactions on information theory*, 44(6):2325–2383, 1998.

[GPGL13] Albert Gordo, Florent Perronnin, Yunchao Gong, and Svetlana Lazebnik. Asymmetric distances for binary embeddings. *IEEE transactions on pattern analysis and machine intelligence*, 36(1):33–47, 2013.

[GSB16] Raja Giryes, Guillermo Sapiro, and Alex M Bronstein. Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Transactions on Signal Processing*, 64(13):3444–3457, 2016.

[GZK05] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: a review. *ACM Sigmod Record*, 34(2):18–26, 2005.

[Hal05] Alastair R Hall. *Generalized method of moments*. Oxford University Press, 2005.

[HAP20] Frederik Harder, Kamil Adamczewski, and Mijung Park. Differentially private mean embeddings with random features (dp-merf) for simple & practical synthetic data generation. *arXiv preprint arXiv:2002.11603*, 2020.

[HEM19] LN Hoang and EM El Mhamdi. The fabulous endeavor: Making artificial intelligence robustly beneficial. *EDP Sciences, published in Nov*, 2019.

[HR21] Moritz Hardt and Benjamin Recht. Patterns, predictions, and actions: A story about machine learning. 2021.

[HSS15] Nikhila Haridas, V Sowmya, and KP Soman. Gurls vs libsvm: Performance comparison of kernel methods for hyperspectral image classification. *Indian Journal of Science and Technology*, 8(24):1, 2015.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[Jai10] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[JC17] Laurent Jacques and Valerio Cambareri. Time for dithering: fast and quantized random embeddings via the restricted isometry property. *Information and Inference: A Journal of the IMA*, 6(4):441–476, 2017.

[JDJ19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

[JDS10] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

[JFL15] Oren N Jaspan, Roman Fleysher, and Michael L Lipton. Compressed sensing mri: a review of the clinical literature. *The British journal of radiology*, 88(1056):20150487, 2015.

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8580–8589, 2018.

[JL84] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[JLBB13] Laurent Jacques, Jason N. Laska, Petros T. Boufounos, and Richard G. Baraniuk. Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. *IEEE Transactions on Information Theory*, 59(4):2082–2102, apr 2013.

[JMF19] Ibrahim Jubran, Alaa Maalouf, and Dan Feldman. Introduction to coresets: accurate coresets. *arXiv preprint arXiv:1910.08707*, 2019.

[JTD11] Prateek Jain, Ambuj Tewari, and Inderjit S Dhillon. Orthogonal matching pursuit with replacement. *arXiv preprint arXiv:1106.2774*, 2011.

[JVB+09] Laurent Jacques, Pierre Vandergheynst, Alexandre Bibet, Vahid Majidzadeh, Alexandre Schmid, and Yusuf Leblebici. Cmos compressed imaging by random convolution. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1113–1116. IEEE, 2009.

[KBGP18] Nicolas Keriven, Anthony Bourrier, Rémi Gribonval, and Patrick Pérez. Sketching for large-scale learning of mixture models. *Information and Inference: A Journal of the IMA*, 7(3):447–508, 2018.

[KDL18] Nicolas Keriven, Antoine Deleforge, and Antoine Liutkus. Blind source separation using mixtures of alpha-stable distributions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 771–775. IEEE, 2018.

[KG18] Nicolas Keriven and Rémi Gribonval. Instance optimal decoding and the restricted isometry property. In *Journal of Physics: Conference Series*, volume 1131, page 012002. IOP Publishing, 2018.

[KKMM13] Krishnaram Kenthapadi, Aleksandra Korolova, Ilya Mironov, and Nina Mishra. Privacy via the Johnson-Lindenstrauss Transform. *Journal of Privacy and Confidentiality*, 5(1), 2013.

[KOV17] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. *IEEE Transactions on Information Theory*, 63(6):4037–4049, 2017.

[KR16] Anurag Kumar and Bhiksha Raj. Features and kernels for audio event recognition. *arXiv preprint arXiv:1607.05765*, 2016.

[KT61] AN Kolmogorov and VM Tihomirov. ϵ-entropy and ϵ-capacity of sets in functional space. *Amer. Math. Soc. Transl.(2)*, 17:277–364, 1961.

[KT18] Krishnaram Kenthapadi and Thanh TL Tran. Pripearl: A framework for privacy-preserving analytics and reporting at linkedin. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2183–2191, 2018.

[KTTG17] Nicolas Keriven, Nicolas Tremblay, Yann Traonmilin, and Rémi Gribonval. Compressive K-means. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6369–6373. IEEE, 2017.

[L+18] Alice Lucas et al. Using deep neural networks for inverse problems in imaging: Beyond analytical methods. *IEEE Signal Processing Magazine*, 35(1):20–36, 2018. URL: `https://doi.org/10.1109%2Fmsp.2017.2760358`, `doi:10.1109/msp.2017.2760358`.

[LCB] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`. Accessed: 2018-01-25.

[LDSP08] Michael Lustig, David L Donoho, Juan M Santos, and John M Pauly. Compressed sensing mri. *IEEE signal processing magazine*, 25(2):72–82, 2008.

[Leg06] Adrien Marie Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes: avec un supplément contenant divers*

perfectionnemens de ces méthodes et leur application aux deux comètes de 1805. Courcier, 1806.

[LGEC17] Alexandra L'heureux, Katarina Grolinger, Hany F Elyamany, and Miriam AM Capretz. Machine learning with big data: Challenges and approaches. *IEEE Access*, 5:7776–7797, 2017.

[LHCS20] Fanghui Liu, Xiaolin Huang, Yudong Chen, and Johan AK Suykens. Random features for kernel approximation: A survey in algorithms, theory, and beyond. *arXiv preprint arXiv:2004.11154*, 2020.

[Li19] Ping Li. Sign-full random projections. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4205–4212, 2019.

[LL19] Xiaoyun Li and Ping Li. Random projections with asymmetric quantization. In *Advances in Neural Information Processing Systems*, pages 10857–10866, 2019.

[Llo82] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[LS19] Zhigang Lu and Hong Shen. A convergent differentially private k-means clustering algorithm. In Qiang Yang, Zhi-Hua Zhou, Zhiguo Gong, Min-Ling Zhang, and Sheng-Jun Huang, editors, *Advances in Knowledge Discovery and Data Mining*, volume 11439, pages 612–624. Springer International Publishing, 2019. URL: `http://link.springer.com/10.1007/978-3-030-16148-4_47`, `doi:10.1007/978-3-030-16148-4_47`.

[LSST+02] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444, 2002.

[LSZ15] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727, 2015.

[LTOS19] Zhu Li, Jean-Francois Ton, Dino Oglic, and Dino Sejdinovic. Towards a unified analysis of random fourier features. In

International Conference on Machine Learning, pages 3905–3914. PMLR, 2019.

[LV07] John A Lee and Michel Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[M+17] Morteza Mardani et al. Deep generative adversarial networks for compressed sensing automates MRI. *arXiv preprint:1706.00051*, 2017.

[Mac11] Chris A Mack. Fifty years of Moore's law. *IEEE Transactions on semiconductor manufacturing*, 24(2):202–207, 2011.

[Mal99] Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

[MB04] Farid Melgani and Lorenzo Bruzzone. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on geoscience and remote sensing*, 42(8):1778–1790, 2004.

[MBDJ20] Amirafshar Moshtaghpour, José M Bioucas-Dias, and Laurent Jacques. Close encounters of the binary kind: Signal reconstruction guarantees for compressive hadamard sampling with haar wavelet basis. *IEEE Transactions on Information Theory*, 66(11):7253–7273, 2020.

[MBP14] Julien Mairal, Francis Bach, and Jean Ponce. Sparse modeling for image and vision processing. *Foundations and Trends® in Computer Graphics and Vision*, 8(2-3):85–283, 2014.

[MCRR20] Giacomo Meanti, Luigi Carratino, Lorenzo Rosasco, and Alessandro Rudi. Kernel methods through the roof: handling billions of points efficiently. *arXiv preprint arXiv:2006.10350*, 2020.

[MFSS17] K Muandet, K Fukumizu, B Sriperumbudur, and B Schölkopf. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends in Machine Learning*, 10(1-2):1–144, 2017.

[MIO11] Giorgos Mountrakis, Jungho Im, and Caesar Ogole. Support vector machines in remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(3):247–259, 2011.

[MLM+20] Brian McFee, Vincent Lostanlen, Alexandros Metsai, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Frank Zalkow, Ayoub Malek, Dana, Kyungyun Lee, Oriol Nieto, Jack Mason, Dan Ellis, Eric Battenberg, Scott Seyfarth, Ryuichi Yamamoto, Keunwoo Choi, viktorandreevichmorozov, Josh Moore, Rachel Bittner, Shunsuke Hidaka, Ziyao Wei, nullmightybofo, Darío Hereñú, Fabian-Robert Stöter, Pius Friesch, Adam Weiss, Matt Vollrath, and Taewoon Kim. librosa/librosa: 0.8.0, July 2020. doi:10.5281/zenodo.3955228.

[MM09] Odalric-Ambrym Maillard and Rémi Munos. Compressed least-squares regression. In *NIPS 2009*, 2009.

[Moo96] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[Mor78] Robert Morris. Counting large numbers of events in small registers. *Communications of the ACM*, 21(10):840–842, 1978.

[MPTJ08] Shahar Mendelson, Alain Pajor, and Nicole Tomczak-Jaegermann. Uniform uncertainty principle for bernoulli and subgaussian ensembles. *Constructive Approximation*, 28(3):277–289, 2008.

[MT07] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103. IEEE, 2007.

[Mül97] Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

[Mur12] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[MZ93] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

[Nat95] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

[NCBN16] Richard Nock, Raphaël Canyasse, Roksana Boreli, and Frank Nielsen. k-variates++: more pluses in the k-means++. In *International Conference on Machine Learning*, pages 145–154, 2016.

[Nie20] Frank Nielsen. An elementary introduction to information geometry. *Entropy*, 22(10):1100, 2020.

[NKB+19] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*, 2019.

[NRS] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84. ACM.

[NS08] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125. IEEE, 2008.

[NSW17] Deanna Needell, Rayan Saab, and Tina Woolf. Simple classification using binary data. *CoRR*, abs/1707.01945, 2017. `arXiv:1707.01945`.

[OA11] Daniel Otero and Gonzalo R Arce. Generalized restricted isometry property for alpha-stable random projections. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3676–3679. IEEE, 2011.

[PB14] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

[PC+19] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.

[PDG17] Gilles Puy, Mike E Davies, and Rémi Gribonval. Recipes for stable linear embeddings from Hilbert spaces to $\mathbb{R}^m$. *IEEE Transactions on Information Theory*, 63(4):2171–2187, 2017.

[PFCW16] Mijung Park, Jimmy Foulds, Kamalika Chaudhuri, and Max Welling. DP-EM: Differentially private expectation maximization. 2016. URL: http://arxiv.org/abs/1605.06995, arXiv:1605.06995.

[Phi16] Jeff M Phillips. Coresets and sketches. *arXiv preprint arXiv:1601.00617*, 2016.

[Pic15] Karol J Piczak. ESC: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018, 2015. Available online at https://github.com/karolpiczak/ESC-50.

[Pis99] Gilles Pisier. *The volume of convex bodies and Banach space geometry*, volume 94. Cambridge University Press, 1999.

[PKP06] Joel B Predd, Sanjeev B Kulkarni, and H Vincent Poor. Distributed learning in wireless sensor networks. *IEEE Signal Processing Magazine*, 23(4):56–69, 2006.

[Poo13] H Vincent Poor. *An Introduction to Signal Detection and Estimation*. Springer Science & Business Media, 2013.

[PRK93] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pages 40–44. IEEE, 1993.

[PV16] Yaniv Plan and Roman Vershynin. The generalized lasso with non-linear observations. *IEEE Transactions on information theory*, 62(3):1528–1537, 2016.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[QWD+16] Junfei Qiu, Qihui Wu, Guoru Ding, Yuhua Xu, and Shuo Feng. A survey of machine learning for big data processing.

EURASIP Journal on Advances in Signal Processing, 2016(1):67, 2016.

[QYL13] Wahbeh Qardaji, Weining Yang, and Ninghui Li. Differentially private grids for geospatial data. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 757–768. IEEE, 2013.

[Rao45] C Radhakrishna Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Reson. J. Sci. Educ*, 20:78–90, 1945.

[Rau10] Holger Rauhut. Compressive sensing and structured random matrices. *Theoretical foundations and numerical methods for sparse recovery*, 9:1–92, 2010.

[RB13] Shantanu Rane and Petros T Boufounos. Privacy-preserving nearest neighbor methods: Comparing signals without revealing them. 30(2):18–28, 2013.

[RC+17] JH Rick Chang et al. One network to solve them all–solving linear inverse problems using deep projection models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5888–5897, 2017.

[RJ03] Jehyuk Rhee and Youngjoong Joo. Wide dynamic range CMOS image sensor with pixel level ADC. *Electronics Letters*, 39(4):360–361, 2003.

[RKL19] Nick Ryder, Zohar Karnin, and Edo Liberty. Asymmetric random projections. *arXiv preprint arXiv:1906.09489*, 2019.

[RL09] Maxim Raginsky and Svetlana Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Advances in neural information processing systems*, pages 1509–1517, 2009.

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[RR08] Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. Curran Associates, Inc., 2008.

[RR17] Alessandro Rudi and Lorenzo Rosasco. Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems*, pages 3215–3225, 2017.

[RT19] Amir Rosenfeld and John K Tsotsos. Intriguing properties of randomly weighted networks: Generalizing while learning next to nothing. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 9–16. IEEE, 2019.

[Rud62] Walter Rudin. *Fourier Analysis on Groups*. Interscience Publishers, 1962.

[S+10] Bharath K. Sriperumbudur et al. Hilbert Space Embeddings and Metrics on Probability Measures. *J. Mach. Learn. Res.*, 11:1517–1561, August 2010. URL: `http://dl.acm.org/citation.cfm?id=1756006.1859901`.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[SBS05] Dave Steinkraus, Ian Buck, and PY Simard. Using GPUs for machine learning algorithms. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 1115–1120. IEEE, 2005.

[SC13] Anand D Sarwate and Kamalika Chaudhuri. Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data. 30(5):86–94, 2013.

[SCC+16] Alaa Saade, Francesco Caltagirone, Igor Carron, Laurent Daudet, Angélique Drémeau, Sylvain Gigan, and Florent Krzakala. Random projections through multiple optical scattering: Approximating kernels at the speed of light. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6215–6219. IEEE, 2016.

[Sch17] Vincent Schellekens. Compressive clustering of high-dimensional datasets by 1-bit sketching, 2017. Master thesis.

[SCH+19a] Vincent Schellekens, Antoine Chatalic, Florimond Houssiau, Y-A De Montjoye, Laurent Jacques, and Rémi Gribonval. Differentially private compressive k-means. In *ICASSP 2019-*

2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7933–7937. IEEE, 2019.

[SCH+19b] Vincent Schellekens, Antoine Chatalic, Florimond Houssiau, Yves-Alexandre de Montjoye, Laurent Jacques, and Rémi Gribonval. Compressive k-means with differential privacy. In *SPARS Workshop. July*, volume 1, 2019.

[Sch20] Vincent Schellekens. Pycle: a python compressive learning toolbox, 2020.

[SCL+16] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. Differentially private k-means clustering. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 26–37. ACM, 2016.

[Set09] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[SFG+09] Bharath K Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert RG Lanckriet. On integral probability metrics,\phi-divergences and binary classification. *arXiv preprint arXiv:0901.2698*, 2009.

[SGD19] Michael P Sheehan, Antoine Gonon, and Mike E Davies. Compressive learning for semi-parametric models. *arXiv preprint arXiv:1910.10024*, 2019.

[SGM19] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[SGSS07] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*, pages 13–31. Springer, 2007.

[SGV98] Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. 1998.

[SH19] Viraj Shah and Chinmay Hegde. Signal reconstruction from modulo observations. In *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1–5. IEEE, 2019.

[Sha48] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[SHS01] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

[SJ18a] Vincent Schellekens and Laurent Jacques. Compressive classification (machine learning without learning). *arXiv preprint arXiv:1812.01410*, 2018.

[SJ18b] Vincent Schellekens and Laurent Jacques. Quantized compressive k-means. *IEEE Signal Processing Letters*, 25(8):1211–1215, 2018.

[SJ20a] Vincent Schellekens and Laurent Jacques. Breaking the waves: asymmetric random periodic features for low-bitrate kernel machines. *arXiv preprint arXiv:2004.06560*, 2020.

[SJ20b] Vincent Schellekens and Laurent Jacques. Compressive learning of generative networks. In *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN, 2020)*, 2020.

[SJ20c] Vincent Schellekens and Laurent Jacques. When compressive learning fails: blame the decoder or the sketch? *arXiv preprint arXiv:2009.08273*, 2020.

[SJ21] Vincent Schellekens and Laurent Jacques. Asymmetric compressive learning guarantees with applications to quantized sketches. *arXiv preprint XXXX*, 2021.

[SKD19] Michael P Sheehan, Madeleine S Kotzagiannidis, and Mike E Davies. Compressive independent component analysis. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5. IEEE, 2019.

[SMF10] Jean-Luc Starck, Fionn Murtagh, and Jalal M Fadili. *Sparse image and signal processing: wavelets, curvelets, morphological diversity*. Cambridge university press, 2010.

[SS15a] Bharath Sriperumbudur and Zoltán Szabó. Optimal rates for random Fourier features. In *Advances in Neural Information Processing Systems*, pages 1144–1152, 2015.

[SS15b] Dougal J Sutherland and Jeff Schneider. On the error of random Fourier features. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence (UAI*, pages 862–871, 2015.

[SSB+02] Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. 2002.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[Ste56] Hugo Steinhaus. Sur la division des corps matériels en parties. *Bulletin de l'Académie polonaise des sciences*, 1(804):801, 1956.

[Ste06] Douglas Steinley. K-means clustering: a half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, 59(1):1–34, 2006.

[STN96] Richard Schreier, Gabor C Temes, and Steven R Norsworthy. *Delta-sigma data converters: theory, design, and simulation*. IEEE press, 1996.

[Swe02] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[Szá37] Otto Szász. Fourier series and mean moduli of continuity. *Transactions of the American Mathematical Society*, 42(3):366–395, 1937.

[T+17] DP Team et al. Learning with privacy at scale. *Online at: https://machinelearning. apple. com/2017/12/06/learning-with-privacy-at-scale. html*, 2017.

[TA19] Yann Traonmilin and Jean-François Aujol. The basins of attraction of the global minimizers of the non-convex sparse spike estimation problem. *Inverse Problems*, 2019.

[TGLM20] Neil C Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F Manso. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 2020.

[Tur50] Alan Mathison Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

[TVBM19] Matteo Testa, Diego Valsesia, Tiziano Bianchi, and Enrico Magli. *Compressed Sensing for Privacy-Preserving Data Processing*. Springer, 2019.

[TVV+17] Abhradeep Guha Thakurta, Andrew H Vyrros, Umesh S Vaishampayan, Gaurav Kapoor, Julien Freudinger, Vipul Ved Prakash, Arnaud Legendre, and Steven Duplinsky. Emoji frequency detection and deep link frequency, July 11 2017. US Patent 9,705,908.

[TYD+20] Dat Thanh Tran, Mehmet Yamaç, Aysen Degerli, Moncef Gabbouj, and Alexandros Iosifidis. Multilinear compressive learning. *IEEE transactions on neural networks and learning systems*, 2020.

[Val84] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Vap99] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

[VC71] Vladimir Naumovich Vapnik and Aleksei Yakovlevich Chervonenkis. On uniform convergence of the frequencies of events to their probabilities. *Teoriya Veroyatnostei i ee Primeneniya*, 16(2):264–279, 1971.

[VD20] Jan Van Dijk. *The network society*. Sage, 2020.

[VdMH08] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[VEB10] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854, 2010.

[Ver12] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. In Yonina C. Eldar and Gitta Kutyniok, editors, *Compressed Sensing*, pages 210–268. Cambridge University Press, 2012.

[Ver18] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[Vil08] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[VL07] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[VLB18] Andrea Vedaldi, Mathias Lux, and Marco Bertini. Matconvnet: Cnns are also for matlab users. *ACM SIGMultimedia Records*, 10(1):9–9, 2018.

[War65] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[WE18] Isabel Wagner and David Eckhoff. Technical privacy metrics: a systematic survey. *ACM Computing Surveys (CSUR)*, 51(3):57, 2018.

[Wik72] Ingemar Wik. Criteria for absolute convegence of Fourier series of functions of bounded variation. *Transactions of the American Mathematical Society*, 163:1–24, 1972.

[Wil98] Edward O Wilson. *Consilience: The Unity of Knowledge*. 1998.

[WWP+16] Yuncheng Wu, Yao Wu, Hui Peng, Juru Zeng, Hong Chen, and Cuiping Li. Differentially private density estimation via gaussian mixtures model. In *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, pages 1–6. IEEE, 2016.

[WYX17] Di Wang, Minwei Ye, and Jinhui Xu. Differentially private empirical risk minimization revisited: Faster and more general. In *Advances in Neural Information Processing Systems*, pages 2722–2731, 2017.

[WYZ16] W. Wang, L. Ying, and J. Zhang. On the relation between identifiability, differential privacy, and mutual-information privacy. *IEEE Transactions on Information Theory*, 62(9):5018–5029, 2016. doi:10.1109/TIT.2016.2584610.

[WZ10] Larry Wasserman and Shuheng Zhou. A statistical framework for differential privacy. *Journal of the American Statistical Association*, 105(489):375–389, 2010.

[XJ20] Chunlei Xu and Laurent Jacques. Quantized compressive sensing with rip matrices: The benefit of dithering. *Information and Inference: A Journal of the IMA*, 9(3):543–586, 2020.

[XLX17] Haozhe Xie, Jie Li, and Hanqing Xue. A survey of dimensionality reduction techniques based on random projection. *arXiv preprint arXiv:1706.04371*, 2017.

[Xu14] Zhi-Wei Xu. Cloud-sea computing systems: Towards thousand-fold improvement in performance per watt for the coming zettabyte era. *Journal of Computer Science and Technology*, 29(2):177–181, 2014.

[YKJC08] Young-Gyu Yoon, Jaewook Kim, Tae-Kwang Jang, and SeongHwan Cho. A time-based bandpass ADC using time-interleaved voltage-controlled oscillators. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 55(11):3571–3581, 2008.

[YLM+12] Tianbao Yang, Yu-Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. *Advances in neural information processing systems*, 25:476–484, 2012.

[ZBH+17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=Sy8gdB9xx.

[ZLW09] S. Zhou, K. Ligett, and L. Wasserman. Differential privacy with compression. In *2009 IEEE International Symposium on Information Theory*, pages 2718–2722, 2009. doi:10.1109/ISIT.2009.5205863.

[ZMDR18] Jian Zhang, Avner May, Tri Dao, and Christopher Ré. Low-Precision Random Fourier Features for Memory-Constrained Kernel Approximation. *arXiv preprint arXiv:1811.00155*, 2018.

[ZXY+13] Jun Zhang, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, and Marianne Winslett. PrivGene: differentially private model fitting using genetic algorithms. In *Proceedings of the 2013 international conference on Management of data - SIGMOD '13*, page 665. ACM Press, 2013. URL: `http://dl.acm.org/citation.cfm?doid=2463676.2465330`, `doi:10.1145/2463676.2465330`.

A

Useful quantities about periodic functions

In Part I, we study random features (and the resulting sketch operator) defined by a generic periodic function f. In the context of this analysis, several quantities about f must be computed: its Fourier Series (FS) coefficients $\{F_k\}_{k\in\mathbb{Z}}$, several of its L_p norms (in particular for $p = 2$ and ∞), and its mean Lipschitz smoothness L_f^μ, introduced in Section 3.4. Among others, we apply those results to particular cases where f is one of the following functions (the definitions are reminded below): the cosine, the complex exponential, the (real- or complex-valued) one-bit universal quantization, and the (real- or complex-valued) normalized modulo function. For convenience, this appendix gathers all those results, along with short proofs when relevant.

A.1 Reminder on the relevant definitions

We consider generic (possibly complex-valued) periodic functions $f : \mathbb{R} \to \mathbb{C}$, that are assumed to be bounded. Without loss of generality (more on this in the next subsection), we focus in particular on the case where f is normalized such that its fundamental period is 2π, and it is centered (zero-mean) $\int_0^{2\pi} f(t)\,\mathrm{d}t = 0$.

A.1.1 Quantities to compute

We can then decompose $f(t) = \sum_{k=-\infty}^{\infty} F_k e^{ikt}$ as a **Fourier Series** (FS), where the coefficients F_k are

$$F_k := \frac{1}{2\pi} \int_0^{2\pi} f(t) e^{-ikt} \, \mathrm{d}t, \quad \forall k \in \mathbb{Z}. \tag{A.1}$$

Note that if f is centered, we have $F_0 = 0$.

Another quantity of interest is the **2-norm** of f, noted $\|f\|_2$ (or simply written $\|f\|$). It is defined as

$$\|f\|_2 := \left(\frac{1}{2\pi} \int_0^{2\pi} |f(t)|^2 \, \mathrm{d}t \right)^{1/2}, \tag{A.2}$$

and can moreover be expressed as $\|f\|_2^2 = \sum_k |F_k|^2$ (Parseval's theorem). Another norm of interest is the **infinity norm**, *i.e.*, the largest absolute value of f,

$$\|f\|_\infty := \sup_{t \in [0,2\pi]} |f(t)|. \tag{A.3}$$

Finally, we recall the **mean Lipschitz smoothness** constant L_f^μ we introduced in Chapter 3, defined as[1] the largest local deviation of f *on average*:

$$L_f^\mu := \sup_{0<\delta\leq\pi} \frac{1}{\delta} \frac{1}{2\pi} \int_0^{2\pi} \sup_{-\delta \leq r \leq \delta} |f(t+r) - f(t)| \, \mathrm{d}t. \tag{A.4}$$

A.1.2 Normalization properties

Many different renormalizations of f can be performed. Since we know from experience that it is easy to get lost in the different normalizations, we remind some basic facts here.

Lemma A.1 (Rescaling). *For some real scalars α, β, t_0, if $g(t) := \alpha f(t - t_0) + \beta$, we have*

- *(Fourier Series) $G_k = \alpha e^{-ikt_0} F_k + \beta \delta_k$;*
- *(norms) $\|g\|_p = \alpha \|f\|_p$, provided $\beta = 0$;*
- *(mean Lipschitz smoothness) $L_g^\mu = \alpha L_f^\mu$.*

One operation we also often perform is to compose a complex-valued function g by adding two phase-shifted copies of a real-valued function f (which implies conjugate symmetry in the Fourier domain, *i.e.*, $F_{-k} = F_k^*$).

[1] Here we seek the *lowest* possible constant such that the mean Lipschitz property holds.

Lemma A.2 (Complex version of a real-valued periodic function). *If* $g(t) := f(t) + \mathrm{i}f(t - \frac{\pi}{2})$ *for* $f : \mathbb{R} \to \mathbb{R}$, *we have*

- *(Fourier Series)* $G_k = F_k(1 + e^{\mathrm{i}\frac{\pi}{2}(1-k)})$, *and in particular* $G_1 = 2F_1$ *and* $G_{-1} = 0$;
- *(norms)* $\|g\|_p \leq 2\|f\|_p$ *(notice the inequality sign);*
- *(mean Lipschitz smoothness)* $L_g^\mu \leq 2L_f^\mu$ *(idem).*

A.2 Computing the constants

A.2.1 Trigonometric functions

Let us start with the basic cosine function.

Lemma A.3 (Cosine). *Let* $f(t) = \cos(t)$, *it satisfies*

- *(Fourier Series)* $F_k = \frac{1}{2}\delta_{k,-1} + \frac{1}{2}\delta_{k,+1}$ (i.e., $1/2$ *for* $k = \pm 1$ *and* 0 *for all other* k), *in particular* $F_1 = \frac{1}{2}$;
- *(2-norm)* $\|f\|_2 = \frac{1}{\sqrt{2}}$;
- *(infinity norm)* $\|f\|_\infty = 1$;
- *(mean Lipschitz smoothness)* $L_f^\mu = \frac{1}{2\pi}\int_0^{2\pi} |\sin(t)|\,\mathrm{d}t = \frac{2}{\pi} \leq 1$.

For the complex exponential, we have

Lemma A.4 (Complex exponential). *Let* $f(t) = \exp(\mathrm{i}t)$, *it satisfies*

- *(Fourier Series)* $F_k = \delta_{k,1}$ (i.e., $F_1 = 1$ *and* 0 *for all other* k);
- *(2-norm)* $\|f\|_2 = 1$;
- *(infinity norm)* $\|f\|_\infty = 1$;
- *(mean Lipschitz smoothness)* $L_f^\mu = 1$.

A.2.2 One-bit universal quantization

We recall the (real) one-bit universal quantization function $q : \mathbb{R} \mapsto \mathbb{R}$

$$q(t) := \mathrm{sign} \circ \cos(t) = \begin{cases} +1 & \text{if } 2\pi k - \frac{\pi}{2} \leq t \leq 2\pi k + \frac{\pi}{2} \text{ for some } k \in \mathbb{Z}, \\ -1 & \text{else.} \end{cases} \tag{A.5}$$

It is, up to a re-scaling (see Lemma A.1), the least significant bit of a standard scalar quantizer. Or geometrically, it is a square wave. One can show the following.

Lemma A.5 (Periodic quantization, real)**.** *Let $q(t)$ be as in* (A.5)*, we have*

- *(Fourier Series) given by*

$$Q_k = \begin{cases} \frac{2}{k\pi}(-1)^{\frac{k-1}{2}} & \text{if } k \text{ odd}, \\ 0 & \text{if } k \text{ even}, \end{cases} \tag{A.6}$$

  and in particular, $Q_1 = \frac{2}{\pi}$;

- *(infinity norm) $\|q\|_\infty = 1$;*

- *(mean Lipschitz smoothness) $L_f^\mu = \frac{4}{\pi}$ (proof in Prop. 3.19).*

When considering the complex extension of this quantization function, $q_{\mathbb{C}}(t) := \operatorname{sign}(\exp(\mathrm{i}t)) = q(t) + \mathrm{i}q(t - \frac{\pi}{2})$, with sign acting independently on the real and imaginary part, we obtain the following quantities.

Lemma A.6 (Periodic quantization, complex)**.** *Let $q_{\mathbb{C}}(t) := q(t) + \mathrm{i}q(t - \frac{\pi}{2})$, we have*

- *(Fourier Series) given by*

$$Q_{\mathbb{C},k} = \begin{cases} \frac{4}{k\pi} & \text{if } k = 4l + 1 \text{ for some } l \in \mathbb{Z} \\ 0 & \text{if } k \text{ even}, \end{cases} \tag{A.7}$$

  and in particular, $Q_{\mathbb{C},1} = \frac{4}{\pi}$;

- *(infinity norm) $\|q_{\mathbb{C}}\|_\infty = \sqrt{2}$;*

- *(mean Lipschitz smoothness) $L_{q_{\mathbb{C}}}^\mu = \frac{8}{\pi}$.*

Proof. For the Fourier Series, use Lemma A.2. For the infinity norm, $\|\widetilde{q}\|_\infty = \sqrt{1^2 + 1^2} = \sqrt{2}$, since $q(t) \in \{\pm 1 \pm \mathrm{i}\}$ for all t. The mean Lipschitz smoothness can be computed as in Prop. 3.19, but where the "stumps" in I_δ appear twice as much (one stump appearing around each value $l\frac{\pi}{2}$ for $l \in \mathbb{Z}$). □

A.2.3 Modulo function

We recall the (real) normalized modulo function, $\mathrm{mod} : \mathbb{R} \mapsto \mathbb{R}$,

$$\mathrm{mod}(t) := \mathrm{mod}_{2\pi}(t) \quad \text{with} \quad \mathrm{mod}_T(t) := 2\big(\tfrac{t}{T} - \lfloor \tfrac{t}{T} \rfloor\big) - 1. \tag{A.8}$$

Geometrically, it is a "sawtooth wave". It satisfies the following.

Lemma A.7 (Normalized modulo, real)**.** *Let* $\mathrm{mod}(t)$ *be as in* (A.5)*, we have*

- *(Fourier Series) given by* $M_0 = 0$ *and*

$$M_k = \tfrac{\mathrm{i}}{k\pi} \quad \text{if } k \neq 0, \tag{A.9}$$

  and in particular, $M_1 = \frac{\mathrm{i}}{\pi}$*;*
- *(infinity norm)* $\|\mathrm{mod}\|_\infty = 1$*;*
- *(mean Lipschitz smoothness)* $L^\mu_{\mathrm{mod}} = \frac{3}{\pi}$ *(proof in Prop. 3.22).*

Moreover, its complex extension, $\mathrm{mod}_{\mathbb{C}}(t) := \mathrm{mod}(t) + \mathrm{i} \cdot \mathrm{mod}(t - \frac{\pi}{2})$, satisfies the following.

Lemma A.8 (Normalized modulo, complex)**.** *Let* $\mathrm{mod}_{\mathbb{C}}(t) = \mathrm{mod}(t) + \mathrm{i} \cdot \mathrm{mod}(t - \frac{\pi}{2})$*, we have*

- *(Fourier Series) given by* $M_{\mathbb{C},0} = 0$ *and*

$$M_{\mathbb{C},k} = \tfrac{\mathrm{i}}{k\pi}(1 + \mathrm{i}^{(1-k)}) \quad \text{if } k \neq 0, \tag{A.10}$$

  and in particular, $M_{\mathbb{C},1} = \frac{2\mathrm{i}}{\pi}$*;*
- *(infinity norm)* $\|\mathrm{mod}_{\mathbb{C}}\|_\infty = \sqrt{5/4}$*;*
- *(mean Lipschitz smoothness)* $L^\mu_{\mathrm{mod}_{\mathbb{C}}} = \frac{4+\sqrt{2}}{\pi}$.

Proof. For the Fourier Series, use Lemma A.2. For the infinity norm, it is easy to verify that $\|\mathrm{mod}_{\mathbb{C}}\|_\infty = \sqrt{1^2 + (\frac{1}{2})^2} = \sqrt{5/4}$, a value which is reached between the successive "peaks" of the real and imaginary modulo functions (see Fig. 4.3). The mean Lipschitz smoothness can be computed as in Prop. 3.22, but special care must be given to handle the interplay between the real and imaginary components. More formally, the integral in (3.18), *i.e.*, $J_\delta := \int_0^{2\pi} \sup_{|r| \leq \delta} |\mathrm{mod}(t + r) - \mathrm{mod}(t)| \mathrm{d}t$, can be upper bounded as (for $\delta \leq \frac{\pi}{4}$ the equality is reached)

$$J_\delta \leq (2\pi - 4\delta) \cdot \tfrac{\delta\sqrt{2}}{\pi} + 2 \cdot 2\delta \cdot \tfrac{1}{2}(2 + 2\sqrt{(1 - \tfrac{\delta}{2\pi})^2 + \tfrac{\delta^2}{4\pi^2}}).$$

Therefore, since $(1-\frac{\delta}{2\pi})^2+\frac{\delta^2}{4\pi^2}\leq 1$ (with equality if $\delta\in\{0,2\pi\}$), $\frac{I_\delta}{\delta}\leq g(\delta):=8-6\sqrt{2}+8\sqrt{2}(1-\frac{\delta}{2\pi})$. This function g reaches its maximum in $\delta=0$, where the equality holds, which means that $L^{\mu}_{\mathrm{mod}}=\sup_{0<\delta\leq\pi}\frac{J_\delta}{2\pi\delta}=\frac{4+\sqrt{2}}{\pi}$. □

B

The `pycle` toolbox

THIS appendix is a short guide to `pycle`, the **Py**thon **c**ompressive **le**arning) toolbox [Sch20], which was developed in the context of this thesis. It briefly explains how the toolbox is organized (*e.g.*, to facilitate further contributions). A tutorial then highlights how to use its main features (*e.g.*, to directly use the toolbox). Complementary practical examples are available as jupyter notebooks on the main github page of `pycle`: `https://github.com/schellekensv/pycle`.

The primary goals for the `pycle` toolbox are that it should be:

- **intuitive to use:** practitioners with minimal knowledge in compressive learning and little experience in Python should be able to use it to implement compressive learning in their own projects;
- **flexible to new features:** researchers with interest in compressive learning (that want to try out new methods/techniques in CL) should be able to easily extend this code to suit their own needs, without having to re-write things from scratch (and eventually, suggesting to add some features to the toolbox);
- **efficient to run:** the main motivation of compressive learning is based on the fact that it can be much more memory- and time-efficient than traditional learning methods, so the performances of the tool-

box should fulfill that promise (personal note: this goal is still a challenging for me, this item is rather wishful thinking).

B.1 Preliminaries

B.1.1 Reminder on the compressive learning workflow

In usual machine learning, we fit some parameters $\boldsymbol{\theta}$ (e.g., a parametric curve in regression, centroids in k-means clustering, weights in a neural network...) to a given set of training data $\mathcal{X}$. The actual algorithms for such tasks usually necessitate to access this training set multiple times (one complete pass on the dataset is sometimes called an "epoch"). This can be cumbersome when the dataset is extremely large, and/or distributed across different machines. Compressive learning (CL, also called sketched learning) seeks to circumvent this issue by compressing the whole dataset into one lightweight "sketch" vector, that requires only one pass on the dataset and can be computed in parallel. Learning is then done using only this sketch instead of the inconveniently large dataset (that can be discarded). This allows to significantly reduce the computational resources that are needed to handle massive collections of data.

More precisely, CL comprises two steps:

1. **Sketching:** The dataset $\mathcal{X} = \{\boldsymbol{x}_i \,|\, i = 1, ..., n\}$ (where we assume $\boldsymbol{x}_i \in \mathbb{R}^d$) is compressed as a sketch vector that we note $\boldsymbol{z}_\mathcal{X}$, defined as the average over the dataset of some features $\Phi(\boldsymbol{x}_i)$ (the function $\Phi : \mathbb{R}^d \mapsto \mathbb{C}^m$ or $\mathbb{R}^m$ computes m features, possibly complex):

$$\boldsymbol{z}_\mathcal{X} := \frac{1}{n} \sum_{i=1}^{n} \Phi(\boldsymbol{x}_i). \tag{B.1}$$

   Since this is a simple averaging, sketching can be done on different chunks of $\mathcal{X}$ independently, which is quite handy in distributed or streaming applications.

2. **Learning:** The target model parameters $\boldsymbol{\theta}$ are then obtained by some algorithm Δ that operates *only* on this sketch,

$$\boldsymbol{\theta} = \Delta(\boldsymbol{z}_\mathcal{X}). \tag{B.2}$$

   Typically, this involves an optimization problem $\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}; \boldsymbol{z}_\mathcal{X})$.

In the following, these steps are explained intuitively, the formal details being introduced only when needed; for a more solid/exhaustive overview of compressive learning, see [GBKT17].

B.1.2 Requirements

The `pycle` package is built on standard Python scientific computing libraries: `numpy`, `scipy` and `matplotlib`; if you don't already have them installed, follow the instructions at `https://www.scipy.org/install.html`.

B.1.3 Toolbox organization

The `pycle` toolbox is a collection of several submodules:

1. The `sketching.py` module instantiates feature maps then computes the sketch of datasets with it.
2. The `compressive_learning.py` module contains the actual "learning" methods, extracting the desired parameters from the sketch.
3. The `utils.py` module contains miscellaneous auxiliary functions, for, amongst others, generating synthetic datasets and evaluate the obtained solutions (as well quantitatively with well-defined metrics as qualitatively with visualization tools).
4. Finally, the `third_party.py` module serves to group code chunks used by `pycle` that are written by other developers but not published as independent packages.

B.2 A tutorial tour of `pycle`

Let's explore the core submodules of `pycle`; our focus here is understanding, so this section is a high-level tutorial rather than an exhaustive enumeration of what's inside the toolbox.

B.2.1 Sketching datasets: the `sketching.py` submodule

To use the `sketching` submodule, you first need to import it; I often use `sk` as shorthand. Sketching, as defined in (B.1), is done by simply calling `sk.computeSketch`, as follows (we'll fill in the dots later):

```
import pycle.sketching as sk # import the sketching submodule

X = ...   # load a numpy array of dimension (n,d)
```

```
Phi = ... # sketch feature map, see later

z = sk.computeSketch(X,Phi) # z is a numpy array containing the
    sketch
```

As shown in the code snippet, `sk.computeSketch` requires two arguments: the dataset $\mathcal{X}$ (given as a numpy array of dimensions (n, d)) and the feature map Φ, which must be an instance of a `sk.FeatureMap` object[1]. Actually, all feature maps used up to now in CL are of the following "Simple Feature Map" form:

$$\Phi(\boldsymbol{x}) = f(\Omega^T \boldsymbol{x} + \boldsymbol{\xi}), \tag{B.3}$$

where

$$\Omega = [\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_m] \in \mathbb{R}^{d \times m}, \ \boldsymbol{\xi} = [\xi_1, \cdots, \xi_m]^T \in \mathbb{R}^m, \tag{B.4}$$

and f is a point-wise nonlinearity (i.e., $\Phi_j(\boldsymbol{x}) = f(\boldsymbol{\omega}_j^T \boldsymbol{x} + \xi_j)$ for all j)[2]. You can instantiate such a nonlinearity in `pycle` using the a `SimpleFeatureMap` child class:

```
import pycle.sketching as sk

f = ...      # nonlinearity (Python function, tuple or string)
Omega = ...  # (d,m) numpy array
xi = ...     # (m,) numpy array
Phi = sk.SimpleFeatureMap(f, Omega, xi)
```

We now explain how to set those three arguments.

- **Nonlinearity `f`:** you can simply pass as a string the name of standard nonlinearities used in CL, such as:
  - `"complexExponential"`, for the complex exponential $f(\cdot) = \exp(\mathrm{i}\cdot)$ (corresponds to the random Fourier features sketch);
  - `"cosine"`, simply the cosine $f(\cdot) = \cos(\cdot)$ (the real part of the random Fourier features sketch);

[1] Why do we use `FeatureMap` objects instead of (Python) functions to represent... well, (mathematical) functions? The reason is that we often require additional information about Φ (such as the ambient dimension d, the target dimension m, a method to compute the jacobian $\nabla\Phi$,...). All these parameters and methods are thus conveniently packaged inside the `FeatureMap` objects.

[2] One way to interpret this map is to associate it with a one-layer neural network (without learning the "weights" Ω).

  - `"universalQuantization"`, for the one-bit square wave with normalization in $\{\pm 1\}$, given by $f(\cdot) = \operatorname{sign} \circ \cos(\cdot)$ and used for quantized sketching (see Chapter 4). The complex equivalent is also available as `"universalQuantization_complex"`, which corresponds to $f(\cdot) = \operatorname{sign} \circ \cos(\cdot) + \mathrm{i} \cdot \operatorname{sign} \circ \sin(\cdot)$.

  Alternatively, you can pass in any Python function $\mathbb{R} \mapsto \mathbb{C}$ or $\mathbb{R}$. However, since computing the gradient[3] $\nabla\Phi$ requires $\frac{\mathrm{d}f(t)}{\mathrm{d}t}$, you can also pass in a *tuple* of functions $(f, \frac{\mathrm{d}f(t)}{\mathrm{d}t})$ in order to use gradient-based CL methods later.

- **Projections/frequencies `Omega`:** A (d, m) numpy array, typically randomly generated. Without entering into the details, common choices are instantiated by `sk.drawFrequencies(drawType,d,m,Sigma)`, with `drawType` is a string describing the sampling pattern (`Gaussian`, and `FoldedGaussian` or `AdaptedRadius` which perform better in higher dimensions, see [KBGP18]) and `Sigma` is the associated scale parameter (in the simplest case, it is a scalar parameter corresponding to the bandwidth σ^2 of the associated Gaussian kernel).

- **Dither `xi`:** this optional parameter expects an $(m,)$ numpy array (not providing anything is equivalent to setting $\boldsymbol{\xi} = \mathbf{0}$). To draw *i.i.d.* values uniformly from $[0, 2\pi]$, you can use `sk.drawDithering(m)`.

To summarize, here is a typical example of the creation of a sketch:

```
import pycle.sketching as sk

# Load the dataset
X = ...
(n,d) = X.shape

# Instantiate the feature map
m = 10*d # Sketch size
Omega = sk.drawFrequencies("FoldedGaussian",d,m,Sigma = 0.01) #
    Kernel bandwidth = 0.1
Phi = sk.SimpleFeatureMap("complexExponential", Omega) # No
    dithering used here
```

[3] Noting the Jacobian matrix $\nabla\Phi(\boldsymbol{x}) = [\nabla\Phi_1(\boldsymbol{x}), \cdots, \nabla\Phi_m(\boldsymbol{x})] \in \mathbb{R}^{d\times m}$ and $f'(t) = \frac{\mathrm{d}f(t)}{\mathrm{d}t}$ applied component-wise, we have $\nabla\Phi(\boldsymbol{x}) = \operatorname{diag}\left(f'(\Omega^T\boldsymbol{x} + \boldsymbol{\xi})\right) \cdot \Omega$.

```
# Compute the sketch
z = sk.computeSketch(X,Phi) # z is a numpy array containing the
    sketch
```

B.2.2 Learning from the sketch: the compressive_learning.py submodule

For now, pycle features mainly one algorithm called CLOMPR for mixtures of Gaussians [KBGP18] and k-means clustering [KTTG17]. In pycle, learning methods are wrapped into instances of the abstract Solver class, which gives for example:

```
import pycle.compressive_learning as cl

# Beforehand, define the feature map Phi and compute the sketch z

# Bounds for the data
bounds = np.array([X.min(axis=0),X.max(axis=0)])


# Number of centroids/gaussian modes
K = ...

# Create the (task-specific) solver (children of the Solver
    class)
solver = cl.CLOMP_CKM(Phi,K,bounds,z) # For k-means
# ... OR ...
solver = cl.CLOMP_dGMM(Phi,K,bounds,z) # For GMM

# Learn from the sketch
solver.fit_once()
# ... OR ...
solver.fit_several_times(n_times) # Selects the best out of
    n_times independent trials

# Access the solution
centroids = solver.get_centroids() # For k-means
# ... OR ...
GMM = solver.get_GMM() # For GMM

# Do something with the model (prediction, visualization, ...)
```

The way Solver instances are created depend on the specific task at hand, but for the defines the CLOMP_CKM and CLOMP_dGMM objects (for k-means or (diagonal) Gaussian Mixture Modeling, respectively), you pass in a FeatureMap object, a number of components K, data bounds and the sketch vector.

Then, fit_once or fit_several_times can be called to actually run the learning algorithm. With fit_several_times, several independent trials are run and the best one is selected.

Finally, the achieved solution can be recovered with the solver.current_sol attribute, or by calling specific functions which improve the output format, such as

- solver.get_centroids(), outputs the centroids only (in a (K, d) numpy array).
- solver.get_GMM(), returns a tuple of three elements (w,mus,covs), where w are the mixture coefficients, and mus and covs are the centers (resp. covariance matrices) of the Gaussian modes, in a (K, d) numpy array (resp. (K, d, d) numpy array).

Detailed examples of those tasks are available on the main github page of the pycle toolbox. With that and the documentation[4] of the different functions, you should have a good idea of how to use the toolbox in practice. In the next section, some advanced functionalities of the toolbox are described.

B.3 Advanced features of pycle

B.3.1 Helpful tools from the utils.py submodule

Dataset generation tools

Several methods allow to generate synthetic datasets, the most notable being generatedataset_GMM for datasets sampled from Gaussian mixture models (with a large set of tunable parameters). Moreover, other toy examples datasets can be generated with generateCirclesDataset which generates concentric circles, generateSpiralDataset which generates a spiral dataset, and generatedataset_Ksparse for K-sparse vectors.

[4] Just type help(nameOfAFunction) to see all the available options.

Performance metrics

The toolbox provides several metrics to assess the quality of the learned models. For k-means, `SSE` computes the Sum of Squared Errors of a set of centroids. For Gaussian Mixture Models, `loglikelihood_GMM` assesses the log-likelihood of the provided Gaussian mixture on a dataset. Moreover, if a ground-truth GMM is known, `symmKLdivergence_GMM` estimates the (symmetrized) Kullback-Leibler divergence between two GMMs [KBGP18].

Visualization tools

To visualize the quality of fit of a GMM to the dataset, `plotGMM` plots the contour curves of the given GMMs density, along with the optional dataset.

B.3.2 Designing the sampling pattern parameters when drawing the feature map

In `sketching.py`, a strategy to estimate `Sigma` (used to draw the frequencies) from a small preliminary sketch, described in [KBGP18], is implemented under the name `sk.estimateSigma`.

B.3.3 Sketching with Differential Privacy

As described in Chapter 5, a layer of differential privacy can easily be incorporated on top of the sketch. In `pycle`, this is achieved by replacing `computeSketch` with its variant `computeSketch_DP`.

C

Patching "Representation and Coding of Signal Geometry"

OUR approach from Chapter 3 can be related to the context of geometry-preserving embedding (or coding) developed in "Representation and Coding of Signal Geometry" [BRM17]. This allows us to provide another version of one of their central results, [BRM17, Thm 3.2], whose proof is incorrect (as described below). While the alternative result we propose looks slightly different, it fulfills the same goal: a non-asymptotic guarantee for the geometry-preserving capabilities of the embedding (3.6) with discontinuous f, which holds on infinite signal sets. This section can be seen as a first (theoretical) application of Prop. 3.15.

C.1 Geometry-preserving embedding: the initial approach

In [BRM17] the authors study when a mapping $\boldsymbol{\varphi} : \Sigma \to \mathbb{C}^m$ (such as $\boldsymbol{z}_f$ defined in (3.6) for $f \in \mathrm{PF}$) defines an embedding of Σ into $\mathbb{C}^m$ approximately preserving the proximity of vectors in Σ. This proximity is measured by the (local) preservation a distance associated with a $\ell_\sharp$-norm $\|\cdot\|_\sharp$ (*e.g.*,

the ℓ_1 or the ℓ_2-norm). Adapting their setting to our conventions[1], given some $\epsilon, \delta > 0$, and an invertible function, or *distance map*, $\gamma : \mathbb{R}_+ \to \mathbb{R}_+$, they study the conditions ensuring that $\boldsymbol{\varphi}$ is a $(\gamma, \delta, \epsilon)$-embedding of Σ (endowed with the $\ell_\sharp$-norm) into $\mathbb{C}^m$; or mathematically, such that $\boldsymbol{\varphi}$ respects

$$(1-\delta)\,\gamma(\|\boldsymbol{x}-\boldsymbol{y}\|_\sharp) - \epsilon \leq \|\boldsymbol{\varphi}(\boldsymbol{x}) - \boldsymbol{\varphi}(\boldsymbol{y})\|^2 \leq (1+\delta)\,\gamma(\|\boldsymbol{x}-\boldsymbol{y}\|_\sharp) + \epsilon, \quad \text{(C.1)}$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \Sigma$.

In (C.1), γ maps distances in Σ to (squared) distances in $\mathbb{C}^m$, and δ and ϵ quantify the multiplicative and the additive error, respectively, of the embedding associated with the map γ. For instance, if $\boldsymbol{\varphi}$ is linear with $\boldsymbol{\varphi}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$, there exists many random constructions of the $m \times d$ matrix $\boldsymbol{A}$ with appropriate scaling (*e.g.*, random Gaussian matrix or random partial Fourier matrix [FR17]) for which (C.1) holds with high probability with $\ell_\sharp \equiv \ell_2$, $\epsilon = 0$, and $\gamma(t) = t^2$ for $\Sigma = \Sigma_k$ and $m = O(\delta^{-2} k \log(n/k))$. Similarly, in the context of one-bit compressive sensing where $\boldsymbol{\varphi}(\boldsymbol{x}) = (cm)^{-1/2} \operatorname{sign}(\boldsymbol{A}\boldsymbol{x})$ (for some suitable $c > 0$), $\|\boldsymbol{\varphi}(\boldsymbol{x}) - \boldsymbol{\varphi}(\boldsymbol{y})\|^2$ represents the (scaled) Hamming distance between the two binary vectors $\boldsymbol{\varphi}(\boldsymbol{x})$ and $\boldsymbol{\varphi}(\boldsymbol{y})$, and (C.1) is verified with high probability over $\Sigma_k \cap \mathbb{B}^d$ with $m = O(\epsilon^{-2} k \log(n/k))$, $\ell_\sharp \equiv \ell_2$, $\delta = 0$, and $\gamma(t) = t$ [JLBB13]. The work [BRM17] extends this analysis to general nonlinear feature maps $\boldsymbol{\varphi}(\cdot) = \boldsymbol{z}_f(\cdot)$ for some periodic function f (such as the universal quantizer q). In such a context, the authors show that (C.1) holds with a map γ that often displays two regimes: a linear regime for small distances in Σ ($\boldsymbol{x} \approx \boldsymbol{y}$), and a saturation regime where γ quickly flattens after a certain distance.

As explained in [BRM17, Sec. 4.5], this approach is connected to the approximation of a kernel $\kappa : \Sigma \times \Sigma \to \mathbb{R}_+$ from the inner product of the images of two vectors, namely, for which $\langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_f(\boldsymbol{y}) \rangle \approx \kappa(\boldsymbol{x}, \boldsymbol{y})$. Assuming $\|\boldsymbol{z}_f\| = 1$ for simplicity, which is the case for complex exponential and universal quantization features, we find $\|\boldsymbol{z}_f(\boldsymbol{x}) - \boldsymbol{z}_f(\boldsymbol{y})\|^2 = 2\,(1 - \langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_f(\boldsymbol{y}) \rangle)$. Therefore, if $\boldsymbol{z}_f$ is a $(\gamma, 0, 2\epsilon)$-embedding of Σ into $\mathbb{C}^m$, then

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) - \epsilon \leq \langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_f(\boldsymbol{y}) \rangle \leq \kappa(\boldsymbol{x}, \boldsymbol{y}) + \epsilon, \quad \text{(C.2)}$$

[1] Hereafter, departing from the general approach of [BRM17], we always consider the simplified case where $\mathbb{C}^m$ is equipped with the Euclidean distance, with a squaring of the corresponding distance in (C.1).

for all $\boldsymbol{x}, \boldsymbol{y} \in \Sigma$, provided we define the kernel

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) := 1 - \tfrac{1}{2}\gamma(\|\boldsymbol{x} - \boldsymbol{y}\|_\sharp). \tag{C.3}$$

The smoothness of κ is thus directly connected to the one of γ; for instance, if γ is Lipschitz continuous with constant $L_\gamma \geq 0$ of $\mathbb{R}_+$, then, from the invertibility of γ over $\mathbb{R}^+$, κ is Lipschitz continuous with constant $L_\gamma/2$ with respect to any of its argument. Note that, from Lemma 3.11 and Lemma 3.14, we also know that if f is mean smooth with constant L_f^μ, then $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \kappa_{f,f}^\Delta(\boldsymbol{x} - \boldsymbol{y})$ is Lipschitz continuous with constant $L_\kappa \leq C_\Lambda L_f^\mu$ with respect to any of its argument (as proved from the bound on δ_4 in the proof of Prop. 3.15). This shows that, despite their different origin, the smoothness of γ (in the approach [BRM17]) and the one of f (in ours) control the one of κ.

Compared to our approach, [BRM17] imposes the periodic function f to be "Lipschitz continuous by part" (rather than being mean smooth), as defined hereafter in a setting adapted to our needs.

Definition C.1 (T-part Lipschitz continuity [BRM17, Def. 2.1])**.** A function $f : \Sigma \to \mathbb{C}$ is T-part Lipschitz continuous over $\mathcal{S} \subset \Sigma$ with constant $\bar{L}_f \geq 0$, if there exists a finite partition $\{\mathcal{S}_t\}_{t=1}^T$ of $\mathcal{S}$ into T disjoint sets (*i.e.*, $\bigcup_{t=1}^T \mathcal{S}_t = \mathcal{S}$) such that

$$\forall t \in [T],\ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}_t, \quad |f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq \bar{L}_f \cdot \|\boldsymbol{x} - \boldsymbol{y}\|_\sharp. \tag{C.4}$$

Moreover, f is *exactly* T-part Lipschitz continuous over $\mathcal{S}$ with constant $\bar{L}_f \geq 0$, which we write $f \in \overline{\mathrm{Lip}}(\mathcal{S}, T, \bar{L}_f)$, if it is T-part Lipschitz continuous with that constant but is not $(T-1)$-part Lipschitz continuous with the same constant.

Note that (C.4) both generalizes (3.17) to functions from $\Sigma \subset \mathbb{R}^d$ to $\mathbb{C}$, and localizes (3.17) on $\mathcal{S}$.

Following the convention of our paper, the authors of [BRM17] then prove the following result. We simplify it to the case $\delta = 0$ and where each component of $\boldsymbol{z}_f$ has at most T parts of continuity (despite its randomness).

Theorem C.2 (Adapted from [BRM17, Thm 3.2])**.** *Given $0 < \epsilon < 1$, an L_γ-Lipschitz continuous distance map $\gamma : \mathbb{R}_+ \to \mathbb{R}_+$, and a signal set Σ with finite covering number $\mathcal{C}_\eta(\Sigma)$ for any radius $\eta > 0$, let us assume that, for any* fixed *pair of vectors $\boldsymbol{x}, \boldsymbol{y} \in \Sigma$, the mapping $\boldsymbol{z}_f$ defined in* (3.6) *satisfies the embedding relation* (C.1) *(for $\delta = 0$) with probability exceeding $1 - Ce^{-cm\epsilon^2}$.*

Let us suppose that there exists a constant $\bar{L}_f \geq 0$ such that, for any $\boldsymbol{x} \in \Sigma$, integer $t \geq 1$, radius $\eta > 0$, and given $\mathcal{S}_{\boldsymbol{x}}(\eta) := \{\boldsymbol{u} \in \Sigma : \|\boldsymbol{u} - \boldsymbol{x}\|_\sharp \leq \eta\}$ (a neighborhood of $\boldsymbol{x}$ of radius η),

$$\mathbb{P}\left[(\boldsymbol{z}_f(\cdot))_k \in \overline{\mathrm{Lip}}(\mathcal{S}_{\boldsymbol{x}}(\eta), T, \bar{L}_f)\right] \;\leq\; p_t(\eta), \tag{C.5}$$

with p_t independent of $\boldsymbol{x}$, and $p_t(\eta) = 0$ if $t > T$ for some integer $T \geq 0$.

In this context, defining $\rho(\eta, T) := \sum_{t=2}^{T} p_t(\eta) \log t$ and $\nu := \frac{1}{4}(L_\gamma + \bar{L}_f)^{-1}$, provided that $\epsilon^2 \geq C\rho(2\nu\epsilon^2, T)$ and

$$m \geq C\epsilon^{-2}\big(\mathcal{H}_{\nu\epsilon^2}(\Sigma) + \log T\big), \tag{C.6}$$

the mapping $\boldsymbol{z}_f$ is a $(\gamma, 0, 2\epsilon)$-embedding with probability exceeding $1 - Ce^{-c\epsilon^2 m}$, i.e., *$\boldsymbol{z}_f$ respects*

$$\gamma(\|\boldsymbol{x} - \boldsymbol{y}\|_\sharp) - 2\epsilon \leq \|\boldsymbol{z}_f(\boldsymbol{x}) - \boldsymbol{z}_f(\boldsymbol{y})\|^2 \leq \gamma(\|\boldsymbol{x} - \boldsymbol{y}\|_\sharp) + 2\epsilon, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \Sigma.$$

The statement of this theorem is an easy adaptation of [BRM17, Thm 3.2] where we set $\delta = 0$, $w(\epsilon, \delta) = c\epsilon^2$, $c_0 = c\epsilon$, $T_{\max} = T$ (so that $P_F = 0$), and $\alpha = \epsilon^2 \leq \epsilon \leq 1$.

Note that the first assumption of this theorem (regarding the fact that (C.1) holds with high probability for any fixed pair of signals) is proven in a separate result, namely [BRM17, Thm 4.1]. This theorem is similar to our Prop. 3.12 (up to an easy extension of this proposition to a finite set of pairs by union bound). Since the flaw developed below is independent of that separate result, we abstract this specific assumption away in this work. As explained in [BRM17, App. E], the conditions of this theorem can thus be met for instance in the case where f is the universal quantizer. One can then show that $T = 2$, and defining $p_2(\eta) := D\eta$, with $D > 0$ function of d and Λ, is appropriate for the bound (C.5). Therefore, $\rho(2\nu\epsilon^2, T) \leq 2D\nu\epsilon^2 \leq \epsilon^2/C$ for an appropriate $C > 0$.

The statement of this theorem bears similarities with our Prop. 3.15 in the case where $f = g$; in essence, keeping in mind the equivalence (C.2), up to a smaller covering radius scaling as $\epsilon^2 < \epsilon < 1$ in (C.6), the constraint (C.6) is similar to (3.19) if we consider that the T-part Lipschitz continuity of f replaces its mean smoothness.

However, the proof of Theorem C.2 in [BRM17, App. B] is incorrect. Let us see why by sketching their arguments in our system of notations

and using $\ell_\sharp = \ell_2$ for the sake of simplicity. Given $\eta > 0$, $\boldsymbol{x} \in \Sigma$, and $t \in [T]$, the authors first (implicitly) note that if the random variable $Z(\boldsymbol{x})$ counts the number of components of $\boldsymbol{z}_f(\cdot)$ that are exactly t-part Lipschitz over $\mathcal{S}_{\boldsymbol{x}}(\frac{\eta}{2})$ with a given constant $\bar{L}_f$, then $\mathbb{E}Z \leq mp_t$. Therefore, given $c_0 > 0$ and invoking Hoeffding's inequality, they can upper bound the probability that $Z(\boldsymbol{x}) \geq mp_t(1+c_0) \geq \mathbb{E}Z(\boldsymbol{x}) + mp_t c_0$ with

$$\mathbb{P}[Z(\boldsymbol{x}) \geq mp_t(1+c_0)] \leq \exp(-2c_0^2 m).$$

From this bound (using a union bound over all $t \in [T]$), they then determine that $\mathcal{S}_{\boldsymbol{x}}(\frac{\eta}{2})$ is partitioned in at most $S := \exp((1+c_0)\,\rho(\eta, T)m)$ cells with probability greater than $1 - Te^{-2c_0^2 m}$. Each cell of this partition of $\mathcal{S}_{\boldsymbol{x}}(\frac{\eta}{2})$ has thus a diameter of at most η. Moreover, by definition, $\boldsymbol{z}_f$ is guaranteed to be Lipschitz continuous with constant $\bar{L}_f$ over every such cell.

The author then consider the possibility to pick one point per such cell, called cell center, and to gather them in a finite set of at most S elements. One can then repeat this construction for all vectors $\boldsymbol{x}$ of a $\frac{\eta}{2}$-covering $\Sigma_{\eta/2}$ of Σ, and collect, for each such vector, all cell centers of its related neighborhood into a global set $\mathcal{G}$ of centers of at most $S \times \mathcal{C}_{\frac{\eta}{2}}(\Sigma)$ elements. By definition, $\mathcal{G}$ is thus a $\eta/2$-covering of Σ with the additional property that $\boldsymbol{z}_f$ is $\bar{L}_f$-Lipschitz continuous over each cell.

The authors then leverage this local continuity as follows. Since, by hypothesis, the mapping $\boldsymbol{z}_f$ defined in (3.6) satisfies the embedding relation (C.1) (for $\delta = 0$) with probability exceeding $1 - Ce^{-cm\epsilon^2}$ over any *fixed* pair of vectors $\boldsymbol{x}, \boldsymbol{y} \in \Sigma$, they first expand this property over all pairs of vectors taken in $\mathcal{G} \times \mathcal{G}$. This is ensured with probability exceeding $1 - CS^2\mathcal{C}^2_{\eta/2}(\Sigma)e^{-cm\epsilon^2}$, by union bound, since $|\mathcal{G} \times \mathcal{G}| \leq S^2\mathcal{C}^2_{\eta/2}(\Sigma)$. Next, they extend this property to all $\boldsymbol{x}, \boldsymbol{y} \in \Sigma$ by continuity, exploiting the (local) Lipschitz continuity of f over each cell.

The flaw, which happens in the first step above, is analogous to how we cannot show the wrong statement $\mathbb{P}[\|\boldsymbol{g}\|^2 < 0] = 1/2$ for a Gaussian vector $\boldsymbol{g} \sim \mathcal{N}^d(0,1)$ by assigning another vector $\boldsymbol{u}$ to $\boldsymbol{g}$ in the correct equality $\mathbb{P}[\langle \boldsymbol{u}, \boldsymbol{g}\rangle < 0] = 1/2$, valid for $\boldsymbol{u}$ fixed. Indeed, the vectors of $\mathcal{G}$, the collection of all cell centers, are built from the random mapping $\boldsymbol{z}_f$ — each center must be taken in a cell whose frontiers are controlled by the discontinuities of the components of $\boldsymbol{z}_f$. These vectors are thus dependent of both $\boldsymbol{\Omega} \sim \Lambda^m$ and the dither $\boldsymbol{\xi} \sim \mathcal{U}^m([0, 2\pi))$, through their dependence in

z_f. Therefore, one cannot ensure that the probability that (C.1) holds (with $\delta = 0$) on two cell centers $x = x(\Omega, \xi), y = y(\Omega, \xi)$ exceeds $1 - Ce^{-cm\epsilon^2}$, since that probability is itself taken over Ω, ξ. This flaw breaks the proof of [BRM17, Thm 3.2].

C.2 An alternative geometry-preserving embedding

One can use Prop. 3.15 to get a variant of Thm C.2 relying on the equivalence between (C.1) and (C.2). This variant achieves the same high-level goal (*i.e.*, a non-asymptotic guarantee on the approximation error achieved by the embedding z_f that holds infinite signal sets even for discontinuous f), but the assumptions it relies on differ in two aspects. First, the smoothness of the distance map γ is not anymore characterized by its Lipschitz smoothness directly, but by the constant C_Λ, defined by the sampling scheme Λ driving the random projections Ω. Second, we use the mean Lipschitz property instead of the T-part Lipschitz property as notion of "generalized smoothness" for the map f. It is not clear if this change is fundamentally necessary to be able to prove a variant of Thm C.2, but we leave an investigation of this issue for future work (the universal quantization q satisfies both properties anyway).

In fact, the following corollary shows that one can define novel *asymmetric embeddings* from Σ into $\mathbb{C}^m$; we can map two vectors of Σ with different random feature mappings z_f and z_g achieved with distinct periodic functions f and g, respectively, and still show that, under certain conditions on f, g, and the frequency distribution Λ, $\|z_f(x) - z_g(y)\|^2$ approximates a distortion of the distance between any $x, y \in \Sigma$ provided m is large compared to the complexity of Σ. Then, setting $f = g$ provides a specific embedding of Σ into $\mathbb{C}^m$, in the sense described by [BRM17].

Corollary C.3 (Asymmetric geometry-preserving embedding)**.** *Let Σ be a compact set with finite covering number, $f, g \in \mathrm{PF}$ be two real 2π-periodic functions and finite mean smoothness constants $L^\mu_f > 0$ and $L^\mu_g > 0$, respectively. We assume that the frequency distribution Λ is such that $C_\Lambda < \infty$, and there exists a real, one-dimensional p.d. kernel $\kappa^\Delta_0 : \mathbb{R}_+ \to [0, 1]$ such that $[\mathcal{F}^{-1}\Lambda](u) = \kappa^\Delta_0(\|u\|_\sharp)$ for some norm $\|\cdot\|_\sharp$.*
For all error level $\epsilon > 0$, provided the feature dimension is larger than

$$m \geq 128 \cdot \tfrac{1}{\epsilon^2} \cdot \mathcal{H}_{\epsilon/c}(\Sigma), \tag{C.7}$$

with constant $c = 4C_\Lambda(L_f^\mu + L_g^\mu + 2\min(L_f^\mu, L_g^\mu))$*, we have, with probability exceeding* $1 - 9\exp(-\frac{m\epsilon^2}{64})$*,*

$$\gamma_{f,g}(\|\boldsymbol{x}-\boldsymbol{y}\|_\sharp) - 4\epsilon \le \|\boldsymbol{z}_f(\boldsymbol{x}) - \boldsymbol{z}_g(\boldsymbol{y})\|^2 \le \gamma_{f,g}(\|\boldsymbol{x}-\boldsymbol{y}\|_\sharp) + 4\epsilon,\ \forall \boldsymbol{x},\boldsymbol{y} \in \Sigma, \tag{C.8}$$

according to the distance map $\gamma_{f,g}$ *defined by*

$$\gamma_{f,g} : s \in \mathbb{R}_+ \to \gamma_{f,g}(s) = \|f\|^2 + \|g\|^2 - 2\textstyle\sum_{k\in\mathbb{Z}} F_k G_k^* \kappa_0^\Delta(|k|s) \in \mathbb{R}_+.$$

Therefore, if $f = g$ *and if* $\gamma_{f,f}$ *is invertible,* $\boldsymbol{z}_f$ *is a* $(\gamma_{f,f}, 0, 4\epsilon)$*-embedding of* Σ *(equipped with the norm* $\|\cdot\|_\sharp$*) into* $\mathbb{C}^m$*, with* $\gamma_{f,f}(0) = 0$ *and* $\gamma_{f,f}(s) \in [0,2]$*.*

Proof. Under the hypothesis of this corollary and remembering that $\widehat{\kappa}_{f,g}(\boldsymbol{x},\boldsymbol{y}) = \langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_g(\boldsymbol{y})\rangle$, Prop. 3.15 tells us that the event

$$|\widehat{\kappa}_{f,g}(\boldsymbol{x},\boldsymbol{y}) - \kappa_{f,g}(\boldsymbol{x},\boldsymbol{y})| \le \epsilon,\ \forall\, \boldsymbol{x},\boldsymbol{y} \in \Sigma,$$

holds with probability exceeding $1 - 3\exp(-\frac{m\epsilon^2}{64})$. Similarly, with the same probability, $|\widehat{\kappa}_{f,f}(\boldsymbol{x},\boldsymbol{y}) - \kappa_{f,f}(\boldsymbol{x},\boldsymbol{y})| \le \epsilon$ and $|\widehat{\kappa}_{g,g}(\boldsymbol{x},\boldsymbol{y}) - \kappa_{g,g}(\boldsymbol{x},\boldsymbol{y})| \le \epsilon$, for all $\boldsymbol{x},\boldsymbol{y} \in \Sigma$. Therefore, by union bound, these three events jointly hold with probability larger than $1 - 9\exp(-\frac{m\epsilon^2}{64})$.

Conditionally to this combined occurrence, since $\kappa_{f,f}(\boldsymbol{x},\boldsymbol{x}) = \kappa_{f,f}^\Delta(\boldsymbol{0}) = \|f\|^2$ and $\kappa_{g,g}(\boldsymbol{y},\boldsymbol{y}) = \kappa_{g,g}^\Delta(\boldsymbol{0}) = \|g\|^2$, we find

$$\left|\langle \boldsymbol{z}_f(\boldsymbol{x}), \boldsymbol{z}_f(\boldsymbol{x})\rangle - \|f\|^2\right| \le \epsilon, \quad \left|\langle \boldsymbol{z}_g(\boldsymbol{y}), \boldsymbol{z}_g(\boldsymbol{y})\rangle - \|g\|^2\right| \le \epsilon,$$

and

$$\begin{aligned}\|\boldsymbol{z}_f(\boldsymbol{x}) - \boldsymbol{z}_g(\boldsymbol{y})\|^2 &\le \|f\|^2 + \|g\|^2 - 2\kappa_{f,g}(\boldsymbol{x},\boldsymbol{y}) + 4\epsilon \\ &= \|f\|^2 + \|g\|^2 - 2\kappa_{f,g}^\Delta(\boldsymbol{x}-\boldsymbol{y}) + 4\epsilon.\end{aligned}$$

Moreover, from Prop. 3.8, since $\kappa_{f,g}^\Delta(\boldsymbol{u}) = \sum_{k\in\mathbb{Z}} F_k G_k^* \kappa^\Delta(k\boldsymbol{u})$ with $\kappa^\Delta(\boldsymbol{u}) = (\mathcal{F}^{-1}\Lambda)(\boldsymbol{u}) = \kappa_0^\Delta(\|\boldsymbol{u}\|_\sharp)$ and $\boldsymbol{u} \in \mathbb{R}^d$, the definition of $\gamma_{f,g}$ provides

$$\gamma_{f,g}(\|\boldsymbol{u}\|_\sharp) = \|f\|^2 + \|g\|^2 - 2\kappa_{f,g}^\Delta(\boldsymbol{u}),$$

which proves the upper bound of (C.8), the lower bound being established similarly.

Since $f, g, \kappa_0^\Delta \in \mathbb{R}$, $\sum_k F_k G_k^* \beta_k \in \mathbb{R}$ for any real coefficients β_k, we show easily that $\gamma_{f,g} \in \mathbb{R}$ with $\gamma_{f,g}(0) = \|f\|^2 + \|g\|^2 - 2\langle f, g\rangle \ge \|f\|^2 + \|g\|^2 -$

$2\|f\|\,\|g\| \geq 0$. Moreover, if $f = g$, we get $\gamma_{f,f}(0) = 0$ since $\kappa_0^\Lambda(0) = 0$, and $\gamma_{f,f}(s) \in [0,2]$ since $0 \leq \kappa_0^\Lambda(s) \leq 1$ for all $s \geq 0$ and $\sum_{k\in\mathbb{Z}} |F_k|^2 \kappa_0^\Lambda(|k|s) \leq \|f\|^2$. □

In this corollary, the existence of a norm $\|\cdot\|_\sharp$ controlling the behavior of $\mathcal{F}^{-1}\Lambda$ is ensured, for instance, if Λ is a centered Gaussian distribution, in which case the $\ell_\sharp$-norm is the ℓ_2-norm. If Λ is the Cartesian product of d Cauchy distributions in $\mathbb{R}^d$ (with zero location parameter and scale parameter $\tau > 0$), *i.e.*,

$$\Lambda(\boldsymbol{\omega}) = \tfrac{1}{\pi^d \tau^d} \prod_{k=1}^d \tfrac{\tau^2}{\omega_k^2 + \tau^2}, \tag{C.9}$$

then $\mathcal{F}^{-1}\Lambda$ amounts to the Laplace distribution and $\|\cdot\|_\sharp = \|\cdot\|_1$ [BRM17, Sec. 4.2.2.]. Moreover, if Λ is set to any α-stable distribution with $\alpha \geq 1$, *i.e.*, a distribution with characteristic function $(\mathcal{F}^{-1}\Lambda)(\boldsymbol{x}) \propto \exp(-c\|\boldsymbol{x}\|_\alpha^\alpha)$ with the Gaussian and the Cauchy distributions as special cases, we can reach an (asymmetric) embedding associated with the norm $\|\cdot\|_\alpha$ [OA11].

Regarding the distance map $\gamma_{f,g}$, we observe that it does not necessarily vanish at the origin, when $\boldsymbol{x} = \boldsymbol{y}$ in (C.8). As soon as $f \neq g$, a bias exists since

$$\gamma_{f,g}(0) = \|f\|^2 + \|g\|^2 - 2\sum_{k\in\mathbb{Z}} F_k G_k^* = \|f\|^2 + \|g\|^2 - 2\langle f, g\rangle = \|f - g\|^2, \tag{C.10}$$

using $\kappa_0^\Lambda(0) = \int_{\mathbb{R}^d} \Lambda(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} = 1$. For instance, if $f = q$ (with q the universal quantizer defined in (3.8)), and $g(\cdot) = \cos(\cdot)$, $\gamma_{q,\cos}(s) = \frac{3}{2} - 2\Re(F_1)\,\kappa_0^\Lambda(s) = \frac{3}{2} - \frac{4}{\pi}\kappa_0^\Lambda(s)$ since $\|f\|^2 = 1$, $\|g\|^2 = 1/2$, $2G_k = \delta_{k,1} + \delta_{k,-1}$, and $F_1 = F_{-1} = \frac{2}{\pi}$ from (3.8). Therefore, if Λ is a Gaussian distribution with unit standard deviation,

$$\|\boldsymbol{z}_f(\boldsymbol{x}) - \boldsymbol{z}_g(\boldsymbol{y})\|^2 \approx \gamma_{q,\cos}(\|\boldsymbol{x} - \boldsymbol{y}\|) = \tfrac{3}{2} - \tfrac{4}{\pi}\exp(-\tfrac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2).$$

Compared to the case $f(\cdot) = g(\cdot) = \cos(\cdot)$ where

$$\gamma_{\cos,\cos}(\|\boldsymbol{x} - \boldsymbol{y}\|) = 1 - \exp(-\tfrac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2),$$

and $\gamma_{\cos,\cos}(0) = 0$, we thus observe a systematic bias $\gamma_{q,\cos}(0) = \frac{3}{2} - \frac{4}{\pi} \approx 0.2268$ at the origin.

This non-vanishing bias[2] in the case $f \neq g$ is not a drawback per se,

[2]This bias is here demonstrated when the feature space $\mathbb{C}^m$ is equipped with the squared

since $\gamma_{f,g}$ can still be invertible. For $\gamma_{q,\cos}$ and a Gaussian Λ with unit variance, we find

$$\gamma_{q,\cos}^{-1}(s') = \big(-2\ln(\tfrac{3\pi}{8} - \tfrac{\pi}{4}s')\big)^{1/2}, \text{ with } s' \in [\tfrac{3}{2} - \tfrac{4}{\pi}, \tfrac{3}{2}].$$

This shows that, if ϵ is small enough, we can still reliably infer the distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ from $\|\boldsymbol{z}_f(\boldsymbol{x}) - \boldsymbol{z}_g(\boldsymbol{y})\|$ provided that $\boldsymbol{x} \approx \boldsymbol{y}$. Indeed, estimating $\gamma_{q,\cos}^{-1}(\|\boldsymbol{z}_f(\boldsymbol{x}) - \boldsymbol{z}_g(\boldsymbol{y})\|^2) \approx \|\boldsymbol{x} - \boldsymbol{y}\|$ leads to a first order error [BRM17] proportional to

$$\big(\tfrac{\mathrm{d}}{\mathrm{d}s}\gamma_{q,\cos}(s)\big)^{-1}\epsilon = \tfrac{4}{\pi s}\,\epsilon\exp(\tfrac{s^2}{2}),$$

for $s = \|\boldsymbol{x} - \boldsymbol{y}\|$. As expected from the local nature of the embedding, this error quickly explodes when s is large.

Remark C.4. When $f = g$, $\gamma(s) = 2\|f\|^2 - 2\sum_k |F_k|^2\kappa_0^\Lambda(|k|s)$ is invertible iff $\sum_k |F_k|^2\kappa_0^\Lambda(|k|s)$ is invertible. This occurs, for instance, if the one-dimensional kernel κ_0^Λ is differentiable with $\frac{\mathrm{d}}{\mathrm{d}s}\kappa_0^\Lambda(s) < 0$ for all $s > 0$, which is the case of any symmetric α-stable distribution Λ for which $(\mathcal{F}^{-1}\Lambda)(\boldsymbol{x}) \propto \exp(-c\|\boldsymbol{x}\|_\alpha^\alpha)$. In this case, we easily verify that $\frac{\mathrm{d}}{\mathrm{d}s}\gamma(s) > 0$ for $s > 0$, and γ is monotonically increasing when s increases, starting from $\gamma(0) = 0$. This ensures the injectivity of γ.

ℓ_2-distance; the question of its existence for other metrics, such as the ℓ_1-distance, remains open.

List of symbols and acronyms

δ_c — Dirac measure (delta) located at a point $\boldsymbol{c}$ (p. 57)

$\mathcal{P}_0$ — True ("ideal") distribution of the data. (p. 15)

$\mathcal{X}$ — Dataset, (multi)set of n samples in Σ. (p. 15)

$\boldsymbol{I}_n$ — Identity matrix of size $n \times n$. (p. 21)

ℓ — Loss function $\Sigma \times \Theta \to \mathbb{R}$ (p. 16)

Δ_N — Probability simplex in dimension N (p. 56)

$\mathcal{R}$ — Risk function (average loss given a distribution) (p. 16)

$\mathcal{M}$ — Set of probability measure on the set Σ (p. 57)

Σ — Signal space. (p. 15)

$\mathcal{A}_\Phi$ — Sketching operator on measures, defined by Φ (p. 62)

$z_{\Phi,\mathcal{X}}$ — Sketch vector of the dataset $\mathcal{X}$ defined by the map Φ (p. 62)

$\boldsymbol{\theta}$ — Parameter vector. (p. 16)

$\widetilde{\boldsymbol{\theta}}$ — Minimizer of the empirical risk (p. 17)

$\boldsymbol{\theta}^*$ — Minimizer of the true risk $\mathcal{R}(\boldsymbol{\theta}; \mathcal{P}_0)$ (p. 16)

i.i.d. — Independent and identically distributed. (p. 15)

ADC — Analog-to-Digital Converter (p. 46)

ADP — Approximate Differential Privacy (p. 165)

AQP — Approximate Query Processing (p. 50)

CL — Compressive Learning (p. 13)

CS — Compressive Sensing (p. 43)

DP — Differential Privacy (p. 161)

ERM — Empirical Risk Minimization (p. 16)

GeMM — Generalized Method of Moments (p. 61)

GMM — Gaussian Mixture Model (p. 37)

GN — Generative Network (p. 194)

IPM — Integral Probability Metric (p. 58)

KRR — Kernel Ridge Regression (p. 29)

LRIP — Lower Restricted Isometry Property (p. 45)

MAP — Maximum A Posteriori estimator. (p. 22)

ML — Machine Learning (p. 14)

MMD — Maximum Mean Discrepancy (p. 59)

PDF — Probability Density Function (p. 56)

QCS — Quantized Compressive Sensing (p. 47)

RFF — Random Fourier Features (p. 31)

RIP — Restricted Isometry Property (p. 45)

RKHS — Reproducible Kernel Hilbert Space (p. 28)

RPF — Random Periodic Features (p. 83)

SL — Statistical Learning framework (p. 14)

SP — Signal Processing (p. 39)