HAL open science

Efficient Sequential Learning in Structured and Constrained Environments

Daniele Calandriello

To cite this version:

Daniele Calandriello. Efficient Sequential Learning in Structured and Constrained Environments. Machine Learning [cs.LG]. Inria Lille Nord Europe - Laboratoire CRIStAL - Université de Lille, 2017. English. NNT : . tel-01816904

HAL Id: tel-01816904

https://theses.hal.science/tel-01816904v1

Submitted on 15 Jun 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Efficient Sequential Learning in Structured and Constrained Environments

Daniele Calandriello

THÈSE DE DOCTORAT
UNIVERSITÉ LILLE 1

Thése de doctorat

Spécialité : **Informatiques**

présentée par

Daniele Calandriello

Efficient Sequential Learning in Structured and Constrained Environments

sous la direction de M. Michal **VALKO**
et le co-encadrement de M. Alessandro **LAZARIC**

Rapporteurs :	M.	Nicolò	**CESA-BIANCHI**	Università di Milano
	M.	Michael	**MAHONEY**	UC Berkeley

Soutenue publiquement le **18 Decembre 2017** devant le jury composé de

M.	Nicolò	**CESA-BIANCHI**	Università di Milano	Rapporteur
M.	Michael	**MAHONEY**	UC Berkeley	Rapporteur
M.	Francis	**BACH**	INRIA-ENS	Examinateur
Mme.	Mylène	**MAÏDA**	Université Lille 1	Examinateur
M.	Michal	**VALKO**	INRIA	Directeur
M.	Alessandro	**LAZARIC**	INRIA	Co-Directeur

To whom it shall matter

Abstract

The main advantage of non-parametric models is that the accuracy of the model (degrees of freedom) adapts to the number of samples. The main drawback is the so-called "curse of kernelization": to learn the model we must first compute a similarity matrix among all samples, which requires quadratic space and time and is unfeasible for large datasets.
Nonetheless the underlying effective dimension (effective d.o.f.) of the dataset is often much smaller than its size, and we can replace the dataset with a subset (dictionary) of highly informative samples. Unfortunately, fast data-oblivious selection methods (e.g., uniform sampling) almost always discard useful information, while data-adaptive methods that provably construct an accurate dictionary, such as ridge leverage score (RLS) sampling, have a quadratic time/space cost.
In this thesis we introduce a new single-pass streaming RLS sampling approach that sequentially construct the dictionary, where each step compares a new sample only with the current intermediate dictionary and not all past samples. We prove that the size of all intermediate dictionaries scales only with the effective dimension of the dataset, and therefore guarantee a per-step time and space complexity independent from the number of samples. This reduces the overall time required to construct provably accurate dictionaries from quadratic to near-linear, or even logarithmic when parallelized.
Finally, for many non-parametric learning problems (e.g., K-PCA, graph SSL, online kernel learning) we we show that we can can use the generated dictionaries to compute approximate solutions in near-linear that are both provably accurate and empirically competitive.

Résumé

L'avantage principal des méthodes d'apprentissage non-paramétriques réside dans le fait que la nombre de degrés de libertés du modèle appris s'adapte automatiquement au nombre d'échantillons. Ces méthodes sont cependant limitées par le "fléau de la kernelisation": apprendre le modèle requière dans un premier temps de construire une matrice de similitude entre tous les échantillons. La complexité est alors quadratique en temps et espace, ce qui s'avère rapidement trop coûteux pour les jeux de données de grande dimension.
Cependant, la dimension "effective" d'un jeu de donnée est bien souvent beaucoup plus petite que le nombre d'échantillons lui-même. Il est alors possible de substituer le jeu de donnée réel par un jeu de données de taille réduite (appelé "dictionnaire") composé exclusivement d'échantillons informatifs. Malheureusement, les méthodes avec garanties théoriques utilisant des dictionnaires comme "Ridge Leverage Score" (RLS) ont aussi une complexité quadratique.
Dans cette thèse nous présentons une nouvelle méthode d'échantillonage RLS qui met à jour le dictionnaire séquentiellement en ne comparant chaque nouvel échantillon qu'avec le dictionnaire actuel, et non avec l'ensemble des échantillons passés. Nous montrons que la taille de tous les dictionnaires ainsi construits est de l'ordre de la dimension effective du jeu de données final, guarantissant ainsi une complexité en temps et espace à chaque étape indépendante du nombre total d'échantillons. Cette méthode présente l'avantage de pouvoir être parallélisée.
Enfin, nous montrons que de nombreux problèmes d'apprentissage non-paramétriques peuvent être résolus de manière approchée grâce à notre méthode.

Acknowledgements

My gratitude goes to my advisors, Alessandro Lazaric and Michal Valko. Thanking them for teaching me how to be a good scientist would be telling only half of the story, as showing me how to be professional has been priceless for my growth. The hardest part of sequential learning is credit assignment, but it is not difficult for me to see that it is thanks to your advice that I can now face and solve many more problems, in academia and life.

My thanks extend to the whole SequeL team. Thank you Frédéric, Tomáš, Julien, Jean-Bastien, Ronan, Matteo, Alexandre, Florian, and all the other PhDs. It is much more fun to march for a deadline when you have the right company. Thank you to Philippe, Olivier, Jérémie, Romaric, Emilie, Daniil, Odalric, and the other permanents. It has been an exciting experience to work in a group where every day you could discuss about something new with an expert on the topic.

A special thank to Yiannis Koutis, whose work on graphs set me off on my research, and whose help gave me an extra push at critical moments of my PhD. I am also thankful to my reviewers, Michael Mahoney and Nicolò Cesa-Bianchi, for laying the foundations of the problems discussed in this thesis, and to the whole thesis committee.

My PhD would not have been possible today without all the support I received in the past. Grazie Mom, Dad, Franci and Stefi for pushing me when I needed and keeping me grounded. This thesis might be finished, but I hope you will keep asking me how my algorithms are doing for many years to come. And grazie also to my friends Andrea, Matteo, Piero, Lillo, Lanci and Luca that balanced days spent between abstractions.

Most of all, this thanks can not be finished without thanking my Ştefana, just as a day would not be finished without her bringing me inspiration and ungrumpification. Having you made me a better person, and showed me that for some things taking risks is worth it.

Contents

1 Introduction 9
1.1 Why Efficient Machine Learning? 9
1.2 Sequential dictionary learning in a streaming setting 10
1.3 Our contributions 11
1.4 Thesis structure 12
1.5 Notation 13

2 Dictionary Learning for Covariance Matrices 15
2.1 Nyström in an Euclidean space: leverage score sampling 15
2.1.1 Subspace approximation 17
2.1.2 Nyström sampling 19
2.1.3 Uniform sampling 21
2.1.4 Oracle leverage score sampling 23
2.2 Nyström in a RKHS: ridge leverage score sampling 25
2.2.1 Regularized subspace approximation 26
2.2.2 Oracle ridge leverage score sampling 29
2.3 Extended proofs 31

3 Sequential RLS Sampling 37
3.1 Multi-Pass RLS sampling 38
3.1.1 Two-Step RLS sampling 38
3.1.2 Many-Step RLS sampling 42
3.1.3 Recursive RLS sampling 44
3.2 Sequential RLS sampling without removal 46
3.2.1 Oracle RLS sampling in the streaming setting 47

3.2.2 Implementing the oracle and guarantees . . . 50
3.3 Sequential RLS Sampling with removal 51
3.3.1 Oracle RLS sampling with removal . . . 52
3.3.2 Implementing the oracle and guarantees . . . 55
3.3.3 Comparison of all RLS sampling algorithms . . . 60
3.3.4 Straightforward extensions and practical tricks . . . 64
3.3.5 Open questions: lifelong learning and deterministic dictionary learning . . . 64
3.4 Proof of main results 65
3.4.1 Decomposing the merge tree . . . 65
3.4.2 Bounding the projection error $\|\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}\|$ . . . 68
3.4.3 Bounding the martingale $\mathbf{Y}_h$ . . . 70
3.4.4 Bound on predictable quadratic variation $\mathbf{W}_h$ . . . 75
3.4.5 Space complexity bound . . . 84
3.5 Extended proofs 85

4 Sequential RLS Sampling in the Batch Setting . . . 91
4.1 Sequential RLS sampling for kernel matrix approximation 91
4.1.1 Kernel matrix reconstruction . . . 92
4.2 Efficient learning with kernels 94
4.2.1 Kernel ridge regression . . . 94
4.2.2 Kernel principal component analysis . . . 96
4.3 Sequential RLS sampling for graph spectral sparsification 98
4.3.1 Graph spectral sparsification in the semi-streaming setting . . . 98
4.3.2 SQUEAK for graph Laplacian approximation . . . 100
4.4 Efficient graph semi-supervised learning 102
4.4.1 Semi-supervised learning with graph sparsifiers . . . 105
4.4.2 Bounding the stability . . . 106
4.4.3 Experiments on spam detection . . . 109
4.4.4 Recap and open questions: dynamic sparsifiers . . . 112

5 Sequential RLS Sampling in the Online Setting . . . 115
5.1 Online kernel learning and the kernelized online Newton step 118
5.1.1 Kernelized online newton step . . . 119
5.2 Sketched KONS 122
5.2.1 Limitations of budgeted OKL algorithms . . . 125
5.3 The PROS-N-KONS algorithm 127
5.3.1 Regret guarantees . . . 129
5.3.2 Experiments on online regression and classification . . . 132
5.3.3 Recap and open questions: do we need restarts? . . . 134
5.4 Extended proofs 135

References . . . 147

1. Introduction

This thesis focuses on developing efficient algorithms for constrained environments. Here by *efficient* we mean that the algorithm should be able to scale to large problem instances, and the *constraints* on the environment can be *computational*, such as limited memory to store the problem and time to find a solution, or *statistical*, such as limited amount of labeled data to learn on.
Consider an algorithm that can find good solutions for small instances but is too expensive to run on large instances. The final goal of this thesis is to develop new algorithms that, using only a fraction of the resources, can compute an approximate solution that is provably close to the one of the original algorithm.

1.1 Why Efficient Machine Learning?

The recent widespread adoption of machine learning in many fields is leading to the emergence of new challenges. In some applications, we are now collecting data at a faster rate than ever. It is not uncommon for a dataset to contain million of samples, and even more massive dataset are constructed starting from social networks and web crawls composed of billions or even trillions of edges.
Faced with the challenge of scalability, the most commonly used solution is to simply increase the engineering effort. For a long time, it was enough to add more memory as datasets increased. Similarly, if the runtime of the algorithms grew too large, one could wait for Moore's law to bring it back down to feasibility.

The core motivation of this thesis is that, when the size of a problem surpasses a certain threshold, simple hardware or software improvements cannot counterbalance an intrinsically inefficient algorithm. To cope with the growing size of data in our times, the only way forward is the development of more efficient algorithms.

Since the computational problem often lies in the size of the dataset, it seems reasonable

to solve the problem by throwing away a part of our samples and reduce the size of the dataset. Unfortunately, throwing away information can negatively affect the performance of our learning algorithms.
In this thesis we present two new algorithms that can find a small *dictionary* of highly informative samples in a *single pass* over the data. We also prove that the samples contained in these dictionaries can be used in place of the whole dataset to compute approximate solutions to many learning problems, and provably find a solution that is close to the one computed using all the data available.
In particular, we show that this is the case when our goal is to learn a regression function in a linear space, complete the labeling of a graph, or compete against an adversary in an online optimization game.

1.2 Sequential dictionary learning in a streaming setting

Dictionary learning. Given a large collection of objects, that we will call atoms, in a *dictionary learning* problem our goal is to approximate the whole collection using only a small subset of atoms. Simple practical examples of this problem is learning to represent colors, where we only need to combine red, blue and green to cover the whole space, or learning a language, where we can approximately express a word as a combination of other words (e.g., a star is a far sun).
We consider the case where the atoms that we have to choose are vectors belonging to a linear space, which can be infinite-dimensional, i.e., a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$. Given a RKHS and a dataset of samples arbitrarily drawn from this space, we are interested in approximating the spectral information contained in this dataset, using only a small subset of the whole dataset.
This can be done by looking at the covariance matrix of these samples, that contains information on the geometry of the data (eigenvectors), as well as which directions are more important and which less (eigenvalues).
To be accurate, our method will have to adapt to the difficulty of the data. If all samples are very unique and lie in all different directions, we want to keep them all, if they are very redundant and they are all grouped in a few directions, we want to keep only a few representatives.
Unfortunately, deciding the uniqueness of a sample naively is too expensive, since we need to compare each sample against every other sample in the dataset, resulting in a quadratic space and time complexity. Nonetheless, if we already had a small dictionary of representative samples we could *efficiently* compute the uniqueness of each sample in the dataset simply by comparing it to the dictionary.
This is a classical chicken-and-egg problem: we need accurate estimation of the sample's importance to construct a good dictionary, but we need a good dictionary to compute accurate estimations.
To break the cycle, we propose to take a *sequential* sampling approach. We begin by processing a *small* subset of the dataset, making it possible to extract an accurate dictionary out of it efficiently. When we process the next portion of the dataset, we only compare the new data to previously selected dictionary, and update our dictionary if necessary. If the previous dictionary is accurate, we will again choose the right samples and construct an accurate dictionary, repeating this process until the whole dataset is examined.
This will allow us to break the *quadratic barrier* of previous dictionary learning algorithm, since we do not compare each object to each other, and translate this improvements to downstream problems that also could not be solved in linear time. Moreover it will allow

us to apply our algorithms to more stringent computational settings, in particular online and streaming settings.

Streaming algorithms An important metric to measure algorithmic efficiency is how many passes over the dataset the algorithm performs during its execution. The main difference is between algorithms that can operate in a *single* pass over the input vs. algorithm that require multiple passes. The former can be applied not only in cases where all the data is ready beforehand, but also to more difficult setting where the data arrives sequentially over a *stream*, and the algorithm can only access past data if it explicitly stored it. Being able to operate in a streaming setting has a number of practical efficiency consequences, in order to avoid bottlenecks when dealing with with data storage and centralization.
Moreover, streaming algorithms are naturally applicable to sequential and online learning problems, because they provide the option of continuing to update the current solution when more data arrives rather than restarting from scratch.

1.3 Our contributions

Dictionary learning for covariance matrices We propose two new sequential sampling algorithms, KORS and SQUEAK, that efficiently construct a core dictionary capable of approximating the covariance matrix of the dataset up to a small *constant* additive and multiplicative error. This small constant error is achieved with high probability without any assumption on the geometry of the data.
To guarantee this, our methods automatically *adapt* the size of the dictionary until it captures all relevant directions in the data, where a direction is relevant if its associated eigenvalue is larger than our chosen error threshold. The total number of these relevant directions, called the *effective* dimension of the dataset, can be potentially much smaller than the *ambient* dimension of the linear space. If the atoms are very spread out these two dimensions coincide, but if the majority of the atoms follow some structure and lie on a small subspace, the size of our dictionary becomes independent on the number of samples and their dimensionality.
For the problem of spectral approximation in a RKHS, we introduce the first dictionary learning streaming algorithm that operates in a single-pass over the dataset, with a time complexity linear in the number of samples, and a space complexity independent from the number of samples that scales only with the effective dimension of the dataset. Previous results had either a quadratic time complexity, or a space complexity that scaled with the coherence of the dataset, a quantity always larger than the effective dimension.
In addition, we present a modification to our algorithm that can be run in a distributed computation environment. This allows us to introduce the first dictionary learning algorithm that, with enough machines to parallelize the dictionary construction, can achieve a runtime logarithmic in the number of samples.

Learning with Kernels Reconstructing the covariance matrix of the samples allow us to provide reconstruction guarantees for low-rank kernel matrix approximation. Similarly, many existing error analysis for fundamental problems such as kernel ridge regression (KRR) or kernel principal component analysis (PCA) can be extended to work with our dictionary learning methods. Therefore, the approximate solutions computed computed using the dictionary are provably close to the exact ones, and overall both dictionary and approximate solution can be computed in space constant and time linear in the number of samples. This greatly outperforms computing the exact solution, that requires at least

quadratic time and space, and is also an improvement over all other existing dictionary learning algorithms with comparable accuracy guarantees.

Learning on Graph We show that our general dictionary learning approach can be successfully applied to the task of *graph spectral sparsification*, recovering a well known algorithm by Kelner and Levin, (2013). In doing so, we rectify a flaw in the original analysis of the accuracy and computational complexity of the algorithm.
In addition, similarly to how dictionaries are used for kernel learning, we argue that graph sparsifiers can be used to accelerate graph learning algorithms without compromising learning performance. As an example, we show that a semi-supervised learning (SSL) solution computed on the sparsified graph will have almost the same performance as a solution computed on the whole graph, but can be computed in a time that is linear in the number of edges of the graph, and in a space linear in the number of nodes. This is also experimentally validated on graphs with tens of thousands of nodes.

Online learning We present Sketched-KONS and PROS-N-KONS, two approximate second-order algorithms for online kernel learning (OKL). They both achieve the optimal logarithmic regret of exact second-order methods, while reducing their quadratic complexity. Sketched-KONS is the first algorithm to show that it is possible to approximate the second-order updates of an exact second-order OKL method, without losing the logarithmic regret rate. Computationally, Sketched-KONS offers a favorable regret-performance trade-off. For a given factor $\beta \leq 1$, we can increase the regret by a linear $1/\beta$ factor while obtaining a quadratic β^2 improvement in runtime compared to exact methods.
PROS-N-KONS improves on Sketched-KONS, and is the first second-order OKL method that guarantees both logarithmic regret and a sub-quadratic runtime, taking only a near-constant per-step time and space cost that scales only with the effective dimension of the problem.

1.4 Thesis structure

Chapter 2. We formally introduce the problem of dictionary learning for covariance matrices, and provide two definitions that quantify when a dictionary accurately reconstructs the spectrum of a covariance matrix, either in an Euclidean space (Section 2.1) or in a general RKHS (Section 2.2).
We also survey existing batch sampling algorithms, and show how using the *ridge leverage score* (RLS) of a sample to quantify its uniqueness provably constructs small and accurate dictionaries. Unfortunately computing ridge leverage scores is too computationally expensive.
While there exists efficient methods to approximate them when we restrict our problems to Euclidean spaces, prior to the results presented in this manuscript no method could approximate leverage scores in a general RKHS taking less than quadratic time.

Chapter 3. We continue our survey of sampling methods in a general RKHS, presenting two batch algorithms, the first an new result of this thesis and the second a recent result from Musco and Musco, (2017), that can approximate RLS efficiently but require multiple passes over the data (Section 3.1).
Unfortunately multiple passes are extremely expensive or impossible (i.e., streaming setting) on large dataset. We address this problem with two new *sequential* RLS sampling algorithms, that represent the main contribution of the thesis. Both of these algorithms process the dataset incrementally, alternating between using an intermediate dictionary to estimate

the RLSs of new samples, and using the estimate to choose which informative new sample should be inserted in the dictionary and which samples became redundant and must be removed.
We first introduce (Section 3.2) the simpler of the two, KORS, that takes an insertion-only sampling approach, where a sample already included in the dictionary is never removed. Afterwards (Section 3.3), we show with SQUEAK how adding the possibility of removing already sampled atoms from the dictionary allows us to parallelize the sampling process and distribute it across multiple machines.
Rigorously proving that KORS and SQUEAK succeed with high probability (Section 3.4) requires a careful stochastic dominance argument that is of independent interest for adaptive importance sampling algorithms.

This conclude the first part of the thesis dedicated to introducing improved dictionary learning methods. We continue in the second part showing how they can be applied to solve important batch and online machine learning problems.

Chapter 4 We show how our fast approximation scheme can be applied to many batch learning problems that are based on kernels. In particular, using KORS and SQUEAK, we derive linear time algorithms that can compute low-rank approximations of kernel matrices (Section 4.1), and approximate solutions for kernel ridge regression and kernel principal component analysis problems (Section 4.2).
Similarly, we show how a specialized variant of SQUEAK can be used to compute spectral graph sparsifiers (Section 4.3), and how spectral graph sparsifiers can be applied to machine learning problems to derive efficient algorithms (Section 4.4). In particular, we focus on the problem of graph semi-supervised learning, proving that computing an approximate solution on a sparse approximation of the graph constructed using SQUEAK performs almost as well as the exact solution computed on the whole graph.

Chapter 5 We introduce the problem of online kernel learning (Section 5.1) and survey existing first-order and second-order gradient methods proposed to solve them. All of these methods suffer from the *curse of kernelization*, and require at least $\mathcal{O}(t)$ per-step time to guarantee low regret, resulting in a quadratic $\mathcal{O}(T^2)$ runtime.
For the first time, we break this quadratic barrier, while provably maintaining logarithmic regret, using two novel approximate second-order OKL algorithms. Sketched-KONS (Section 5.2) takes the approach of approximating gradient updates, while PROS-N-KONS (Section 5.3) takes the approach of approximating the RKHS where the gradient descent takes place.

1.5 Notation

We now outline the general notation used in this thesis. We will also add more specific notation when the corresponding concept is introduced. Any deviation from this notation is explicitly stated in the text.
We use curly bracket notation $\{a_i\}_{i=1}^n$ to indicate sets of arbitrary elements, possibly with duplicates, and the union operator to represent adding (combining) two sets $\{a_i\}_{i=1}^n \cup \{b_i\}_{i=1}^n$, or a set and a single element $\{a_i\}_{i=1}^n \cup b$. Finally, we use round bracket notation $(\cdot,\cdot,\cdot)$ to indicate tuples, and the set of integers between 1 and n is denoted by $[n] := \{1,\dots,n\}$.
We use upper-case bold letters $\mathbf{A}$ for matrices and maps, lower-case bold letters $\mathbf{a}$ for vectors and points in an RKHS, lower-case letters a for scalars. We denote by $[\mathbf{A}]_{i,j}$ and $[\mathbf{a}]_i$ the (i,j) element of a matrix and i-th element of a vector respectively, with $[\mathbf{A}]_{i,*}$ the

i-th columns of a matrix, and with $[\mathbf{A}]_{*,j}$ its j-th row.
We denote by $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ the identity matrix of dimension n and by $\mathrm{Diag}(\{a_i\}_{i=1}^n) \in \mathbb{R}^{n\times n}$ the diagonal matrix with the elements of the set $\{a_i\}_{i=1}^n$ on the diagonal. We use $\mathbf{e}_{n,i} \in \mathbb{R}^n$ to denote the indicator vector of dimension n for element i. When the dimension of $\mathbf{I}$ and $\mathbf{e}_i$ is clear from the context, we omit the n.
We use $\mathbf{A} \succeq \mathbf{B}$ to indicate the Löwner ordering (Horn and Johnson, 2012), i.e., that $\mathbf{A} - \mathbf{B}$ is a positive semi-definite (PSD) operator. We use $\mathbb{I}\{A\}$ for the $\{0,1\}$ indicator function of event A, and $\#\{\{A_i\}\} = \sum_{i=1}^n \mathbb{I}\{A_i\}$ for the count of how many events are realized in a set of events.

Norms and decompositions. Throughout the thesis norms of vector and matrices are ℓ-2 and induced ℓ-2 norm (operator norm), unless when explicitly indicated. For a vector $\mathbf{a}$, $\|\mathbf{a}\| = \sqrt{\mathbf{a}^\intercal\mathbf{a}}$, and for a matrix $\|\mathbf{A}\| = \sup\{\|\mathbf{A}\mathbf{a}\| : \|\mathbf{a}\| = 1\}$.
Given a matrix $\mathbf{A}$, we indicate with $\mathbf{A} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^\intercal$ its singular value decomposition (SVD), where the matrix $\mathbf{\Sigma} = \mathrm{Diag}(\{\sigma_i\}_{i=1}^{\mathrm{Rank}(\mathbf{A})})$ contains the singular value of $\mathbf{A}$, with $\sigma_n \geq \sigma_{n-1} \geq \cdots \geq \sigma_1 > 0$. Moreover, we indicate with $\sigma_{\max} = \sigma_n$ the largest singular value, and with $\sigma_{\min}$ the smallest.
Similarly, given a symmetric matrix $\mathbf{A}$, we indicate with $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\intercal$ its eigenvalue decomposition (eigendecomposition), where the matrix $\mathbf{\Lambda} = \mathrm{Diag}(\{\lambda_i\}_{i=1}^{\mathrm{Rank}(\mathbf{A})})$ contains the singular value of $\mathbf{A}$, with $\lambda_n \geq \lambda_{n-1} \geq \cdots \geq \lambda_1$. Moreover, we indicate with $\lambda_{\max} = \lambda_n$ the largest eigenvalue. and with $\lambda_{\min}$ the smallest (possibly negative).
Note that the induced ℓ-2 norm of a matrix is equivalent to its largest singular value $\sigma_{\max}$, which is equal to the eigenvalue with largest magnitude in self-adjoin operators (symmetric matrices).

Computational complexity. We use $\mathcal{O}(\cdot)$ to indicate the usual big-O notation, and Ω for the big-Omega notation. We use $\widetilde{\mathcal{O}}(\cdot)$ for a big-O notation that ignores polylog$(\cdot)$ terms.

2. Dictionary Learning for Covariance Matrices

In this chapter we formally introduce the problem of dictionary learning, first for finite dimensional spaces $\mathbb{R}^d$ and then for general Reproducing Kernel Hilbert Spaces (RKHS) $\mathcal{H}$. In particular, we introduce a metric to quantify the *accuracy* of a dictionary, and survey existing *batch sampling* methods that can construct small and accurate dictionaries. Unfortunately, these batch sampling methods are too computationally expensive and require an *oracle* to execute. How to implement such an oracle in practice will be the focus of Chapter 3.

2.1 Nyström in an Euclidean space: leverage score sampling

Consider a collection $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$ of n vectors $\mathbf{x}_i$ selected *arbitrarily*[1] from $\mathbb{R}^d$, possibly with duplicates, and let $\mathbf{X} \in \mathbb{R}^{d \times n}$ be the matrix with $\mathbf{x}_i$ as its i-th column. Throughout this section we will focus on the case $n \gg d$ where we have many more samples than dimensions. Associated with $\mathbf{X}$ we have the covariance matrix

$$\mathbf{X}\mathbf{X}^\intercal = \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\intercal.$$

In machine learning the covariance matrix is a recurrent element, which can be used among other things to extract principal components, fit (non-)parametric linear models and perform density estimation (Hastie et al., 2009). The space and time complexity of all of these methods scale proportionally to both the dimension of the covariance matrix d, as well as the number n of rank-1 matrices $\mathbf{x}_i\mathbf{x}_i^\intercal$ contained in it. When n is very large, it is computationally very hard to process $\mathbf{X}\mathbf{X}^\intercal$, so we need to reduce this dependency. Given our dataset $\mathcal{D}$, in this section we are interested in finding a dictionary $\mathcal{I}$ that satisfies two goals:

[1] i.e., not necessarily according to a sampling distribution.

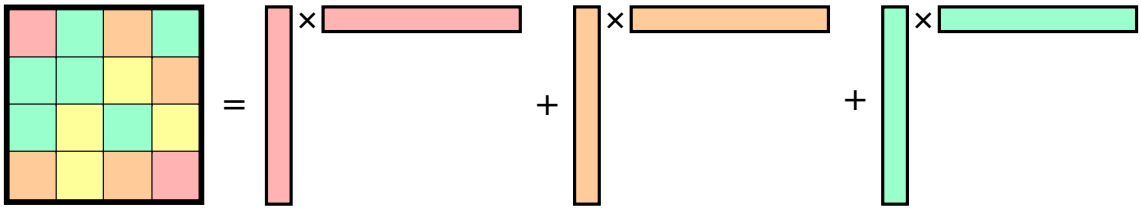


Figure 2.1: Dictionary Learning for Covariance Matrices

1. we can use $\mathcal{I}$ to represent $\mathcal{D}$,
2. the dictionary's size $|\mathcal{I}|$ is small and does not scale with n,

A *dictionary* is a generic collection of weighted atoms $\mathcal{I} := \{(\text{weight}, \text{atom})\} = \{(s_i, \mathbf{a}_i)\}$, where the *atoms* $\mathbf{a}_i$ are arbitrary elements of $\mathbb{R}^d$, and s_i are non-negative weights[2]. The size of the dictionary $|\mathcal{I}|$ is the number of distinct atoms with a non-zero weight. We say that a dictionary $\mathcal{I}$ can be used to exactly *represent* dataset $\mathcal{D}$ when it is possible to exactly reconstruct the covariance matrix $\mathbf{X}\mathbf{X}^\intercal = \sum s_i \mathbf{a}_i \mathbf{a}_i^\intercal$ as a weighted sum of the elements in the dictionary (see Fig. 2.1). If the dictionary correctly reconstructs $\mathbf{X}\mathbf{X}^\intercal$ but its size does not scale with n (i.e., it is *small*), we can use it in place of $\mathbf{X}\mathbf{X}^\intercal$ to answer our machine learning questions (e.g., fit a linear model).

Clearly the dictionary necessary to represent $\mathbf{X}\mathbf{X}^\intercal$ is not unique, and many alternatives are possible, that differ in the choice of atoms and weights. We will now consider some options:

Without learning. The simplest choice for a dictionary that needs to represent $\mathcal{D}$ is $\mathcal{D}$ itself, that is we do not change neither atoms or weights and set $\mathcal{I} = \{(1, \mathbf{x}_i)\}_{i=1}^n$. This perfectly reconstructs $\mathcal{D}$ since $\sum_{i=1}^n 1\mathbf{x}_i\mathbf{x}_i^\intercal = \mathbf{X}\mathbf{X}^\intercal$, but does not save any space.

Learn atoms.[3] As soon as $n > d$, $\mathcal{I} = \mathcal{D}$ is not the most succinct (in number of atoms) representation possible. Consider the thin Singular Value Decomposition of $\mathbf{X} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^\intercal$ and let r be the rank of $\mathbf{X}$

1. $\mathbf{V} \in \mathbb{R}^{d\times r}$ contains the orthonormal left singular vectors $\mathbf{v}_i \in \mathbb{R}^d$ as columns, such that $\mathbf{V}^\intercal\mathbf{V} = \mathbf{I}_r$ and $\mathbf{V}\mathbf{V}^\intercal$ is the projection on $\text{Im}(\mathbf{X})$.
2. $\boldsymbol{\Sigma} : \mathbb{R}^{r\times r}$ is the diagonal matrix where $[\boldsymbol{\Sigma}]_{i,i} = \sigma_i$ is the i-th singular value, and $\sigma_r \geq \sigma_{r-1} \geq \cdots \geq \sigma_1 > 0$.
3. $\mathbf{U} : \mathbb{R}^{n\times r}$ contains the orthonormal right singular vectors $\mathbf{u}_i \in \mathbb{R}^n$ as columns, such that $\text{Im}(\mathbf{U}) = \text{Im}(\mathbf{X}^\intercal)$.

Then we can rewrite

$$\mathbf{X}\mathbf{X}^\intercal = \mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^\intercal\mathbf{U}\boldsymbol{\Sigma}^\intercal\mathbf{V}^\intercal\mathbf{V}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^\intercal\mathbf{V}^\intercal = \sum_{i=1}^r \sigma_i^2 \mathbf{v}_i\mathbf{v}_i^\intercal$$

and use $\mathcal{I} = \{1, \sigma_i\mathbf{v}_i\}$ as a dictionary for $\mathcal{D}$, replacing the atoms $\mathbf{x}_i$ with $\sigma_i\mathbf{v}_i$. Clearly the number of atoms in the SVD-based dictionary is smaller than the one in the original dataset, but unfortunately when n is very large computing the SVD is prohibitively expensive.

Learn weights but not atoms. This is the family of dictionaries $\mathcal{I} = \{(s_i, \mathbf{x}_i)\}_{i=1}^n$. When $s_i = 1$ for all i (i.e., $\mathcal{I} = \mathcal{D}$) we saw that this is not the most succinct representation

[2] Since the matrix $\mathbf{X}\mathbf{X}^\intercal$ is PSD, non-negative weights are sufficient for reconstruction.

[3] This approach is equivalent to learning both atoms and weights, since we can always rescale the learned atoms to include any necessary weight.

possible, and there must be some redundancies that we can eliminate.
A simple example of redundancy can be seen when $\mathbf{x}_i$ and $\mathbf{x}_{i'}$ are equal. In this case, we can simply remove $\mathbf{x}_{i'}$ from the dictionary setting $s_{i'} = 0$ and compensate by setting $s_i = 2$. This leaves $\sum_{i=1}^n s_i \mathbf{x}_i \mathbf{x}_i^\intercal = \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\intercal = \mathbf{X}\mathbf{X}^\intercal$ unchanged. In other words we can leave the atoms $\mathbf{x}_i$ unchanged, but adapt the weights s_i to the data in order to reduce the number of non-zero weights and the size of the dictionary.

While general dictionary learning consider the overall problem of jointly learning the weights and atoms, in the remainder of this thesis we will focus only on this last approach of selecting and reweighting a subset of $\mathcal{D}$ itself, sometime called *sparse coding* or sparse approximation. Given the selected subset, we can define the selection matrix $\mathbf{S}$ associated with a dictionary as $\mathbf{S} = \mathrm{Diag}(\{\sqrt{s_i}\}_{i=1}^n) \in \mathbb{R}^{n\times n}$. Then $\mathbf{X}\mathbf{S}$ is a $\mathbb{R}^{d\times n}$ matrix where the atoms contained in $\mathcal{I}$ are properly reweighted so that

$$\mathbf{X}\mathbf{S}\mathbf{S}^\intercal\mathbf{X}^\intercal = \sum_{i=1}^n s_i \mathbf{x}_i \mathbf{x}_i^\intercal.$$

Unfortunately (even when $n > d$) it is not always possible to find a subset of input samples $\mathbf{x}_i$ that is strictly smaller than $\mathcal{D}$ and exactly reconstructs $\mathcal{D}$. If we want to reduce the size of the dictionary we must incur in some *approximation error*. We will consider two metrics to quantify our approximation error. We present the first here, and postpone the second to Section 2.2.1.

2.1.1 Subspace approximation

As we mentioned before, the covariance matrix $\mathbf{X}\mathbf{X}^\intercal$ plays a central role in many linear problems, such as linear regression or PCA, that try to find the vector or vectors in a linear space that best fit the data collected. More formally, the solution to all of these problems must lie in $\mathrm{Im}(\mathbf{X}) = \mathrm{Im}(\mathbf{X}\mathbf{X}^\intercal)$, and is computed using the spectral information contained in $\mathbf{X}\mathbf{X}^\intercal$, in the form of eigenvectors and eigenvalues. Therefore if the span and the spectrum of $\mathbf{X}\mathbf{S}\mathbf{S}^\intercal\mathbf{X}$ is *close* to that of $\mathbf{X}\mathbf{X}^\intercal$, we can use only the data contained in the dictionary to compute an accurate approximate solution. Chapters 4 and 5 will quantify more rigorously how an accurate dictionary translates into a good approximate solution, while we will now give our definition of accurate dictionaries.

We begin by introducing the orthogonal projection matrix on $\mathrm{Im}(\mathbf{X})$ as a proxy for the spectrum of $\mathbf{X}$,

Definition 2.1 The orthogonal projection $\boldsymbol{\Pi}$ on $\mathrm{Im}(\mathbf{X})$ is defined as

$$\boldsymbol{\Pi} = (\mathbf{X}\mathbf{X}^\intercal)^{+/2}\mathbf{X}\mathbf{X}^\intercal(\mathbf{X}\mathbf{X}^\intercal)^{+/2} = \sum_{i=1}^n \boldsymbol{\pi}_i\boldsymbol{\pi}_i^\intercal, \tag{2.1}$$

with $\boldsymbol{\pi}_i = (\mathbf{X}\mathbf{X}^\intercal)^{+/2}\mathbf{x}_i$. The approximate orthogonal projection $\widetilde{\boldsymbol{\Pi}}$ for a dictionary $\mathcal{I}$ and its selection matrix $\mathbf{S}$ is defined as

$$\widetilde{\boldsymbol{\Pi}} = (\mathbf{X}\mathbf{X}^\intercal)^{+/2}\mathbf{X}\mathbf{S}\mathbf{S}^\intercal\mathbf{X}^\intercal(\mathbf{X}\mathbf{X}^\intercal)^{+/2} = \sum_{i=1}^n s_i\boldsymbol{\pi}_i\boldsymbol{\pi}_i^\intercal. \tag{2.2}$$

Note that $\boldsymbol{\Pi} = \mathbf{I}_d$ when $\mathbf{X}$ is full rank, and that we can also express $\boldsymbol{\Pi} = \sum_{i=1}^r \mathbf{v}_i\mathbf{v}_i^\intercal$, i.e., the SVD dictionary can represent $\boldsymbol{\Pi}$ using only r atoms, while the $\{\boldsymbol{\pi}_i\}_{i=1}^n$ dictionary

requires n atoms. Finally, note that the approximate orthogonal projection $\widetilde{\mathbf{\Pi}}$ is not the orthogonal projection on $\mathrm{Im}(\mathbf{XS})$

$$\widetilde{\mathbf{\Pi}} \neq \mathbf{\Pi}_{\mathcal{I}} := (\mathbf{XSS}^\intercal\mathbf{X}^\intercal)^{+/2}\mathbf{XSS}^\intercal\mathbf{X}^\intercal(\mathbf{XSS}^\intercal\mathbf{X}^\intercal)^{+/2}.$$

We chose this definition for $\widetilde{\mathbf{\Pi}}$ because we are not only interested in reconstructing the subspace spanned by $\mathbf{XX}^\intercal$, but we also want to reconstruct its defining spectral properties. To show that $\widetilde{\mathbf{\Pi}}$ is the right tool for the job, we use Definition 2.2 and Proposition 2.1.

Definition 2.2 A dictionary $\mathcal{I} = \{(s_i, \mathbf{x}_i)\}_{i=1}^n$ and its associated selection matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$ are ε-accurate w.r.t. a matrix $\mathbf{X}$ if

$$\|\mathbf{\Pi} - \widetilde{\mathbf{\Pi}}\| = \|(\mathbf{XX}^\intercal)^{+/2}\mathbf{X}(\mathbf{I}_n - \mathbf{SS}^\intercal)\mathbf{X}^\intercal(\mathbf{XX}^\intercal)^{+/2}\| \leq \varepsilon \tag{2.3}$$

Proposition 2.1 — (Kelner and Levin, 2013).

$$\begin{gathered}\|(\mathbf{XX}^\intercal)^{+/2}(\mathbf{XX}^\intercal - \mathbf{XSS}^\intercal\mathbf{X}^\intercal)(\mathbf{XX}^\intercal)^{+/2}\| \leq \varepsilon \\ \Updownarrow \\ (1-\varepsilon)\mathbf{XX}^\intercal \preceq \mathbf{XSS}^\intercal\mathbf{X}^\intercal \preceq (1+\varepsilon)\mathbf{XX}^\intercal\end{gathered} \tag{2.4}$$

Due to the definition of Löwner PSD ordering, if the dictionary $\mathcal{I}$ is ε-accurate Proposition 2.1 guarantees that, for any $\mathbf{a} \in \mathbb{R}^d$, $\mathbf{a}^\intercal\mathbf{XSS}^\intercal\mathbf{X}^\intercal\mathbf{a}$ is at most a $(1 \pm \varepsilon)$ constant multiplicative error away from $\mathbf{a}^\intercal\mathbf{XX}^\intercal\mathbf{a}$. Since $\mathbf{a}^\intercal\mathbf{XX}^\intercal\mathbf{a} = 0$ only if $\mathbf{a} \in \mathrm{Ker}(\mathbf{XX}^\intercal)$, this implies that $\mathrm{Ker}(\mathbf{XX}^\intercal) = \mathrm{Ker}(\mathbf{XSS}^\intercal\mathbf{X})$, $\mathrm{Im}(\mathbf{XX}^\intercal) = \mathrm{Im}(\mathbf{XSS}^\intercal\mathbf{X})$, and the dictionary perfectly reconstructs the projection $\mathbf{\Pi}$

$$\mathbf{\Pi} = (\mathbf{XX}^\intercal)^{+/2}\mathbf{XX}^\intercal(\mathbf{XX}^\intercal)^{+/2} = (\mathbf{XSS}^\intercal\mathbf{X}^\intercal)^{+/2}\mathbf{XSS}^\intercal\mathbf{X}^\intercal(\mathbf{XSS}^\intercal\mathbf{X}^\intercal)^{+/2} = \mathbf{\Pi}_{\mathcal{I}}.$$

We can also show that the spectrum of $\mathbf{XX}^\intercal$ is preserved. In particular, ε-accuracy and Proposition 2.1 implies that all eigenvalues of $\mathbf{XX}^\intercal$ are approximated by $\mathbf{XSS}^\intercal\mathbf{X}^\intercal$ up to the same constant factor. Therefore, ε-accuracy not only preserves the subspace that contains $\mathbf{X}$, but also approximates well the importance of each direction in that subspace.
As we will see this translates in stronger guarantees for complex problems such as PCA and regression. In addition, since $\mathbf{XSS}^\intercal\mathbf{X}^\intercal$ has to approximate at least r eigenvalues and eigenvectors, it is easy to see that an ε-accurate dictionary must contain at least r atoms (e.g., to approximate a diagonal matrix such as $\mathbf{I}_r$, we need at least r atoms).

Now that we defined an accuracy measure, there are many possible ways to try to find a dictionary and selection matrix $\mathbf{S}$ that satisfy it.
Gradient descent. We can rewrite Eq. 2.3 as a function of the weights s_i as

$$\min_{\{s_i\}_{i=1}^n} \left\| \sum_{i=1}^n (1 - s_i)\boldsymbol{\pi}_i\boldsymbol{\pi}_i^\intercal \right\|$$

A simple approach would be to use gradient descent mixed with an ℓ-0 or ℓ-1 penalty on the weights s_i to minimize this objective. Unfortunately computing the vectors $\boldsymbol{\pi}_i$ requires $\mathcal{O}(nd^2)$ time and is not efficient. Moreover computing a gradient w.r.t. n weights s_i, and repeating this operation at each step again makes the algorithm's complexity scale poorly with n.
Greedy choice with barrier function. Instead of performing expensive full gradient

Algorithm 1 Nyström method

Input: Budget $\overline{q}$, set of probabilities $\{p_i\}_{i=1}^n$ such that for all i $0 < p_i \leq 1$
Output: $\mathcal{I}$

1: Initialize $\mathcal{I} = \emptyset$
2: **for** $i = \{1, \dots, n\}$ **do**
3: Draw the Binomial r.v. $q_i \sim \mathcal{B}(p_i, \overline{q})$
4: If $q_t \neq 0$, add $\left(\frac{1}{p_i}\frac{q_i}{\overline{q}}, \mathbf{x}_i\right)$ to $\mathcal{I}$
5: **end for**

updates, several papers (Batson et al., 2012; Silva et al., 2016) propose to use simpler greedy methods, providing increasingly stronger guarantees for these approaches. Given a partial dictionary, their strategy is to sequentially compute which would be the vector $\boldsymbol{\pi}_i$ that would further the most the approximation goal while keeping the existing dictionary fixed.

Batson et al., (2012) show that using appropriately constructed barrier functions to quantify the usefulness of each $\boldsymbol{\pi}_i$ vector, it is possible to prove that this sequential approach generates an ε-accurate and in many cases optimally small dictionary. Unfortunately each step of this strategy requires solving a linear system in $\mathbf{X}$ and it is not computationally efficient, unless $\mathbf{X}$ satisfies stringent structural assumptions, e.g., when $\mathbf{X}\mathbf{X}^\intercal$ is the Laplacian of a graph (Lee and Sun, 2015).

Random sampling. A traditional (Williams and Seeger, 2001) and common heuristic to construct a dictionary is to build an unbiased estimator of $\boldsymbol{\Pi}$ choosing the weights with uniform sampling. In particular, we sample a small subset of q atoms uniformly at random (i.e., with probability $p = 1/n$), and set their weight to $s_i = 1/(pq) = n/q$. This strategy is simple, requires no knowledge of $\boldsymbol{\pi}_i$, and almost no computation.

Unfortunately, it often fails to construct an ε-accurate dictionary (Bach, 2013). Nonetheless if we replace the *data oblivious* uniform distribution with a more accurate *data adaptive* distribution, we can improve our accuracy. In the next section we will focus on this family of sampling methods, and discuss how to find a good data adaptive distribution.

2.1.2 Nyström sampling

In Section 2.1 we restricted our attention to dictionaries that select and reweight a subset of the input matrix. The Nyström method is a general name for all methods that choose this subset according to a random process. The blueprint for batch Nyström sampling is reported in Algorithm 1.

The algorithm iterates over each atom $\mathbf{x}_i$ and draws a Binomial random variable $q_i \sim \mathcal{B}(p_i, \overline{q})$ containing $\overline{q}$ independent copies of a Bernoulli $Ber(p_i)$. While we refer to q_i as the number of copies of atom $\mathbf{x}_i$, Algorithm 1 does not explicitly add duplicate copies of $\mathbf{x}_i$ to the dictionary $\mathcal{I}$, but a single copy with weight $\frac{1}{p_i}\frac{q_i}{\overline{q}}$.

Notice that traditionally Nyström sampling draws $\overline{q}(\sum_{i=1}^n p_i)$ samples without (Williams and Seeger, 2001) or with (Alaoui and Mahoney, 2015) replacement from the multinomial distribution $\mathcal{M}(\{p_i/\sum_{i=1}^n p_i\})$. Algorithm 1 draws $\overline{q}$ samples from each of the n Binomials $\mathcal{B}(p_i, \overline{q})$, which can be seen as a "Bernoulli with replacement". On a statistical level this sampling scheme differs from Multinomial sampling in two aspects.

The first difference is that the size of the dictionary $|\mathcal{I}| = q = \sum_{i=1}^n q_i$ is not fixed in advance to $\overline{q}(\sum_{i=1}^n p_i)$ but random and equal to the number of non-zero weights contained in the dictionary $|\mathcal{I}| = \sum_{i=1}^n \mathbb{I}\{\frac{1}{p_i}\frac{q_i}{\overline{q}} \neq 0\} = \sum_{i=1}^n \mathbb{I}\{q_i \neq 0\}$. Nonetheless, we can bound w.h.p.

the final size of the dictionary to show that it is at most $3\overline{q}(\sum_{i=1}^n p_i)$.

Proposition 2.2 Given probabilities $\{p_i\}$ and budget $\overline{q}$, let $\mathcal{I}$ be the dictionary generated by Algorithm 1. Then

$$\mathbb{P}\left(|\mathcal{I}| \geq 3\overline{q}\left(\sum_{i=1}^n p_i\right)\right) \leq \exp\left\{-\overline{q}\left(\sum_{i=1}^n p_i\right)\right\}.$$

Proof sketch of Proposition 2.2. While the true size of the dictionary is the number of non-zero weights, it is easier to analyze the upper bound $|\mathcal{I}| \leq \sum_{i=1}^n q_i$, as if we assumed that all the copies included in the dictionaries belonged to different atoms, or we explicitly stored each copy of an atom separately. Then we can use a Hoeffding-like bound, proved in Proposition 2.13, to bound the sum of independent, non-identically distributed Binomials $\sum_{i=1}^n q_i$. ■

The second difference is that, although in expectation both the number of drawn samples for each atom ($\mathbb{E}[q_i] = \overline{q}p_i$) and in total ($\mathbb{E}[q] = \overline{q}\sum_{i=1}^n p_i$) match between the two approaches, our sequential Binomial sampling is not equivalent (same distribution) as a series of $\overline{q}(\sum_{i=1}^n p_i)$ Multinomial draws with replacement[4].
On an algorithmic level, unlike traditional Nyström, we do not need to explicitly compute the normalization constant $\sum_{i=1}^n p_i$ to perform the sampling procedure. This will become useful in Section 2.2.2, where the p_i will be computed starting from data.
Other algorithm that use this Binomial sampling approach are present in the literature (Kelner and Levin, 2013), as well as variants that draw q_i proportionally to an overestimated Bernoulli $Ber(p_i\overline{q})$ rather than a Binomial $\mathcal{B}(p_i, \overline{q})$ (Cohen et al., 2015a; Cohen et al., 2015b; Cohen et al., 2016; Musco and Musco, 2017). It is easy to see that

$$\mathbb{P}(Ber(p_i\overline{q}) = 0) = 1 - p_i\overline{q} \leq (1 - p_i)^{\overline{q}} = \mathbb{P}(\mathcal{B}(p_i, \overline{q}) = 0),$$

and drawing from the Binomial will produce smaller dictionaries.
From the definition of sampling probabilities p_i and weights $\frac{1}{p_i}\frac{q_i}{\overline{q}}$ we can see that the dictionary $\mathcal{I}$ is an unbiased estimator of $\mathcal{D}$ since

$$\mathbb{E}[\mathbf{X}\mathbf{S}\mathbf{S}^\intercal\mathbf{X}^\intercal] = \sum_{i=1}^n \frac{1}{p_i}\frac{\mathbb{E}[q_i]}{\overline{q}}\mathbf{x}_i\mathbf{x}_i^\intercal = \sum_{i=1}^n \cancelto{1}{\frac{p_i\overline{q}}{p_i\overline{q}}}\,\mathbf{x}_i\mathbf{x}_i^\intercal = \mathbf{X}\mathbf{X}^\intercal.$$

This holds for any valid set of probabilities p_i and any $\overline{q}$ but it is easy to see that some choices are better than others. For example selecting extremely small p_i for all i and $\overline{q} = 1$ will almost certainly result in an empty (and very inaccurate) dictionary. From an accuracy perspective, we want to know when $\mathbf{X}\mathbf{S}\mathbf{S}^\intercal\mathbf{X}^\intercal$, and in turn $\widetilde{\mathbf{\Pi}}$, are close to their mean $\mathbf{X}\mathbf{X}^\intercal$ and $\mathbf{\Pi}$, or in other words when $\mathbf{X}\mathbf{S}\mathbf{S}^\intercal\mathbf{X}^\intercal$ concentrates. From a computational improvement perspective, we want to know when the generated dictionary $\mathcal{I}$ is small. Two factors impact these two quantities.

The probabilities

The probabilities p_i used to sample the Binomials tells us *which* atoms to choose and *how many*. More in general, the probability p_i should tell us how unique or redundant each sample is, and the sum of these probabilities $\sum p_i$ encodes how many atoms overall are

[4] In particular, drawing $\overline{q}$ elements from a Multinomial correspond to the sequence of Binomial draws $q_1 \sim \mathcal{B}(p_1, \overline{q}), q_2|q_1 \sim \mathcal{B}(p_2/(1-p_1), \overline{q} - q_1), \ldots, q_{n-1} \sim \mathcal{B}(p_{n-1}/(1 - \sum_{i=1}^{n-2} p_i), \overline{q} - \sum_{i=1}^{n-2} q_i)$

necessary to represent the dataset. The concentration of $\mathbf{X}\mathbf{S}\mathbf{S}^\intercal\mathbf{X}^\intercal$ around its mean will be guided by how closely the distribution p_i captures the true underlying redundancies.
Remember from the previous section that we can find a small dictionary that accurately reconstruct the dataset because there are redundancies between samples. Consider the example dataset $\mathcal{D} = \{\mathbf{x}, \mathbf{x}, \mathbf{x}', \mathbf{x}', \mathbf{x}'\}$ with two identical copies of $\mathbf{x}$, three identical copies of $\mathbf{x}'$, and $\mathbf{x}^\intercal\mathbf{x}' = 0$ orthogonal. The dictionary that perfectly reconstruct this dataset is $\{(2, \mathbf{x}), (3, \mathbf{x}')\}$ with minimal size 2. How would we set our probabilities in Algorithm 1 to generate this dictionary? To obtain 2 and 3 as our weights, we set $p_1 = p_2 = 1/2$ and $p_3 = p_4 = p_5 = 1/3$, with $\sum p_i = 2$ total number of needed atoms.
This example also highlights why we need randomization: selecting greedily the two most unique atoms associated with the highest probabilities ($\mathbf{x}_1$ and $\mathbf{x}_2$) will not generate a good dictionary.

The number of copies

The budget $\overline{q}$ tells us the *extra budget* we have to pay to compensate the intrinsic randomness induced by the random sampling, and any mistake in choosing the probability p_i (mismatch between chosen p_i and true redundancies).
Take again the example $\mathcal{D} = \{\mathbf{x}, \mathbf{x}, \mathbf{x}', \mathbf{x}', \mathbf{x}'\}$ with $p_1 = p_2 = 1/2$ and $p_3 = p_4 = p_5 = 1/3$. If we set $\overline{q} = 1$, simply due to the way q_1 and q_2 are randomly selected, we have a $1/4$ possibility to generate a bad dictionary that does not contain $\mathbf{x}$. Increasing $\overline{q}$ reduces the probability of this event happening.
Similarly, if we underestimate the importance of a sample (e.g., set $p_1 = p_2 = 1/4$), increasing $\overline{q}$ will bring the final dictionary close to the mean no matter what kind of error we make. While increasing $\overline{q}$ helps with accuracy, we saw that our dictionary size increases as $\overline{q}$, so raising it excessively might result in a very large dictionary that will not improve our computational complexity.

2.1.3 Uniform sampling

To see an example of how the probabilities and budget $\overline{q}$ impact accuracy and dictionary size, we begin by considering *uniform* probabilities $p_i = 1/n$. We will refer to Algorithm 1 using uniform probabilities as Uniform.
Historically the first Nyström methods introduced in machine learning (Williams and Seeger, 2001) used uniform sampling, and it remains a popular choice for its conceptual simplicity and the fact that computing this distribution takes no computation at all. More recently, Bach, (2013) provided sufficient conditions under which uniform sampling generates an ε-accurate dictionary, using the dataset's coherence to quantify the number of samples necessary.

Definition 2.3 — (Drineas et al., 2012). Given a matrix $\mathbf{X}$, the coherence μ of $\mathbf{X}$ is

$$\mu = \max_{i=1,\dots,n} \|\boldsymbol{\pi}_i\boldsymbol{\pi}_i^\intercal\| = \max_{i=1,\dots,n} \|\boldsymbol{\pi}_i\|^2 \qquad 2.5$$

Proposition 2.3 — (Bach, 2013, App. B1, Lem. 2). Consider an arbitrary matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ with $\mathbf{x}_i$ as its i-th column and rank r. Let $\boldsymbol{\Pi}$ and $\widetilde{\boldsymbol{\Pi}}$ be defined according to

Definition 2.1. Let $\mathcal{I}$ be the dictionary generated by Algorithm 1. If

$$p_i = 1/n, \qquad \overline{q} \geq n\mu \frac{4\log(2d/\delta)}{\varepsilon^2},$$

with $\sum_{i=1}^n p_i = 1$, then for any $0 < \varepsilon < 1$ and w.p.$1-\delta$

Accuracy: $\mathcal{I}$ is ε-accurate.

Space: the size of the dictionary is bounded by $|\mathcal{I}| \leq 3\overline{q}$.

Proposition 2.3 tells us that to construct an ε-accurate dictionary, we should sample roughly $\mathcal{O}(n\mu)$ atoms, which can correspond to almost the whole dataset when μ is close to 1. Unfortunately, estimating the coherence μ is a hard problem in itself (Drineas et al., 2012), therefore running uniform sampling with guarantees is not easy. As a solution, it is common in the Nyström literature to make an *assumption* of low coherence.
Since the coherence μ is directly related to how much samples in the dataset are orthogonal with each other, Proposition 2.3 also gives us intuition on how uniform sampling can fail. For example, consider the case where a single sample $\mathbf{x}_i$ is completely orthogonal to the others ($\mu = 1$), and we try to use uniform probabilities in Algorithm 1 but set $\overline{q}$ to a constant (w.l.o.g. $\overline{q} = 1$) to have a dictionary size that does not scale with n. Then the probability of not including the orthogonal sample is going to be roughly $(1 - 1/n) \sim 1$. And if we do not include the sample, it is easy to see that $\|\mathbf{\Pi} - \widetilde{\mathbf{\Pi}}\| \sim 1$ simply by testing these two matrices along the sample direction, where $\mathbf{x}_i^\intercal \mathbf{\Pi} \mathbf{x}_i = \mathbf{x}_i^\intercal \mathbf{x}_i$ and $\mathbf{x}_i^\intercal \widetilde{\mathbf{\Pi}} \mathbf{x}_i = 0$ since $\mathbf{x}_i$ is orthogonal to all other samples. Note that this problem will not happen with high probability only when a single sample is orthogonal. As an example, imagine that we have $n^{1/2}$ vectors identical and orthogonal to the rest, we are still going to miss them with probability roughly $(1 - 1/n^{1/2})$.
This also is in line with our intuition of $\overline{q}$ as an extra budget to compensate inaccuracies in choosing p_i. When the coherence is small, and no sample is orthogonal to the others, all samples look the same and have the same importance of roughly $1/n$, making uniform sampling very accurate. But if the coherence is high, uniform probabilities are very likely to not correspond at all to the true underlying importance of each samples (they are chosen before even looking at the samples), and to concentrate we need a large number of copies $\overline{q}$ that potentially scales with n.

To see how to remove this dependency on the coherence, let us first give a sketch of Proposition 2.3's proof. The proof is based on the following general concentration for matrices,

Proposition 2.4 — (Tropp, 2012, Theorem 1.6). Consider a sequence $[\mathbf{X}_j]$ of independent random symmetric matrices with dimension t. Assume that $\mathbb{E}[\mathbf{X}_j] = 0$, $\|\mathbf{X}_j\| \leq R$, and let $\mathbf{Y} = \sum_j \mathbf{X}_j$. Furthermore, assume that there exists $\sigma > 0$ that bounds the matrix variance statistic of the sum $\|\mathbb{E}[\mathbf{Y}^2]\|_2 \leq \sigma^2$. Then

$$\mathbb{P}(\|\mathbf{Y}\| \geq \varepsilon) \leq t \exp\left\{-\frac{\varepsilon^2/2}{\sigma^2 + R\varepsilon/3}\right\} .$$

Proof sketch of Proposition 2.3. We provide a sketch of the proof here and include the full result in Section 2.3. Intuitively, the core of the proof consist in finding appropriate values for R and σ^2 to apply Proposition 2.4.
In our case we use uniform sampling with probabilities $p_i = 1/n$ and the matrix we are

interested in is $\mathbf{Y} = \mathbf{\Pi} = \sum_{i=1}^{n} \boldsymbol{\pi}_i \boldsymbol{\pi}_i^\intercal$. We can show that

$$R := \frac{1}{\bar{q}} \max_i \frac{\|\boldsymbol{\pi}_i\|^2}{p_i} = \frac{n \max_i \|\boldsymbol{\pi}_i\|^2}{\bar{q}} = \frac{n\mu}{\bar{q}},$$
$$\sigma^2 := \frac{1}{\bar{q}} \max_i \frac{\|\boldsymbol{\pi}_i\|^2}{p_i} \|\mathbf{\Pi}\| = \frac{n}{\bar{q}} \max_i \|\boldsymbol{\pi}_i\|^2 \|\mathbf{\Pi}\| \leq \frac{n\mu}{\bar{q}} \|\mathbf{\Pi}\| \leq \frac{n\mu}{\bar{q}},$$

satisfy the necessary conditions, and then apply the proposition. ■

2.1.4 Oracle leverage score sampling

From the proof sketch of Proposition 2.3 we saw that the norms of the individual atoms $\|\boldsymbol{\pi}_i\|^2 = \|(\mathbf{X}\mathbf{X}^\intercal)^{+/2}\mathbf{x}_i\mathbf{x}_i^\intercal(\mathbf{X}\mathbf{X}^\intercal)^{+/2}\|$ plays a large role in bounding the range and variance of the random process induced by Algorithm 1. These quantities are well known in Statistics under the name of leverage scores.

Definition 2.4 — (Drineas et al., 2012). Given a matrix $\mathbf{X}$, the leverage score (LS) of column $\mathbf{x}_i \in \mathbf{X}$ is

$$\tau_i = \|\boldsymbol{\pi}_i\boldsymbol{\pi}_i^\intercal\| = \|(\mathbf{X}\mathbf{X}^\intercal)^{+/2}\mathbf{x}_i\mathbf{x}_i^\intercal(\mathbf{X}\mathbf{X}^\intercal)^{+/2}\| = \mathbf{x}_i^\intercal(\mathbf{X}\mathbf{X}^\intercal)^{+}\mathbf{x}_i. \tag{2.6}$$

Furthermore, the sum of the leverage scores

$$\sum_{i=1}^{n} \tau_i = \mathrm{Tr}(\mathbf{\Pi}) = \mathrm{Tr}\left(\mathbf{X}\mathbf{X}^\intercal(\mathbf{X}\mathbf{X}^\intercal)^{+}\right) = \mathrm{Rank}(\mathbf{X}\mathbf{X}^\intercal) = r,$$

is equal to the rank of $\mathbf{X}\mathbf{X}^\intercal$.

The leverage score τ_i intuitively captures how orthogonal $\mathbf{x}_i$ is w.r.t. the rest of the samples $\mathbf{X}$, and serves to rigorously measure what we so far called redundancy w.r.t. to preserving the spectrum of $\mathbf{X}\mathbf{X}^\intercal$. LSs have the following properties:

- LSs are always smaller than 1. The maximum LS for sample $\mathbf{x}_i$ is achieved when $\mathbf{x}_i$ is orthogonal to all other samples: then $\mathbf{x}_i^\intercal(\mathbf{X}\mathbf{X}^\intercal)^{+}\mathbf{x}_i = \mathbf{x}_i^\intercal(\mathbf{x}_i\mathbf{x}_i^\intercal)^{+}\mathbf{x}_i = 1$.
- LSs are always larger than 0. The minimum LS is achieved when $\mathbf{x}_i$ is identical to all other samples (all samples are equal): then $\mathbf{x}_i^\intercal(\mathbf{X}\mathbf{X}^\intercal)^{+}\mathbf{x}_i = \mathbf{x}_i^\intercal(n\mathbf{x}_i\mathbf{x}_i^\intercal)^{+}\mathbf{x}_i = 1/n$.
- The sum of the leverage scores captures the overall orthogonal directions contained in the data, making it equal to the rank of $\mathbf{X}$.
- The coherence of the matrix $\mathbf{X}$ is equal to the maximum LS ($\max_i \tau_i$). Therefore, the low coherence assumption commonly used when applying uniform sampling is saying that no leverage score should be higher than the other, or alternatively that the leverage scores should be uniform.
- Computing the LSs is computationally expensive, requiring nd^2 operations to construct the $\mathbf{X}\mathbf{X}^\intercal$ matrix. For simplicity we assume that an efficient oracle can compute them for us. In practice there are many efficient algorithms (Cohen et al., 2015a; Drineas et al., 2012) to approximate them in $\mathcal{O}(nd\log(d))$ or $\mathcal{O}(\mathrm{nnz}(\mathbf{X})\log(d) + d^3\log(d))$ time.

Finally, note that the LS of sample $\mathbf{x}_i$ is equal to the squared norm $\|\boldsymbol{\pi}_i\|^2$ of the corresponding atom in $\mathbf{\Pi}$. Intuitively, this means that samples with larger LS contribute more to the spectrum of $\mathbf{\Pi}$. Since our goal is to reconstruct $\mathbf{\Pi}$, it seems therefore reasonable to use LS as *data adaptive* probabilities p_i in Algorithm 1 instead of the *data oblivious* probabilities

used in uniform sampling. The following proposition help us quantify how this help us in approximating $\mathbf{\Pi}$.

Proposition 2.5 — (Alaoui and Mahoney, 2015, App. A, Lem. 1). Consider an arbitrary matrix $\mathbf{A} \in \mathbb{R}^{d\times n}$ with $\mathbf{a}_i$ as its i-th column and assume $\max_{i=1,\dots,n} \|\mathbf{a}_i\|^2 \leq 1$. Let $\mathcal{I}$ be the dictionary randomly generated by Algorithm 1 using probabilities $p_i \geq \|\mathbf{a}_i\|^2$. For any $t > 0$ we have

$$\mathbb{P}(\|\mathbf{A}\mathbf{A}^\intercal - \mathbf{A}\mathbf{S}\mathbf{S}^\intercal\mathbf{A}^\intercal\| \geq \varepsilon \leq 2d \exp\left(-\frac{\overline{q}\varepsilon^2/2}{(\|\mathbf{A}\mathbf{A}^\intercal\| + \varepsilon/3)}\right) \tag{2.7}$$

Proof. The proof remains substantially the same as (Alaoui and Mahoney, 2015, Thm.2) up to two differences. (1) We consider the spectral norm $\|\cdot\|$ rather than the largest eigenvalue since in general $\mathbf{A}\mathbf{A}^\intercal - \mathbf{A}\mathbf{S}\mathbf{S}^\intercal\mathbf{A}^\intercal$ is not guaranteed to be PSD, and (2) we use our "Bernoulli with replacement" rather than Multinomial with replacement. As a sketch, the proof uses the same approach as Proposition 2.3, applying Proposition 2.4 to $\mathbf{\Pi}$, but this time using $p_i \geq \|\mathbf{a}_i\|^2$ instead of $p_i = 1/n$. This lets us develop a tighter bound

$$R := \frac{1}{\overline{q}} \max_i \frac{\|\mathbf{a}_i\|^2}{p_i} = \frac{1}{\overline{q}} \max_i \frac{p_i}{p_i} = \frac{1}{\overline{q}}, \qquad \sigma^2 := \frac{\|\mathbf{A}\mathbf{A}^\intercal\|}{\overline{q}} \max_i \frac{\|\mathbf{a}_i\|^2}{p_i} \leq \frac{\|\mathbf{A}\mathbf{A}^\intercal\|}{\overline{q}},$$

Intuitively, what this proposition tells us is that to preserve in spectral norm the sum of rank-1 matrices $\mathbf{A}\mathbf{A}^\intercal = \sum_{i=1}^n \mathbf{a}_i\mathbf{a}_i^\intercal$, we want to sample more often matrices with larger norm. Under the hood of the proof, this is because in the random sum $\mathbf{A}\mathbf{S}\mathbf{S}^\intercal\mathbf{A}^\intercal = \sum_{i=1}^n s_i\mathbf{a}_i\mathbf{a}_i^\intercal$ rank-1 matrices with larger norm cause larger variance, and sampling them more often help us to reduce the variance and concentrate around the expectation. ■

Noting that the LS are the norm of the columns of $\mathbf{\Pi}$, we can instantly apply Proposition 2.5 to obtain

Corollary 2.6 — Proposition 2.5, Proposition 2.13. Consider an arbitrary matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ with $\mathbf{x}_i$ as its i-th column and rank r. Let $\mathbf{\Pi}$ and $\widetilde{\mathbf{\Pi}}$ be defined according to Definition 2.1, and the LS τ_i according to Definition 2.4. Let $\mathcal{I}$ be the dictionary generated by Algorithm 1 run with parameters

$$p_i = \tau_i, \qquad \overline{q} \geq \frac{4\log(4r/\delta)}{\varepsilon^2},$$

and $\sum_{i=1}^n p_i = r$. Then for any $0 < \varepsilon < 1$ and w.p.$1-\delta$

Accuracy: $\mathcal{I}$ is ε-accurate.

Space: the size of the dictionary is bounded by $|\mathcal{I}| \leq 3\overline{q}r$

Proof of Corollary 2.6. The proof follows directly from Proposition 2.5. To replace the dependency on d with the dependency on r, it is sufficient to decompose $\mathbf{\Pi}$ before applying the concentration using the SVD decomposition

$$\begin{aligned}\|\mathbf{\Pi} - \widetilde{\mathbf{\Pi}}\| &= \|(\mathbf{X}\mathbf{X}^\intercal)^{+/2}\mathbf{X}(\mathbf{I}_n - \mathbf{S}\mathbf{S}^\intercal)\mathbf{X}^\intercal(\mathbf{X}\mathbf{X}^\intercal)^{+/2}\| \\ &= \|(\mathbf{\Sigma}\mathbf{\Sigma}^\intercal)^{+/2}\mathbf{\Sigma}\mathbf{U}^\intercal(\mathbf{I}_n - \mathbf{S}\mathbf{S}^\intercal)\mathbf{U}\mathbf{\Sigma}^\intercal(\mathbf{\Sigma}\mathbf{\Sigma}^\intercal)^{+/2}\|,\end{aligned}$$

where $\mathbf{\Sigma}\mathbf{\Sigma}^\intercal \in \mathbb{R}^{r\times r}$. This leaves unchanged the norms $\|\mathbf{a}_i\|^2$ and all other quantities. ■

Algorithm 2 Nyström method in $\mathcal{H}$

Input: Budget $\overline{q}$, set of probabilities $\{p_i\}_{i=1}^n$ such that for all i $0 < p_i \leq 1$
Output: $\mathcal{I}$

1: Initialize $\mathcal{I} = \emptyset$
2: **for** $i = \{1, \dots, n\}$ **do**
3: Draw the binomial r.v. $q_i \sim \mathcal{B}(p_i, \overline{q})$
4: If $q_t \neq 0$, add $\left(\frac{1}{p_i}\frac{q_i}{\overline{q}}, \phi_i = \varphi(\mathbf{x}_i)\right)$ to $\mathcal{I}$
5: **end for**

Note that this result was first derived by Drineas et al., (2008). With this corollary we have answered the question on which and how many atoms we need to sample to guarantee an accurate reconstruction of . We have finally satisfied the two goals of this section. For example, setting $\overline{q} = \mathcal{O}(\log(r))$ gives us w.h.p. an ε-accurate dictionary that can reconstruct $\mathbf{\Pi}$ and $\mathbf{X}\mathbf{X}^\intercal$, with a total size smaller than $\mathcal{O}(r\log(r))$ that only scales with the actual rank of the problem, and independent from n. With this dictionary, we can solve downstream tasks in a time independent from n, making it feasible to scale to problems on very large datasets.
Compared to uniform sampling, we do not need to assume a low coherence or select a large $\overline{q}$ to guarantee accuracy, because we will be actively looking for those orthogonal samples that made uniform sampling fail. In other words, our p_i now capture much more closely our true underlying importance, and we can use a small logarithmic $\overline{q}$ because we only need to compensate the intrinsic randomness of the sampling process and not wrong probabilities. Moreover, the size of the dictionary automatically scales as $\overline{q}r$, adapting to the rank and difficulty of the problem.

Nonetheless the result is still not fully satisfactory. In particular, we have no way to automatically estimate r, so we typically resort to setting $\overline{q} = \mathcal{O}(\log(d))$. In addition, two obvious shortcomings emerge from looking at the LS sampling approach

Fast-decaying eigenvalues. Sometimes the rank of the matrix r is large and close to d, but only due to small numerical fluctuations. Or alternatively the smaller eigenvalues of $\mathbf{X}\mathbf{X}^\intercal$ are negligible compared to the larger and we would like to ignore them.

Large n, larger d. In many cases of interest d itself is also very large, and potentially larger than n. This is commonly the case when we project our point in some high dimensional space using tile-coding, wavelets, one-hot encodings or other combinatorial features. While these higher dimensional spaces have room for richer dictionaries, using LS to measure importance becomes vacuous, since it becomes possible for $\mathbf{X}$ to have n orthogonal vectors with $\tau_i = 1$ and Algorithm 1 would simply keep all elements.

In both of these cases, not only it is impossible to exactly reconstruct $\mathbf{X}\mathbf{X}^\intercal$ using a dictionary smaller than $n \leq d$, even ε approximation becomes impossible without storing all n input samples in the dictionary. In the next section, we will see how to further relax the approximation condition to tackle this setting, and how to generalize the LS to Regularized Leverage Scores.

2.2 Nyström in a RKHS: ridge leverage score sampling

To move from $\mathbb{R}^d$ to a Reproducing Kernel Hilbert Space $\mathcal{H}$ we must introduce some additional notation. First we do not restrict our samples $\mathbf{x}_i$ to come from $\mathbb{R}^d$ anymore,

but make no assumption on the input space $\mathcal{X}$. Then, given our dataset $\mathcal{D}$ of n arbitrarily chosen points[5] $\mathbf{x}_i \in \mathcal{X}$ we use a feature map $\phi(\cdot) : \mathcal{X} \to \mathcal{H}$ to map the samples into the RKHS as $\boldsymbol{\phi}_i = \varphi(\mathbf{x}_i)$. In the remainder of the thesis, the points $\boldsymbol{\phi}_i$ as will play the roles of points in a RKHS, samples from a dataset, columns of a map, or atoms in a dictionary, and we will refer to them accordingly depending on the context.
To simplify the notation, we indicate with $\boldsymbol{\phi}_i^\intercal \boldsymbol{\phi}_j$ the inner product $\langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_j \rangle_{\mathcal{H}}$ in $\mathcal{H}$. Also for simplicity and without loss of generality we assume that for all $\varphi(\mathbf{x}_i)$ the feature map φ induces a unit norm $\boldsymbol{\phi}_i^\intercal \boldsymbol{\phi}_i = \|\boldsymbol{\phi}_i\|^2 = 1$. This is commonly satisfied in practice, for example by using Gaussian or cosine similarities for our $\mathcal{H}$, or by normalizing the inputs $\mathbf{x}_i$ in the $\mathbb{R}^d$ case. Alternatively, all our results can be extended to non-unit norm, with extra multiplicative terms of the form $(\max_i \|\boldsymbol{\phi}_i\|^2)/(\min_i \|\boldsymbol{\phi}_i\|^2)$ appearing in space and time complexities. When evaluating computational costs, we also assume that computing the inner product $\boldsymbol{\phi}_i^\intercal \boldsymbol{\phi}_j$ takes a time independent from n (i.e., constant).
We also replace the matrix $\mathbf{X}$ with the map $\Phi : \mathbb{R}^n \to \mathcal{H}$ composed by stacking together all points $\boldsymbol{\phi}_i$. To simplify again the notation, we indicate with $\Phi^\intercal : \mathcal{H} \to \mathbb{R}^n$ its adjoint map. It is easy to see that the self-adjoint operator $\Phi\Phi^\intercal : \mathcal{H} \to \mathcal{H}$ that replaces the covariance matrix $\mathbf{X}\mathbf{X}^\intercal$ remains PSD and it can again be expressed as $\Phi\Phi^\intercal = \sum_{i=1}^n \boldsymbol{\phi}_i \boldsymbol{\phi}_i^\intercal$. The thin SVD of $\Phi = \mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^\intercal$ must also be appropriately redefined. Let r be the rank of Φ, we have that

1. The projection matrix becomes a projection operator on $\mathrm{Im}(\Phi)$

$$\boldsymbol{\Pi} = (\Phi\Phi^\intercal)^{+/2}\Phi\Phi^\intercal(\Phi\Phi^\intercal)^{+/2} = \mathbf{V}\mathbf{V}^\intercal$$

   based on the map $\mathbf{V} : \mathbb{R}^r \to \mathcal{H}$, where $\mathbf{v}_i$ are the orthonormal eigenfunctions that span $\mathrm{Im}(\Phi)$, such that $\mathbf{V}^\intercal\mathbf{V} = \mathbf{I}_r$. Note that $\boldsymbol{\Pi}$ also acts as the identity for all vectors in Φ ($\Phi = \boldsymbol{\Pi}\Phi$)
2. $\boldsymbol{\Sigma} : \mathbb{R}^{r\times r}$ remains the diagonal matrix where $[\boldsymbol{\Sigma}]_{i,i} = \sigma_i$ is the i-th singular value, and $\sigma_r \geq \sigma_{r-1} \geq \cdots \geq \sigma_1 > 0$
3. $\mathbf{U} : \mathbb{R}^{n\times r}$ remain the orthonormal vectors that span $\mathrm{Im}(\Phi^\intercal)$

Compared to the simpler leverage score sampling case in $\mathbb{R}^d$, we have two new difficulties. First, we cannot store or manipulate directly the vectors $\boldsymbol{\phi}_i$, but only access them through inner products. Therefore it is not clear how to compute the LS $\boldsymbol{\phi}^\intercal(\Phi\Phi^\intercal)^+\boldsymbol{\phi}$.
Second, when we choose our RKHS we prefer those with large dimension, because we want a rich space that can contain good solutions for our downstream task problems. As a consequence, it is very easy to end up constructing an $\mathcal{H}$ where $\mathrm{Rank}(\Phi) = n$. For example, using the Gaussian similarity the only way for the rank of the matrix to be smaller than the number of points is for some of the points to be exactly identical to each other.
When the rank of the matrix is n, even if we could compute $\boldsymbol{\phi}^\intercal(\Phi\Phi^\intercal)^+\boldsymbol{\phi}$ we would end up with all LS equal to 1, and the sampling would be meaningless.
Nonetheless, in many cases the eigenvalues of $\Phi\Phi$ decay very fast, or in other words only a few of the many (infinite) directions in $\mathcal{H}$ capture most of the information contained in Φ.

2.2.1 Regularized subspace approximation

We can encode this intuition using a *regularized* projection, and a new (ε, γ)-accuracy definition.

[5] i.e., we do not assume the points are sampled from a distribution, but since they are part of a dataset we still refer to them as samples.

Definition 2.5 The γ-regularized projection $\mathbf{\Gamma}$ on $\mathrm{Im}(\Phi)$ is defined as

$$\mathbf{\Gamma} = (\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1/2}\Phi\Phi^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1/2} = \sum_{i=1}^{n} \psi_i\psi_i^\intercal, \tag{2.8}$$

with $\psi_i = (\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1/2}\phi_i$. The approximate γ-regularized projection $\widetilde{\mathbf{\Gamma}}$ for a dictionary $\mathcal{I}$ and its selection matrix $\mathbf{S}$ is defined as

$$\widetilde{\mathbf{\Gamma}} = (\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1/2}\Phi\mathbf{S}\mathbf{S}^\intercal\Phi^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1/2} = \sum_{i=1}^{n} s_i\psi_i\psi_i^\intercal. \tag{2.9}$$

Definition 2.6 A dictionary $\mathcal{I} = \{(s_i, \phi_i)\}_{i=1}^n$ and its associated selection matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$ are (ε, γ)-accurate w.r.t. a dataset Φ if

$$\|\mathbf{\Gamma} - \widetilde{\mathbf{\Gamma}}\| = \|(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1/2}\Phi(\mathbf{I}_n - \mathbf{S}\mathbf{S}^\intercal)\Phi^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1/2}\| \leq \varepsilon \tag{2.10}$$

Associated with Definition 2.6 we have another reconstruction result

Lemma 2.7

$$\|(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1/2}(\Phi\Phi^\intercal - \Phi\mathbf{S}\mathbf{S}^\intercal\Phi^\intercal)(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1/2}\| \leq \varepsilon$$
$$\Updownarrow$$
$$(1-\varepsilon)\Phi\Phi^\intercal - \varepsilon\gamma\mathbf{\Pi} \preceq \Phi\mathbf{S}\mathbf{S}^\intercal\Phi^\intercal \preceq (1+\varepsilon)\Phi\Phi^\intercal + \varepsilon\gamma\mathbf{\Pi} \tag{2.11}$$

Looking at the equation, we see that an (ε, γ)-accurate dictionary gives us the same $(1 \pm \varepsilon)$ multiplicative error guarantees of an ε-accurate dictionary, but also adds a $\varepsilon\gamma$ *additive* error. This means for example that only eigenvalues larger than $\varepsilon\gamma$ will be accurately reconstructed by the dictionary, and that the equivalence $\mathrm{Ker}(\Phi\Phi^\intercal) = \mathrm{Ker}(\Phi\mathbf{S}\mathbf{S}^\intercal\Phi)$ does not hold. Nonetheless, in many interesting problems (e.g., PCA, regularized regression) we were not interested in approximating the directions associated with smaller eigenvalues anyway, and as we will see in Chapter 4 and Chapter 5 this extra small error does influence too much the quality of the solutions computed on the dictionary. In exchange for this new mixed multiplicative-additive error, we can use $\mathbf{\Gamma}$ to define a generalization of the LS to this new $d \gg n$ setting

Definition 2.7 — (Alaoui and Mahoney, 2015). Given a map $\Phi : \mathbb{R}^n \to \mathcal{H}$, the γ-ridge leverage score (RLS) of column $i \in [n]$ is

$$\tau_i(\gamma) = \phi_i^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i = \sum_{j=1}^{n} \frac{\sigma_j^2}{\sigma_j^2 + \gamma}[\mathbf{U}]_{i,j}. \tag{2.12}$$

Furthermore, the effective dimension $d_{\mathrm{eff}}(\gamma)$ of the operator is defined as

$$d_{\mathrm{eff}}(\gamma) = \sum_{i=1}^{n} \tau_i(\gamma) = \mathrm{Tr}\left(\Phi\Phi^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\right) = \sum_{i=1}^{n} \frac{\sigma_i^2}{\sigma_i^2 + \gamma}. \tag{2.13}$$

Ridge leverage scores

The RLSs $\tau_i(\gamma)$ generalize the way LSs captures the overall orthogonality of a sample in the dataset, and LSs can be seen as a special case $\tau_i(0)$. In particular, the directions

associated with smaller eigenvalues are non-linearly penalized by the regularization γ, and being orthogonal along those direction will not make the sample important. Since the value of γ will be often clear from the context, we will refer to both LSs and RLSs as τ_i and only include the full notation $\tau_i(\gamma)$ when important.
We can now look at two opposite cases to get an idea of how RLSs behave. When all samples are identical, the RLS of ϕ_i is at its minimum

$$0 \leq \phi_i^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i = \phi_i^\intercal(n\phi\phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i = \frac{\phi_i^\intercal\phi_i}{n\phi_i^\intercal\phi_i + \gamma} = \frac{1}{n+\gamma},$$

even lower than the corresponding lowest LS. When ϕ_i is orthogonal to all other samples its RLS is at its maximum

$$\phi_i^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i = \phi_i^\intercal(\phi_i\phi_i^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i = \frac{\phi_i^\intercal\phi_i}{\phi_i^\intercal\phi_i + \gamma} = \frac{1}{1+\gamma} \leq 1.$$

But when we have many near-orthogonal samples the difference between LSs and RLSs emerge. While all the LSs would be very close to 1 (uniform sampling), looking at Eq. 2.12 we see that as γ grows, the RLSs are non-uniformly shrunk. In particular samples ϕ_i correlated with the larger eigenvalues of $\Phi\Phi$ remain close to 1, while the ones correlated with smaller eigenvalues are shrunk (nonuniform sampling). For example a sample orthogonal to the others and fully aligned with the smallest non-zero eigenvalue $\lambda_{\min}$ would be shrunk by roughly a $\lambda_{\min}/\gamma$ factor.
In the linear case where $\varphi(\cdot) : \mathcal{X} \to \mathbb{R}^d$ ($\phi_i = \mathbf{x}_i$), the RLS can be explicitly computed as $\mathbf{x}_i^\intercal(\mathbf{X}\mathbf{X}^\intercal + \gamma\mathbf{I}_d)^{-1}\mathbf{x}_i$ (Cohen et al., 2015b; Cohen et al., 2016). In the general $\mathcal{H}$ setting, to compute RLS in practice we can use the following equality

Lemma 2.8 — (Calandriello et al., 2017a). The RLSs can be reformulated as

$$\phi_i^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i = \frac{1}{\gamma}\left(\phi_i^\intercal\phi_i - \phi_i^\intercal\Phi(\Phi^\intercal\Phi + \gamma\mathbf{I}_n)^{-1}\Phi^\intercal\phi_i\right) \tag{2.14}$$

While evaluating Eq. 2.14 is now possible, since $\Phi^\intercal\Phi \in \mathbb{R}^{n\times n}$ and $\Phi^\intercal\phi_i \in \mathbb{R}^n$ are real matrices and vectors, it is still to computationally expensive. Operations involving $\Phi^\intercal\Phi$ comprise the dominant part of this cost: it takes $\mathcal{O}(n^2)$ space and operations to construct and store it, $\mathcal{O}(n^3)$ operations to invert $(\Phi^\intercal\Phi + \gamma\mathbf{I}_n)^{-1}$, and $\mathcal{O}(n^2)$ operations to compute each $\phi_i^\intercal\Phi(\Phi^\intercal\Phi + \gamma\mathbf{I}_n)^{-1}\Phi^\intercal\phi_i$ using the precomputed inverse. For now, we will assume that an efficient oracle can compute $\tau_i(\gamma)$ for us, and we postpone the implementation of such an oracle to Chapter 3.

Ridge leverage scores are also known under a number of different names in Statistics, often appearing together with the Gaussian distribution. Consider the problem of Gaussian process regression (Rasmussen and Williams, 2005), where our function is drawn from a $f \sim GP(0, \varphi(\mathbf{x})^\intercal\varphi(\mathbf{x}'))$ with mean function 0 and covariance function induced by $\varphi(\cdot)$. The target variables are generated as $y_i = f(\mathbf{x}_i) + \epsilon_i$, with ϵ_i drawn i.i.d. from $\mathcal{N}(0,1)$. Then, after drawing n samples the posterior variance of a new point $\mathbf{x}_*$ w.r.t. the sample is

$$\sigma_n(\mathbf{x}_*) = \phi_*^\intercal\phi_* - \phi_*\Phi(\Phi^\intercal\Phi + \mathbf{I}_n)^{-1}\Phi^\intercal\phi_*.$$

This is almost equivalent to computing the RLS of ϕ_* w.r.t. Φ using the reformulation from Lemma 2.8. The only difference is that ϕ_* is not included in the inverse, or in other words that we compute the RLS w.r.t. Φ rather than w.r.t. $[\Phi, \phi_*]$ as we did so far.
Taking this interpretation, Srinivas et al., (2010) shows that for Gaussian processes, points that maximize RLSs (large variance) also approximately maximize the *information gain* $I(\{y_i\}|f)$ (Cover and Thomas, 2012).

Effective dimension

We saw that, compared to LSs, the RLSs shrink depending on the overall spectrum of Φ. This is also reflected in the sum of the RLSs: although the LSs sum to the *ambient* dimensionality (and rank) of Φ in $\mathcal{H}$, which might be large, the RLSs sum to its *effective* dimensionality, which is potentially much smaller.
Intuitively $d_{\text{eff}}(\gamma)$ scales with the number of *relevant* directions $\Phi\Phi^\intercal$, where a direction is relevant if it is associated with an eigenvalue larger than the regularization γ. We can also see $d_{\text{eff}}(\gamma)$ as the rank of $\Phi\Phi^\intercal$ after *thresholding* all eigenvalues smaller than γ, and it is easy to see that $d_{\text{eff}}(\gamma) < r$ for any $\gamma > 0$, and $d_{\text{eff}}(0) = r$. The effective dimension is also monotone in γ, which means we can always make it smaller by increasing the regularization.
While in general $d_{\text{eff}}(\gamma)$ can greatly vary depending on the specific $\Phi\Phi^\intercal$, if we know something on the structure of the operator (e.g., because of the way ϕ_i are generated, or some property of φ) we can still find bounds on it. Note that the partial derivative of $d_{\text{eff}}(\gamma)$ w.r.t. the values λ_i is

$$\frac{\partial}{\partial \lambda_i} \sum_{j=1}^{n} \frac{\lambda_j}{\lambda_j + \gamma} = \frac{\partial}{\partial \lambda_i} \sum_{j=1}^{n} 1 - \frac{\gamma}{\lambda_j + \gamma} = \frac{\gamma}{(\lambda_i + \gamma)^2},$$

with the same partial derivative for all λ_i and the maximum increase in $\lambda_i = 0$. Therefore, when the trace $\mathrm{Tr}(\Phi\Phi^\intercal) = \sum_{i=1}^{n} \lambda_i$ is bounded (e.g., $\mathrm{Tr}(\Phi\Phi^\intercal) = n$ as in our case) the maximum $d_{\text{eff}}(\gamma)$ is achieved when all λ_i are equal, which corresponds to the case where all samples are orthogonal to each other. Conversely, the minimum is achieved when all samples are equal and only a single λ_i is non-zero, resulting in $d_{\text{eff}}(\gamma) = \frac{n\phi_i^\intercal \phi_i}{n\phi_i^\intercal \phi_i + \gamma} \sim 1$.
In general, knowing that the operator has a bounded trace $\mathrm{Tr}(\Phi\Phi^\intercal) = n$ gives us a bound $d_{\text{eff}}(\gamma) \leq n/\gamma$ which scales as $\mathcal{O}(n)$ when γ is a constant, but a smaller $\mathcal{O}(\sqrt{n})$ when $\gamma = \sqrt{n}$. Stronger bounds are possible when we can control each individual eigenvalue. For example if the eigenvalues decay polynomially $\lambda_i = i^{-1}$ then $d_{\text{eff}}(\gamma) = \sum_{i=1}^{n} \frac{i^{-1}}{i^{-1}+\gamma} = \sum_{i=1}^{n} \frac{1}{1+\gamma i} \leq \log(n + \min\{1, \gamma^{-1}\})/\gamma$, while clearly the rank is much larger and equal to n.

Together with the RLSs, the effective dimension appears in many statistical problems. If we assume that the points are drawn according to a distribution $\mu(\mathbf{x})$ rather than arbitrarily, then the effective dimension is the empirical equivalent of the *capacity* of the RKHS $\mathcal{H}$ under μ (Caponnetto and De Vito, 2005).
One of the contributions of this thesis, reported in Section 3.2, will be showing that the log-determinant $\log(\mathrm{Det}(\Phi^\intercal\Phi/\gamma + \mathbf{I}_n))$ of the covariance matrix can be upper bounded using its effective dimension. This log-determinant appears, among other topics, as the sampling function in determinantal point processes (Kulesza and Taskar, 2012), as an upper bound for the total information gain $I(\{y_i\}|f)$ in Gaussian process regression (Srinivas et al., 2010), and as an upper bound for the regret of online learning algorithms (Hazan et al., 2006).

2.2.2 Oracle ridge leverage score sampling

Using RLSs as probabilities in Algorithm 2, we can construct an (ε, γ)-accurate dictionary w.h.p.. We refer to this algorithm as Oracle-RLS.

Corollary 2.9 — Lemma 2.10, Proposition 2.5, Proposition 2.13. Consider an arbitrary map $\Phi : \mathbb{R}^n \to \mathcal{H}$ with ϕ_i as its i-th column and rank r. Given $\gamma > 0$, let $\mathbf{\Gamma}$ and $\widetilde{\mathbf{\Gamma}}$ be defined according to Definition 2.5, and the RLS τ_i according to Definition 2.7. Let $\mathcal{I}$ be the

dictionary generated by Algorithm 2 run with parameters

$$p_i = \tau_i(\gamma), \qquad \overline{q} \geq \frac{4\log(4r/\delta)}{\varepsilon^2},$$

and $\sum_{i=1}^n p_i = d_{\text{eff}}(\gamma)$, i.e., ORACLE-RLS. Then for any $0 < \varepsilon < 1$ and w.p.$1-\delta$

Accuracy: $\mathcal{I}$ is (ε, γ)-accurate.

Space: the size of the dictionary is bounded by $|\mathcal{I}| \leq 3\overline{q}d_{\text{eff}}(\gamma)$

Differently from Corollary 2.6 in the previous section, Corollary 2.9 cannot always guarantee a constant (in n) error and at the same time guarantee a constant (again in n) dictionary size. If we set $\varphi(\cdot)$ to be the identity map and $\mathcal{X} = \mathbb{R}^d$, we can compare better the two settings.
When $n \gg d$, we know that $d_{\text{eff}}(\gamma) \leq r \leq d$ no matter the γ, and therefore we can always find an ε-accurate dictionary that does not grow with n. But in the $d \gg n$ setting (or in a general $\mathcal{H}$) the regularization γ drives the trade-off between dictionary size and accuracy. Going back to the bounded trace $\text{Tr}(\Phi\Phi^\intercal) = n$, in the worst case (all orthogonal samples and diagonal $\Phi^\intercal\Phi$) we can only trade-off a n/γ dictionary size for a γ approximation error. Nonetheless, we still achieved the goal that we set at the beginning of the section: when the spectrum of $\Phi\Phi^\intercal$ is not uniform, RLSs sampling will automatically adapt to the difficulty of the problem and construct $\widetilde{\mathcal{O}}(d_{\text{eff}}(\gamma))$ sized dictionaries that are not only much smaller than d or n, but also of the numerically high rank r.

Finally, note that for simplicity we chose our bounds to scale with $\overline{q} \geq \mathcal{O}(\log(r))$, in order to continue using the simple inequalities of Proposition 2.4. Using more refined concentrations based on the stable rank rather than the rank it is possible to obtain tighter bounds that scale with $\overline{q} \geq \mathcal{O}(\log(d_{\text{eff}}(\gamma)))$. Unfortunately, estimating both n and $d_{\text{eff}}(\gamma)$ is computationally expensive, and in practice $\overline{q}$ is often set proportionally to $\log(n)$, since scaling logarithmically w.r.t. to n is quite mild and still acceptable.

Proof of Corollary 2.9. For the proof we cannot directly apply Proposition 2.5, since $\mathbf{\Gamma}$ is an operator and not a matrix and Proposition 2.4 cannot be applied. Nonetheless, we can define a proxy for the regularized projection $\mathbf{\Gamma}$ that is a matrix.

Definition 2.8 The γ-regularized (approximate) projection $\mathbf{P} \in \mathbb{R}^{n\times n}$ on $\text{Im}(\Phi^\intercal)$ is defined as

$$\mathbf{P} = \Phi^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\Phi = \mathbf{C}\mathbf{C}^\intercal, \qquad \widetilde{\mathbf{P}} = \mathbf{C}\mathbf{S}\mathbf{S}^\intercal\mathbf{C}^\intercal. \tag{2.15}$$

with

$$\mathbf{C} = \mathbf{U}(\mathbf{\Sigma}\mathbf{\Sigma}^\intercal + \gamma\mathbf{I}_r)^{-1/2}\mathbf{\Sigma}\mathbf{U}^\intercal, \qquad \mathbf{c}_i = \mathbf{U}(\mathbf{\Sigma}\mathbf{\Sigma}^\intercal + \gamma\mathbf{I}_r)^{-1/2}\mathbf{\Sigma}\mathbf{U}^\intercal\mathbf{e}_i. \tag{2.16}$$

Among other important things, we can use this definition as an alternative formulation for RLSs

$$\begin{aligned}\tau_i &= \|\psi_i\|^2 = \psi_i^\intercal\psi_i = \phi_i^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i = \mathbf{e}_i^\intercal\Phi^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\Phi\mathbf{e}_i \\ &= \|\mathbf{c}_i\|^2 = \mathbf{c}_i^\intercal\mathbf{c}_i = \mathbf{e}_i^\intercal\mathbf{C}^\intercal\mathbf{C}\mathbf{e}_i = \mathbf{e}_i^\intercal\mathbf{U}\mathbf{\Sigma}^\intercal(\mathbf{\Sigma}\mathbf{\Sigma}^\intercal + \gamma\mathbf{I}_r)^{-1/2}\mathbf{U}^\intercal\mathbf{U}(\mathbf{\Sigma}\mathbf{\Sigma}^\intercal + \gamma\mathbf{I}_r)^{-1/2}\mathbf{\Sigma}\mathbf{U}^\intercal\mathbf{e}_i,\end{aligned}$$

and to reformulate the definition of (ε, γ)-accurate dictionaries

Lemma 2.10 — (Calandriello et al., 2017a). Let $\mathbf{\Gamma}, \widetilde{\mathbf{\Gamma}}$ be defined according to Definition 2.5, and $\mathbf{P}, \widetilde{\mathbf{P}}$ according to Definition 2.8. Then

$$\begin{aligned}\|\mathbf{\Gamma} - \widetilde{\mathbf{\Gamma}}\| &= \|(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1/2}\Phi(\mathbf{I}_n - \mathbf{S}\mathbf{S}^\intercal)\Phi^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1/2}\| \\ &= \|\mathbf{U}(\mathbf{\Sigma}\mathbf{\Sigma}^\intercal + \gamma\mathbf{I}_r)^{-1/2}\mathbf{\Sigma}\mathbf{U}^\intercal(\mathbf{I}_n - \mathbf{S}\mathbf{S}^\intercal)\mathbf{U}\mathbf{\Sigma}^\intercal(\mathbf{\Sigma}\mathbf{\Sigma}^\intercal + \gamma\mathbf{I}_r)^{-1/2}\mathbf{U}^\intercal\| = \|\mathbf{P} - \widetilde{\mathbf{P}}\|.\end{aligned}$$

Adding also Proposition 2.2 we conclude the proof. ■

In the next chapter we address the computational cost of computing RLSs, in order to remove the necessity of an oracle and derive sampling algorithms that can be implemented in practice.

2.3 Extended proofs

We report some linear algebra identities that will be useful through the thesis.

Proposition 2.11 — (Horn and Johnson, 2012). For any matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ and any $\gamma > 0$

$$\mathbf{A}(\mathbf{A}^\intercal\mathbf{A} + \gamma\mathbf{I}_m)^{-1}\mathbf{A}^\intercal = \mathbf{A}\mathbf{A}^\intercal(\mathbf{A}\mathbf{A}^\intercal + \gamma\mathbf{I}_n)^{-1}.$$

For any symmetric matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ and diagonal matrix $\mathbf{B} \in \mathbb{R}^{n\times n}$ such that $\mathbf{B}$ has $n - s$ zero entries, and s non-zero entries, define $\mathbf{C} \in \mathbb{R}^{n\times s}$ as the matrix obtained by removing all zero columns in $\mathbf{B}$. Then

$$\mathbf{A}\mathbf{B}(\mathbf{B}\mathbf{A}\mathbf{B} + \gamma\mathbf{I}_m)^{-1}\mathbf{B}\mathbf{A} = \mathbf{A}\mathbf{C}(\mathbf{C}^\intercal\mathbf{A}\mathbf{C} + \gamma\mathbf{I}_s)^{-1}\mathbf{C}^\intercal\mathbf{A}.$$

For any appropriately shaped matrix $\mathbf{A}, \mathbf{B}, \mathbf{C}$, with $\mathbf{A}$ and $\mathbf{B}$ invertible, the Woodbury matrix identity states

$$(\mathbf{A} + \mathbf{C}\mathbf{B}\mathbf{C}^\intercal)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{C}\left(\mathbf{C}^\intercal\mathbf{A}^{-1}\mathbf{C} + \mathbf{B}^{-1}\right)^{-1}\mathbf{C}^\intercal\mathbf{A}^{-1}.$$

Proposition 2.12 — (Horn and Johnson, 2012). For any map $\mathbf{A} : \mathbb{R}^n \to \mathcal{H}$, projection operator $\mathbf{\Pi}$ such that $\mathbf{A} = \mathbf{\Pi}\mathbf{A}$, and any $\gamma > 0$

$$\mathbf{A}(\mathbf{A}^\intercal\mathbf{A} + \gamma\mathbf{I}_n)^{-1}\mathbf{A}^\intercal = \mathbf{A}\mathbf{A}^\intercal(\mathbf{A}\mathbf{A}^\intercal + \gamma\mathbf{\Pi})^{-1}.$$

For any map $\mathbf{A} : \mathbb{R}^n \to \mathcal{H}$ and diagonal matrix $\mathbf{B} \in \mathbb{R}^{n\times n}$ such that $\mathbf{B}$ has $n - s$ zero entries, and s non-zero entries, define $\mathbf{C} \in \mathbb{R}^{n\times s}$ as the matrix obtained by removing all zero columns in $\mathbf{B}$. Then

$$\mathbf{A}\mathbf{B}(\mathbf{B}\mathbf{A}^\intercal\mathbf{A}\mathbf{B} + \gamma\mathbf{I}_n)^{-1}\mathbf{B}\mathbf{A}^\intercal = \mathbf{A}\mathbf{C}(\mathbf{C}^\intercal\mathbf{A}\mathbf{C} + \gamma\mathbf{I}_s)^{-1}\mathbf{C}^\intercal\mathbf{A}.$$

For the proof of Proposition 2.2 we need a preliminary result

Proposition 2.13 — (Calandriello et al., 2017a). Let $\{z_s\}_{s=1}^t$ be independent Bernoulli random variables, each with success probability p_s, and denote their sum as $d =$

$\sum_{s=1}^{t} p_s \geq 1$. Then,

$$\mathbb{P}\left(\sum_{s=1}^{t} z_s \geq gd\right) \leq e^{-d} \exp\left\{-g\left(\log\left(g\right)-1\right)-1\right\}$$

Proof of Proposition 2.13. We can bound the sum $\sum_{s=1}^{t} z_s$ with a Hoeffding-like bound using Markov inequality,

$$\begin{aligned}
\mathbb{P}\left(\sum_{s=1}^{t} z_s \geq gd\right) &= \inf_{\theta>0} \mathbb{P}\left(e^{\sum_{s=1}^{t}\theta z_s} \geq e^{\theta g d}\right) \\
&\leq \inf_{\theta>0} \frac{\mathbb{E}\left[e^{\sum_{s=1}^{t}\theta z_s}\right]}{e^{\theta g d}} = \inf_{\theta>0} \frac{\mathbb{E}\left[\prod_{s=1}^{t} e^{\theta z_s}\right]}{e^{\theta g d}} = \inf_{\theta>0} \frac{\prod_{s=1}^{t}\mathbb{E}\left[e^{\theta z_s}\right]}{e^{\theta g d}} \\
&= \inf_{\theta>0} \frac{\prod_{s=1}^{t}(p_s e^{\theta}+(1-p_s))}{e^{\theta g d}} = \inf_{\theta>0} \frac{\prod_{s=1}^{t}(1+p_s(e^{\theta}-1))}{e^{\theta g d}} \\
&\leq \inf_{\theta>0} \frac{\prod_{s=1}^{t} e^{p_s(e^{\theta}-1)}}{e^{\theta g d}} \leq \inf_{\theta>0} \frac{e^{(e^{\theta}-1)\sum_{s=1}^{t} p_s}}{e^{\theta g d}} = \inf_{\theta>0} e^{(d(e^{\theta}-1)-\theta g d)}
\end{aligned}$$

where we use the fact that $1+x \leq e^{x}$, $z_s \sim Ber(p_s)$ and $\sum_{s=1}^{t} p_s = d$. The choice of θ minimizing the previous expression is obtained as

$$\frac{d}{d\theta} e^{\left(d(e^{\theta}-1)-\theta g d\right)} = e^{\left(d(e^{\theta}-1)-\theta g d\right)}\left(de^{\theta}-gd\right) = 0,$$

and thus $\theta = \log(g)$. Plugging this in the previous bound,

$$\begin{aligned}
\inf_{\theta} \exp\left\{d(e^{\theta}-1)-\theta g d)\right\} &= \exp\left\{gd-d-gd\log\left(g\right)\right\} \\
&= e^{-d}\exp\left\{-g\left(\log\left(g\right)-1\right)-1\right\}.
\end{aligned}$$

■

Proof of Proposition 2.13. Let $z_{i,j}$ be the j-th Bernoulli r.v. independently sampled when computing q_i, such that $q_i = \sum_{j=1}^{\overline{q}} z_{i,j}$. To bound the size $|\mathcal{I}| \leq \sum_{i=1}^{n} q_i = \sum_{i=1}^{n}\sum_{j=1}^{\overline{q}} z_{i,j}$, we simply apply Proposition 2.13 with $g=3$ and $d = \overline{q}\sum_{i=1}^{n} p_i$. ■

Proof of Corollary 2.3. We will use Proposition 2.4 from Tropp, (2015). Denote again with $z_{i,j}$ the Bernoulli random variables contained in each Binomial random variable q_i with $q_i = \sum_{j=1}^{\overline{q}} z_{i,j}$. Then we can rewrite

$$\mathbf{A}\mathbf{S}\mathbf{S}^{\mathsf{T}}\mathbf{A}^{\mathsf{T}} = \frac{1}{\overline{q}}\sum_{i=1}^{n}\sum_{j=1}^{\overline{q}}\frac{1}{p_i} z_{i,j}\mathbf{a}_i\mathbf{a}_i^{\mathsf{T}} = \frac{1}{\overline{q}}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{n}\frac{1}{p_i} z_{i,j}\mathbf{a}_i\mathbf{a}_i^{\mathsf{T}}$$

and it's easy to see that $z_{i,j} = 1$ with probability p_i and 0 otherwise. Let $\mathbf{Y} = \mathbf{A}\mathbf{A}^{\mathsf{T}} - \mathbf{A}\mathbf{S}\mathbf{S}^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}$, then

$$\mathbf{Y} = \frac{1}{\overline{q}}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{n}\left(1-\frac{z_{i,j}}{p_i}\right)\mathbf{a}_i\mathbf{a}_i^{\mathsf{T}}$$

We choose $\mathbf{X}_{i,j}$ to be $\frac{1}{\overline{q}}\left(1-\frac{z_{i,j}}{p_i}\right)\mathbf{a}_i\mathbf{a}_i^\top$ for every $i \in [n], j \in [\overline{q}]$. Now we verify the assumptions of the above theorem. The matrices $\mathbf{X}_{i,j}$ are all independent due to the independence of all $z_{i,j}$. We can see

$$\mathbb{E}[\mathbf{X}_{i,j}] = \frac{1}{\overline{q}}\left(1-\frac{\mathbb{E}[z_{i,j}]}{p_i}\right)\mathbf{a}_i\mathbf{a}_i^\top = \frac{1}{\overline{q}}\left(1-\frac{\cancel{p_i}}{\cancel{p_i}}\right)\mathbf{a}_i\mathbf{a}_i^\top = 0$$

And

$$\|\mathbf{X}_{i,j}\| \leq \frac{1}{\overline{q}}\left\|\left(1-\frac{z_{i,j}}{p_i}\right)\mathbf{a}_i\mathbf{a}_i^\top\right\| \leq \frac{1}{\overline{q}}\frac{1}{p_i}\|\mathbf{a}_i\mathbf{a}_i^\top\| = \frac{1}{\overline{q}}\frac{\|\mathbf{a}_i\|^2}{p_i} \leq \frac{1}{\overline{q}}\max_i\frac{\|\mathbf{a}_i\|^2}{p_i} = \frac{n\max_i\|\mathbf{a}_i\|^2}{\overline{q}} := R$$

Now we need to control the spectral norm of the second moment of $\mathbf{Y}$. Again with

$$\begin{aligned}\mathbb{E}[\mathbf{Y}^2] &= \sum_{(i,j)}\sum_{(i',j')}\mathbb{E}[\mathbf{X}_{i,j}\mathbf{X}_{i',j'}] = \sum_{i=1}^{n}\sum_{j=1}^{\overline{q}}\mathbb{E}[\mathbf{X}_{i,j}^2] + \sum_{(i,j)}\sum_{(i',j')\neq(i,j)}\mathbb{E}[\mathbf{X}_{i,j}\mathbf{X}_{i',j'}] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{\overline{q}}\mathbb{E}[\mathbf{X}_{i,j}^2] + \sum_{(i,j)}\sum_{(i',j')\neq(i,j)}\mathbb{E}[\mathbf{X}_{i,j}]\,\mathbb{E}[\mathbf{X}_{i',j'}] = \sum_{i=1}^{n}\sum_{j=1}^{\overline{q}}\mathbb{E}[\mathbf{X}_{i,j}^2]\end{aligned}$$

where the expectation decomposes due to the independence of $\mathbf{X}_{i,j}$ and $\mathbf{X}_{i,j'}$. Now for all i and j

$$\begin{aligned}\mathbb{E}[\mathbf{X}_{i,j}^2] &= \frac{1}{\overline{q}^2}\mathbb{E}\left[\left(1-\frac{z_{i,j}}{p_i}\right)^2\mathbf{a}_i\mathbf{a}_i^\top\mathbf{a}_i\mathbf{a}_i^\top\right] = \frac{1}{\overline{q}^2}\mathbb{E}\left[\left(1-2\frac{z_{i,j}}{p_i}+\frac{z_{i,j}^2}{p_i^2}\right)\right]\mathbf{a}_i\mathbf{a}_i^\top\mathbf{a}_i\mathbf{a}_i^\top \\ &= \frac{1}{\overline{q}^2}\left(1-2\frac{\mathbb{E}[z_{i,j}]}{p_i}+\frac{\mathbb{E}[z_{i,j}^2]}{p_i^2}\right)\mathbf{a}_i\mathbf{a}_i^\top\mathbf{a}_i\mathbf{a}_i^\top = \frac{1}{\overline{q}^2}\left(1-2\frac{\mathbb{E}[z_{i,j}]}{p_i}+\frac{\mathbb{E}[z_{i,j}]}{p_i^2}\right)\mathbf{a}_i\mathbf{a}_i^\top\mathbf{a}_i\mathbf{a}_i^\top \\ &= \frac{1}{\overline{q}^2}\left(1-2\frac{\cancel{p_i}}{\cancel{p_i}}+\frac{p_i}{p_i^2}\right)\mathbf{a}_i\mathbf{a}_i^\top\mathbf{a}_i\mathbf{a}_i^\top = \frac{1}{\overline{q}^2}\left(\frac{1}{p_i}-1\right)\mathbf{a}_i\mathbf{a}_i^\top\mathbf{a}_i\mathbf{a}_i^\top\end{aligned}$$

Where we used the fact that $\mathbb{E}[z_{i,j}] = p_i$ and that $z_{i,j}^2 = z_{i,j}$. We can now derive an upper bound

$$\begin{aligned}\|\,\mathbb{E}[\mathbf{Y}^2]\|_2 &= \left\|\sum_{i=1}^{n}\sum_{j=1}^{\overline{q}}\mathbb{E}[\mathbf{X}_{i,j}^2]\right\| = \left\|\frac{1}{\overline{q}}\sum_{i=1}^{n}\left(\frac{1}{p_i}-1\right)\|\mathbf{a}_i\|^2\mathbf{a}_i\mathbf{a}_i^\top\right\| \\ &\leq \frac{1}{\overline{q}}\left\|\sum_{i=1}^{n}\frac{\|\mathbf{a}_i\|^2}{p_i}\mathbf{a}_i\mathbf{a}_i^\top\right\| \leq \frac{1}{\overline{q}}\max_i\frac{\|\mathbf{a}_i\|^2}{p_i}\left\|\sum_{i=1}^{n}\mathbf{a}_i\mathbf{a}_i^\top\right\| \leq \frac{n\max_i\|\mathbf{a}_i\|^2}{\overline{q}}\|\mathbf{A}\mathbf{A}^\top\| := \sigma^2\end{aligned}$$

We can now apply the theorem

$$\mathbb{P}\left(\|\mathbf{Y}\| \geq \varepsilon\right) \leq d\exp\left\{-\frac{\varepsilon^2/2}{\frac{n\max_i\|\mathbf{a}_i\|^2}{\overline{q}}\|\mathbf{A}\mathbf{A}^\top\| + \frac{n\max_i\|\mathbf{a}_i\|^2\varepsilon}{3\overline{q}}}\right\}$$

Whenever $\varepsilon \leq 1$ and $\|\mathbf{A}\mathbf{A}^\top\|^2 \geq 1$ (as it is in our case) this can be simplified to

$$\mathbb{P}\left(\|\mathbf{Y}\| \geq \varepsilon\right) \leq d\exp\left\{-\frac{\varepsilon^2\overline{q}}{4n\max_i\|\mathbf{a}_i\|^2\|\mathbf{A}\mathbf{A}^\top\|^2}\right\}$$

■

Proof of Corollary 2.5. The first part of the proof is identical to the proof of Proposition 2.3. The main change is in the bounding of R and σ^2.

We begin with R, and using the definition of the probabilities $p_i \geq \|\mathbf{a}_i\|^2$ we have

$$\|\mathbf{X}_{i,j}\| \leq \frac{1}{q}\left\|\left(1-\frac{z_{i,j}}{p_i}\right)\mathbf{a}_i\mathbf{a}_i^\top\right\| \leq \frac{1}{q}\frac{1}{p_i}\|\mathbf{a}_i\mathbf{a}_i^\top\| = \frac{1}{q}\frac{\|\mathbf{a}_i\|^2}{p_i} \leq \frac{1}{q}\frac{\cancel{\|\mathbf{a}_i\|^2}}{\cancel{\|\mathbf{a}_i\|^2}} \leq \frac{1}{q} := R$$

To bound σ^2 We can now derive an upper bound

$$\begin{aligned}\|\mathbb{E}[\mathbf{Y}^2]\|_2 &= \left\|\sum_{i=1}^{n}\sum_{j=1}^{\overline{q}}\mathbb{E}[\mathbf{X}_{i,j}^2]\right\| = \left\|\frac{1}{\overline{q}}\sum_{i=1}^{n}\left(\frac{1}{p_i}-1\right)\|\mathbf{a}_i\|^2\mathbf{a}_i\mathbf{a}_i^\top\right\| \\ &\leq \frac{1}{\overline{q}}\left\|\sum_{i=1}^{n}\frac{\|\mathbf{a}_i\|^2}{p_i}\mathbf{a}_i\mathbf{a}_i^\top\right\| \leq \frac{1}{\overline{q}}\left\|\sum_{i=1}^{n}\frac{\cancel{\|\mathbf{a}_i\|^2}}{\cancel{\|\mathbf{a}_i\|^2}}\mathbf{a}_i\mathbf{a}_i^\top\right\| = \frac{1}{\overline{q}}\left\|\sum_{i=1}^{n}\mathbf{a}_i\mathbf{a}_i^\top\right\| = \frac{1}{\overline{q}}\|\mathbf{A}\mathbf{A}^\top\| := \sigma^2\end{aligned}$$

We can now apply the theorem

$$\mathbb{P}\left(\|\mathbf{Y}\| \geq \varepsilon\right) \leq d\exp\left\{-\frac{\varepsilon^2/2}{\frac{1}{\overline{q}}\|\mathbf{A}\mathbf{A}^\top\| + \frac{\varepsilon}{3\overline{q}}}\right\}$$

Whenever $\varepsilon \leq 1$ and $\|\mathbf{A}\mathbf{A}^\top\|^2 \geq 1$ (as it is in our case) this can be simplified to

$$\mathbb{P}\left(\|\mathbf{Y}\| \geq \varepsilon\right) \leq d\exp\left\{-\frac{\varepsilon^2\overline{q}}{4\|\mathbf{A}\mathbf{A}^\top\|^2}\right\}$$

■

Proof of Lemma 2.7. We begin by noting that since $\mathrm{Im}(\Phi\Phi^\top + \gamma\mathbf{\Pi}) = \mathrm{Im}(\mathbf{\Pi})$

$$\mathbf{\Pi} = (\Phi\Phi^\top + \gamma\mathbf{\Pi})^{1/2}(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{-1/2},$$

and that

$$\begin{aligned}\mathbf{\Gamma} &= (\Phi\Phi^\top + \gamma\mathbf{\Pi})^{-1/2}(\Phi\Phi^\top - \Phi\mathbf{S}\mathbf{S}^\top\Phi^\top)(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{-1/2} \\ &= (\Phi\Phi^\top + \gamma\mathbf{\Pi})^{-1/2}\Phi(\mathbf{I}_n - \mathbf{S}\mathbf{S}^\top)\Phi^\top(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{-1/2}\end{aligned}$$

is Hermitian all its eigenvalues are real. Consider an arbitrary $\phi_a \in \mathcal{H}$. We have

$$\begin{aligned}&\phi_a^\top(\Phi\Phi^\top - \Phi\mathbf{S}\mathbf{S}^\top\Phi^\top)\phi_a = \phi_a^\top\Phi(\mathbf{I}_n - \mathbf{S}\mathbf{S}^\top)\Phi^\top\phi_a = \phi_a^\top\mathbf{\Pi}\Phi(\mathbf{I}_n - \mathbf{S}\mathbf{S}^\top)\Phi^\top\mathbf{\Pi}\phi_a \\ &= \phi_a^\top(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{1/2}(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{-1/2}\Phi(\mathbf{I}_n - \mathbf{S}\mathbf{S}^\top)\Phi^\top(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{-1/2}(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{1/2}\phi_a \\ &= \phi_a^\top(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{1/2}\mathbf{\Gamma}(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{1/2}\phi_a \leq \lambda_{\max}(\mathbf{\Gamma})\phi_a^\top(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{1/2}(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{1/2}\phi_a \\ &\leq \|\mathbf{\Gamma}\|\phi_a^\top(\Phi\Phi^\top + \gamma\mathbf{\Pi})\phi_a \leq \varepsilon\phi_a^\top(\Phi\Phi^\top + \gamma\mathbf{\Pi})\phi_a,\end{aligned}$$

and

$$\phi_a^\top\Phi\Phi^\top\phi_a - \phi_a^\top\Phi\mathbf{S}\mathbf{S}^\top\Phi^\top\phi_a \leq \varepsilon\phi_a^\top\Phi\Phi^\top\phi_a + \varepsilon\gamma\phi_a^\top\mathbf{\Pi}\phi_a.$$

Similarly

$$\begin{aligned}&\phi_a^\top(\Phi\mathbf{S}\mathbf{S}^\top\Phi^\top - \Phi\Phi^\top)\phi_a = \phi_a^\top(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{1/2}(-\mathbf{\Gamma})(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{1/2}\phi_a \\ &\leq \lambda_{\max}(-\mathbf{\Gamma})\phi_a^\top(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{1/2}(\Phi\Phi^\top + \gamma\mathbf{\Pi})^{1/2}\phi_a \\ &\leq \|-\mathbf{\Gamma}\|\phi_a^\top(\Phi\Phi^\top + \gamma\mathbf{\Pi})\phi_a = \|\mathbf{\Gamma}\|\phi_a^\top(\Phi\Phi^\top + \gamma\mathbf{\Pi})\phi_a \leq \varepsilon\phi_a^\top(\Phi\Phi^\top + \gamma\mathbf{\Pi})\phi_a.\end{aligned}$$

and

$$\phi_a^\intercal \Phi \mathbf{S}\mathbf{S}^\intercal \Phi^\intercal \phi_a - \phi_a^\intercal \Phi\Phi^\intercal \phi_a \leq \varepsilon \phi_a^\intercal \Phi\Phi^\intercal \phi_a + \varepsilon\gamma \phi_a^\intercal \mathbf{\Pi} \phi_a.$$

Putting them together and using the definition of Positive Semi-Definiteness (Horn and Johnson, 2012) concludes the proof. ■

Proof of Lemma 2.8. Using Proposition 2.11

$$\begin{aligned}
&\phi_i^\intercal (\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i = \phi_i^\intercal \left(\frac{\gamma}{\gamma}\mathbf{\Pi}\right)(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i \\
&= \frac{1}{\gamma}\phi_i^\intercal (\gamma\mathbf{\Pi})(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i = \frac{1}{\gamma}\phi_i^\intercal (\Phi\Phi^\intercal + \gamma\mathbf{\Pi} - \Phi\Phi^\intercal)(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i \\
&= \frac{1}{\gamma}\left(\phi_i^\intercal (\Phi\Phi^\intercal + \gamma\mathbf{\Pi})(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i - \phi_i^\intercal \Phi\Phi^\intercal (\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i\right) \\
&= \frac{1}{\gamma}\left(\phi_i^\intercal \mathbf{\Pi}\phi_i - \phi_i^\intercal \Phi\Phi^\intercal (\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i\right) \\
&= \frac{1}{\gamma}\left(\phi_i^\intercal \phi_i - \phi_i^\intercal \Phi\Phi^\intercal (\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i\right) = \frac{1}{\gamma}\left(\phi_i^\intercal \phi_i - \phi_i^\intercal \Phi(\Phi^\intercal\Phi + \gamma\mathbf{I}_n)^{-1}\Phi^\intercal \phi_i\right)
\end{aligned}$$

■

Proof of Lemma 2.10. Using the SVD decomposition and the invariance of the operator norm to orthonormal transformations

$$\begin{aligned}
\|\mathbf{\Gamma} - \widetilde{\mathbf{\Gamma}}\| &= \|(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1/2}\Phi(\mathbf{I}_n - \mathbf{S}\mathbf{S}^\intercal)\Phi^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1/2}\| \\
&= \|\Phi(\Phi^\intercal\Phi + \gamma\mathbf{I}_n)^{-1/2}(\mathbf{I}_n - \mathbf{S}\mathbf{S}^\intercal)(\Phi^\intercal\Phi + \gamma\mathbf{I}_n)^{-1/2}\Phi^\intercal\| \\
&= \|\mathbf{V}\mathbf{\Sigma}^\intercal(\mathbf{\Sigma}\mathbf{\Sigma}^\intercal + \gamma\mathbf{I}_r)^{-1/2}\mathbf{U}^\intercal(\mathbf{I}_n - \mathbf{S}\mathbf{S}^\intercal)\mathbf{U}(\mathbf{\Sigma}\mathbf{\Sigma}^\intercal + \gamma\mathbf{I}_r)^{-1/2}\mathbf{\Sigma}\mathbf{V}^\intercal\| \\
&= \|\mathbf{U}(\mathbf{\Sigma}\mathbf{\Sigma}^\intercal + \gamma\mathbf{I}_r)^{-1/2}\mathbf{\Sigma}\mathbf{U}^\intercal(\mathbf{I}_n - \mathbf{S}\mathbf{S}^\intercal)\mathbf{U}\mathbf{\Sigma}^\intercal(\mathbf{\Sigma}\mathbf{\Sigma}^\intercal + \gamma\mathbf{I}_r)^{-1/2}\mathbf{U}^\intercal\| = \|\mathbf{P} - \widetilde{\mathbf{P}}\|.
\end{aligned}$$

■

3. Sequential RLS Sampling

In the previous section we showed that batch LS and RLS sampling can be used to construct small and accurate dictionaries. Unfortunately this requires an oracle that returns the RLSs.
One possibility way to implement RLS sampling without the oracle is to use Definition 2.4 and the equivalence from Lemma 2.8 to compute all τ_i exactly. Unfortunately, this algorithm, that we will call Exact-RLS, requires decomposing or inverting an $n \times n$ matrix $\Phi^\intercal \Phi$, which takes $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space.

In this chapter we survey existing *sequential* sampling processes that try to reduce this computational cost, and introduce two new algorithms for sequential RLS sampling without (KORS) and with (SQUEAK) the capability to remove samples already included in the dictionary. We show that these approach outperform all previous methods, ultimately reducing the overall computational cost of RLS sampling from $\mathcal{O}(n^3)$ time, $\mathcal{O}(n^2)$ space and repeated passes over the dataset to a near-linear $\widetilde{\mathcal{O}}(nd_{\text{eff}}(\gamma)^3)$ time, near-constant $\widetilde{\mathcal{O}}(d_{\text{eff}}(\gamma)^2)$ space and a *single pass* over the data.

This allows us to run our algorithms not only in the traditional *random* memory access computational model, but also in the more constrained *sequential* memory access model. The main difference between data storage solutions in the real world is whether they support random access, i.e., the capacity of reading *any* entry in the dataset in constant time regardless of their position in the dataset; or only support sequential access, i.e., they can only read the entries of the dataset in order, or pay a much larger time penalty for skipping to another position.
Notable examples of the former is the aptly named Random Access Memory (RAM) in modern computers, while an example of the latter are magnetic hard-drives, where due to the spin times of the magnetic disk random access reading throughput is 2 to 3 orders of magnitude slower than sequential access throughput.
Sequential access memories are usually cheaper and larger than random access memories not only due to technological and material constraints, but also fundamental constraint,

Algorithm 3 Oracle Approximate RLSs sampling

Input: Regularization γ, accuracy ε, accuracy ρ, budget $\overline{q}$
Output: $\mathcal{I}$

1: Initialize $\mathcal{I} = \emptyset$
2: **for** $i = \{1, \dots, n\}$ **do**
3: Compute ρ-accurate approximate RLSs $\widetilde{\tau}_i$
4: Set approximate probability $\widetilde{p}_i = \widetilde{\tau}_i$
5: Draw the binomial r.v. $q_i \sim \mathcal{B}(\widetilde{p}_i, \overline{q})$
6: If $q_t \neq 0$, add $\left(\frac{1}{\widetilde{p}_i}\frac{q_i}{\overline{q}}, \phi_i\right)$ to $\mathcal{I}$
7: **end for**

since as the size of the storage grows indexing logic and circuitry becomes more complex and slow.
Restricting ourselves to perfectly sequential access (a single pass over the dataset) places us in a *streaming* setting. In other words, we assume that we can see each sample only once, and we must either choose to explicitly store it in the dictionary to access it later, or discard it and permanently lose access to it. As single-pass algorithms with a constant memory requirement, KORS and SQUEAK can operate in a streaming setting.

Restrictions similar to the streaming setting also apply to the *distributed* computing setting, where the data is split across multiple databases or datacenters, and communication costs across centers are much higher. To avoid these costs, algorithms must run in a *distributed* manner, with all computations restricted to data *local* to each site. To counterbalance this drawback, they can now exploit the additional computational power of multiple machines, and any parallelism that the distributed algorithm allows.
We will show that SQUEAK can operate in a distributed setting, running efficiently in parallel across multiple machine. This will further reduce its runtime to $\widetilde{\mathcal{O}}(d_{\text{eff}}(\gamma)^3)$.

Note that although the sampling schemes reviewed in this section were proposed in chronological order Alaoui and Mahoney, (2015), Calandriello et al., (2017a), Calandriello et al., (2017c) and Musco and Musco, (2017)[1], we slightly change the order in the presentation to capture all the methods in a general framework.

3.1 Multi-Pass RLS sampling

We begin by presenting three multi-pass algorithms that will help us introduce concepts necessary to build our single-pass, time-efficient RLS sampling algorithm.

3.1.1 Two-Step RLS sampling

The first algorithm that we present was introduced by Alaoui and Mahoney, (2015), together with the definition of RLSs and the first results on oracle RLSs sampling. To begin with, Alaoui and Mahoney, (2015) note that the proof of Proposition 2.5 still follow through if instead of sampling according to exact RLSs τ_i, we sample according to ρ-accurate RLSs.

[1] A preliminary version of the paper, (Musco and Musco, 2016), was available while (Musco and Musco, 2017) was under submission. (Calandriello et al., 2017a) and (Calandriello et al., 2017c) were developed and submitted independently from this preliminary version.

Algorithm 4 Two-Step RLS sampling (Two-Step-RLS)

Input: Map Φ, regularization γ, accuracy ε, budget $\overline{q}_1$, $\overline{q}_2$

1: Initialize $\overline{\mathcal{I}} = \emptyset$
2: **for** $i = \{1, \dots, n\}$ **do**
3: draw $q_i \sim \mathcal{B}(1/n, \overline{q}_1)$ and if $q_i \neq 0$, add $\left(n\frac{q_i}{\overline{q}_1}, \phi_i\right)$ to $\overline{\mathcal{I}}$
4: **end for**
5: compute $\overline{\mathbf{\Gamma}} = \Phi\overline{\mathbf{S}}(\overline{\mathbf{S}}^\intercal\Phi^\intercal\Phi\overline{\mathbf{S}} + \gamma\mathbf{I}_n)^{-1}\overline{\mathbf{S}}^\intercal\Phi^\intercal$
6: Initialize $\mathcal{I} = \emptyset$
7: **for** $i = \{1, \dots, n\}$ **do**
8: compute $\widetilde{\tau}_i = \phi_i^\intercal\overline{\mathbf{\Gamma}}^{1/2}(\overline{\mathbf{\Gamma}}^{1/2}\Phi\Phi^\intercal\overline{\mathbf{\Gamma}}^{1/2} + \gamma\overline{\mathbf{\Pi}})^{-1}\overline{\mathbf{\Gamma}}^{1/2}\phi_i$
9: compute $\widetilde{p}_i = \widetilde{\tau}_i$
10: draw $q_i \sim \mathcal{B}(\widetilde{p}_i, \overline{q}_2)$ and if $q_i \neq 0$, add $\left(\frac{1}{\widetilde{p}_i}\frac{q_i}{\overline{q}_2}, \phi_i\right)$ to $\mathcal{I}$
11: **end for**

Definition 3.1 An approximate RLS $\widetilde{\tau}_i$ is called ρ-accurate for $\rho \geq 1$ if it satisfies

$$\frac{1}{\rho}\tau_i \leq \widetilde{\tau}_i \leq \tau_i.$$

In other words, to be ρ-accurate, an approximation $\widetilde{\tau}_i$ of a RLS τ_i must be smaller than τ_i, so as to keep being a probability, while at the same time be larger than τ_i/ρ to avoid under-representing the importance of ϕ_i. We have the following guarantees for oracle ρ-approximate RLS sampling, reported in Algorithm 3.

Corollary 3.1 — Lemma 2.10, Proposition 2.5, Proposition 2.13. Consider an arbitrary map $\Phi : \mathbb{R}^n \to \mathcal{H}$ with ϕ_i as its i-th column and rank r. Given $\gamma > 0$, let $\mathbf{\Gamma}$ and $\widetilde{\mathbf{\Gamma}}$ be defined according to Definition 2.5, and the RLS τ_i according to Definition 2.7. Let $\mathcal{I}$ be the dictionary generated by Algorithm 3 run with parameters

$$\widetilde{p}_i = \widetilde{\tau}_i, \qquad \overline{q} \geq \frac{4\rho\log(4r/\delta)}{\varepsilon^2},$$

where the approximate RLSs $\widetilde{\tau}_i$ are ρ-accurate w.r.t. τ_i, then for any $0 < \varepsilon < 1$ and w.p.$1 - \delta$

Accuracy: $\mathcal{I}$ is (ε, γ)-accurate.

Space: the size of the dictionary is bounded by $|\mathcal{I}| \leq 3\overline{q}\sum_{i=1}^n \widetilde{\tau}_i \leq 3\overline{q}d_{\text{eff}}(\gamma)$.

This is coherent with our interpretation of the extra budget $\overline{q}$: if we make a constant error on the importance $\widetilde{\tau}_i$ of ϕ_i, i.e., running Algorithm 3 with ρ-accurate RLSs estimate, we only need a constant increase in $\overline{q}$ and in final dictionary size to obtain the desired (ε, γ)-accurate dictionary.

Knowing that ρ-accurate RLSs suffice to sample, Alaoui and Mahoney, (2015) propose a simple two-step algorithm Two-Step-RLS, reported in Algorithm 4:

1 Do a first sampling pass with probabilities that are simple to compute (e.g., uniform) to get a mildly accurate dictionary $\overline{\mathcal{I}}$, and use $\overline{\mathcal{I}}$ to build ρ-accurate approximations $\widetilde{\tau}_i$.
2 Do a second sampling pass using $\widetilde{\tau}_i$, and build an (ε, γ)-accurate dictionary $\mathcal{I}$.

Proposition 3.2 — (Alaoui and Mahoney, 2015). Consider an arbitrary map $\Phi : \mathbb{R}^n \to \mathcal{H}$ with ϕ_i as its i-th column and rank r. Given $\gamma > 0$, let $\mathbf{\Gamma}$ and $\widetilde{\mathbf{\Gamma}}$ be defined according to Definition 2.5, and the RLS τ_i according to Definition 2.7. Let $\mathcal{I}$, $\{\widetilde{\tau}_i\}$ be the dictionary and approximate RLSs generated by Two-Step-RLS (Algorithm 4) using Eq. 3.1. If

$$\overline{q}_1 \geq \frac{n}{\gamma}\frac{4\log(4r/\delta)}{\varepsilon^2}, \qquad \overline{q}_2 \geq \left(\frac{\lambda_{\max}-\gamma}{\lambda_{\min}-\gamma}\right)\frac{4\log(4r/\delta)}{\varepsilon^2},$$

then for any $0 < \varepsilon < 1$ and w.p.$1-\delta$

Accuracy: $\mathcal{I}$ and $\overline{\mathcal{I}}$ are (ε,γ)-accurate

Space: the size of the dictionaries are bounded by $|\overline{\mathcal{I}}| \leq 3\overline{q}_1$ and $|\mathcal{I}| \leq 3\overline{q}_2 d_{\text{eff}}(\gamma)$

Time: the runtime of the algorithm is $\mathcal{O}(n\overline{q}_1^2)$

Passes: the algorithm requires 3 passes over Φ

Proof. Note that since $\tau_i(\gamma) \leq \frac{1}{1+\gamma} \leq 1/\gamma$, setting $\overline{q}_1 \geq \frac{n}{\gamma}\frac{4\log(4r/\delta)}{\varepsilon^2}$ guarantees that $\overline{q}_1 \geq n\max_i \tau_i(\gamma)\frac{4\log(4r/\delta)}{\varepsilon^2}$ as requested by Proposition 2.3. We can then use Proposition 2.3 to guarantee that the dictionary built during Step 1 is an (ε,γ)-accurate dictionary with $|\overline{\mathcal{I}}| \leq 3\overline{q}_1$ atoms.
In their original work, rather than uniform sampling Alaoui and Mahoney, (2015) use the slightly more refined probabilities $p_i = \phi_i^\intercal\phi_i/\operatorname{Tr}(\Phi\Phi^\intercal)$, which provides a slightly tighter dictionary size bound of $4\frac{\operatorname{Tr}(\Phi\Phi)}{\gamma}\log(n/\delta)/\varepsilon^2$. Nonetheless, in many cases the samples are already in unitary norm $\phi_i^\intercal\phi_i = 1$, (e.g., all self normalized kernels such as the Gaussian kernel) which for simplicity we assumed in this thesis, and the adaptive distribution reverts to uniform sampling.
Given $\overline{\mathcal{I}}$, Alaoui and Mahoney, (2015) propose the following estimator for $\widetilde{\tau}_i$, that we report using our notation,

$$\widetilde{\tau}_i(\gamma') = \phi_i^\intercal\overline{\mathbf{\Gamma}}^{1/2}(\overline{\mathbf{\Gamma}}^{1/2}\Phi\Phi^\intercal\overline{\mathbf{\Gamma}}^{1/2} + \gamma'\overline{\mathbf{\Pi}})^{-1}\overline{\mathbf{\Gamma}}^{1/2}\phi_i \tag{3.1}$$

with $\overline{\mathbf{\Gamma}} = \Phi\overline{\mathbf{S}}(\overline{\mathbf{S}}^\intercal\Phi^\intercal\Phi\overline{\mathbf{S}} + \gamma\mathbf{I}_n)^{-1}\overline{\mathbf{S}}^\intercal\Phi^\intercal$ being the exact regularized projection matrix on the approximate $\operatorname{Im}(\Phi\overline{\mathbf{S}})$ (not to be confused with the approximate regularized projection matrix $\widetilde{\mathbf{\Gamma}}$). Note also that γ' inside the inverse does not need to be equal to the γ used in $\overline{\mathbf{\Gamma}}$. Regardless of γ', the ρ multiplicative approximation constant for $\widetilde{\tau}_i$ given by Alaoui and Mahoney, (2015, Theorem 4) is $\rho = \left(\frac{\lambda_{\max}-\gamma}{\lambda_{\min}-\gamma}\right)$ resulting in

$$\left(\frac{\lambda_{\min}-\gamma}{\lambda_{\max}-\gamma}\right)\tau_i(\gamma') \leq \widetilde{\tau}_i \leq \tau_i(\gamma').$$

Plugging this into an approximate RLSs sampling in Step 2, invoking Corollary 3.1, and setting $\overline{q}_2$ appropriately, we obtain a final (ε,γ)-accurate dictionary $\mathcal{I}$ containing only $|\mathcal{I}| \leq \overline{q}_2 d_{\text{eff}}(\gamma')$ atoms. ■

Unfortunately the definition of $\overline{q}_2$ include a $(\lambda_{\min}-\gamma)^{-1}$ factor due to the rough ρ-accuracy of the RLSs $\widetilde{\tau}_i$ computed using Eq. 3.1. This forces us to set $\gamma < \lambda_{\min}$, and $\lambda_{\min}$ is typically smaller than 1, resulting in $\overline{q}_1 \geq n/\gamma > n/\lambda_{\min} > n$ and an extremely large first-step dictionary that keeps all the input atoms, resulting in no computational improvement over exact RLSs calculation.

To make comparisons with following algorithms easier, we present a slightly modified version of Alaoui and Mahoney's algorithm that replaces Eq. 3.1 with an improved estimator introduced by Calandriello et al., (2017a). Given a dictionary $\mathcal{I}$, let $\mathbf{I}_\mathcal{I}$ be the $\mathbb{R}^{n\times n}$ diagonal matrix with $[\mathbf{I}_\mathcal{I}]_{i,i} = 1$ if $q_i \neq 0$ and 0 otherwise.

Lemma 3.3 — (Calandriello et al., 2017a, Lemma 2). Given an (ε, γ)-accurate dictionary $\mathcal{I}$, compute the approximate RLSs $\widetilde{\tau}_i$ as

$$\begin{aligned}\widetilde{\tau}_i &= (1-\varepsilon)\phi_i^\intercal(\Phi\mathbf{S}\mathbf{S}^\intercal\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i \\ &= \tfrac{1-\varepsilon}{\gamma}(\phi_i^\intercal\phi_i - \phi_i^\intercal\Phi\mathbf{S}(\mathbf{S}^\intercal\Phi^\intercal\Phi\mathbf{S} + \gamma\mathbf{I}_\mathcal{I})^{-1}\mathbf{S}^\intercal\Phi^\intercal\phi_i). \qquad 3.2\end{aligned}$$

Then for all i in $\mathcal{D}$, the estimator in Eq. 3.2 is ρ-accurate, with $\rho = (1+\varepsilon)/(1-\varepsilon)$. Moreover, $\widetilde{\tau}_i$ can be computed in $\mathcal{O}(|\mathcal{I}|^3)$ time and $\mathcal{O}(|\mathcal{I}|^2)$ space using only ϕ_i and atoms present in $\mathcal{I}$ itself.

This estimator follows naturally from the reformulation of the RLS presented in Lemma 2.8. Ideally we would like to keep all $\Phi\Phi^\intercal$ columns in the inverse $(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}$ for maximum accuracy, but applying Lemma 2.8 would give us an $\mathbb{R}^{n\times n}$ matrix to invert, which requires $\mathcal{O}(n^3)$ time and space. Instead, similarly to what Cohen et al., (2016) did for the special Euclidean case $\phi_i = \mathbf{x}_i$, we only keep a reweighted subset $(\Phi\mathbf{S}\mathbf{S}^\intercal\Phi^\intercal + \gamma\mathbf{\Pi})$ so that after applying Lemma 2.8 and eliminating the zeroed-out columns (Proposition 2.12) we are left with only a $|\mathcal{I}| \times |\mathcal{I}|$ matrix that can be inverted efficiently in $\mathcal{O}(|\mathcal{I}|^3)$ time and $\mathcal{O}(|\mathcal{I}|^2)$ space.

Proof sketch of Lemma 3.3. The full proof is reported in Section 3.5. To bound the accuracy, we know from the dictionary that

$$\Phi\mathbf{S}\mathbf{S}^\intercal\Phi^\intercal + \gamma\mathbf{\Pi} \preceq (1+\varepsilon)\Phi\Phi^\intercal + \varepsilon\gamma\mathbf{\Pi} + \gamma\mathbf{\Pi} = (1+\varepsilon)(\Phi\Phi^\intercal + \gamma\mathbf{\Pi}),$$

and therefore $(\Phi\mathbf{S}\mathbf{S}^\intercal\Phi^\intercal + \gamma\mathbf{\Pi})^{-1} \preceq \frac{1}{1+\varepsilon}(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}$. Using a similar approach to obtain a lower bound, and adjusting constants, we can prove the $\rho = \frac{(1+\varepsilon)}{(1-\varepsilon)}$-accuracy. Note that Lemma 2.8 is crucial to the proof. We are first showing that $(1-\varepsilon)\phi_i^\intercal(\Phi\mathbf{S}\mathbf{S}^\intercal\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i$ is ρ-accurate w.r.t. $\phi_i^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})^{-1}\phi_i$, and only afterwards we apply the equivalence of Eq. 2.14 to obtain a reformulation of the estimator that can be computed using only inner products. Proving directly the ρ-accuracy of the computable version is significantly more difficult. ■

Plugging Lemma 3.3 in the first step of Alaoui and Mahoney's two-step algorithm we have

Corollary 3.4 — Proposition 2.3, Corollary 3.1, Lemma 3.3. Consider an arbitrary map $\Phi : \mathbb{R}^n \to \mathcal{H}$ with ϕ_i as its i-th column and rank r. Given $\gamma > 0$, let $\mathbf{\Gamma}$ and $\widetilde{\mathbf{\Gamma}}$ be defined according to Definition 2.5, and the RLS τ_i according to Definition 2.7. Let $\mathcal{I}$, $\{\widetilde{\tau}_i\}$ be the dictionary and approximate RLSs generated by Two-Step-RLS (Algorithm 4) using Eq. 3.2 in place of Eq. 3.1. If

$$\bar{q}_1 \geq \frac{n}{\gamma}\frac{4\log(4r/\delta)}{\varepsilon^2}\tau_i, \qquad \bar{q}_2 \geq \left(\frac{1+\varepsilon}{1-\varepsilon}\right)\frac{4\log(4r/\delta)}{\varepsilon^2},$$

then for any $0 < \varepsilon < 1$ and w.p.$1-\delta$

Accuracy: $\mathcal{I}$ and $\overline{\mathcal{I}}$ are (ε, γ)-accurate
Space: the size of the dictionaries are bounded by $|\overline{\mathcal{I}}| \leq 3\overline{q}_1$ and $|\mathcal{I}| \leq 3\overline{q}_2 d_{\text{eff}}(\gamma)$
Time: the runtime of the algorithm is $\mathcal{O}(n\overline{q}_1^2)$
Passes: the algorithm requires 3 passes over Φ

Compared to Proposition 3.2, we have greatly improved the $\left(\frac{\lambda_{\max}-\gamma}{\lambda_{\min}-\gamma}\right)$ constant in $\overline{q}_2$ to $\left(\frac{1+\varepsilon}{1-\varepsilon}\right)$, which in turn leave us free to choose γ more freely without impacting the n/γ factor in $\overline{q}_1$.
The total runtime of the algorithm is dominated by the RLSs estimation, that takes $\mathcal{O}(|\overline{\mathcal{I}}|^3) = \widetilde{\mathcal{O}}(\overline{q}_1^3)$ to precompute the inverse or a decomposition, and $\widetilde{\mathcal{O}}(n\overline{q}_1^2)$ time to compute all individual $\widetilde{\tau}_i$. Since $\overline{q}_1$ must scale with n/γ, this results in a $\mathcal{O}(n^3/\gamma^2)$ time and $\mathcal{O}(n^2/\gamma^2)$ space complexity, down from the $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space of exact RLSs sampling. As Alaoui and Mahoney, (2015) note, when γ is large, this is a significantly smaller than the exact approach. As we will see, a typical value for γ is $n^{1/2}$, resulting in a $\mathcal{O}(n^2)$ time and $\mathcal{O}(n)$ space complexity.

To recap, Alaoui and Mahoney's Two-Step-RLS gave us two key ideas to quickly construct a dictionary: (1) sample according to approximate RLSs instead of exact RLSs, and (2) instead of directly sampling proportionally to the final RLSs (second step) build an intermediate approximation that is easier to compute (first step).
While Lemma 3.3 is already close to the best we can do for RLSs estimation given an (ε, γ)-accurate dictionary, the idea of sequentially approximating better and better RLSs can be pushed further. In the next sections we present two algorithms that perform *multiple* approximation steps rather than one.

3.1.2 Many-Step RLS sampling

In the analysis of Two-Step-RLS Alaoui and Mahoney, (2015) use guarantees for uniform sampling (Proposition 2.3) to decide how many samples are necessary in the first sampling step. In this section we take the alternative approach of showing that the estimator $\widetilde{\tau}_i = 1/(n+\gamma)$ can be used as an accurate ρ-accurate approximation, with constant ρ, but only for specific values of γ.
It is easy to show that this is not true for all γ: if a sample is orthogonal to all others and $\gamma = 1$ we have $\tau_i = 1/2$ and $\widetilde{\tau}_i = 1/(n+1)$, which is an arbitrarily poor approximation. Nonetheless, using the generic upper and lower bounds $\frac{1}{n+\gamma} \leq \tau_i(\gamma) \leq \frac{1}{1+\gamma}$ we know that $\widetilde{\tau}_i$ must satisfy

$$\frac{1}{\rho}\tau_i(\gamma) \leq \frac{1}{\rho}\frac{1}{1+\gamma} \leq \widetilde{\tau}_i = \frac{1}{n+\gamma} \leq \tau_i(\gamma),$$

which means that in general $\frac{1}{n+\gamma}$ achieve $\rho = \frac{n+\gamma}{1+\gamma}$ accuracy. This accuracy depends on n and γ, therefore uniform $\widetilde{\tau}_i$ might be ρ-accurate w.r.t. $\tau_i(\gamma)$ with constant ρ for certain values of γ, and not for others. For example, setting $\gamma = n$ tells us that uniform sampling is 2-accurate, w.r.t. $\tau_i(n)$.
In other words, when γ is small, uniform sampling is only accurate when the RLSs $\tau_i(\gamma)$ are uniform due to the low coherence in the data; but using a high enough γ we can shrink all probabilities so much that uniform sampling becomes accurate *regardless* of the data.
Indeed, an additive error of $\varepsilon n \mathbf{\Pi}$ on the reconstruction for an operator with norm $\|\Phi\Phi^\intercal\| \leq n$ means that we do not need to reconstruct much at all to meet (ε, n)-accuracy. In particular,

Algorithm 5 Many-Step RLS sampling (MANY-STEP-RLS)

Input: Map Φ, regularization γ, accuracy ε, budget $\overline{q}$
Output: $\mathcal{I}$

```
Initialize p̃_{0,i} = τ̃_{0,i} = 1/(n + γ) for all i, , γ_0 = n
for t = {1, ..., ⌈log_2(n/γ) + 1⌉} do
    Initialize γ_t = γ_{t-1}/2, I_t = ∅
    for i = {1, ..., n} do
        draw q_{t,i} ~ B(p̃_{t-1,i}, q̄) and if q_{t,i} ≠ 0, add (1/p̃_{t-1,i} q_{t,i}/q̄, φ_i) to I_t
    end for
    construct S_t from I_t
    for i = {1, ..., n} do
        set τ̃_{t,i} = (1 − ε)φ_i^T(ΦS_tS_t^TΦ^T + (1 + ε)γ_tΠ)^{-1}φ_i
        set p̃_{t,i} = τ̃_{t,i}
    end for
end for
```

applying Corollary 3.1 tells us that we can construct an (ε, n)-accurate dictionary having only $24 d_{\text{eff}}(n)\log(n/\delta)/\varepsilon^2 \leq 24\log(n/\delta)/\varepsilon^2$ atoms since $d_{\text{eff}}(\gamma) \leq n/\gamma = 1$ when $\gamma = n$.[2]. Despite the low accuracy of a (ε, n)-accurate dictionary, the following lemma, based on Lemma 3.3 shows how we can use this inaccurate dictionary to compute slightly better estimates.

Lemma 3.5 Given an $(\varepsilon, 2\gamma)$-accurate dictionary $\mathcal{I}$, compute the approximate RLSs $\widetilde{\tau}_i$ as

$$\begin{aligned}\widetilde{\tau}_i &= (1-\varepsilon)\phi_i^\intercal(\Phi\mathbf{S}\mathbf{S}^\intercal\Phi^\intercal + (1+\varepsilon)\gamma\mathbf{\Pi})^{-1}\phi_i \\ &= \tfrac{1-\varepsilon}{(1+\varepsilon)\gamma}(\phi_i^\intercal\phi_i - \phi_i^\intercal\Phi\mathbf{S}(\mathbf{S}^\intercal\Phi^\intercal\Phi\mathbf{S} + (1+\varepsilon)\gamma\mathbf{I}_{\mathcal{I}})^{-1}\mathbf{S}^\intercal\Phi^\intercal\phi_i). \qquad 3.3\end{aligned}$$

Then for all i in $\mathcal{D}$, the estimator in Eq. 3.3 is ρ-accurate w.r.t. $\tau_i(\gamma)$, with accuracy $\rho = (1+3\varepsilon)/(1-\varepsilon)$. Moreover, $\widetilde{\tau}_i$ can be computed in $\mathcal{O}(|\mathcal{I}|^3)$ time and $\mathcal{O}(|\mathcal{I}|^2)$ space using only ϕ_i and atoms present in $\mathcal{I}$ itself.

Therefore, using Lemma 3.5 and the (ε, n)-accurate dictionary, we can compute an $(\varepsilon, n/2)$ dictionary with only $12(\frac{1+3\varepsilon}{1-\varepsilon})d_{\text{eff}}(n/2)\log(n/\delta)/\varepsilon^2$ atoms, that can be used to compute an $(\varepsilon, n/4)$ dictionary with only $12(\frac{1+3\varepsilon}{1-\varepsilon})d_{\text{eff}}(n/4)\log(n/\delta)/\varepsilon^2$ atoms, and so on until after $\lceil\log_2(n/\gamma)+1\rceil$ steps we have a $12(\frac{1+3\varepsilon}{1-\varepsilon})d_{\text{eff}}(\gamma)\log(n/\delta)/\varepsilon^2$ dictionary. The resulting algorithm, MANY-STEP-RLS is reported in Algorithm 5.

Using a union bound on the steps we obtain the following result.

Theorem 3.6 — Lemma 3.5, Corollary 3.1, Proposition 2.13. Consider an arbitrary map $\Phi : \mathbb{R}^n \to \mathcal{H}$ with ϕ_i as its i-th column and rank r. Given $\gamma > 0$, let $\mathbf{\Gamma}$ and $\widetilde{\mathbf{\Gamma}}$ be defined according to Definition 2.5, and the RLS τ_i according to Definition 2.7. Let $\mathcal{I}_t$ be the dictionaries generated by MANY-STEP-RLS (Algorithm 5). If

$$\overline{q} \geq \left(\frac{1+3\varepsilon}{1-\varepsilon}\right)\frac{4}{\varepsilon^2}\log\left(\frac{4r\lceil\log_2(n/\gamma)+1\rceil}{\delta}\right),$$

[2] Uniform sampling (i.e., Proposition 2.3) would give us the same result, since $n\mu = n\max_i \tau_i(n) \leq n\frac{1}{1+n} \leq 1$.

then for any $0 < \varepsilon < 1$ and w.p.$1 - \delta$

Accuracy: Each $\mathcal{I}_t$ is $(\varepsilon, n/2^{t-1})$-accurate w.r.t. Φ, and $\mathcal{I}_{\lceil \log_2(n/\gamma)+1 \rceil}$ is (ε, γ)-accurate w.r.t. Φ

Space: the size of each of the dictionaries is bounded by $|\mathcal{I}_t| \leq 3\overline{q} d_{\text{eff}}(\gamma)$

Time: the runtime of the algorithm is $\mathcal{O}(n\overline{q}^2)$

Passes: the algorithm requires $2\lceil \log_2(n/\gamma) + 1 \rceil$ passes over Φ or $4n$ random accesses

Using Many-Step-RLS we can compute an (ε, γ)-accurate dictionary with a size that is bounded by $12 d_{\text{eff}}(\gamma) \log(n \lceil \log_2(n/\gamma) + 1 \rceil / \delta)/\varepsilon^2$. Up to a $\log(\lceil \log_2(n/\gamma) + 1 \rceil)$ factor, this is the same result as Oracle-RLS, but with a computational complexity of $\mathcal{O}(n\overline{q}^2) \leq \widetilde{\mathcal{O}}(n d_{\text{eff}}(\gamma)^2)$.
Since we are interested in cases where $d_{\text{eff}}(\gamma)$ is small, this is essentially a *linear* runtime in n, which is an enormous improvement over the $\mathcal{O}(n^3)$ cost of the exact oracle and finally breaks the *quadratic barrier* that blocked previous methods. We refer to algorithms that achieve this time complexity as near-linear time algorithms.
On the other hand, when $d_{\text{eff}}(\gamma)$ is large, potentially as large as n, Many-Step-RLS recovers the $\mathcal{O}(n^3)$ runtime of computing exact RLSs using the definition and sampling. Nonetheless, in the cases where $d_{\text{eff}}(\gamma)$ is large even an oracle algorithm would generate a large dictionary, that would not be very useful to accelerate downstream applications. In this case we can either accept that the problem is intrinsically hard, i.e., there is no free lunch, or start considering increasing γ to trade-off computational cost and accuracy.
We postpone a full discussion of Theorem 3.6 to Section 3.3.3, where we will make an in-depth comparison across all near-linear time oracles. We now present an alternative approach that eliminates the extra $\log(\log(n/\gamma))$ factor in the size of the dictionary.

3.1.3 Recursive RLS sampling

Another approach to multiple-step RLS sampling, called Recursive-RLS and reported in Algorithm 6, was recently proposed by Musco and Musco, (2017). For clarity, we also report an equivalent iterative version. Similarly to Many-Step-RLS, the algorithm's goal is to approximate sufficiently well τ_i in order to be able to perform approximate RLSs sampling and use Corollary 3.1.
While Many-Step-RLS is a strictly bottom-up approach that starts with a large γ (small $d_{\text{eff}}(\gamma)$ and dictionary) and increasingly shrinks it (large $d_{\text{eff}}(\gamma)$ and dictionary), Recursive-RLS has both a top-down and bottom-up phase.
In the top-down phase it recursively randomly halves the number of columns in the map Φ: at each level t for each column in Φ_{t-1} a $1/2$ coin flip is performed, and if it is successful the column is included in Φ_t. As a consequence we have a top-down chain of maps where Φ_t w.h.p. contains $\mathcal{O}(n/2^t)$ atoms, and all atoms in Φ_t are included in Φ_{t-1}. The goal of the recursive chain is to reduce the size of Φ_t sufficiently so that it can be used to compute approximate RLSs with a small computational cost. This base-case is encountered after $\log(n)$ recursion, where the algorithm constructs a dictionary $\mathcal{I}_{\log_2(n)} = \{(1, \phi_i)\}$ based on $\Phi_{\log_2(n)}$ that contains only a constant number of atoms.
Starting from there, the algorithm performs a bottom-up pass. At each level t the dictionary $\mathcal{I}_{t+1}$ from the lower level $t+1$ is used to estimate approximate RLSs $\widetilde{\tau}_{t,i}$ for Φ_t. The RLSs estimator used by Musco and Musco, (2017) is identical to Eq. 3.2, although in their independent discovery Musco and Musco, (2017) derived it starting from $\mathbf{P}$, $\mathbf{C}$ and $\mathbf{c}_i$ (Definition 2.8) rather than $\mathbf{\Gamma}$, Φ, and ϕ_i (our definition). This makes it hard to see

Algorithm 6 RECURSIVE-RLS sampling (Musco and Musco, 2017) (Recursive version)

Input: map Φ, regularization γ, failure probability δ
Output: Dictionary $\mathcal{I}$

1: Sample $\overline{\Phi}$ by randomly including each column of Φ w.p. $1/2$
2: **if** $\overline{\Phi}$ has less than 16 columns **then**
3: Set $\overline{\mathcal{I}} = \{(1, \phi_i)\}$ for all ϕ_i in $\overline{\Phi}$
4: **else**
5: Invoke Algorithm 6 on $\overline{\Phi}$ adjusting $\delta \leftarrow \delta/2$
6: Set $\overline{\mathcal{I}}$ to the result returned by the recursive invocation
7: **end if**
8: For all ϕ_i in Φ compute $\widetilde{\tau}_{t,i}$ using $\overline{\mathcal{I}}$ and Eq. 3.2 with $\varepsilon = 1/2$.
9: Compute approximate probabilities $\widetilde{p}_{t,i} = \min\{1, 16\widetilde{\tau}_{t,i} \log(\sum \widetilde{\tau}_{t,i}/\delta)$
10: Construct $\mathcal{I}$ of Φ by approx. RLSs sampling (Algorithm 2) with $p_i = \widetilde{p}_{t,i}$ and $\overline{q} = 1$

Algorithm 6 RECURSIVE-RLS sampling (Musco and Musco, 2017) (Iterative version)

Input: map Φ, regularization γ, budget $\overline{q}$
Output: Dictionary $\mathcal{I}_1$

1: **for** $t = \{1, \dots, \log_2(n)\}$ **do** ▷Top-down
2: Sample Φ_t by randomly including each column of Φ_{t-1} w.p. $1/2$
3: **end for**
4: Set $\mathcal{I}_{\log_2(n)} = \{(1, \phi_i)\}$ for all ϕ_i in $\Phi_{\log_2(n)}$ ▷Base case
5: **for** $t = \{\log_2(n) - 1, \dots, 1\}$ **do** ▷Bottom-up
6: For all ϕ_i in Φ_t compute $\widetilde{\tau}_{t,i}$ using $\mathcal{I}_{t+1}$ and Eq. 3.2 with $\varepsilon = 1/2$.
7: Compute approximate probabilities $\widetilde{p}_{t,i} = \min\{1, 32\widetilde{\tau}_{t,i} \log(\sum_i \widetilde{\tau}_{t,i}/(2^{-t}\delta))$
8: Construct $\mathcal{I}$ of Φ by approx. RLSs sampling (Algorithm 2) with $p_i = \widetilde{p}_{t,i}$ and $\overline{q} = 1$
9: **end for**

that Eq. 3.2 is actually a generalization (and a member) of a well known family of RLSs estimators from the linear ($\mathbb{R}^d$) setting (Cohen et al., 2015b; Cohen et al., 2016), and harder to see variations such as Lemma 3.5 or Lemma 3.14.

Once the approximate RLSs are computed, the dictionary $\mathcal{I}_t$ is constructed using approximate RLS sampling (Algorithm 2) with $\varepsilon = 1/2$, $\widetilde{p}_i = \widetilde{p}_{t,i} = 32\widetilde{\tau}_{t,i} \log(\sum_i \widetilde{\tau}_{t,i}/(2^{-t}\delta))$ and $\overline{q} = 1$. In other words, each atom in Φ_t is added to $\mathcal{I}_t$ with probability $\widetilde{p}_{t,i}$ and weight $1/\widetilde{p}_{t,i}$. This is essentially the approach of drawing a Bernoulli $Ber(32\widetilde{\tau}_{t,i} \log(\sum_i \widetilde{\tau}_{t,i}/(2^{-t}\delta)))$ rather than a Binomial $\mathcal{B}(\widetilde{\tau}_{t,i}, 32 \log(\sum_i \widetilde{\tau}_{t,i}/(2^{-t}\delta)))$, which as we saw in Section 2.1.2 generates slightly larger dictionaries. Note that in the run of the algorithm it is possible for an atom ϕ_i to be present in dictionary $\mathcal{I}_t$, not present in $\mathcal{I}_{t-1}$, and again present in $\mathcal{I}_{t-2}$, since the drawings at each level are performed independently.

Proposition 3.7 — (Musco and Musco, 2017). Consider an arbitrary map $\Phi : \mathbb{R}^n \to \mathcal{H}$ with ϕ_i as its i-th column and rank r. Given $\gamma > 0$, let $\mathbf{\Gamma}$ and $\widetilde{\mathbf{\Gamma}}$ be defined according to Definition 2.5, and the RLS τ_i according to Definition 2.7. Let $\mathcal{I} = \mathcal{I}_1$ be the final dictionary generated by RECURSIVE-RLS (Algorithm 6). With probability $1 - 2\delta$

Accuracy: the dictionary $\mathcal{I}$ is $(1/2, \gamma)$-accurate

Space: the size of each of the dictionaries is bounded by $|\overline{\mathcal{I}}| \leq 384 d_{\text{eff}}(\gamma) \log(d_{\text{eff}}(\gamma)/\delta)$

Time: the runtime of the algorithm is $\widetilde{\mathcal{O}}(n d_{\text{eff}}(\gamma)^2)$

Passes: the algorithm requires $2\log(n)$ passes over Φ or $4n$ random accesses

Proof sketch. We provide a sketch of the proof for comparison, and refer the reader to Musco and Musco, (2017) for details. At its core, Recursive-RLS is a recursive scheme where the top-down pass uses Φ_t to approximate Φ_{t-1} through uniform sampling, and a bottom-up pass where $\mathcal{I}_t$ is used to approximate Φ_t through approximate RLSs sampling. The goal is to construct more and more accurate approximations $\widetilde{\tau}_{t,i}$ during the bottom-up pass, in the same spirit of Two-Step-RLS and Many-Step-RLS.
Note that none of the Φ_t are reweighted, therefore $\mathbb{E}[\Phi_t] = (1/2^t)\Phi$ and the top-down sampling process that generates Φ_t is not unbiased. Therefore, we cannot use concentration inequalities to show that Φ_t is an (ε, γ)-accurate approximation of Φ_{t-1}. Moreover, even if we did reweight Φ_t, our results on uniform sampling (Proposition 2.3) suggest that $n/2$ uniform samples are not sufficient to construct an (ε, γ)-accurate dictionary w.h.p.
For the bottom-up pass, since (1) we use $\mathcal{I}_{t+1}$ to approximate RLSs, (2) $\mathcal{I}_{t+1}$ approximates Φ_{t+1}, and (3) Φ_{t+1} does not approximate well Φ_t, the approximate RLSs $\widetilde{\tau}_{t,i}$ are not ρ-accurate w.r.t. Φ_t and the sampling guarantees of Corollary 3.1 do not apply.
Surprisingly and non-trivially, Musco and Musco, (2017) could transfer the proof proposed in Cohen et al., (2015a) for a similar RLSs sampling algorithm in $\mathbb{R}^d$ to the RKHS setting. With a very technical argument they show that although each $\widetilde{\tau}_{t,i}$ does not approximate each $\tau_{t,i}$ the following invariant holds at each level

(1) $\tau_{t,i} \leq \widetilde{\tau}_{t,i}$ which guarantees that we will never undersample.
(2) $\sum_i \widetilde{\tau}_{t,i} \leq d_{\text{eff}}(\gamma) \log(d_{\text{eff}}(\gamma)/\delta)$ which guarantees that *across all atoms* we will not oversample too much.

This is sufficient to show that w.h.p. all dictionaries satisfy $|\mathcal{I}_t| \leq \mathcal{O}(\sum_i \widetilde{\tau}_{t,i}) \leq \widetilde{\mathcal{O}}(d_{\text{eff}}(\gamma))$ and will not be too large, and that although all intermediate matrix Φ_t and dictionary $\mathcal{I}_t$ are not close to Φ, the final dictionary $\mathcal{I}$ is (ε, γ)-accurate w.r.t. Φ. ■

Like Many-Step-RLS, Recursive-RLS produces an accurate dictionary in an small space and time complexity, with Proposition 3.7 improving by a $\log(\log(n/\gamma))$ factor over Theorem 3.6 in space (and $\text{polylog}(\log(n/\gamma))$ in time). We will again postpone a full evaluation of this result to Section 3.3.3, where we compare all near-linear time oracles.

3.2 Sequential RLS sampling without removal

While both Many-Step-RLS and Musco and Musco's Recursive-RLS achieve near-linear runtime, they have two major drawbacks: (1) they require multiple passes over the data or alternatively random access to the dataset, and (2) they have inherent bottlenecks that make it difficult to parallelize them.
In this section we focus on solving the first of these drawbacks with a simple sequential sampling approach that can operate in a *single pass* over the dataset, processing the dataset as if it was coming from a continuous *stream* of data.
The parallelization problems will be addressed in the following section by showing how to generalize sequential sampling to operate on multiple streams at the same time.

To describe this new streaming setting, we will slightly modify our notation. Given our dataset $\mathcal{D} = \{\phi_t\}_{t=1}^n$ we can define the partial dataset $\mathcal{D}_t$ that contains the first t samples. Associated with $\mathcal{D}_t$ we have all the corresponding quantities Φ_t, Ψ_t, $\mathbf{\Pi}_t$, $\mathbf{\Gamma}_t$. Similarly, our goal is now to generate dictionaries $\mathcal{I}_t$ that are (ε, γ)-accurate w.r.t. to their respective Φ_t. We can also define RLSs $\tau_{t,i}$ and effective dimension $d^t_{\text{eff}}(\gamma)$ w.r.t. the partial dataset Φ_t. For notational convenience, we still refer to $\tau_{t,i}$ as a RLSs when $t < i$ (when we have not

Algorithm 7 Streaming Oracle RLS sampling without removal (SOracle-RLS-NoRem)

Input: Regularization ρ, accuracy ε, budget $\overline{q}$

1: Initialize $\mathcal{I}_0 = \emptyset$
2: **for** $t = \{1, \dots, n\}$ **do**
3: receive ϕ_t
4: compute $p_t = \tau_{t,t}$ using an oracle limited to $\mathcal{D}_t$
5: draw $q_t \sim \mathcal{B}(p_t, \overline{q})$
6: if $q_t \neq 0$, add $\left(\frac{1}{p_t}\frac{q_t}{\overline{q}}, \phi_t\right)$ to $\mathcal{I}_t$ ▷Expand
7: **end for**

seen the sample yet), but in this case we will arbitrarily set $\tau_{t,i} = 1$ for all $t < i$.
Remember also that in the streaming setting if a past sample ϕ_i is not explicitly stored in the current dictionary $\mathcal{I}_t$, we cannot access it, i.e., we delete from memory all samples that are not explicitly stored at each step.

3.2.1 Oracle RLS sampling in the streaming setting

From the previous sections' results on RLS sampling, we know that choosing whether to include sample ϕ_t or not in $\mathcal{I}_t$ with probability p_t based on the sample's *final* RLS $\tau_{n,t}$ will be enough to generate an (ε, γ)-accurate dictionary. This is essentially what Oracle-RLS does, with a computationally unbounded oracle providing the exact RLSs $\tau_{n,t}$.
Unfortunately, in the streaming setting at time t we did not yet see the $n-t$ future samples remaining in the stream, and not even the oracle can compute these quantities. We will now present SOracle-RLS-Rem, a simple oracle-based algorithm, reported in Algorithm 7, that generalizes Oracle-RLS to the streaming setting. We call this approach sequential RLS sampling without removal because once an atom is added to the dictionary $\mathcal{I}_t$, it is never subsequently removed.
The basic idea behind SOracle-RLS-NoRem is to use the sample's *current* RLS $\tau_{t,t}$ to decide whether or or not to include it *without* waiting to see the rest of the stream. Even when using exact RLS $\tau_{t,t}$, this approach can be grouped in the family of approximate RLS sampling methods: since we cannot (this time in an absolute sense) compute the probabilities $\tau_{n,t}$, and we try to approximate them with $\tau_{t,t}$. Due to the oracle, all the samples in the dictionary remain independent, and if we can satisfy the invariant $p_i = p_t = \tau_{t,t} \geq \tau_{n,t}$ required by Proposition 2.5 we will generate an (ε, γ)-accurate dictionary. Moreover, if the sum $\sum_{i=1}^{t} \tau_{i,i}$ is not much larger than $d_{\text{eff}}^{t}(\gamma)$, we can use Proposition 2.2 to show that we generate a small dictionary.

We now focus on characterizing how RLSs and $d_{\text{eff}}^{t}(\gamma)$ evolve over time as new samples arrive (e.g., the relationship between $\tau_{t,i}$ and $\tau_{t+1,i}$).

Lemma 3.8 — (Calandriello et al., 2017a). For any map Φ_{t-1} at time $t-1$ and its extension Φ_t at time t, we have that the RLS are monotonically decreasing and the effective dimension is monotonically increasing,

$$\frac{1}{\tau_{t-1,i}+1}\tau_{t-1,i} \leq \tau_{t,i} \leq \tau_{t-1,i}, \qquad d_{\text{eff}}^{t}(\gamma) \geq d_{\text{eff}}^{t-1}(\gamma).$$

It is easier to show that RLSs shrink over time once we define them in terms of Φ and ϕ_i

rather than $\mathbf{C}$ and $\mathbf{c}_i$[3] (Alaoui and Mahoney, 2015) which change size when we add the new sample. It suffices to see that

$$\tau_{t,i} = \phi_i \left(\sum_{s=1}^{t} \phi_s \phi_s^\intercal + \gamma \mathbf{\Pi}_t \right)^{-1} \phi_i \leq \phi_i \left(\sum_{s=1}^{t-1} \phi_s \phi_s^\intercal + \gamma \mathbf{\Pi}_t \right)^{-1} \phi_i = \tau_{t-1,i},$$

since $\sum_{i=1}^{t-1} \phi_i \phi_i^\intercal \preceq \sum_{i=1}^{t} \phi_i \phi_i^\intercal$ has one less term in the summation. Intuitively, adding a new sample ϕ_t to Φ_{t-1} can either decrease the importance of samples observed before (i.e., if they are correlated with the new point) or leave it unchanged (i.e., if they are orthogonal) and thus for any $i \leq t$, the RLSs will shrink. Each RLS will shrink by a different factor, depending on the degree of orthogonality of each sample with the new point and the rest of the dataset, but there is a maximum shrinkage factor $\frac{1}{\tau_{t-1,i}+1} \leq \frac{1}{2}$, that roughly corresponds to the case where the new sample ϕ_i added to Φ_t is *identical* to a sample $\phi_{i'}$ already present in Φ_t.
As for the effective dimension, relating $d_{\text{eff}}^t(\gamma) = \sum_{i=1}^{t} \tau_{t,i}$ with $d_{\text{eff}}^{t-1}(\gamma) = \sum_{i=1}^{t-1} \tau_{t-1,i}$ is not as easy, since adding $\tau_{t,i}$ increases the total sum, but each of the other $\tau_{t,i}$ might shrink compared to $\tau_{t-1,i}$. We can nonetheless show that the effective dimension $d_{\text{eff}}^t(\gamma)$ always grows over time, since it represents the total amount of directions larger than the regularization γ present in Φ_t and adding a sample will either strengthen existing directions or add a new one. Quantifying or even bounding this increase is much more difficult than in the case of RLSs, and therefore estimating the changes in effective dimension without estimating each RLSs is much more difficult than estimating changes in RLSs (Calandriello et al., 2016).
Using the fact that $\tau_{t,t} \geq \tau_{n,t}$, and that all sampling is performed independently accordingly to the oracle, we can invoke Proposition 2.5 with $p_t = \tau_{t,t}$ and $\overline{q} \geq 4\log(2r/\delta)/\varepsilon^2$ to guarantee that $\mathcal{I}_n$ is (ε, γ)-accurate w.r.t. Φ_n. Unfortunately, to union bound the failure (generating an inaccurate or too large dictionary) probability across n iteration, compared to Oracle-RLS we must choose a larger $\overline{q} \geq 4\log(2n/\delta)/\varepsilon^2 \geq 4\log(2r/\delta)/\varepsilon^2$. This is necessary to guarantee that each of the dictionaries $\mathcal{I}_t$ is (ε, γ)-accurate w.r.t. Φ_t with $1 - \delta/n$ probability, and provide accuracy guarantees for the overall process.

Following a similar path as the one used for Corollary 2.9 in the batch setting, we now need to use Proposition 2.2 to bound the size of the dictionaries $\mathcal{I}_t$. Unfortunately, to apply it we must look at the sum $\sum_{i=1}^{t} p_i = \sum_{i=1}^{t} \tau_{i,i}$ of the probabilities used by SOracle-RLS-NoRem, and it is not clear how this quantity relates to $d_{\text{eff}}^t(\gamma)$. Knowing that $\tau_{t,i} \geq \tau_{t-1,i}$, it is easy to give a lower bound

$$d_{\text{eff}}^t(\gamma) = \sum_{i=1}^{t} \tau_{t,i} = \tau_{t,t} + \sum_{i=1}^{t-1} \tau_{t,i} \leq \tau_{t,t} + \sum_{i=1}^{t-1} \tau_{t-1,i} = \tau_{t,t} + d_{\text{eff}}^{t-1}(\gamma) \leq \sum_{i=1}^{t} \tau_{i,i},$$

which shows that we may pay a price in space complexity when we replace $\tau_{n,i}$ with $\tau_{i,i}$. Finding an upper bound to quantify this price is much more difficult. The simplest case is when all samples are orthogonal. Then $\tau_{1,1} = \tau_{2,1} = \tau_{t,1} = \frac{1}{1+\gamma}$, and the two quantities $\sum_{i=1}^{t} \tau_{i,i} = \sum_{i=1}^{t} \tau_{t,i} = d_{\text{eff}}^t(\gamma)$ coincide. But in the opposite case when all samples are identical $\tau_{1,1} = \frac{1}{1+\gamma}$ while $\tau_{t,1} = \frac{1}{t+\gamma}$, and as a result, $\tau_{1,1}$ contributes to the sum n times more than it should. Nonetheless, remaining in the all-identical sample case, if we sum

[3] see Definition 2.8

across all samples,

$$\sum_{i=1}^{t} \tau_{i,i} = \sum_{i=1}^{t} \frac{1}{i+\gamma} \leq \log(t+\gamma)+1, \qquad d_{\text{eff}}^{t}(\gamma) = \frac{t}{t+\gamma},$$

then one quantity is constant while the other is only $\log(n)$ times larger. Fortunately, we can show that $\log(t)$ is the largest gap that we can incur.

Definition 3.2 The *online* effective dimension of a map $\Phi : \mathbb{R}^t \to \mathcal{H}$ is

$$d_{\text{onl}}^{t}(\gamma) := \sum_{i=1}^{t} \tau_{i,i}(\gamma).$$

We call $d_{\text{onl}}^{t}(\gamma)$ the *online* effective dimension because it is influenced by the order in which we see the samples.

Lemma 3.9 — (Calandriello et al., 2017c). For any map $\Phi : \mathbb{R}^t \to \mathcal{H}$ and any $\gamma > 0$ we have that its online effective dimension $d_{\text{onl}}^{t}(\gamma) = \sum_{i=1}^{t} \tau_{t,t}$ is bounded by

$$d_{\text{onl}}^{t}(\gamma) \leq \log(\text{Det}(\Phi_t^\intercal \Phi_t/\gamma + \mathbf{I}_t)) \leq d_{\text{eff}}^{t}(\gamma)(1+\log(\|\Phi_t^\intercal \Phi_t\|/\gamma+1)) \leq 2d_{\text{eff}}^{t}(\gamma)\log(n/\gamma).$$

The first inequality relates $d_{\text{onl}}^{t}(\gamma)$ to the log-determinant of the matrix $\Phi_t^\intercal \Phi_t$. As we mentioned when we introduced the concept of effective dimension (see Section 2.2.1), this quantity appears in a large number of works on linear models, where it is connected to the maximal mutual information gain in Gaussian processes (Srinivas et al., 2010), or the regret in online optimization (Hazan et al., 2006). Finally, the second inequality shows that in general the complexity of measuring this capacity sequentially, i.e., the ratio between the online and offline sum of the RLS, is only a $\log(n)$ factor (in the worst case). This allows us to prove the following corollary

Corollary 3.10 — (Lemma 3.9, Corollary 2.9). Consider an arbitrary map $\Phi : \mathbb{R}^n \to \mathcal{H}$ with ϕ_i as its i-th column and rank r. Given $\gamma > 0$, let $\mathbf{\Gamma}$ and $\widetilde{\mathbf{\Gamma}}$ be defined according to Definition 2.5, and the RLS τ_i according to Definition 2.7. Let $\mathcal{I}_t$ be the dictionaries generated by SOracle-RLS-NoRem (Algorithm 7). If

$$p_t = \tau_{t,t}, \qquad \overline{q} \geq \frac{4\log(4n/\delta)}{\varepsilon^2},$$

with $\sum_{i=1}^{n} p_i = d_{\text{onl}}(\gamma)$, then for any $0 < \varepsilon < 1$ and w.p.$1-\delta$

Accuracy: Each $\mathcal{I}_t$ is (ε, γ)-accurate w.r.t. Φ_t

Space: the size of each of the dictionaries is bounded by $|\overline{\mathcal{I}_t}| \leq 3\overline{q}d_{\text{onl}}^{t}(\gamma) \leq 6\overline{q}d_{\text{eff}}^{t}(\gamma)\log(n)$

With this result, we have essentially matched the $\widetilde{\mathcal{O}}(d_{\text{eff}}^{n}(\gamma))$ space complexity of oracle batch RLS sampling (i.e., sampling according to $\tau_{n,i}$), up to a small increase in the logarithmic factors (from $\log(r)$ to $\log(n)$) due to the union bound across iterations, and an extra $\log(n)$ factor due to the gap between online and offline effective dimension. At the same time, SOracle-RLS-NoRem can be run in a streaming setting, since the algorithm does not need to know in advance the final RLSs $\tau_{n,i}$ to execute.

Algorithm 8 Kernel online row sampling (KORS)

Input: Regularization ρ, accuracy ε, budget $\overline{q}$

1: Initialize $\mathcal{I}_0 = \emptyset$
2: **for** $t = \{1, \dots, n\}$ **do**
3: receive ϕ_t
4: construct temporary dictionary $\overline{\mathcal{I}}_{t,*} := \mathcal{I}_{t-1} \cup (1, \phi_t)$
5: compute $\widetilde{\tau}_{t,t} = (1-\varepsilon)\phi_i^\intercal(\Phi_t \mathbf{S}_{t,*}\mathbf{S}_{t,*}^\intercal \Phi_t^\intercal + \gamma \mathbf{\Pi}_t)^{-1}\phi_i$ using the Eq. 3.2 estimator
6: compute $\widetilde{p}_t = \widetilde{\tau}_{t,t}$
7: draw $q_t \sim \mathcal{B}(\widetilde{p}_t, \overline{q})$
8: if $q_t \neq 0$, add $\left(\frac{1}{\widetilde{p}_t}\frac{q_t}{\overline{q}}, \phi_t\right)$ to $\mathcal{I}_t$ ▷EXPAND
9: **end for**

3.2.2 Implementing the oracle and guarantees

We can now remove the necessity of an oracle, and introduce KORS, reported in Algorithm 8, our first single-pass RLS sampling algorithm that achieves near-linear runtime and is not based on an oracle. Note that since computing exact RLSs $\tau_{t,t}$ would be too expensive, we will again resort to approximate RLSs $\widetilde{\tau}_{t,t}$ and approximate probabilities $\widetilde{p}_t = \widetilde{\tau}_{t,t}$.
Starting from an empty dictionary $\mathcal{I}_0$, at each time step the algorithm loads sample ϕ_t into memory and combines the new sample $(1, \phi_t)$ to the dictionary $\mathcal{I}_{t-1}$ to constructs the temporary dictionary $\mathcal{I}_{t,*}$. Using the associated selection matrix $\mathbf{S}_{t,*}$, this augmented dictionary can be effectively used to compute $\widetilde{\tau}_{t,t}$ using the Eq. 3.2 RLS estimator,

$$\begin{aligned}\widetilde{\tau}_{t,i} &= (1-\varepsilon)\phi_i^\intercal(\Phi_t \mathbf{S}_{t,*}\mathbf{S}_{t,*}^\intercal \Phi_t^\intercal + \gamma \mathbf{\Pi})^{-1}\phi_i \\ &= \tfrac{1-\varepsilon}{\gamma}(\phi_i^\intercal\phi_i - \phi_i^\intercal \Phi_t \mathbf{S}_{t,*}(\mathbf{S}_{t,*}^\intercal \Phi_t^\intercal \Phi_t \mathbf{S}_{t,*} + \gamma \mathbf{I}_{\mathcal{I}_{t,*}})^{-1}\mathbf{S}_{t,*}^\intercal \Phi_t^\intercal \phi_i). \end{aligned} \tag{3.4}$$

Note that combining two dictionaries $\mathcal{I}$ and $\mathcal{I}'$, each (ε, γ)-accurate w.r.t. to disjoint datasets $\mathcal{D}$ and $\mathcal{D}'$ generates a dictionary that is (ε, γ)-accurate w.r.t. $\mathcal{D} \cup \mathcal{D}'$. Therefore, if $\mathcal{I}_{t-1}$ is (ε, γ)-accurate, combining it with $(1, \phi_t)$ makes $\mathcal{I}_*$ an (ε, γ)-accurate dictionary, because $(1, \phi_t)$ can be seen as a $(0, 0)$-accurate dictionary w.r.t. to ϕ_t.
Taking into account Lemma 3.3, we see that if $\mathcal{I}_*$ is (ε, γ)-accurate, then $\widetilde{\tau}_{t,t}$ is ρ-accurate. This cycle of using an accurate dictionary to estimate accurate approximate RLSs, and accurate approximate RLSs to compute accurate dictionaries will be at the core of the analysis of the algorithm in Theorem 3.11.
After computing the approximate RLS, KORS sets the approximate probability $\widetilde{p}_t = \widetilde{\tau}_{t,t}$ and draws a Binomial $q_t \sim \mathcal{B}(\widetilde{p}_t, \overline{q})$. If $q_t \neq 0$, we choose to EXPAND $\mathcal{I}_{t-1}$ and ϕ_t is added with weight $\frac{1}{\widetilde{p}_t}\frac{q_t}{\overline{q}}$ to the next dictionary $\mathcal{I}_t$. If $q_t = 0$, the sample is not added and will be discarded at the end of the iteration to free space.
Similarly to TWO-STEP-RLS or MANY-STEP-RLS, all rows and columns for which $\mathbf{S}_{t,*}$ is zero (all points outside the temporary dictionary $\mathcal{I}_{t,*}$) do not influence the estimator, so they can be excluded from the computation. Therefore, once a sample ϕ_t is dropped from memory we do not need to re-load it anymore for the rest of the algorithm's execution. This means that we only need to load the dataset once and sequentially, making KORS a single-pass algorithm. This is vital in the streaming setting where we cannot go back and resample a discarded atom.
Moreover, since the difference between $\mathcal{I}_{t-1}$ and $\mathcal{I}_{t-1,*}$ or $\mathcal{I}_t$ is a single atom ϕ_t, $\widetilde{\tau}_{t,i}$ can be efficiently computed in $\mathcal{O}(|\mathcal{I}_{t,*}|^2)$ space and $\mathcal{O}(|\mathcal{I}_{t,*}|^2)$ time using an incremental version of Eq. 3.4 that exploits rank-1 updates. Overall, the algorithm will take $\mathcal{O}(n \max_t |\mathcal{I}_t|^2)$ time and $\mathcal{O}(\max_t |\mathcal{I}_t|^2)$ space to run.

We are now ready to state the main result of this section.

Theorem 3.11 — (Calandriello et al., 2017c). Given parameters $0 < \varepsilon \leq 1$, $0 < \gamma$, $0 < \delta < 1$, let $\rho = \frac{1+\varepsilon}{1-\varepsilon}$ and run KORS (Algorithm 8) with **budget** $\overline{q} \geq 4\log(2n/\delta)/\varepsilon^2$. Then w.p. $1-\delta$,

Accuracy: each $\mathcal{I}_t$ is (ε,γ)-accurate w.r.t. Φ_t and each $\widetilde{\tau}_{t,t}$ is ρ-accurate w.r.t. $\tau_{t,t}$.

Space: the size of each dictionary is bounded by $|\mathcal{I}_t| \leq 3\overline{q}d^t_{\text{onl}}(\gamma) \leq 6d^t_{\text{eff}}(\gamma)\overline{q}\log(2n/\gamma)$.

Time: the algorithm runs in $\widetilde{\mathcal{O}}(d^t_{\text{eff}}(\gamma)^2)$ per-iteration time, and $\widetilde{\mathcal{O}}(nd^n_{\text{eff}}(\gamma)^2)$ overall time.

Passes: the algorithm requires a single pass over Φ

Theorem 3.11 shows that we can construct an (ε,γ)-accurate dictionary containing only $\mathcal{O}(d^n_{\text{eff}}(\gamma)\log^2(n))$ atoms. Although this is a $\mathcal{O}(\log(n))$ factor larger than SOracle-RLS-NoRem and batch methods like Many-Step-RLS or Recursive-RLS, KORS does not require an oracle or multiple passes over the data.
Moreover, unlike Recursive-RLS, it provides accuracy guarantees for all intermediate dictionaries. Therefore, if we choose to terminate the algorithm early, we still have gained an (ε,γ)-accurate dictionary of the data we have processed up to that point.
This is especially important because the size of the dictionary scales with $d^t_{\text{eff}}(\gamma)$, which is not known in advance. Therefore, if during KORS's execution we see that the dictionary is growing too large and decide to terminate it, we can use the intermediate dictionary to study the spectral properties of Φ_t and choose a better regularization γ for future runs.
Finally, note that KORS is a generalization of ORS (online row sampling) by Cohen et al., (2016) from Euclidean to Hilbert spaces. While we drew inspiration from their algorithm to derive KORS (e.g., the estimator in Lemma 3.3), moving to an infinite-dimensional space introduces new difficulties, such us finding a way to compute the estimator, how to replace the dimensionality d with $d^t_{\text{onl}}(\gamma)$ in the algorithm analysis, and how to bound it.

Proof sketch: The first step in the proof is to reformulate the event failure $\|\mathbf{\Gamma}_t - \widetilde{\mathbf{\Gamma}}_t\| \geq \varepsilon$ using the equivalent finite-dimensional $\|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\| \geq \varepsilon$ failure condition. Notice that the matrices $\mathbf{P}_t - \widetilde{\mathbf{P}}_t$ change size over time. To avoid this, we carefully decompose the failure event across the whole process into separate failure events for each matrix $\|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\| \geq \varepsilon$, and construct a dedicated random process $\mathbf{Y}_t$ that models how KORS generates the dictionary $\widetilde{\mathbf{P}}_t$. These processes are sequential in nature and their steps are not i.i.d., since $\widetilde{\tau}_{t,t}$, and therefore $\widetilde{\mathbf{P}}_t$, depends on $\widetilde{\tau}_{t-1,t-1}$, q_{t-1} and $\widetilde{\mathbf{P}}_{t-1}$. As a consequence we cannot use concentrations for i.i.d. r.v. such as Proposition 2.4. Instead, following an approach introduced in (Cohen et al., 2016), we use Freedman's inequality and a deterministic bound on the predictable quadratic variation (variance) $\mathbf{W}_t$ of the process to bound each failure probability.
Note also that KORS can be seen as a special case of SQUEAK, an algorithm that will be introduced in the next section. Therefore, we will only provide a proof for SQUEAK, highlighting when necessary how things can be specialized to address KORS. For a whole proof dedicated solely to KORS, we direct the reader to Calandriello et al., (2017c).

3.3 Sequential RLS Sampling with removal

KORS has two clear drawbacks. First, over-sampling according to $\tau_{t,t}$ rather than $\tau_{n,t}$ increases the dictionary size by a $\log(n)$ factor. This is due to the fact that without a

Algorithm 9 Streaming Oracle RLS sampling with removal (SOracle-RLS-Rem)

Input: Regularization ρ, accuracy ε, budget $\overline{q}$

Initialize $\mathcal{I}_0 = \emptyset$
for $t = \{1, \dots, n\}$ **do**
  receive ϕ_t
  add $(1, \overline{q}, \phi_t)$ to $\mathcal{I}_{t-1}$ ▷Expand
  for $s = \{i \in [t] : q_{t-1,i} \neq 0\}$ **do**
    compute $p_{t,s} = \tau_{t,s}$ using an oracle limited to $\mathcal{D}_t$
    draw $q_{t,s} \sim \mathcal{B}\left(p_{t,s}/p_{t-1,s}, q_{t-1,s}\right)$
    if $q_{t,s} \neq 0$, update $(p_{t,s}, q_{t,s}, \phi_s)$ in $\mathcal{I}_t$
    if $q_{t,s} = 0$, remove $(p_{t-1,s}, q_{t-1,s}, \phi_s)$ in $\mathcal{I}_t$ ▷Shrink
  end for
end for

way to remove samples once we notice that they are not as relevant as we thought, we cannot recover from these overestimation mistakes. Second, the algorithm cannot be easily parallelized, due to the sequential nature of the RLSs estimation. In this section, we will show how adding the possibility of removing samples from the dictionary to the sampling process can cope with both drawbacks.

3.3.1 Oracle RLS sampling with removal

We begin with a simple oracle algorithm to illustrate the problem of RLS sampling with removal.

From the analysis of Lemma 3.8 we know that $\tau_{t,t} \geq \tau_{t+1,t}$. Therefore, sampling q_t proportionally to $\tau_{t,t}$ in SOracle-RLS-NoRem is an overestimate of the current $\tau_{t',t}$ for all step $t' > t$. To correct this, a simple approach is to continue updating $q_t := q_{t,t}$ over time, i.e., also for samples already included in $\mathcal{I}_{t-1}$, and generate $q_{t+1,t}, \dots q_{n,t}$ proportional to the correct $\tau_{t+1,t}, \dots, \tau_{n,t}$. We can formalize this using the following *adaptive importance sampling* chain

$$
\begin{aligned}
&q_{t,t} \sim \mathcal{B}\left(p_{t,t}, \overline{q}\right), \\
&q_{t+1,t} | q_{t,t} \sim \mathcal{B}\left(p_{t+1,t}/p_{t,t}, q_{t,t}\right), \\
&q_{t+2,t} | q_{t+1,t} \sim \mathcal{B}\left(p_{t+2,t}/p_{t+1,t}, q_{t+1,t}\right), \\
&\dots, \\
&q_{n,t} | q_{n-1,t} \sim \mathcal{B}\left(p_{n,t}/p_{n-1,t}, q_{n-1,t}\right),
\end{aligned}
$$

where the conditioning $q_{t+1,t}|q_{t,t}$ is necessary to make $\mathcal{B}\left(p_{t+1,t}/p_{t,t}, q_{t,t}\right)$ a well defined Binomial.
Before continuing, we also extend the dictionary notation $\left\{\left(\frac{1}{p_{t,s}}\frac{q_{t,s}}{\overline{q}}, \phi_s\right)\right\}$ to explicitly keep track of the probabilities and the number of copies $\{(p_{t,s}, q_{t,s}, \phi_s)\}$, where the weights $\frac{1}{p_{t,s}}\frac{q_{t,s}}{\overline{q}}$ can be recomputed from the dictionary when necessary.

We can now introduce our oracle sequential RLSs sampling algorithm with removal, reported in Algorithm 9, where "with removal" highlights that now atoms previously added to the dictionary might be removed by the adaptive importance sampling process.
At each step, the algorithm receives a new point ϕ_t, but unlike SOracle-RLS-NoRem it always Expand the previous dictionary $\mathcal{I}_{t-1}$ to include it with probability and number

of copies $(p_{t-1,t} = 1, q_{t-1,t} = \overline{q}, \phi_t)$. Remember that our convention is to set $\tau_{t-1,t} = 1$ for samples that we did not see yet. Therefore, we are still initializing our importance sampling chain with a draw from a Binomial $\mathcal{B}(p_{t-1,t}, \overline{q}) = \mathcal{B}(\tau_{t-1,t}, \overline{q}) = \mathcal{B}(1, \overline{q})$ whose outcome is always $\overline{q}$.

Since SOracle-RLS-Rem, unlike SOracle-RLS-NoRem, always Expand the dictionary at each step, the size of the dictionary and our memory usage always increases. In order to compensate this and decrease the size of the next dictionary $\mathcal{I}_t$, the algorithm iterates over all atoms still in the dictionary $\mathcal{I}_{t-1}$ (all atoms with a non-zero $q_{t-1,i}$) to adjust their number of copies $q_{t,i}$ using importance sampling.

The Shrink step draws a sample from the Binomial $\mathcal{B}(p_{t,i}/p_{t-1,i}, q_{t-1,i})$, where the fact that $p_{t,i} = \tau_{t,i} \geq \tau_{t-1,i} = p_{t-1,i}$ ensures that the Binomial probability is well defined (i.e., $p_{t,i}/p_{t-1,i} \leq 1$). This resampling step *tracks* the changes in the RLS and constructs a new dictionary $\mathcal{I}_t$, which is *as if* it was created from scratch using all the correct RLS $\tau_{t,i}$ at time t (with high probability).

We can now clarify the relationship between the role of Expand in SOracle-RLS-Rem and the previous SOracle-RLS-NoRem. Since ϕ_t was added with $p_{t-1,t} = 1$ and $q_{t-1,q}$, the Shrink step inserts it in the new dictionary $\mathcal{I}_t$ by drawing from $\mathcal{B}(p_{t,t}/p_{t-1,t}, q_{t-1,t}) = \mathcal{B}(\tau_{t,t}/1, \overline{q})$, which makes the Expand-Shrink combination in SOracle-RLS-Rem equivalent to the Expand of SOracle-RLS-NoRem.

In other words, the new element ϕ_t is kept in the dictionary ($q_{t,t} \neq 0$) only if its RLS $p_{t,t} = \tau_{t,t}$ is high. At the same time, the number of copies of the older atoms already in $\mathcal{I}_{t-1}$ are stochastically reduced using Shrink to reflect the reductions of the RLSs. The lower $p_{t,i}$ is w.r.t. $p_{t-1,i}$, the lower the number of copies $q_{t,i}$ will be w.r.t. $q_{t-1,i}$. As the probability $p_{t,i}$ continues to decrease over time, $q_{t,i}$ may become zero, and the atom ϕ_i is completely dropped from the dictionary (by setting its weight to zero).

We show that using this adaptive importance sampling $q_{n,t}$ follows a Binomial distribution $\mathcal{B}(\tau_{n,t}, \overline{q})$. To see this, rewrite $q_{t,i} = \sum_{j=1}^{\overline{q}} z_{t,i,j}$ as the sum of $\overline{q}$ individual coin flip chains $z_{t,t,j}$. The j-th coin flip chain evolves following the rule

$$z_{t,i,j} \sim \begin{cases} Ber\left(\frac{p_{t,i}}{p_{t-1,i}}\right) & \text{if } z_{t-1,i,j} = 1, \\ 0 & \text{otherwise,} \end{cases}$$

or in other words we continue flipping coins (drawing from a Bernoulli) only if the previous coin flip succeeded, and set the rest of the chain to 0 at the first failure. As a consequence of this definition, we have

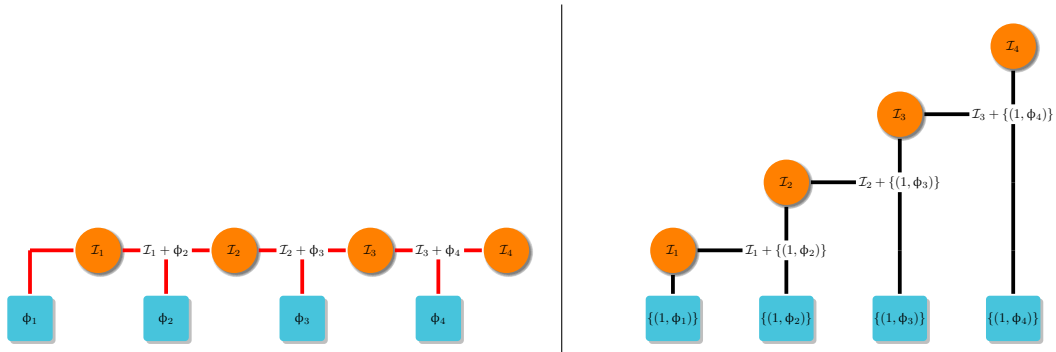
$$\begin{aligned} \mathbb{P}(z_{t+1,t,j} = 1) &= \mathbb{P}(z_{t+1,t,j} = 1 | z_{t,t,j} = 1)\mathbb{P}(z_{t,t,j} = 1) + \mathbb{P}(z_{t+1,t,j} = 1 | z_{t,t,j} = 0)\mathbb{P}(z_{t,t,j} = 0) \\ &= \frac{p_{t+1,t}}{p_{t,t}} p_{t,t} + \underbrace{\mathbb{P}(z_{t+1,t,j} = 1 | z_{t,t,j} = 0)}_{0} \quad \mathbb{P}(z_{t,t,j} = 0), \end{aligned}$$

and iterating

$$\begin{aligned} \mathbb{P}(z_{n,t,j} = 1) &= \mathbb{P}(z_{t,t,j} = 1)\mathbb{P}(z_{t+1,t,j} = 1 | z_{t,t,j} = 1) \ldots \mathbb{P}(z_{n,t,j} = 1 | z_{n-1,t,j} = 1) \\ &= p_{t,t} \frac{p_{t+1,t}}{p_{t,t}} \cdots \frac{p_{n,t}}{p_{n-1,t}} = \cancel{p_{t,t}} \frac{\cancel{p_{t+1,t}}}{\cancel{p_{t,t}}} \cdots \frac{\cancel{p_{n-1,t}}}{\cancel{p_{n-1,t}}} p_{n,t}. \end{aligned}$$

Since $q_{n,t} = \sum_{j=1}^{\overline{q}} z_{n,t,j}$ is the sum of $\overline{q}$ Bernoullis with probabilities $p_{n,t} = \tau_{n,t}$, it is by definition a Binomial $\mathcal{B}(\tau_{n,t}, \overline{q})$.

Note that, since $q_{t,i}$ depends on $q_{t-1,i}$, to study the dictionaries $\mathcal{I}_t$ we would in general need to use results for random processes. Fortunately, we know that each $q_{t,i}$ is also



(a) Expand-Shrink cycle, chain view. (b) Expand-Shrink cycle, tree view.

Figure 3.1: Different views of the Expand-Shrink cycle.

unconditionally distributed as $\mathcal{B}(p_{t,i}, \overline{q})$. Combining this with the fact that the probabilities $p_{t,i} = \tau_{t,i}$ are provided by an oracle, and the fact that the RLSs $p_{t,i} = \tau_{t,i}$ are decided by the data and not the random process, we have that all of the $q_{t,i}$ are appropriately distributed independent Binomials, and we can simply invoke Proposition 2.5 and a union bound over n steps to guarantee the following.

Corollary 3.12 — Lemma 2.10, Proposition 2.5, Proposition 2.13. Consider an arbitrary map $\Phi : \mathbb{R}^n \to \mathcal{H}$ with ϕ_i as its i-th column and rank r. Given $\gamma > 0$, let $\mathbf{\Gamma}$ and $\widetilde{\mathbf{\Gamma}}$ be defined according to Definition 2.5, and the RLS τ_i according to Definition 2.7. Let $\mathcal{I}_t$ be the dictionaries generated by SOracle-RLS-Rem. If

$$p_{t,t} = \tau_{t,t}, \qquad \overline{q} \geq \frac{4\log(4n/\delta)}{\varepsilon^2},$$

with $\sum_{i=1}^n p_{t,i} = d_{\text{eff}}^t(\gamma)$, then for any $0 < \varepsilon < 1$ and w.p.$1 - \delta$

Accuracy: Each $\mathcal{I}_t$ is (ε, γ)-accurate w.r.t. Φ_t

Space: the size of each of the dictionaries is bounded by $|\overline{\mathcal{I}_t}| \leq 3\overline{q} d_{\text{eff}}^t(\gamma)$

The space and accuracy of oracle sequential RLS sampling with removal (Corollary 3.12) is almost identical to the space and accuracy of oracle batch RLS sampling (Corollary 2.9), with the only difference that we are forced to increase again our $\mathcal{O}(\log(r))$ dependency to $\mathcal{O}(\log(n))$ to balance a union bound over n steps. This is not surprising since SOracle-RLS-Rem is equivalent to running Oracle-RLS over the sequence of maps Φ_t, but paying attention to do so in a single pass over the dataset, thanks to the sequential adaptive importance sampling rather than repeatedly drawing fresh Binomials. Nonetheless, it shows that it is possible (given the oracle) to construct dictionaries as small and accurate as in the batch setting, but using only a single pass on the data.

Before showing how to implement the oracle, we first show how a simple modification of SOracle-RLS-Rem leads to an algorithm amenable to parallel and distributed computing. We begin by highlighting (Fig. 3.1(a)) that SOracle-RLS-Rem can be seen as a chain of n steps that at each step t combines an (ε, γ)-accurate dictionary $\mathcal{I}_{t-1}$ and a new sample ϕ_t to construct a new dictionary $\mathcal{I}_t$.
Although adding an extra Shrink pass improved the space complexity of the algorithm, this is still the same sequential approach of KORS of continuously combining an old dictionary

Algorithm 10 SeQUential Approximation for Kernel sampling (SQUEAK)

Input: Dataset $\mathcal{D}$, parameters $k, \gamma, \varepsilon, \delta$
Output: $\mathcal{I}_{\mathcal{D}}$

1: Partition $\mathcal{D}$ into k disjoint sub-datasets $\mathcal{D}_i$
2: Initialize $\mathcal{I}_{\mathcal{D}_i} = \{(\widetilde{p}_{0,j} = 1, q_{0,j} = \overline{q}, \phi_j) : j \in \mathcal{D}_i\}$
3: Build set $\mathcal{S}_1 = \{\mathcal{I}_{\mathcal{D}_i}\}_{i=1}^k$
4: **for** $h = 1, \dots, k-1$ **do**
5: Pick two dictionaries $\mathcal{I}_{\mathcal{D}}, \mathcal{I}_{\mathcal{D}'}$ from $\mathcal{S}_h$
6: Set $\mathcal{D}_* = \mathcal{D} \cup \mathcal{D}'$
7: $\mathcal{I}_* = \mathcal{I}_{\mathcal{D}} \cup \mathcal{I}_{\mathcal{D}'}$ ▷EXPAND
8: **for all** $(\widetilde{p}_i, q_i, \phi_i) \in \mathcal{I}_*$ **do**
9: Compute $\widetilde{\tau}_i = (1-\varepsilon)\phi_i^\intercal(\Phi_{\mathcal{D}_*}\mathbf{S}_*\mathbf{S}_*^\intercal\Phi_{\mathcal{D}_*}^\intercal + (1+\varepsilon)\gamma\mathbf{\Pi}_{\mathcal{D}_*})^{-1}\phi_i$
10: Set $\widetilde{p}_{i,*} = \min\{\widetilde{\tau}_i, \widetilde{p}_i\}$
11: Set $q_{i,*} \sim \mathcal{B}(\widetilde{p}_{i,*}/\widetilde{p}_i, q_i)$
12: If $q_{i,*} \neq 0$, update $(\widetilde{p}_{i,*}, q_{i,*}, \phi_i) \leftarrow \cancel{(\widetilde{p}_i, q_i, \phi_i)}$ in $\mathcal{I}_*$
13: If $q_{i,*} = 0$, remove $(\widetilde{p}_i, q_i, \phi_i)$ from $\mathcal{I}_*$ ▷SHRINK
14: **end for**
15: Place $\mathcal{I}_{\mathcal{D}\cup\mathcal{D}'} := \mathcal{I}_*$ back into $\mathcal{S}_{h+1}$
16: **end for**
17: Return $\mathcal{I}_{\mathcal{D}}$, the last dictionary in $\mathcal{S}_k$

DICT-MERGE

and new data, as the new data arrives. But in the discussion of Eq. 3.4, we previously argued that what KORS is actually doing is combining two (ε, γ)-accurate dictionaries, since ϕ_t can be seen as a perfect $(0, 0)$-accurate dictionary $\{(1, \phi_t)\}$ w.r.t. the singleton dataset $\{\phi_t\}$.
If we take this generalized approach to SORACLE-RLS-REM, we see that we can generalize the chain structure of Fig. 3.1(a) into the tree structure of Fig. 3.1(b), where at each step we merge the previous dictionary $\mathcal{I}_{t-1}$ with another dictionary $\{(1, \phi_i)\}$. It is therefore reasonable to imagine an even more general algorithm that follows an arbitrary merge tree structure, and recursively performs dictionary merges were the two dictionaries being merged can contain either fresh data or the result of a previous merge.
In the next section, we introduce a new approximate sequential RLS sampling algorithm, based on this dictionary merging strategy, that is amenable to parallelization, and a more sophisticated analysis that shows that it achieves almost the same space and accuracy as SORACLE-RLS-REM without the need of an oracle.

3.3.2 Implementing the oracle and guarantees

Replacing the sequential merges of dictionaries and a single sample in SORACLE-RLS-REM with general dictionary-dictionary merges allows us to parallelize and distribute the computation of the dictionary $\mathcal{I}_n$ over multiple machines, thus reducing even further its time complexity.
Beside the computational advantage, a distributed architecture is needed as soon as the input dimension d and the number of points n is so large that having the dataset on a single machine is impossible. Furthermore, distributed processing can reduce contention on bottleneck data sources such as databases or network connections.

SQUEAK (Algorithm 10) partitions $\mathcal{D}$ over multiple machines and the (small) dictionaries

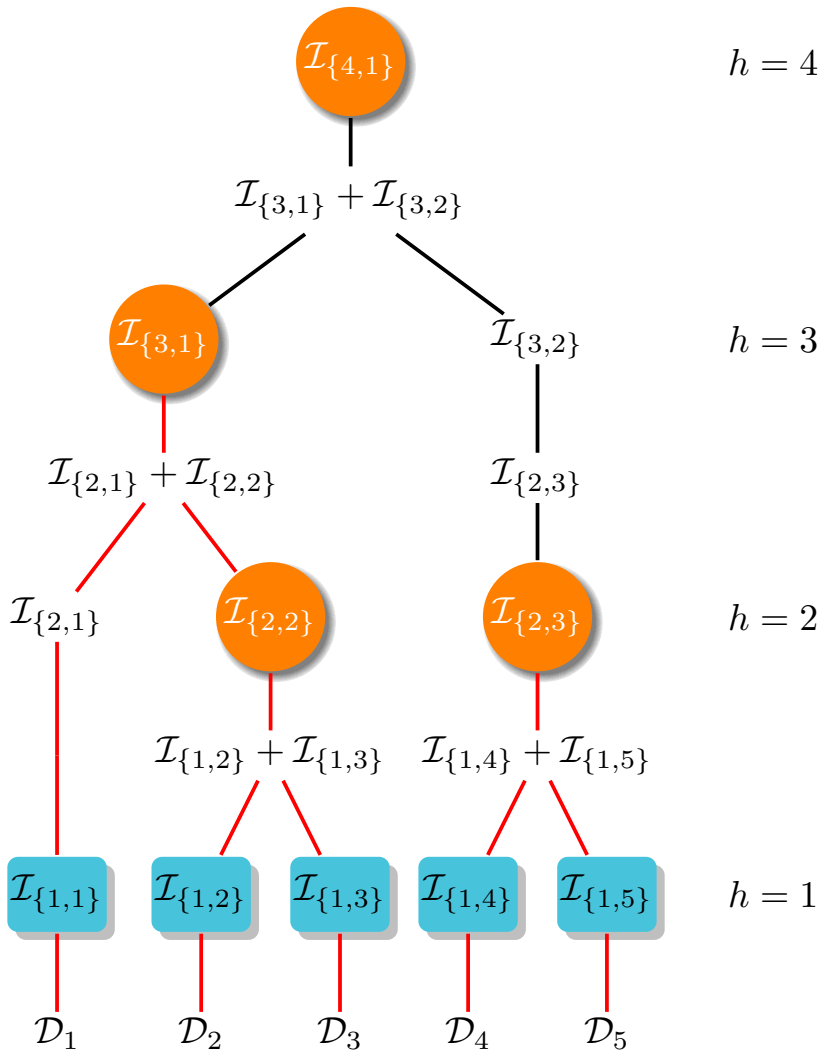


Figure 3.2: Merge tree for Alg. 10 with an arbitrary partitioning and merging scheme.

that are generated from different portions of the dataset are integrated in a hierarchical way. Since we will now be dealing with subsets of the dataset $\mathcal{D}$, we introduce the general notation $\Phi_{\mathcal{D}}$ to indicate the map containing all samples in $\mathcal{D}$. Similarly if we extract a subset $\mathcal{D}'$ out of $\mathcal{D}$, $\Phi_{\mathcal{D}'}$ contains only the samples in $\mathcal{D}'$.
Together with $\Phi_{\mathcal{D}'}$ we also introduce $\mathbf{\Pi}_{\mathcal{D}'}$ as the projection on the span of the samples in the subset $\mathcal{D}'$, the effective dimension $d_{\text{eff}}^{\mathcal{D}'}(\gamma)$ of $\Phi_{\mathcal{D}'}$, and the RLS $\tau_{\mathcal{D}',i}(\gamma)$ of sample $\phi_i \in \mathcal{D}'$ w.r.t. only the other samples in $\mathcal{D}'$.
The initial dataset is partitioned over k disjoint sub-datasets $\mathcal{D}_i$ with $i = 1, \ldots, k$ and k dictionaries $\mathcal{I}_{\mathcal{D}_i} = \{(\widetilde{p}_{0,j} = 1, q_{0,j} = \overline{q}, \phi_j) : j \in \mathcal{D}_i\}$ are initialized simply by placing all samples in $\mathcal{D}_i$ into $\mathcal{I}$ with weight 1 and multiplicity $\overline{q}$. Alternatively, if the datasets $\mathcal{D}_i$ are too large to fit in memory, we can run KORS on each of them separately to generate the initial dictionaries.
The dictionaries $\mathcal{I}_{\mathcal{D}_i}$ are added to a dictionary collection $\mathcal{S}$, and following any predefined binary *merge tree* as in Fig. 3.2, these dictionaries are progressively merged together.
To describe the merge tree, we introduce additional notation: we index each node in the merge tree by its height h and position l, where the root (top) of the tree is at height $k-1$, and the leaves (original datasets) are at height 1. We denote the dictionary associated to node $\{h, l\}$ by $\mathcal{I}_{\{h,l\}}$ and the collection of all dictionaries available at height h of the merge

tree correspond to $\mathcal{S}_h = \{\mathcal{I}_{\{h,l\}}\}$. We also use $\Phi_{\{h,l\}}$ to refer to the map constructed from the datasets $\mathcal{D}_{\{h,l\}}$, which contains all points present in the leaves reachable from node $\{h,l\}$. For instance in Fig. 3.2, node $\{3,1\}$ is associated with $\mathcal{I}_{\{3,1\}}$, which is (as we will prove) an (ε,γ)-accurate dictionary w.r.t. to the map $\Phi_{\{3,1\}}$ constructed from the dataset $\mathcal{D}_{\{3,1\}}$. $\mathcal{D}_{\{3,1\}}$ contains $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$ (descendent nodes are highlighted in red) and it has dimension $(|\mathcal{D}_1|+|\mathcal{D}_2|+|\mathcal{D}_3|)$.
At each iteration of SQUEAK, one node in the tree is scheduled to be merged. Given the two input dictionaries $\mathcal{I}_{\mathcal{D}}$ and $\mathcal{I}_{\mathcal{D}'}$ contained in the two children of the node, we combine them into a single dictionary $\mathcal{I}_*$ (the equivalent of the Expand phase in KORS) and then we Shrink the merged dictionaries to create an updated dictionary $\mathcal{I}_{\mathcal{D}\cup\mathcal{D}'}$, which is placed back in the dictionary collection $\mathcal{S}$. Overall we call this dictionary merging operation Dict-Merge.
To perform the Shrink part of Dict-Merge, we need to estimate how the RLSs have changed when combining $\mathcal{D}$ and $\mathcal{D}'$. Unlike in KORS, Dict-Merge is run on the union of two distinct dictionaries rather than one dictionary and a new single point, and must track the changes following the merge of two datasets rather than a dataset and a new single point. As a result, we need to derive the "distributed" counterparts of Lemma 3.8 and Lemma 3.3 to analyze the behavior of the RLSs and the quality of the estimator used in the algorithm.

Lemma 3.13 — (Calandriello et al., 2017a). Given two disjoint datasets $\mathcal{D},\mathcal{D}'$, for every $i \in \mathcal{D}\cup\mathcal{D}'$, $\tau_{\mathcal{D},i} \geq \tau_{\mathcal{D}\cup\mathcal{D}',i}$ and

$$2d_{\text{eff}}^{\mathcal{D}\cup\mathcal{D}'}(\gamma) \geq d_{\text{eff}}^{\mathcal{D}}(\gamma) + d_{\text{eff}}^{\mathcal{D}'}(\gamma) \geq d_{\text{eff}}^{\mathcal{D}\cup\mathcal{D}'}(\gamma).$$

While in KORS we were merging an (ε,γ)-accurate dictionary $\mathcal{I}_t$ and a new point, which is equivalent to a perfect, $(0,0)$-accurate dictionary, in SQUEAK both dictionaries used in a merge are only (ε,γ)-accurate. We need to slightly adjust our RLS estimator to compensate for this loss in accuracy.

Lemma 3.14 — (Calandriello et al., 2017a). Given two disjoint datasets $\mathcal{D},\mathcal{D}'$, and two (ε,γ)-accurate dictionaries $\mathcal{I}_{\mathcal{D}}$, $\mathcal{I}_{\mathcal{D}'}$, let $\mathcal{I}_* = \mathcal{I}_{\mathcal{D}}\cup\mathcal{I}_{\mathcal{D}'}$ and $\mathbf{S}_*$ be the associated selection matrix. Let $\Phi_{\mathcal{D}_*}$ be the map constructed using $\mathcal{D}_* = \mathcal{D}\cup\mathcal{D}'$, and $\tau_{\mathcal{D}\cup\mathcal{D}',i}$ the RLS of ϕ_i w.r.t. $\mathcal{D}_*$. Compute the approximate RLSs $\widetilde{\tau}_i$ as

$$\begin{aligned}\widetilde{\tau}_i &= \widetilde{\tau}_{\mathcal{D}\cup\mathcal{D}',i}\\ &= (1-\varepsilon)\phi_i^\intercal(\Phi_{\mathcal{D}_*}\mathbf{S}_*\mathbf{S}_*^\intercal\Phi_{\mathcal{D}_*}^\intercal + (1+\varepsilon)\gamma\mathbf{\Pi}_{\mathcal{D}_*})^{-1}\phi_i\\ &= \tfrac{1-\varepsilon}{(1+\varepsilon)\gamma}(\phi_i^\intercal\phi_i - \phi_i^\intercal\Phi_{\mathcal{D}_*}\mathbf{S}_*(\mathbf{S}_*{}^\intercal\Phi_{\mathcal{D}_*}^\intercal\Phi_{\mathcal{D}_*}\mathbf{S}_* + (1+\varepsilon)\gamma\mathbf{I}_{\mathcal{I}_*})^{-1}\mathbf{S}_*{}^\intercal\Phi_{\mathcal{D}_*}^\intercal\phi_i), \qquad 3.5\end{aligned}$$

Then for all ϕ_i in $\mathcal{D}_*$, the estimator in Eq. 3.5 is ρ-accurate w.r.t. $\tau_{\mathcal{D}\cup\mathcal{D}',i}(\gamma)$, with $\rho = (1+3\varepsilon)/(1-\varepsilon)$. Moreover, $\widetilde{\tau}_i$ can be computed in $\mathcal{O}(|\mathcal{I}|^3)$ time and space using only ϕ_i and atoms present in $\mathcal{I}_*$ itself.

We denoted again with $\mathbf{I}_{\mathcal{I}_*}$ the $\mathbb{R}^{|\mathcal{D}_*|\times|\mathcal{D}_*|}$ diagonal matrix with $[\mathbf{I}_{\mathcal{I}_*}]_{i,i} = 1$ if $q_i \neq 0$ in $\mathcal{I}_*$ and 0 otherwise. Notice that Eq. 3.5 is (except for the specific dictionaries involved) identical to Eq. 3.3 that we used in the multi-step batch RLS sampling Algorithm 5 when we had a single $(\varepsilon,2\gamma)$-accurate dictionary and we wanted to estimate γ-RLSs.
We can now make a comparison: in our first proposed estimator Eq. 3.2 we had access to a dictionary with γ error and a new sample with 0 error and achieved $\frac{1+\varepsilon}{1-\varepsilon}$-accuracy, in

Eq. 3.3 we had a single dictionary with double error 2γ and a new sample with 0 error and achieved $\frac{1+3\varepsilon}{1-\varepsilon}$-accuracy here we have two dictionaries with γ error and achieve the same $\frac{1+3\varepsilon}{1-\varepsilon}$-accuracy. This shows the flexibility of Eq. 3.2. For example, it is easy to generalize both Eq. 3.5 and Eq. 3.3 to be able to handle k-way merges of (ε, γ)-accurate dictionaries, or the use of a single $(\varepsilon, k\gamma)$-accurate dictionary and achieve $\frac{1+(kn-1)\varepsilon}{1-\varepsilon}$-accuracy.
To conclude the RLSs estimation step, we also take the minimum $\widetilde{p}_{i,*} = \min\{\widetilde{\tau}_i, \widetilde{p}_i\}$, or in other words the minimum between the current RLS estimate and the one at the lower level of the tree. This is necessary to guarantee that the following importance sampling step $\mathcal{B}(\widetilde{p}_{i,*}/\widetilde{p}_i, q_i)$ remains well defined, and the estimator is not ruined since taking the minimum of two ρ-accurate approximate RLSs w.r.t. to a growing dataset always preserves ρ-accuracy.

Lemma 3.15 Given two approximate RLSs $\widetilde{\tau}_{\mathcal{D}\cup\mathcal{D}',i}$ and $\widetilde{\tau}_{\mathcal{D},i}$, ρ-accurate RLS w.r.t. $\tau_{\mathcal{D}\cup\mathcal{D}',i}$ and $\tau_{\mathcal{D},i}$, the quantity $\min\left\{\widetilde{\tau}_{\mathcal{D}\cup\mathcal{D}',i},\ \widetilde{\tau}_{\mathcal{D},i}\right\}$ is also an ρ-accurate RLS w.r.t. $\tau_{\mathcal{D}\cup\mathcal{D}',i}$.

Since each merge takes as input two dictionaries and returns one, starting from k subdatasets $\mathcal{D}_i$, after k merges we will be left with a single dictionary $\mathcal{I}_{\mathcal{D}}$ that approximates the whole dataset. Note that Dict-Merge only takes the two dictionaries as input and does not require any information on the dictionaries in the rest of the tree. Therefore, we do not need to run the k merges sequentially, but separate branches can be run simultaneously on different machines, and only the resulting (small) dictionary needs to be propagated to the parent node for the future Dict-Merge.

We can now show that the final dictionary $\mathcal{I}_{\mathcal{D}}$ is small and (ε, γ)-accurate w.r.t. $\Phi_{\mathcal{D}}$.

Theorem 3.16 Given parameters $0 < \varepsilon \leq 1$, $0 < \gamma$, $0 < \delta < 1$, an arbitrary dataset $\mathcal{D}$, and a arbitrary merge tree structure of height k, let $\rho = \frac{1+3\varepsilon}{1-\varepsilon}$ and run Algorithm 10 with **budget** $\overline{q} = \frac{26\rho\log(3n/\delta)}{\varepsilon^2}$. Then w.p. $1-\delta$,

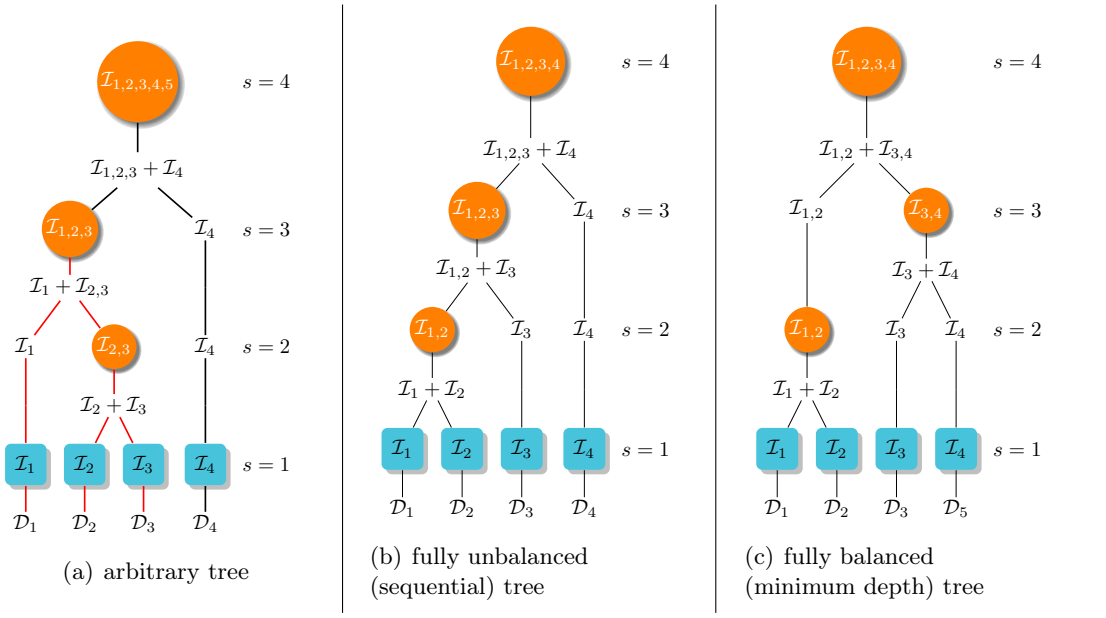
Accuracy: each dictionary $\mathcal{I}_{\{h,l\}}$ is (ε, γ)-accurate w.r.t. $\Phi_{\{h,l\}}$, and each estimate $\widetilde{\tau}_{\mathcal{D},i}$ is ρ-accurate w.r.t. $\tau_{\mathcal{D},i}$.

Space: the size of each dictionary is bounded by $|\mathcal{I}_{\{h,l\}}| \leq 3\overline{q}d_{\text{eff}}^{\mathcal{D}_{\{h,l\}}}(\gamma)$, the overall space requirements of the algorithm depend on the exact shape of the merge tree.

Time: the algorithm runs in $\widetilde{\mathcal{O}}(d_{\text{eff}}^{\mathcal{D}}(\gamma)^3)$ per-merge time, the overall time requirements of the algorithm depend on the exact shape of the merge tree.

Passes: the algorithm requires a single pass over Φ

Theorem 4.7 gives approximation and space guarantees for *every* node of the tree. In other words, it guarantees that each intermediate dictionary processed by SQUEAK is both an (ε, γ)-accurate approximation of the datasets used to generate it, and requires a small space proportional to the effective dimension of the same dataset.
From an accuracy and space perspective, up to a small ρ constant factor SQUEAK provides exactly the same guarantees as the original batch oracle Nyström sampling from Oracle-RLS, without knowing the final RLSs in advance and without multiple passes over the data.
Analysing the runtime space and computational complexity of SQUEAK is however more complex, since the order and arguments of the Dict-Merge operations are determined by the merge tree. We distinguish between the time and work complexity of a tree by defining



(a) arbitrary tree

(b) fully unbalanced (sequential) tree

(c) fully balanced (minimum depth) tree

Figure 3.3: Merge trees for Algorithm 10.

the time complexity as the amount of time necessary to compute the final solution, and the work complexity as the total amount of operations carried out by all machines in the tree in order to compute the final solution.

We consider two special cases, reported in Fig. 3.3, a fully balanced tree (all inner nodes have either two inner nodes as children or two leaves), and a fully unbalanced tree (all inner nodes have exactly one inner node and one leaf as children). For both cases, we consider trees where each leaf dataset contains a single point $\mathcal{D}_i = \{\phi_i\}$, and the tree will contain n merges.

In the fully unbalanced tree, we always merge the current dictionary with a new dataset (a single new point) and no Dict-Merge operation can be carried out in parallel. Unsurprisingly, the sequential algorithm induced by this merge tree is strictly equivalent to sequential RLSs sampling with removal, i.e., SOracle-RLS-Rem without the need of an oracle. The runtime is dominated by the n matrix inversions used to compute Eq. 3.5, each requiring $\mathcal{O}(|\mathcal{I}_{\{h,l\}}|^3)$ time, and by the RLSs estimation also requiring $\mathcal{O}(|\mathcal{I}_{\{h,l\}}|^3)$ time. Therefore, computing a solution in the fully unbalanced tree takes $\mathcal{O}(n\big(\max_{h,l}|\mathcal{I}_{\{h,l\}}|\big)^3) \leq \mathcal{O}(nd_{\text{eff}}^{\mathcal{D}}(\gamma)^3\overline{q}^3)$ time with a total work that is also $\mathcal{O}(nd_{\text{eff}}^{\mathcal{D}}(\gamma)^3\overline{q}^3)$.

On the opposite end, in the fully balanced tree we need to invert multiple $d_{\text{eff}}^{\mathcal{D}_{\{h,l\}}}(\gamma)$ dimensional matrices in parallel at each layer of the tree for a total of $\log(n)$ layers. Bounding all $d_{\text{eff}}^{\mathcal{D}_{\{h,l\}}}(\gamma)$ with $d_{\text{eff}}^{\mathcal{D}}(\gamma)$, gives a complexity for computing the final solution of $\mathcal{O}(\log(n)\overline{q}^3 d_{\text{eff}}^{\mathcal{D}}(\gamma)^3)$ time, with a huge improvement on the unbalanced tree. This is the first result that not only breaks the quadratic time barrier of previous methods, but can extract enough parallelism to break the *linear* barrier, running in *near-constant* time in n. Surprisingly, the total work is only twice $\mathcal{O}(n\overline{q}^3 d_{\text{eff}}^{\mathcal{D}}(\gamma)^3)$, since at each layer h we perform $n/2^h$ inversions (on $n/2^h$ machines), and the sum across all layers is $\sum_{h=1}^{\log(n)} n/2^h \leq 2n$. Therefore, we can compute a solution in a much shorter time than in the sequential sampling case, with a comparable amount of work, but at the expense of requiring much more memory across multiple machines, since at layer h, the sum $\sum_{l=1}^{|\mathcal{S}_h|} d_{\text{eff}}^{\mathcal{D}_{\{h,l\}}}(\gamma)$ can be much larger than $d_{\text{eff}}^{\mathcal{D}}(\gamma)$. Nonetheless, this is partly alleviated by the fact that each node $\{h,l\}$ locally

requires only $d_{\text{eff}}^{\mathcal{D}_{\{h,l\}}}(\gamma)^2 \leq d_{\text{eff}}^{\mathcal{D}}(\gamma)^2$ memory.

Finally, note that independently from this thesis, Kyng et al., (2017) propose an algorithm that is almost equivalent to SQUEAK in its fully sequential variant. Their work focuses on the specific case of sequential graph sparsification (i.e., when the atoms ϕ_i are edges in a graph), and therefore many choices, such as how to estimate RLSs and how to parallelize the algorithm, are tailored to this setting and differ from our work. We will present a more accurate comparison in Section 4.3.2 where we apply SQUEAK to graph spectral sparsification and consider the problem of learning on graph.

Proof sketch: Although SQUEAK is conceptually simple, providing guarantees on its space/time complexity and accuracy is far from trivial.
The first step in the proof is again to carefully decompose the failure event across the whole merge tree into separate failure events for each merge node $\{h, l\}$, and for each node construct a random process $\mathbf{Y}$ that models how Algorithm 10 generates the dictionary $\mathcal{I}_{\{h,l\}}$.
Notice that these processes are again sequential in nature and the various steps (layers in the tree) are not i.i.d.. Furthermore, compared to KORS, the variance of $\mathbf{Y}$ is now much larger due to the additional randomness of updating $q_{t,i}$, and cannot be bounded uniformly. Instead, we take a more refined approach, inspired by Pachocki, (2016), that 1) uses Freedman's inequality to treat $\mathbf{W}$, the variance of process $\mathbf{Y}$, as a random object itself, 2) applies a stochastic dominance argument to $\mathbf{W}$ to reduce it to a sum of i.i.d. r.v.-s and only then we can 3) apply i.i.d. concentrations to obtain the desired result.

One of the core component of the proof is a stochastic dominance argument for adaptive importance sampling processes that might be of independent interest.
Using the notation from Section 3.3.1, consider an adaptive importance sampling chain where the adaptive probabilities $p_{t,i}$ are not i.i.d (i.e., computed using the chain itself), but we can guarantee that the *final* probability $p_{n,i}$ is lower bounded by a fixed (non-random) quantity $\tau_{n,i}$.
Then, no matter which sequence we use for the sampling or the length of the sampling process, we can show that the maximum weight (minimum probability) reached in the chain (which drives the variance of the importance sampling process) is stochastically dominated by a truncated Pareto random variable.
This can help us analyse future importance sampling algorithm. Since the importance sampling chain has correlated steps, it is hard to bound its maximum weight with anything other than its range $1/\tau_{n,i}$. On the other hand it is easy to see that the dominating Pareto r.v. has a much smaller mean $1 + \log(1/\tau_{n,i})$, and a small, bounded variance.

3.3.3 Comparison of all RLS sampling algorithms

We can now recap how all the sampling algorithm we presented so far compare with each other. The results are reported in Table 3.1.
For all algorithms, the comparison is done using the minimum $\overline{q}$ (or equivalent parameter) that guarantees generating an (ε, γ)-accurate dictionary in the end. We report the runtime $\widetilde{\mathcal{O}}(\text{Runtime})$ of the algorithm, ignoring constant and logarithmic terms in the runtime, the size of the dictionary $\mathcal{O}(|\mathcal{I}_n|)$, ignoring constants (we treat ε as a constant), and for multi-pass algorithms the number of passes over the data they require to execute. For algorithms that require an oracle to run, we report an extra runtime cost in the runtime column.
Finally, note that for all algorithms, except for Exact-RLS, the memory required to execute

Multi-Pass

	$\widetilde{\mathcal{O}}$(Runtime)	$\mathcal{O}(\lvert\mathcal{I}_n\rvert)$	Passes
Oracle RLS Sampl.	n +	$d_{\text{eff}}^n(\gamma)\log(n)$	Many
Exact RLS Sampl.	n^3	$d_{\text{eff}}^n(\gamma)\log(n)$	Many
Two-Pass RLS Sampl. (Alaoui and Mahoney, 2015)	n^3/γ^2	$n/\gamma + d_{\text{eff}}^n(\gamma)\log(n)$	3
Many-Pass RLS Sampl.	$nd_{\text{eff}}^n(\gamma)^2$	$d_{\text{eff}}^n(\gamma)\log(n\log(\frac{n}{\gamma}))$	$\log(n/\gamma)$
Recursive RLS Sampl. (Musco and Musco, 2017)	$nd_{\text{eff}}^n(\gamma)^2$	$d_{\text{eff}}^n(\gamma)\log(n)$	$\log(n)$

Single-pass

	$\widetilde{\mathcal{O}}$(Runtime)	$\mathcal{O}(\lvert\mathcal{I}_n\rvert)$
Seq. Oracle RLS Sampl. (w/o removal)	n +	$d_{\text{eff}}^n(\gamma)\log^2(n)$
Uniform Sampl. (Bach, 2013)	$n\mu(\gamma)$ +	$n\mu(\gamma)$
Seq. RLS Sampl. (w/o removal) (Calandriello et al., 2017c)	$nd_{\text{eff}}^n(\gamma)^2$	$d_{\text{eff}}^n(\gamma)\log^2(n)$
Seq. Oracle RLS Sampl. (w/ removal)	n +	$d_{\text{eff}}^n(\gamma)\log(n)$
Seq. RLS Sampl. (w/ removal, fully sequential) (Calandriello et al., 2017a)	$nd_{\text{eff}}^n(\gamma)^3$	$d_{\text{eff}}^n(\gamma)\log(n)$
Seq. RLS Sampl. (w/ removal, parallel on k machines) (Calandriello et al., 2017a)	$(n/k)d_{\text{eff}}^n(\gamma)^3$	$d_{\text{eff}}^n(\gamma)\log(n)$

Table 3.1: Comparison of all sampling algorithms.

the algorithm scales as $\mathcal{O}(|\mathcal{I}_t|^2)$, necessary to store the matrix $\mathbf{S}_t^\intercal\Phi_t^\intercal\Phi_t\mathbf{S}_t$. In Exact-RLS the gap is due to the fact that it takes $\mathcal{O}(n^2)$ space to construct, and store the $\Phi_n^\intercal\Phi_n$ matrix necessary to compute RLS, but the final dictionary size is only $\mathcal{O}(d_{\text{eff}}^n(\gamma)\log(n))$.

Multi-pass algorithms

Among batch algorithms, we begin with Oracle-RLS, where we sample proportionally to the true RLS provided by an oracle. Note that even with the RLS known in advance, sampling without parallelism still require n steps[4], and the final dictionary size is $\mathcal{O}(d_{\text{eff}}^n(\gamma)\log(n))$. This will be the minimum complexity and space that we will refer to as "optimal". Note also that in order for RLS sampling to improve performance in downstream tasks, the dictionary size $\mathcal{O}(d_{\text{eff}}^n(\gamma)\log(n))$ must be small, thus during the comparison we will restrict ourselves to the $d_{\text{eff}}^n(\gamma) \ll n$ regime.
Exact-RLS, which uses Definition 2.4 to compute the RLS and then sample, achieve optimal $\mathcal{O}(d_{\text{eff}}^n(\gamma)\log(n))$ dictionary size, but takes $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ memory to run, and it does not scale to large datasets.
Two-Step-RLS, introduced by Alaoui and Mahoney, (2015), was the first batch approximate RLS sampling to provably improve on Exact-RLS. Space wise, the first-step

[4]Even replacing sequential Binomial sampling with Multinomial cannot reduce this complexity, since drawing a single sample from a Multinomial over n objects still takes $\Omega(n)$ time.

uniformly sampled dictionary has to contain n/γ samples to be (ε, γ)-accurate, much larger than the optimal $\mathcal{O}(d^n_{\text{eff}}(\gamma)\log(n))$ which is achieved only by the second-step dictionary sampled using approximate RLS. Time wise, the first-step uniform sampling has a small $\mathcal{O}(n)$ computational cost, but the $\mathcal{O}(n^3/\gamma^2)$ cost for the RLS estimation phase dominates the time complexity of the algorithm. Although this is an improvement over $\mathcal{O}(n^3)$ it it still quite far away from $\mathcal{O}(n)$.

The first big jump in space and runtime arrives when we extend Two-Step-RLS to make multiple RLS sampling steps, each increasingly more accurate, resulting in Many-Step-RLS and Recursive-RLS by Musco and Musco, (2017). Space wise, Recursive-RLS exactly matches the optimal $d^n_{\text{eff}}(\gamma)\log(n)$ dictionary size, with Many-Step-RLS getting close and only a $\log(\log(n/\gamma))$ factor away from it. Time wise, they both achieve near-linear in n runtime $\widetilde{\mathcal{O}}(nd^n_{\text{eff}}(\gamma)^2)$, which in our $d^n_{\text{eff}}(\gamma) \ll n$ setting essentially matches the optimal $\mathcal{O}(n)$ runtime of Oracle-RLS. Among the two, Recursive-RLS requires slightly less passes, $\log(n)$ vs $\log(n/\gamma)$, when $\gamma < 1$. Nonetheless, for many downstream tasks Many-Step-RLS can improve on the number of passes using a computational regularization approach (Rudi et al., 2015). In particular, we can start with $\overline{q} = \mathcal{O}(\log(n\log(n)))$ and $\gamma = n$, and keep reducing γ at each pass until cross-validation on the downstream task indicates that we should stop. This way, we avoid performing passes unnecessary for the downstream task's performance.

Recap: in the batch setting, we went from the $\mathcal{O}(n^3)$ time complexity of Exact-RLS, to the near-linear $\widetilde{\mathcal{O}}(nd^n_{\text{eff}}(\gamma)^2)$ time complexity of Many-Step-RLS and Recursive-RLS, breaking the *quadratic barrier* required to compare each sample to all other. We also reduce the $\mathcal{O}(n^2)$ memory requirement of Exact-RLS, necessary to compute the exact RLS, to a $\mathcal{O}(d^n_{\text{eff}}(\gamma)^2\log^2(n))$ memory requirement to execute Many-Step-RLS and Recursive-RLS.

Single-pass algorithms w/o removal

To eliminate the need for passes altogether, we moved from RLS sampling to sequential RLS sampling. In this category, we first consider algorithms that take an insertion-only approach to dictionary construction.

Our optimal baseline is SOracle-RLS-NoRem, where we sample proportionally to the *current* RLS provided by an oracle restricted by the streaming setting to seeing only the partial dataset $\mathcal{D}_t$. The total runtime of the algorithm is n, and it generates a dictionary that contains $\mathcal{O}(d^n_{\text{onl}}(\gamma)\log(n)) \leq \mathcal{O}(d^n_{\text{eff}}(\gamma)\log^2(n)$ atoms.

The first algorithm that can operate both in the batch and streaming setting is uniform sampling. Uniform takes only $\mathcal{O}(\overline{q})$ time[5] to sample its dictionary. Unfortunately, Bach, (2013) shows that to guarantee (ε, γ)-accuracy we must choose $\overline{q} = \mathcal{O}(n\mu(\gamma))$ proportional to the regularized coherence $\mu(\gamma) = \max_{i=1}^n \tau_{n,i}(\gamma)$, resulting in a $\mathcal{O}(n\mu(\gamma))$ runtime. Moreover, we need assumptions or an oracle to tell us in advance the value of the coherence $\mu(\gamma)$ to implement the algorithm. Alternatively, we can use the bound $\mu(\gamma) \leq 1/\gamma$ to run the algorithm with $\overline{q} = n/\gamma$, resulting in a $\mathcal{O}(n/\gamma)$ runtime and dictionary size, which matches the n optimal runtime, but is far away from the optimal $d^n_{\text{onl}}(\gamma)\log(n)$ dictionary size.

Using instead our proposed algorithm KORS (Calandriello et al., 2017c) we can achieve (up to constants) the optimal $\mathcal{O}(d^n_{\text{onl}}(\gamma)\log(n))$ dictionary size in a near-linear $\widetilde{\mathcal{O}}(nd^n_{\text{onl}}(\gamma)^2)$ runtime. Therefore, we can achieve essentially the same guarantees as SOracle-RLS-NoRem, without the need of an oracle. Moreover, we get not only guarantees on the final

[5] Drawing a sample from a Binomial $\mathcal{B}(p, \overline{q})$ can be implemented in $\mathcal{O}(p\overline{q})$ w.h.p.. Since in Uniform we have probabilities that sum to one, the overall sampling time is $\mathcal{O}(\overline{q})$.

dictionary $\mathcal{I}_n$, but also on all intermediate dictionaries $\mathcal{I}_t$.
Therefore, if the underlying $d_{\text{eff}}^n(\gamma)$ of the dataset is large and the size of the dictionary grows so much that we have to interrupt the algorithm, we still know that the intermediate dictionary $\mathcal{I}_t$ is (ε, γ)-accurate w.r.t. to the data seen so far, and we can use it to choose a better γ. Similarly, in streaming application we can at any moment take a snapshot of the dictionary to start using it in a downstream task, and continue processing the stream without restarting.
Recap: in the streaming setting with insertion-only updates, we went from the $\log(n)$ passes of Many-Step-RLS and Recursive-RLS to the *single pass* of KORS. The $\widetilde{\mathcal{O}}(nd_{\text{eff}}^n(\gamma)^2)$ time complexity of KORS remain near-linear, although with a worse $\log^4(n)$ dependency when compared to the $\log^2(n)$ dependency of Many-Step-RLS and Recursive-RLS. Similarly, the $\mathcal{O}(d_{\text{eff}}^n(\gamma)^2 \log^4(n))$ memory requirement to execute KORS is only $\log^2(n)$ larger than Many-Step-RLS and Recursive-RLS.

Single-pass algorithms w/ removal

Adding the possibility of removing atoms from intermediate dictionaries brings the requirements of single-pass algorithms close to the ones of multi-pass algorithms. This is clear from looking at SOracle-RLS-Rem, a sequential oracle algorithm, that matches the n time complexity and $\mathcal{O}(d_{\text{eff}}^n(\gamma)\log(n))$ dictionary size of Oracle-RLS, a batch oracle algorithm. In this setting SQUEAK, in its fully sequential variant, runs in a near-linear $\widetilde{\mathcal{O}}(nd_{\text{eff}}^n(\gamma)^3)$ time and construct optimally sized $\mathcal{O}(d_{\text{eff}}^n(\gamma)\log(n))$ dictionaries, essentially matching again both the batch and sequential oracle but without knowing the RLS in advance. The time complexity of the algorithm is $\widetilde{\mathcal{O}}(nd_{\text{eff}}^n(\gamma)^3)$, a small increase compared to the $\widetilde{\mathcal{O}}(nd_{\text{eff}}^n(\gamma)^2)$ complexity of of KORS, Many-Step-RLS and Recursive-RLS.
Compared to KORS, this is because sequential SQUEAK cannot use efficient $\mathcal{O}(d_{\text{eff}}^n(\gamma)^2)$ rank-1 updates when estimating RLSs, due to the fact that it changes many atoms in the dictionary at the same time. Compared to Many-Step-RLS and Recursive-RLS, this is because at each of the n steps KORS needs to perform a $\mathcal{O}(d_{\text{eff}}^n(\gamma)^3)$ inversion followed by $\mathcal{O}(d_{\text{eff}}^n(\gamma))$ RLS estimations each costing $\mathcal{O}(d_{\text{eff}}^n(\gamma)^2)$, while the batch algorithms need to perform a $\mathcal{O}(d_{\text{eff}}^n(\gamma)^3)$ inversion only once for each of the $\log(n)$ passes, while the n RLS estimations cost a cheaper $\mathcal{O}(d_{\text{eff}}^n(\gamma)^2)$ time each.
Nonetheless, if we distribute the computation across k machines, the parallel variant of SQUEAK can compute a solution in $\widetilde{\mathcal{O}}((n/k)d_{\text{eff}}^n(\gamma)^3)$ time, breaking even the near-linear time barrier. This is even better than the batch oracle RLS algorithm, and can only be compared to a parallel oracle algorithm. Note also out of all the near-linear time sampling algorithms, SQUEAK is the only one that can be naturally parallelized.
Finally, compared to the multi-pass algorithms, KORS and SQUEAK are the only near-linear time algorithms that can continue to update their dictionary even beyond the first n samples, without having to restart the computation from scratch.
Recap: the addition of dictionary removal improves the space and time complexity of streaming algorithms. SQUEAK requires only $\mathcal{O}(d_{\text{eff}}^n(\gamma)^2 \log^2(n))$ memory to run, matching Many-Step-RLS and Recursive-RLS and improving on KORS. When run on a single machine, it requires $\widetilde{\mathcal{O}}(nd_{\text{eff}}^n(\gamma)^3)$ time to execute, slightly more than the other near-linear algorithms. When run on k machines, we can improve runtime to $\widetilde{\mathcal{O}}((n/k)d_{\text{eff}}^n(\gamma)^3)$, surpassing all other near-linear time algorithms and reaching near-constant $\widetilde{\mathcal{O}}(d_{\text{eff}}^n(\gamma)^3)$ runtime when enough machines are used.

3.3.4 Straightforward extensions and practical tricks

To simplify the exposition, so far we did not use the tightest possible bounds or the most efficient update rules. In this section we highlight several simple extension to our analysis that can either sharpen our bounds or improve computation.

Improve $\log(n)$ dependency in $\overline{q}$. It is possible to improve the $\log(n)$ dependency in SQUEAK with a milder $\max\{\log(d_{\text{eff}}^n(\gamma)), \log(h)\}$, where h is the height of the merge tree, i.e., the maximum number of merges that a single sample has to go through until it reaches the root.
Proving this requires (1) replacing the concentration inequality used in Proposition 3.18 that scales with n with a stronger version by Minsker, (2011) that scales only with $d_{\text{eff}}^n(\gamma)$, and (2) defining more carefully the union bound over the failure events to scale with the actual path each sample follows in the merge tree rather than the number of nodes in the tree.
When $h = \log(n)$ (fully balanced tree), this means that, if we set $\overline{q} = \max\{\log(d_{\text{eff}}^n(\gamma)), \log(h)\}$, we can improve SQUEAK's final dictionary size to $\mathcal{O}(d_{\text{eff}}^n(\gamma)\log(d_{\text{eff}}^n(\gamma))\log\log(n))$ from the original $\mathcal{O}(d_{\text{eff}}^n(\gamma)\log(n))$, with associated runtime improvements. Similarly, we can tune KORS's $\overline{q}$ to improve final dictionary size to $\mathcal{O}(d_{\text{eff}}^n(\gamma)\log(d_{\text{eff}}^n(\gamma))\log(n))$ from $\mathcal{O}(d_{\text{eff}}^n(\gamma)\log^2(n))$.
Unfortunately, estimating $\log(d_{\text{eff}}^n(\gamma))$ is not easy, and it is impossible to do in a streaming setting since $\overline{q}$ must be set in advance. Thus in practice it is simpler to just set $\overline{q}$ proportional to $\log(n)$.

Time-dependent regularization γ. All the analysis remains identical if we *strictly increase* γ over time. This is helpful, especially when processing large datasets sequentially. For example, in many kernel-related problem the γ that gives us the optimal learning rate is $\sqrt{n}$ (as we will see in the discussion of Proposition 4.5). In a dataset with a million samples this gives us a γ in the thousands. If we split our dataset in small subset of a few hundreds samples, we might end up with extremely small RLS that are numerically hard to estimate. In this case starting with a smaller γ and increasing it as we continue up the merge tree (e.g., $\sqrt{|\mathcal{D}_{\{h,l\}}|}$improves numerical stability, without costing too much space.

Use overestimates of the RLS instead of underestimates. For the definition of ρ-accurate RLS estimate $\widetilde{\tau}_{t,i}$ we chose to make them strict underestimates of the true RLS $\tau_{t,i}$. This was done in order to reinforce their interpretation as probabilities in coin flips.
Another possible solution is to define an estimate as ρ-accurate when it satisfies $\tau_{t,i} \leq \widetilde{\tau}_{t,i} \leq \rho\tau_{t,i}$, i.e., when it overestimates $\tau_{t,i}$. To maintain the algorithm well defined, we can compensate this change using thresholding and define the coin flip probabilities as $\widetilde{p}_{t,i} = \min\{\widetilde{\tau}_{t,i}, 1\}$. Since we know that the true $\tau_{t,i}$ are always smaller than 1, this does not affect ρ-accuracy and our analysis, except for a more complex notation.
On the positive side, we now get more accurate RLS estimates, because we are using prior knowledge on $\tau_{t,i}$ to refine our estimator $\widetilde{\tau}_{t,i}$. A similar argument can be made for other refinements such as $\widetilde{p}_{t,i} = \max\{\widetilde{\tau}_{t,i}, 1/(t+\gamma)\}$ (we know the minimum $\tau_{t,i}$) and for taking the minimum $\widetilde{p}_{t,i} = \min\{\widetilde{\tau}_{t,i}, \widetilde{\tau}_{t-1,i}\}$ (we know $\tau_{t,i} \leq \tau_{t-1,i}$).

3.3.5 Open questions: lifelong learning and deterministic dictionary learning

Two important questions that remain open at the end of this chapter is whether we can continue updating our dictionary indefinitely, and whether randomness is truly necessary for accuracy.

The first question is especially relevant in the context of online learning and learning on streams, where knowing $\log(n)$ in advance might not be possible. It is also central to the problem of lifelong learning (Thrun, 1998), where the learning algorithm is faced with an open-ended goal and must continue to update its model forever.
Unfortunately, both KORS and SQUEAK require to know $\log(n)$ in advance to tune $\overline{q}$. Note that our guarantees apply for any $\overline{q}$ larger than $\log(n)$ and therefore if we know that our dataset might grow over time we can "overbudget" (which is made easy by the logarithmic dependency). For a fixed $\overline{q}$ the guarantees also hold for any $\varepsilon' > \varepsilon$, and therefore if we continue processing samples after the first n our analysis degrades gracefully in guaranteeing (ε', γ)-accuracy.
The main technical obstacle to obtain an algorithm that can truly operate on unbounded streams is the union bound over failure events (total number of dictionary merges). This quantity naturally grows with the number of samples processed, so it is not easy to see if it is possible to avoid this dependency.
Alternatively, we could try to modify our proof to take into account a time-dependant budget $\overline{q}_t$, and use an adaptive schedule $\log(t)$ for it. Unfortunately, this would require going back and re-evaluating already dropped samples when $\overline{q}_t$ increases, violating our single-pass constraint.

Another interesting question is whether it is possible to *deterministically* select the atoms with the highest RLS (e.g., estimated using SQUEAK) and prove that this constructs an (ε, γ)-accurate dictionary. While some early results are available (Papailiopoulos et al., 2014; Patel et al., 2016), they are still very far from the accuracy, space and time performance of randomized methods such as SQUEAK.

3.4 Proof of main results

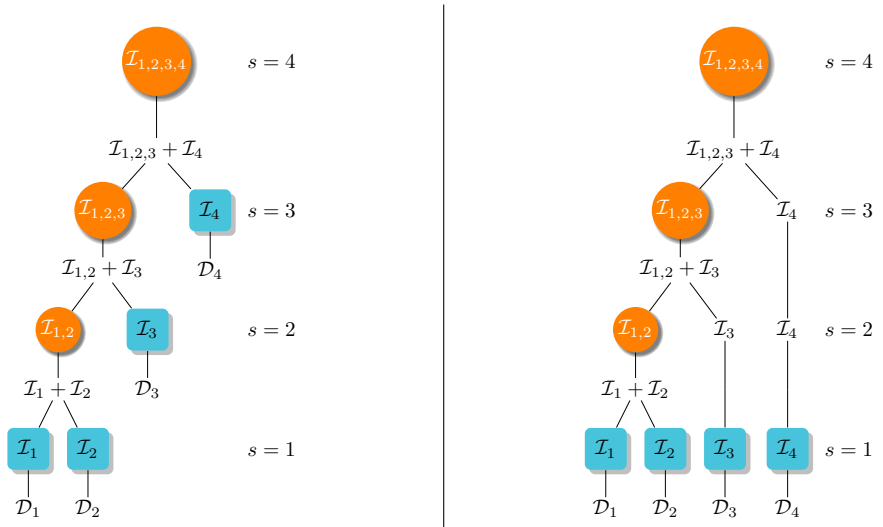
We will provide a single proof of Theorem 4.7, since Algorithm 8 and Theorem 3.11 can be seen as a special case of Algorithm 10 with a fully sequential merge tree, and where we do not update the approximate RLSs after estimating them the first time. It is easy to see that if $\widetilde{\tau}_{t,t} = \widetilde{\tau}_{t',t}$ for all $t' > t$, then the importance sampling will always return $q_{t,t} = q_{t',t}$, and we perform sequential RLS sampling without removal.

3.4.1 Decomposing the merge tree

We begin by describing more in detail some notation introduced in the main paper and necessary for this proof.

Merge trees We first restate and formalize more in detail the random process induced by SQUEAK.

We partition $\mathcal{D}$ into k disjoint sub-datasets $\mathcal{D}_i$ of size n_i, such that $\mathcal{D} = \cup_{i=1}^{k} \mathcal{D}_i$. For each dataset $\mathcal{D}_i$, we construct an initial dictionary $\mathcal{I}_{\{1,i\}} = \{(\widetilde{p}_{0,j} = 1, q_{0,j} = \overline{q}, \phi_j) : j \in \mathcal{D}_i\}$ by inserting all points from $\mathcal{D}_i$ into $\mathcal{I}_{\mathcal{D}_i}$ with weight $\widetilde{p}_{0,j} = 1$ and number of copies $q_{0,j} = \overline{q}$. It is easy to see that $\mathcal{I}_{\{1,i\}}$ is an (ε, γ)-accurate dictionary, and we can split the data in small enough chunks to make sure that it can be easily stored and manipulated in memory. Alternatively, if we want our initial dictionaries to be small and cannot choose the size of $\mathcal{D}_i$, we can run KORS on $\mathcal{D}_i$ to generate $\mathcal{I}_{\{1,i\}}$, and the following proof will remain valid. Regardless of construction, the initial (ε, γ)-accurate dictionaries $\mathcal{I}_{\{1,i\}}$ are included into the dictionary pool $\mathcal{S}_1$.



(a) merge tree (b) stretched merge tree

Figure 3.4: Stretching a merge tree.

At iteration h, the inner loop of SQUEAK arbitrarily chooses two dictionaries from $\mathcal{S}_h$ and merges them into a new dictionary. Any arbitrary sequence of merges can be described by a full binary tree, i.e., a binary tree where each node is either a leaf or has exactly two children. Fig. 3.3 shows several different merge trees corresponding to different choices for the order of the merges.

Note that starting from k leaves, a full binary tree will always have exactly $k-1$ internal nodes, and a maximum depth of k. Therefore, regardless of the structure of the merge tree, we can always *stretch* it into a tree of depth k, with all the initial dictionaries $\mathcal{I}_{1,i}$ as leaves on its deepest layer. Fig. 3.4 reports an example of such a stretching transformation, where all the extra nodes introduced in the trees do not correspond to merges, and simply propagate the dictionary up the tree (e.g., $\mathcal{I}_{\{2,3\}} = \mathcal{I}_{\{3,2\}}$).

After this transformation, we index the tree nodes using their height (longest path from the node to a leaf, also defined as depth of the tree minus depth of the node), where all leaves have height 1 and the root has height k. We can also see that at each layer, there is a single dictionary merge, and the size of $\mathcal{S}_h$ (number of dictionaries present at layer h) is $|\mathcal{S}_h| = k - h + 1$.

Therefore, a node corresponding to a dictionary is uniquely identified with two indices $\{h, l\}$, where h is the height of the layer and $l \leq |\mathcal{S}_h|$ is the index of the node in the layer. For example, in Fig. 3.4, the node containing $\mathcal{I}_{1,2,3}$ is indexed as $\{3, 1\}$, and the highest node containing $\mathcal{I}_4$ is indexed as $\{3, 2\}$.

We also define the dataset $\mathcal{D}_{\{h,l\}}$ as the union of all sub-datasets $\mathcal{D}_{l'}$ that are reachable from node $\{h, l\}$ as leaves. For example, in Fig. 3.4, dictionary $\mathcal{I}_{1,2,3}$ in node $\{3, 1\}$ is constructed starting from all points in $\mathcal{D}_{\{3,1\}} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$, where we highlight in red the descendant tree. We now define $\mathbf{P}^h$ (see Definition 2.8) as the block diagonal regularized projection matrix where each diagonal block $\mathbf{P}_{\{h,l\}}$ is constructed on $\mathcal{D}_{\{h,l\}}$. Again, from Fig. 3.3, $\mathbf{P}^3$ is a $n \times n$ matrix with two blocks on the diagonal, a first $(n_1+n_2+n_3) \times (n_1+n_2+n_3)$ block $\mathbf{P}_{3,1}$ constructed on $\mathcal{D}_{\{3,1\}} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$, and a second $n_4 \times n_4$ block $\mathbf{P}_{3,2}$ constructed on $\mathcal{D}_{\{3,2\}} = \mathcal{D}_4$. Similarly, we can adapt Eq. 2.15 to define a block diagonal $\widetilde{\mathbf{P}}^h$, where block $\widetilde{\mathbf{P}}_{\{h,l\}}$ is defined using $\mathbf{P}_{\{h,l\}}$ and $\mathcal{I}_{\{h,l\}}$, which contains the weights and number of copies

computed by SQUEAK at each node. Finally, we will also denote with $d_{\text{eff}}^{\{h,l\}}(\gamma) = d_{\text{eff}}^{\mathcal{D}_{\{h,l\}}}(\gamma)$ the effective dimension of $\mathcal{D}_{\{h,l\}}$.

The statement. Since $\mathbf{P}^h - \widetilde{\mathbf{P}}^h$ is block diagonal, we have that a bound on its norm implies an equal bound on the norm each matrix on the diagonal, i.e.,

$$\|\mathbf{P}^h - \widetilde{\mathbf{P}}^h\| = \max_l \|\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}\| \leq \varepsilon \Rightarrow \|\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}\| \leq \varepsilon$$

for all blocks l on the diagonal, and since each block corresponds to a dictionary $\mathcal{I}_{\{h,l\}}$, this means that if $\|\mathbf{P}^h - \widetilde{\mathbf{P}}^h\| \leq \varepsilon$, all dictionaries at layer l are ε-accurate approximation of their respective represented datasets. Our goal is to show

$$\begin{aligned}
&\mathbb{P}\Big(\exists h \in [k] : \|\mathbf{P}^h - \widetilde{\mathbf{P}}^h\|_2 \geq \varepsilon \ \cup \max_{l=1,\dots,|\mathcal{S}_h|} |\mathcal{I}_{\{h,l\}}| \geq 3\overline{q} d_{\text{eff}}^{\{h,l\}}(\gamma)\Big) \\
&= \mathbb{P}\Big(\exists h \in [k] : \underbrace{\Big(\max_{l\in[|\mathcal{S}_h|]} \|\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}\|_2\Big) \geq \varepsilon}_{A_h} \ \cup \ \underbrace{\Big(\max_{l\in[|\mathcal{S}_h|]} |\mathcal{I}_{\{h,l\}}| \geq 3\overline{q} d_{\text{eff}}^{\{h,l\}}(\gamma)\Big)}_{B_h}\Big) \leq \delta,
\end{aligned} \tag{3.6}$$

where event A_h refers to the case when some dictionary $\mathcal{I}_{\{h,l\}}$ at an intermediate layer h fails to accurately approximate $\mathcal{D}_{\{h,l\}}$ and event B_h considers the case when the memory requirement is not met (i.e., too many points are kept in one of the dictionaries $\mathcal{I}_{\{h,l\}}$ at a certain layer h). We can conveniently decompose the previous joint (negative) event into two separate conditions, one for accuracy and one for space, as

$$\begin{aligned}
&\mathbb{P}\Big(\bigcup_{h=1}^k A_h \cup B_h\Big) = \mathbb{P}\left(\left\{\bigcup_{h=1}^k A_h\right\} \cup \left\{\bigcup_{h=1}^k B_h\right\}\right) = \mathbb{P}\left(\left\{\bigcup_{h=1}^k A_h\right\}\right) + \mathbb{P}\left(\left\{\bigcup_{h=1}^k B_h\right\} \cap \left\{\bigcup_{h=1}^k A_h\right\}^{\mathsf{C}}\right) \\
&= \mathbb{P}\left(\left\{\bigcup_{h=1}^k A_h\right\}\right) + \mathbb{P}\left(\left\{\bigcup_{h=1}^k B_h\right\} \cap \left\{\bigcap_{h=1}^k A_h^{\mathsf{C}}\right\}\right) = \mathbb{P}\left(\left\{\bigcup_{h=1}^k A_h\right\}\right) + \mathbb{P}\left(\bigcup_{h=1}^k \left\{B_h \cap \left\{\bigcap_{h'=1}^k A_{h'}^{\mathsf{C}}\right\}\right\}\right).
\end{aligned} \tag{3.7}$$

We can now apply a union bound to separate each of the k merge processes. Notice that the union bound is over the nodes, since we want to guarantee that each dictionary in each node is accurate. While we know that stretching the tree (Fig. 3.4) can introduce many artificial nodes, we also know that to merge k sub-datasets we need at most k merges, and therefore we only need to bound over k negative events.
Let $\mathcal{E} = \{(h,l)\}$ be the set of nodes corresponding to actual merges (i.e., nodes before stretching). Then we can reformulate

$$\begin{aligned}
&\mathbb{P}\Big(\exists h \in [k] : \|\mathbf{P}^h - \widetilde{\mathbf{P}}^h\|_2 \geq \varepsilon \ \cup \max_{l=1,\dots,|\mathcal{S}_h|} |\mathcal{I}_{\{h,l\}}| \geq 3\overline{q} d_{\text{eff}}^{\{h,l\}}(\gamma)\Big) \\
&= \mathbb{P}\left(\exists h \in [k] : \left(\max_{l=1,\dots,|\mathcal{S}_h|} \|\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}\|_2\right) \geq \varepsilon\right) \\
&\qquad + \mathbb{P}\Big(\exists h \in [k] : \max_{l=1,\dots,|\mathcal{S}_h|} |\mathcal{I}_{\{h,l\}}| \geq 3\overline{q} d_{\text{eff}}^{\{h,l\}}(\gamma) \cap \Big\{\forall h' \in [h] : \|\mathbf{P}^{h'} - \widetilde{\mathbf{P}}^{h'}\|_2 \leq \varepsilon\Big\}\Big) \\
&\leq \sum_{\{h,l\}\in\mathcal{E}} \mathbb{P}\left(\|\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}\|_2 \geq \varepsilon\right) \\
&\qquad + \sum_{\{h,l\}\in\mathcal{E}} \mathbb{P}\left(|\mathcal{I}_{\{h,l\}}| \geq 3\overline{q} d_{\text{eff}}^{\{h,l\}}(\gamma) \cap \Big\{\forall h' \in [h] : \|\mathbf{P}^{h'} - \widetilde{\mathbf{P}}^{h'}\|_2 \leq \varepsilon\Big\}\right) \leq \delta.
\end{aligned} \tag{3.8}$$

As we highlighted many time, the accuracy of the dictionary (first term in the previous bound) is guaranteed by the fact that given an ε-accurate dictionary we obtain RLS estimates which are at least a fraction of the true RLS, thus forcing the algorithm to sample each column *enough*. On the other hand, the space complexity bound is achieved by exploiting the fact that RLS estimates are always upper-bounded by the true RLS, thus ensuring that SQUEAK does not oversample columns w.r.t. the sampling process following the exact RLS.

In the reminder of the proof, we will show that both events happen with probability smaller than $\delta/(2k)$. The main advantage of splitting the failure probability as we did in Eq. 3.8 is that we can now analyze the processes that generated each $\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}$ (and each dictionary $\mathcal{I}_{\{h,l\}}$) separately. Focusing on a single node $\{h, l\}$ restricts our problem on a well defined dataset $\mathcal{D}_{\{h,l\}}$, where we can analyze the evolution of $\mathcal{I}_{\{h,l\}}$ sequentially.

Challenges. Due to its sequential nature, the "theoretical" structure of the process generating $\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}$ is considerably complicated, where each step in the sampling process (layer in the merge tree) is highly correlated with the previous steps.[6] This prevents us from using concentration inequalities for i.i.d. processes that are at the basis of the analysis of uniform sampling (Bach, 2013), Two-Step-RLS by Alaoui and Mahoney, (2015), and Recursive-RLS by Musco and Musco, (2017).
As a result, we first show that the project error is a martingale process. The main technical difficulty in analyzing how the projection error evolves over iterations is that the projection matrices change dimension every time a new point is processed. In fact, all of the matrices $\mathbf{P}_{\{h',l'\}} - \widetilde{\mathbf{P}}_{\{h',l'\}}$ descending from $\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}$ have potentially a different size since they are based on different datasets $\mathcal{D}_{\{h',l'\}}$. This requires a careful definition of the martingale process to still use matrix concentration inequalities for *fixed-size* matrices (see Section 3.4.2).
Once we have defined the martingale structure, we will use a matrix Freedman inequality to bound the deviation of the martingale from its mean as a function of its predictable quadratic variation $\mathbf{W}$. Here the proof of KORS and SQUEAK will differ. In particular, it is possible to find a small deterministic uniform bound on the variance of each step of KORS, and use it to bound the overall process' variance.
Bounding the per-step variance of SQUEAK is much more difficult, since it continues to update the number of copies $q_{t,i}$ after insertion, as well as their weight $1/\widetilde{p}_{t,i}$ that might grow unbounded over time. As a consequence, uniform bounds on the predictable quadratic variation $\mathbf{W}$ become too large. In addition, $\mathbf{W}$ itself is a complex object since it is composed of many correlated variables.
In order to provide a tighter bound on the total variance of the martingale process of the projection error, we need to first introduce an i.i.d. stochastically dominant process (Section 3.4.4 (step 2)), which finally allows us to use an i.i.d. matrix concentration inequality to bound the total variance (Section 3.4.4-(step 5)). This finally leads to the bound on the accuracy. The bound on the space complexity (Section 3.4.5) follows similar (but simpler) steps.

3.4.2 Bounding the projection error $\|\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}\|$

The sequential process. Thanks to the union bound in Eq. 3.6, instead of having to consider the whole merge tree followed by SQUEAK, we can focus on each individual node

[6] Note once again that although the sampling process is random, the merge tree is deterministic and fixed in advance.

$\{h,l\}$ and study the sequential process that generated its dictionary $\mathcal{I}_{\{h,l\}}$. We will now map more clearly the actions taken by SQUEAK to the process that generated $\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}$. Since the merge tree is decided in advance, the dataset $\mathcal{D}_{\{h,l\}}$ and which samples ϕ_i are involved in the process is not random, and is fixed for the whole process. We begin by focusing on $\widetilde{\mathbf{P}}_{\{h,l\}}$, which is a random matrix defined starting from the fixed map $\Phi_{\{h,l\}}$ and the random dictionary $\mathcal{I}_{\{h,l\}}$, where the randomness influences both which points are included in $\mathcal{I}_{\{h,l\}}$, and the weight with which they are added.

Consider now a point $i \in \mathcal{D}_{\{h,l\}}$. Since the starting datasets in the leaves are disjoint, there is a single path in the tree, with length h, from the leaves to $\{h,l\}$. This means that for all $s < h$, we can properly define a unique $\widetilde{p}_{s,i}$ and $q_{s,i}$ associated with that point. More in detail, if at layer s point i is present in $\mathcal{D}_{\{s,l'\}}$, it means that either

(1) SQUEAK used $\mathcal{I}_{\{s,l'\}}$ to compute $\widetilde{p}_{s,i}$, and $\widetilde{p}_{s,i}$ to compute $q_{s,i}$,
(2) at layer h SQUEAK did not have any merge scheduled for point i, and we simply propagate $\widetilde{p}_{s,i} = \widetilde{p}_{s-1,i}$ and $q_{s,i} = q_{s-1,i}$,
(3) SQUEAK dropped the point at a lower layer, and we set $\widetilde{p}_{s,i}$ to the last value that was computed by SQUEAK before dropping it, i.e., we do not change the probabilities of dropped points.

Consistently with the algorithm, we initialize $\widetilde{p}_{0,i} = 1$ and $q_{0,i} = \overline{q}$.

Denote $\nu_{\{h,l\}} = |\mathcal{D}_{\{h,l\}}|$ so that we can use index $i \in [\nu_{\{h,l\}}]$ to index all points in $\mathcal{D}_{\{h,l\}}$. Using the matrices $\mathbf{C} = \mathbf{C}_{\{h,l\}}$ with vector $\mathbf{c}_i$ as its i-th column (see Definition 2.8), we can again rewrite the regularized projection matrix as that $\mathbf{P}_{\{h,l\}} = \mathbf{C}\mathbf{C}^\mathsf{T} = \sum_{i=1}^{\nu_{\{h,l\}}} \mathbf{c}_i\mathbf{c}_i^\mathsf{T}$, and we can see that the norm $\|\mathbf{c}_i\mathbf{c}_i^\mathsf{T}\|$ is equal to the RLS of the i-th sample ϕ_i w.r.t. to dataset $\mathcal{D}_{\{h,l\}}$. Note that since the starting datasets are disjoint, and there is a single path from the leaves to $\{h,l\}$, i is present only in node l on layer h. Therefore its RLS is uniquely defined w.r.t. $\mathcal{D}_{\{h,l\}}$ and can be shortened as $\tau_{h,i} = \tau_{\mathcal{D}_{\{h,l\}},i}$. Using $\mathbf{c}_i$, we can also introduce the random matrix $\widetilde{\mathbf{P}}_s^{\{h,l\}}$ as

$$\widetilde{\mathbf{P}}_s^{\{h,l\}} = \sum_{i=1}^{\nu_{\{h,l\}}} \frac{q_{s,i}}{\overline{q}\widetilde{p}_{s,i}} \mathbf{c}_i\mathbf{c}_i^\mathsf{T} = \sum_{i=1}^{\nu_{\{h,l\}}} \sum_{j=1}^{\overline{q}} \frac{z_{s,i,j}}{\overline{q}\widetilde{p}_{s,i}} \mathbf{c}_i\mathbf{c}_i^\mathsf{T}.$$

where $z_{s,i,j}$ are $\{0,1\}$ r.v. such that $q_{s,i} = \sum_{j=1}^{\overline{q}} z_{s,i,j}$, and correspond to the coin flip chain (see Section 3.3.1) associated with sample i's j-th copy.
Note that when $s = h$, we have that $\widetilde{\mathbf{P}}_h^{\{h,l\}} = \widetilde{\mathbf{P}}_{\{h,l\}}$ and we recover the definition of the approximate projection matrix from SQUEAK. But, for a general $s \neq h$ $\widetilde{\mathbf{P}}_s^{\{h,l\}}$ does not have a direct interpretation in the context of SQUEAK. It combines the vectors $\mathbf{c}_i$, which are defined using $\Phi_{\{h,l\}}$ at layer h, with the weights $\widetilde{p}_{s,i}$ computed by SQUEAK across multiple nodes at layer s, which are potentially stored in different machines that cannot communicate. Nonetheless, $\widetilde{\mathbf{P}}_s^{\{h,l\}}$ is a useful tool to analyze SQUEAK.

Taking into account that we are now considering a specific node $\{h,l\}$, we can drop the index from the dataset $\mathcal{D}_{\{h,l\}} = \mathcal{D}$, RLS $\tau_{\mathcal{D}_{\{h,l\}},i} = \tau_{h,i}$, and size $\nu_{\{h,l\}} = \nu$. Using this shorter notation, we can reformulate our objective as bounding $\|\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}\|_2 = \|\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_h^{\{h,l\}}\|_2$, and reformulate the process as a sequence of matrices $\{\mathbf{Y}_s\}_{s=1}^h$ defined as

$$\mathbf{Y}_s = \mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_s^{\{h,l\}} = \frac{1}{\overline{q}} \sum_{i=1}^{\nu} \sum_{j=1}^{\overline{q}} \left(1 - \frac{z_{s,i,j}}{\widetilde{p}_{s,i}}\right) \mathbf{c}_i\mathbf{c}_i^\mathsf{T},$$

where $\mathbf{Y}_h = \mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_h^{\{h,l\}} = \mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}$, and $\mathbf{Y}_1 = \mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_0^{\{h,l\}} = \mathbf{0}$ since $\widetilde{p}_{0,i} = 1$ and $q_{0,i} = \overline{q}$.

3.4.3 Bounding the martingale $\mathbf{Y}_h$

We transformed the problem of bounding $\|\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}\|$ into the problem of bounding $\mathbf{Y}_h$, which we modeled as a random matrix process, connected to SQUEAK by the fact that both algorithm and random process $\mathbf{Y}_h$ make use of the same weight $\widetilde{p}_{s,i}$ and multiplicities $q_{s,i}$. There are two main issues in analyzing the process $\{\mathbf{Y}_s\}_{s=1}^h$:

(1) The overall algorithm may fail in generating an accurate dictionary at some intermediate iteration and yet return an accurate dictionary at the end. Moreover, whenever one of the intermediate $\|\mathbf{Y}_s\|$ is larger than ε we lose our guarantees for $\widetilde{p}_{s,i}$ for the whole process, since an inaccurate $\widetilde{p}_{s,i}$ that underestimates too much $p_{s.i}$ will influence all successive $\widetilde{p}_{s',i}$ through the minimum.
To solve this, we consider an alternative (more pessimistic) process which is "frozen" as soon as it constructs an inaccurate dictionary. Freezing the probabilities at the first error gives us a process that fails more often, but that provides strong guarantees up to the moment of failure.

(2) While $\|\mathbf{Y}_h\| = \|\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_h^{\{h,l\}}\| \leq \varepsilon$ guarantees that $\mathcal{I}_{\{h,l\}}$ is an ε-accurate dictionary of $\Phi_{\{h,l\}}$, knowing $\|\mathbf{Y}_s\| = \|\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_s^{\{h,l\}}\| \leq \varepsilon$ for $s < h$ does not guarantee that all descendant $\mathcal{I}_{\{s,l'\}}$ are (ε,γ)-accurate w.r.t. their $\Phi_{\{s,l'\}}$.
Intuitively, this is because $\mathbf{C}_t$ is defined w.r.t. $(\Phi_t^\intercal \Phi_t + \gamma \mathbf{I}_t)^{-1}$, while $\mathbf{C}_s$ is defined w.r.t. $(\Phi_s^\intercal \Phi_s + \gamma \mathbf{I}_s)^{-1}$, with $(\Phi_t \Phi_t^\intercal + \gamma \mathbf{\Pi}_t)^{-1} \preceq (\Phi_s \Phi_s^\intercal + \gamma \mathbf{\Pi}_s)^{-1}$. Since the inverse at step t is "stronger" than at step s, it is possible for $\|\mathbf{C}_t^\intercal(\mathbf{I}_t - \mathbf{S}_s^t(\mathbf{S}_s^t)^\intercal)\mathbf{C}_t^\intercal\|$ to be smaller than ε, while $\|\mathbf{C}_s^\intercal(\mathbf{I}_s - \mathbf{S}_s(\mathbf{S}_s)^\intercal)\mathbf{C}_s^\intercal\|$ is larger than ε.
Nonetheless, we will show that $\|\mathbf{Y}_s\| \leq \varepsilon$ is enough to guarantee that the intermediate estimate $\widetilde{p}_{s,i}$ computed in SQUEAK, and used as weights in $\widetilde{\mathbf{P}}_s^{\{h,l\}}$, are never too small.

The frozen process. We will now replace the process $\mathbf{Y}_s$ with an alternative process $\overline{\mathbf{Y}}_s$ defined as

$$\overline{\mathbf{Y}}_s = \mathbf{Y}_{s-1}\mathbb{I}\left\{\|\overline{\mathbf{Y}}_{s-1}\| \leq \varepsilon\right\} + \overline{\mathbf{Y}}_{s-1}\mathbb{I}\left\{\|\overline{\mathbf{Y}}_{s-1}\| \geq \varepsilon\right\},$$

This process starts from $\overline{\mathbf{Y}}_0 = \mathbf{Y}_0 = \mathbf{0}$, and is identical to $\mathbf{Y}_s$ until a step $\overline{s}$ where for the first time $\|\mathbf{Y}_{\overline{s}}\| \leq \varepsilon$ and $\|\mathbf{Y}_{\overline{s}+1}\| \geq \varepsilon$. After this failure happen the process $\overline{\mathbf{Y}}_s$ is "frozen" at $\overline{s}$ and $\overline{\mathbf{Y}}_s = \mathbf{Y}_{\overline{s}+1}$ for all $\overline{s}+1 \leq s \leq h$. Consequently, if any of the intermediate elements of the sequence violates the condition $\|\mathbf{Y}_s\| \leq \varepsilon$, the last element will violate it too. For the rest, $\overline{\mathbf{Y}}_s$ behaves exactly like $\mathbf{Y}_s$. Therefore,

$$\mathbb{P}\left(\|\mathbf{Y}_h\| \geq \varepsilon\right) \leq \mathbb{P}\Big(\|\overline{\mathbf{Y}}_h\| \geq \varepsilon\Big),$$

and if we can bound $\mathbb{P}\Big(\|\overline{\mathbf{Y}}_h\| \geq \varepsilon\Big)$ we will have a bound for the failure probability of SQUEAK, even though after "freezing" the process $\overline{\mathbf{Y}}_h$ does not make the same choices as the algorithm.

We will see now how to construct the process $\overline{\mathbf{Y}}_s$ starting from $z_{s,i,j}$ and $\widetilde{p}_{s,i,j}$. We recursively define the indicator ($\{0,1\}$) random variable $\overline{z}_{s,i,j}$ as

$$\overline{z}_{s,i,j} = \mathbb{I}\left\{u_{s,i,j} \leq \frac{\overline{p}_{s,i,j}}{\overline{p}_{s-1,i,j}}\right\}\overline{z}_{s-1,i,j},$$

where $u_{s,i,j} \sim \mathcal{U}(0,1)$ is a $[0,1]$ uniform random variable and $\overline{p}_{s,i,j}$ is defined as

$$\overline{p}_{s,i,j} = \widetilde{p}_{s,i}\mathbb{I}\left\{\|\overline{\mathbf{Y}}_{s-1}\| \leq \varepsilon \cap z_{s-1,i,j} = 1\right\} + \overline{p}_{s-1,i,j}\mathbb{I}\left\{\|\overline{\mathbf{Y}}_{s-1}\| \geq \varepsilon \cup z_{s-1,i,j} = 0\right\}.$$

This definition of the process satisfies the freezing condition, since if $\|\mathbf{Y}_{\overline{s}+1}\| \geq \varepsilon$ (we have a failure at step $\overline{s}$), for all $s' \geq \overline{s}+1$ we have $\overline{z}_{s',i,j} = \overline{z}_{\overline{s}+1,i,j}$ with probability 1 ($\overline{p}_{\overline{s}+1,i,j}/\overline{p}_{\overline{s},i,j} = \overline{p}_{\overline{s},i,j}/\overline{p}_{\overline{s},i,j} = 1$), and the weights $1/(\overline{q}\overline{p}_{\overline{s}+1,i,j}) = 1/(\overline{q}\overline{p}_{\overline{s},i,j})$ never change.

Introducing a per-copy weight $\overline{p}_{s,i,j}$ and enforcing that $\overline{p}_{s+1,i,j} = \overline{p}_{s,i,j}$ when $z_{s,i,j} = 0$ avoids subtle inconsistencies in the formulation. In particular, not doing so would semantically correspond to reweighting dropped copies. Although this does not directly affect $\mathbf{Y}_s$ (since the ratio $z_{s,i,j}/\widetilde{p}_{s,i}$ is zero for dropped copies), and therefore the relationship $\mathbb{P}\left(\|\mathbf{Y}_h\| \geq \varepsilon\right) \leq \mathbb{P}\Big(\|\overline{\mathbf{Y}}_h\| \geq \varepsilon\Big)$ still holds, we will see later how maintaining consistency helps us bound the second moment of our process.

We can now arrange the indices s, i, and j into a linear index $r = s$ in the range $[1, \ldots, \nu^2\overline{q}]$, obtained as $r = \{s,i,j\} = (s-1)\nu\overline{q} + (i-1)\overline{q} + j$. We also define the difference matrix as

$$\overline{\mathbf{X}}_{\{s,i,j\}} = \frac{1}{\overline{q}}\left(\frac{z_{s-1,i,j}}{\overline{p}_{s-1,i,j}} - \frac{z_{s,i,j}}{\overline{p}_{s,i,j}}\right)\mathbf{c}_i\mathbf{c}_i^{\mathsf{T}},$$

which allows us to write the cumulative matrix as $\overline{\mathbf{Y}}_{\{s,i,j\}} = \sum_{r=1}^{\{s,i,j\}} \overline{\mathbf{X}}_{\{s,i,j\}}$ where the checkpoints $\{s,\nu,\overline{q}\}$ correspond to $\overline{\mathbf{Y}}_s$,

$$\overline{\mathbf{Y}}_{\{s,\nu,\overline{q}\}} = \overline{\mathbf{Y}}_s = \frac{1}{\overline{q}}\sum_{i=1}^{\nu}\sum_{j=1}^{\overline{q}}\left(1 - \frac{z_{s,i,j}}{\overline{p}_{s,i,j}}\right)\mathbf{c}_i\mathbf{c}_i^{\mathsf{T}}.$$

Let $\mathcal{F}_s$ be the filtration containing all the realizations of the uniform random variables $u_{s,i,j}$ up to the step s, that is $\mathcal{F}_s = \{u_{s',i',j'}, \forall\{s',i',j'\} \leq s\}$. Again, we notice that $\mathcal{F}_s$ defines the state of the algorithm after completing iteration s because, unless a "freezing" happened, SQUEAK and $\overline{\mathbf{Y}}_s$ flip coins with the same probability, and generate the same dictionaries. Since $\widetilde{\tau}_{s,i}$ and $\overline{p}_{s,i,j}$ are computed at the beginning of iteration s using the dictionary $\mathcal{I}_{\{s,l'\}}$ (for some l' unique at layer s), they are fully determined by $\mathcal{F}_{s-1}$. Furthermore, since $\mathcal{F}_{s-1}$ also defines the values of all indicator variables $\overline{z}_{s',i,j}$ up to $\overline{z}_{s-1,i,j}$ for any i and j, we have that all the Bernoulli variables $\overline{z}_{s,i,j}$ at iteration s are conditionally independent given $\mathcal{F}_{s-1}$. In other words, we have that for any i', and j' such that $\{s,1,1\} \leq \{s,i',j'\} < s$ the following random variables are equal in distribution,

$$\overline{z}_{s,i,j}\big|\mathcal{F}_{\{s,i',j'\}} = \overline{z}_{s,i,j}\big|\mathcal{F}_{\{s-1,\nu,\overline{q}\}} \sim Ber\Big(\frac{\overline{p}_{s,i,j}}{\overline{p}_{s-1,i,j}}\Big), \tag{3.9}$$

and for any i', and j' such that $\{s,1,1\} \leq \{s,i',j'\} \leq \{s,\nu,\overline{q}\}$ and $s \neq \{s,i',j'\}$ we have the independence

$$\overline{z}_{s,i,j}\big|\mathcal{F}_{\{s-1,\nu,\overline{q}\}} \perp \overline{z}_{s,i',j'}\big|\mathcal{F}_{\{s-1,\nu,\overline{q}\}}. \tag{3.10}$$

While knowing that $\|\mathbf{Y}_s\| \leq \varepsilon$ is not sufficient to provide guarantees for the approximate probabilities $\widetilde{p}_{s,i}$, we can show that it is enough to prove that the frozen probabilities $\overline{p}_{s,i,j}$ are never too small.

Lemma 3.17 — (Calandriello et al., 2017a). Let $\rho = (1+3\varepsilon)/(1-\varepsilon)$ and $\overline{p}_{s,i,j}$ be the sequence of probabilities generated by the freezing process. Then for any s, i, and j, we have $\overline{p}_{s,i,j} \geq p_{h,i}/\rho = \tau_{h,i}/\rho$.

Proof of Lemma 3.17. Let $\overline{s}$ be the step where the process freezes ($\overline{s} = h$ if it does not freeze), or, in other words, $\|\mathbf{Y}_{\overline{s}}\| < \varepsilon$ and $\|\mathbf{Y}_{\overline{s}+1}\| \geq \varepsilon$. From the definition of $\overline{p}_{s,i,j}$, we have that

$$\begin{aligned}
&\overline{p}_{s,i,j} \geq \overline{p}_{\overline{s},i} = \widetilde{p}_{\overline{s},i} = \max\left\{\min\left\{\widetilde{\tau}_{\overline{s},i},\ \widetilde{p}_{\overline{s}-1,i}\right\},\ \widetilde{p}_{\overline{s}-1,i}/2\right\} \\
&\quad \geq \min\left\{\widetilde{\tau}_{\overline{s},i},\ \widetilde{p}_{\overline{s}-1,i}\right\} = \min\left\{\widetilde{\tau}_{\overline{s},i},\ \widetilde{p}_{\overline{s}-2,i}\right\} = \min\left\{\widetilde{\tau}_{\overline{s},i},\ \widetilde{p}_{\overline{s}-3,i}\right\} \ldots = \min\left\{\widetilde{\tau}_{\overline{s},i},\ \widetilde{p}_{0,i}\right\} = \widetilde{\tau}_{\overline{s},i},
\end{aligned}$$

and therefore $\overline{p}_{s,i,j} \geq \widetilde{\tau}_{\overline{s},i}$. Now let $\{\overline{s}, l'\}$ be the node where $\widetilde{\tau}_{\overline{s},i}$ was computed. We will again drop the $\{h,l\}$ index from $\mathcal{D}_{\{h,l\}}$, and simply refer to it as $\mathcal{D}$. Similarly, we will refer with $\mathcal{D}_i$ to $\mathcal{D}_{\{\overline{s},l'\}}$ (as in, the dataset used to compute $\widetilde{\tau}_{\overline{s},i}$), and with $\overline{\mathcal{D}}_i$ to the samples in $\mathcal{D}$ not contained in $\mathcal{D}_i$ (complement of $\mathcal{D}_i$).
Define $\mathbf{A}$ as the $|\mathcal{D}| \times |\mathcal{D}_i|$ matrix that contains the columns of $\mathbf{S}_{\overline{s}}$ related to points in $\mathcal{D}_i$, and similarly define $\mathbf{B}$ as the $|\mathcal{D}| \times |\overline{\mathcal{D}}_i|$ matrix that contains the columns of $\mathbf{S}_{\overline{s}}$ related to points in $\overline{\mathcal{D}}_i$, where $\mathbf{S}_{\overline{s}}$ can be reconstructed by interleaving columns of $\mathbf{A}$ and $\mathbf{B}$. From its definition in Eq. 3.5, we know that $\widetilde{\tau}_{s,i}$ is computed by SQUEAK as

$$\widetilde{\tau}_{s,i} = (1-\varepsilon)\phi_i^\intercal \left(\Phi_{\mathcal{D}}\mathbf{A}\mathbf{A}^\intercal\Phi_{\mathcal{D}}^\intercal + (1+\varepsilon)\gamma\mathbf{\Pi}_{\mathcal{D}_i}\right)^{-1}\phi_i,$$

using only the points in $\mathbf{A}$ that are available at node $\{\overline{s}, l'\}$. From Lemma 2.10 we know that

$$\begin{aligned}
&\|\mathbf{Y}_{\overline{s}}\| = \left\|\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\overline{s}}^{\{h,l\}}\right\| 2 \\
&= \left\|(\Phi_{\mathcal{D}}\Phi_{\mathcal{D}}^\intercal + \gamma\mathbf{\Pi}_{\mathcal{D}})^{-1/2}(\Phi_{\mathcal{D}}\Phi_{\mathcal{D}}^\intercal - \Phi_{\mathcal{D}}\mathbf{S}_{\overline{s}}\mathbf{S}_{\overline{s}}^\intercal\Phi_{\mathcal{D}}^\intercal)(\Phi_{\mathcal{D}}\Phi_{\mathcal{D}}^\intercal + \gamma\mathbf{\Pi}_{\mathcal{D}_i})^{-1/2}\right\| 2 \\
&= \left\|(\Phi_{\mathcal{D}}\Phi_{\mathcal{D}}^\intercal + \gamma\mathbf{\Pi}_{\mathcal{D}})^{-1/2}(\Phi_{\mathcal{D}}\Phi_{\mathcal{D}}^\intercal - \Phi_{\mathcal{D}}\mathbf{A}\mathbf{A}^\intercal\Phi_{\mathcal{D}}^\intercal - \Phi_{\mathcal{D}}\mathbf{B}\mathbf{B}^\intercal\Phi_{\mathcal{D}}^\intercal)(\Phi_{\mathcal{D}}\Phi_{\mathcal{D}}^\intercal + \gamma\mathbf{\Pi}_{\mathcal{D}})^{-1/2}\right\| 2 \leq \varepsilon
\end{aligned}$$

and we know that this implies

$$\Phi_{\mathcal{D}}\mathbf{A}\mathbf{A}^\intercal\Phi_{\mathcal{D}}^\intercal \preceq \Phi_{\mathcal{D}}\Phi_{\mathcal{D}}^\intercal + \varepsilon(\Phi_{\mathcal{D}}\Phi_{\mathcal{D}}^\intercal + \gamma\mathbf{\Pi}_{\mathcal{D}}) - \Phi_{\mathcal{D}}\mathbf{B}\mathbf{B}^\intercal\Phi_{\mathcal{D}}^\intercal \preceq \Phi_{\mathcal{D}}\Phi_{\mathcal{D}}^\intercal + \varepsilon(\Phi_{\mathcal{D}}\Phi_{\mathcal{D}}^\intercal + \gamma\mathbf{\Pi}_{\mathcal{D}}).$$

Plugging it in the initial definition,

$$\begin{aligned}
\widetilde{\tau}_{s,i} &= (1-\varepsilon)\phi_i^\intercal \left(\Phi_{\mathcal{D}}\mathbf{A}\mathbf{A}^\intercal\Phi_{\mathcal{D}}^\intercal + (1+\varepsilon)\gamma\mathbf{\Pi}_{\mathcal{D}_i}\right)^{-1}\phi_i \\
&\geq (1-\varepsilon)\phi_i^\intercal \left(\Phi_{\mathcal{D}}\mathbf{A}\mathbf{A}^\intercal\Phi_{\mathcal{D}}^\intercal + (1+\varepsilon)\gamma\mathbf{\Pi}_{\mathcal{D}}\right)^{-1}\phi_i \\
&\geq (1-\varepsilon)\phi_i^\intercal(\Phi_{\mathcal{D}}\Phi_{\mathcal{D}}^\intercal + \varepsilon(\Phi_{\mathcal{D}}\Phi_{\mathcal{D}}^\intercal + \gamma\mathbf{\Pi}_{\mathcal{D}}) + (1+\varepsilon)\gamma\mathbf{\Pi}_{\mathcal{D}})^{-1}\phi_i \\
&\geq (1-\varepsilon)\frac{1}{1+3\varepsilon}\phi_i^\intercal(\Phi_{\mathcal{D}}\Phi_{\mathcal{D}}^\intercal + \gamma\mathbf{\Pi}_{\mathcal{D}})^{-1}\phi_i \geq \frac{1-\varepsilon}{1+3\varepsilon}\tau_{h,i} \geq \tau_{h,i}/\rho.
\end{aligned}$$

■

This result is weaker then Lemma 3.3 and Lemma 3.14, since we do not provide an upper bound, and only show that $\widetilde{p}_{s,i} \geq p_{h,i}/\rho$ and not $\widetilde{p}_{s,i} \geq p_{s,i}/\rho$, but it guarantees that the probabilities used at any intermediate layer s are bigger than a fraction $1/\rho$ of the exact *final* $p_{h,i}$ probabilities that we would use at layer h, which will suffice for our purpose.

We now proceed by studying the process $\{\overline{\mathbf{Y}}_s\}_{s=1}^h$ and showing that it is a bounded martingale. In order to show that $\overline{\mathbf{Y}}_s$ is a martingale, it is sufficient to verify the following (equivalent) conditions

$$\mathbb{E}\left[\overline{\mathbf{Y}}_s \mid \mathcal{F}_{s-1}\right] = \overline{\mathbf{Y}}_{s-1} \quad \Leftrightarrow \quad \mathbb{E}\left[\overline{\mathbf{X}}_{\{s,i,j\}} \mid \mathcal{F}_{s-1}\right] = \mathbf{0}.$$

We begin by inspecting the conditional random variable $\overline{\mathbf{X}}_{\{s,i,j\}}|\mathcal{F}_{s-1}$. Given the definition of $\overline{\mathbf{X}}_{\{s,i,j\}}$, the conditioning on $\mathcal{F}_{s-1}$ determines the values of $\overline{z}_{s-1,i,j}$ and the approximate probabilities $\overline{p}_{s-1,i,j}$ and $\overline{p}_{s,i,j}$. In fact, remember that these quantities are fully determined by the realizations in $\mathcal{F}_{s-1}$ which are contained in $\mathcal{F}_{s-1}$. As a result, the only stochastic quantity in $\overline{\mathbf{X}}_{\{s,i,j\}}$ is the variable $\overline{z}_{s,i,j}$. Specifically, if $\|\overline{\mathbf{Y}}_{s-1}\| \geq \varepsilon$, then we have $\overline{p}_{s,i,j} = \overline{p}_{s-1,i,j}$ and $\overline{z}_{s,i,j} = \overline{z}_{s-1,i,j}$ (the process is stopped), and the martingale requirement $\mathbb{E}\left[\overline{\mathbf{X}}_{\{s,i,j\}} \mid \mathcal{F}_{s-1}\right] = \mathbf{0}$ is trivially satisfied. On the other hand, if $\|\overline{\mathbf{Y}}_{s-1}\| \leq \varepsilon$ we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{u_{s,i,j}}\left[\frac{1}{\overline{q}}\left(\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}} - \frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}}\right)\mathbf{c}_i\mathbf{c}_i^\intercal \,\middle|\, \mathcal{F}_{s-1}\right] \\
&\quad = \frac{1}{\overline{q}}\left(\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}} - \frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s,i,j}}\,\mathbb{E}\left[\mathbb{I}\left\{u_{s,i,j} \leq \frac{\overline{p}_{s,i,j}}{\overline{p}_{s-1,i,j}}\right\} \,\middle|\, \mathcal{F}_{s-1}\right]\right)\mathbf{c}_i\mathbf{c}_i^\intercal \\
&\quad = \frac{1}{\overline{q}}\left(\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}} - \frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s,i,j}}\frac{\overline{p}_{s,i,j}}{\overline{p}_{s-1,i,j}}\right)\mathbf{c}_i\mathbf{c}_i^\intercal = \mathbf{0},
\end{aligned}$$

where we use the recursive definition of $\overline{z}_{s,i,j}$ and the fact that $u_{s,i,j}$ is a uniform random variable in $[0,1]$. This proves that $\overline{\mathbf{Y}}_s$ is indeed a martingale. We now compute an upper-bound R on the norm of the values of the difference process as

$$\begin{aligned}
\|\overline{\mathbf{X}}_{\{s,i,j\}}\| &= \frac{1}{\overline{q}}\left|\left(\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}} - \frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}}\right)\right| \|\mathbf{c}_i\mathbf{c}_i^\intercal\| \\
&\leq \frac{1}{\overline{q}}\frac{1}{\overline{p}_{s,i,j}}\|\mathbf{c}_i\mathbf{c}_i^\intercal\| = \frac{1}{\overline{q}}\frac{1}{\overline{p}_{s,i,j}}\tau_{h,i} \leq \frac{1}{\overline{q}}\frac{\rho}{\tau_{h,i}}\tau_{h,i} = \frac{\rho}{\overline{q}} \stackrel{\text{def}}{=} R,
\end{aligned}$$

where we used Lemma 3.17 to bound $\overline{p}_{s,i,j} \leq \tau_{h,i}/\rho$. If instead, $\|\overline{\mathbf{Y}}_{s-1}\| \geq \varepsilon$, the process is stopped and $\|\overline{\mathbf{X}}_s\| = \|\mathbf{0}\| = 0 \leq R$.

We are now ready to use a Freedman matrix inequality from (Tropp, 2011) to bound the norm of $\overline{\mathbf{Y}}$.

Proposition 3.18 — (Tropp, 2011, Theorem 1.2). Consider a matrix martingale $\{\mathbf{Y}_k : k = 0, 1, 2, \dots\}$ whose values are self-adjoint matrices with dimension d, and let $\{\mathbf{X}_k : k = 1, 2, 3, \dots\}$ be the difference sequence. Assume that the difference sequence is uniformly bounded in the sense that

$$\|\mathbf{X}_k\|_2 \leq R \quad \text{almost surely} \quad \text{for } k = 1, 2, 3, \dots.$$

Define the predictable quadratic variation process of the martingale as

$$\mathbf{W}_k \stackrel{\text{def}}{=} \sum_{j=1}^{k} \mathbb{E}\left[\mathbf{X}_j^2 \,\middle|\, \{\mathbf{X}_s\}_{s=0}^{j-1}\right], \quad \text{for } k = 1, 2, 3, \dots.$$

Then, for all $\varepsilon \geq 0$ and $\sigma^2 > 0$,

$$\mathbb{P}\left(\exists k \geq 0 : \|\mathbf{Y}_k\|_2 \geq \varepsilon \ \cap \ \|\mathbf{W}_k\| \leq \sigma^2\right) \leq 2d \cdot \exp\left\{-\frac{\varepsilon^2/2}{\sigma^2 + R\varepsilon/3}\right\}.$$

In order to use the previous inequality, we develop the probability of error for any fixed h as

$$\begin{aligned}
\mathbb{P}\left(\|\mathbf{Y}_h\| \geq \varepsilon\right) &\leq \mathbb{P}\left(\|\overline{\mathbf{Y}}_h\| \geq \varepsilon\right) \\
&= \mathbb{P}\left(\|\overline{\mathbf{Y}}_h\| \geq \varepsilon \cap \|\mathbf{W}_h\| \leq \sigma^2\right) + \mathbb{P}\left(\|\overline{\mathbf{Y}}_h\| \geq \varepsilon \cap \|\mathbf{W}_h\| \geq \sigma^2\right) \\
&\leq \underbrace{\mathbb{P}\left(\|\overline{\mathbf{Y}}_h\| \geq \varepsilon \cap \|\mathbf{W}_h\| \leq \sigma^2\right)}_{\text{(a)}} + \underbrace{\mathbb{P}\left(\|\mathbf{W}_h\| \geq \sigma^2\right)}_{\text{(b)}}.
\end{aligned}$$

Using the bound on $\|\overline{\mathbf{X}}_{\{s,i,j\}}\|_2$, we can directly apply Proposition 3.18 to bound (a) for any fixed σ^2. To bound the part (b), we need to bound w.h.p. the norm $\|\mathbf{W}_h\|$. The proofs for KORS and SQUEAK will now diverge. For KORS we can show the following deterministic result, proved later in Section 3.4.4.

Lemma 3.19 — Uniform bound on norm of the predictable quadratic variation process of KORS.

$$\|\mathbf{W}_h\| \leq \frac{\rho}{\overline{q}}$$

Unfortunately, in the worst case $\|\mathbf{W}_h\|$ for SQUEAK might grow as n, and finding a small deterministic bound is impossible. Instead we use the following high probability result, proved later in Section 3.4.4.

Lemma 3.20 — Low probability of the large norm of the predictable quadratic variation process of SQUEAK.

$$\mathbb{P}\left(\|\mathbf{W}_h\| \geq \frac{6\rho}{\overline{q}}\right) \leq n \cdot \exp\left\{-2\frac{\overline{q}}{\rho}\right\}$$

Therefore for SQUEAK, combining Proposition 3.18 with $\sigma^2 = 6\rho/\overline{q}$, Lemma 3.20, the fact that $2\varepsilon/3 \leq 1$ and the value used by SQUEAK $\overline{q} \geq 26\rho\log(3n/\delta)/\varepsilon^2$ we obtain

$$\begin{aligned}
\mathbb{P}\left(\|\mathbf{P}_{\{h,l\}} - \widetilde{\mathbf{P}}_{\{h,l\}}\|_2 \geq \varepsilon\right) &= \mathbb{P}\left(\|\mathbf{Y}_h\| \geq \varepsilon\right) \leq \mathbb{P}\left(\|\overline{\mathbf{Y}}_h\| \geq \varepsilon \cap \|\mathbf{W}_h\| \leq \sigma^2\right) + \mathbb{P}\left(\|\mathbf{W}_h\| \geq \sigma^2\right) \\
&\leq 2\nu \cdot \exp\left\{-\frac{\varepsilon^2\overline{q}}{\rho}\left(\frac{1}{12 + 2\varepsilon/3}\right)\right\} + n \cdot \exp\left\{-2\frac{\overline{q}}{\rho}\right\} \\
&\leq 3n \cdot \exp\left\{-\frac{\varepsilon^2}{13\rho}\overline{q}\right\} = 3n \cdot \exp\left\{-2\log\left(\frac{3n}{\delta}\right)\right\} \\
&= 3n \cdot \exp\left\{-\log\left(\left(\frac{3n}{\delta}\right)^2\right)\right\} = 3n\frac{\delta^2}{9n^2} \leq \frac{\delta}{2n}.
\end{aligned}$$

This, combined with the fact that $k \leq n$ since at most we can split our dataset in n parts, concludes this part of the proof.

For KORS, using $\sigma^2 = \rho/\overline{q}$, replacing Lemma 3.20 with Lemma 3.19 and the value used by KORS $\overline{q} \geq 4\log(2n/\delta)/\varepsilon^2$ we obtain the same result.

3.4.4 Bound on predictable quadratic variation $\mathbf{W}_h$

Step 1 (applying the definition). We start by writing out $\mathbf{W}_r$ for the process $\overline{\mathbf{Y}}_s$,

$$\mathbf{W}_r = \frac{1}{\overline{q}^2} \sum_{\{s,i,j\} \leq r} \mathbb{E}\left[\left(\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}} - \frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}} \right)^2 \middle| \mathcal{F}_{\{s,i,j\}-1} \right] \mathbf{c}_i \mathbf{c}_i^\mathsf{T} \mathbf{c}_i \mathbf{c}_i^\mathsf{T}.$$

We rewrite the expectation terms in the equation above as

$$\begin{aligned}
&\mathbb{E}\left[\left(\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}} - \frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}} \right)^2 \middle| \mathcal{F}_{\{s,i,j\}-1} \right] \\
&= \mathbb{E}\left[\frac{\overline{z}^2_{s-1,i,j}}{\overline{p}^2_{s-1,i,j}} - 2\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}} \frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}} + \frac{\overline{z}^2_{s,i,j}}{\overline{p}^2_{s,i,j}} \middle| \mathcal{F}_{\{s,i,j\}-1} \right] \\
&\overset{(a)}{=} \mathbb{E}\left[\frac{\overline{z}^2_{s-1,i,j}}{\overline{p}^2_{s-1,i,j}} - 2\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}} \frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}} + \frac{\overline{z}^2_{s,i,j}}{\overline{p}^2_{s,i,j}} \middle| \mathcal{F}_{s-1} \right] \\
&= \frac{\overline{z}^2_{s-1,i,j}}{\overline{p}^2_{s-1,i,j}} - 2\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}} \frac{1}{\overline{p}_{s,i,j}} \mathbb{E}\left[\overline{z}_{s,i,j} \mid \mathcal{F}_{s-1} \right] + \frac{1}{\overline{p}^2_{s,i,j}} \mathbb{E}\left[\overline{z}^2_{s,i,j} \mid \mathcal{F}_{s-1} \right] \\
&\overset{(b)}{=} \frac{\overline{z}_{s-1,i,j}}{\overline{p}^2_{s-1,i,j}} - 2\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}} \frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}} + \frac{1}{\overline{p}^2_{s,i,j}} \mathbb{E}\left[\overline{z}_{s,i,j} \mid \mathcal{F}_{s-1} \right] \\
&= \frac{1}{\overline{p}^2_{s,i,j}} \mathbb{E}\left[\overline{z}_{s,i,j} \mid \mathcal{F}_{s-1} \right] - \frac{\overline{z}_{s-1,i,j}}{\overline{p}^2_{s-1,i,j}} \\
&\overset{(c)}{=} \frac{1}{\overline{p}_{s,i,j}} \frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}} - \frac{\overline{z}_{s-1,i,j}}{\overline{p}^2_{s-1,i,j}} = \frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}} \left(\frac{1}{\overline{p}_{s,i,j}} - \frac{1}{\overline{p}_{s-1,i,j}} \right),
\end{aligned}$$

where in (a) we use the fact that the approximate probabilities $\overline{p}_{s-1,i,j}$ and $\overline{p}_{s,i,j}$ and $\overline{z}_{s-1,i,j}$ are fixed at the end of the previous iteration, while in (b) and (c) we use the fact that $\overline{z}_{s,i,j}$ is a Bernoulli of parameter $\overline{p}_{s,i,j}/\overline{p}_{s-1,i,j}$ (whenever $\overline{z}_{s-1,i,j}$ is equal to 1). Therefore, we can write $\mathbf{W}_r$ at the end of the process as

$$\mathbf{W}_h = \mathbf{W}_{\{h,m,\overline{q}\}} = \frac{1}{\overline{q}^2} \sum_{j=1}^{\overline{q}} \sum_{i=1}^{\nu} \sum_{s=1}^{h} \frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}} \left(\frac{1}{\overline{p}_{s,i,j}} - \frac{1}{\overline{p}_{s-1,i,j}} \right) \mathbf{c}_i \mathbf{c}_i^\mathsf{T} \mathbf{c}_i \mathbf{c}_i^\mathsf{T}.$$

Before continuing with the other steps necessary to bound $\mathbf{W}_h$ for SQUEAK, we take a moment to analyze a simpler problem, showing the deterministic bound on the variance of KORS.

Proof of Lemma 3.19. Since KORS does not update its weights after the insertion, i.e., $\overline{z}_{s,i,j} = \overline{z}_{s',i,j}$ and $\overline{p}_{s,i,j} = \overline{p}_{s',i,j}$ for all $s' > s$, we can greatly simplify the notation.
We only flip coins for atom i at step i, so we can simplify the notation $\overline{z}_{s,i,j}$ to $\overline{z}_{s,j} = \overline{z}_{s,s,j}$. Similarly, the probabilities do not change at all across iteration, so we can drop the per-copy index, leaving only the time index that will serve as both time and atom index $\overline{p}_{s,i,j} = \overline{p}_{s,i} = \overline{p}_{s,s} = \overline{p}_s$.

We can therefore simplify the definition of the predictable quadratic variation

$$\begin{aligned}
\mathbf{W}_h &= \frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\sum_{s=1}^{h}\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}}\left(\frac{1}{\overline{p}_{s,i,j}}-\frac{1}{\overline{p}_{s-1,i,j}}\right)\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal \\
&= \frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\sum_{s=1}^{h}\overline{z}_{i,i,j}\left(\frac{1}{\overline{p}_{s,i,j}\overline{p}_{s-1,i,j}}-\frac{1}{\overline{p}^2_{s-1,i,j}}\right)\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal \\
&= \frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\frac{\overline{z}_{i,j}}{\overline{p}_{i,i,j}}\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal = \frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\frac{\overline{z}_{i,j}}{\overline{p}_i}\tau_{h,i}\mathbf{c}_i\mathbf{c}_i^\intercal,
\end{aligned}$$

where we used the fact that the probabilities $\overline{p}_{h,i}=\overline{p}_{h-1,i}=\overline{p}_{i,i}$ are not updated for $s>i$, and therefore $\frac{1}{\overline{p}_{s,i,j}\overline{p}_{s-1,i,j}}=\frac{1}{\overline{p}^2_{s-1,i,j}}$, and that for $s=i$ we have $\overline{p}_{i-1,i}=1$ by definition. We can now bound this quantity as

$$\begin{aligned}
\left\|\overline{\mathbf{W}}_h\right\| &= \frac{1}{\overline{q}^2}\left\|\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\frac{\overline{z}_{i,j}}{\overline{p}_i}\tau_{h,i}\mathbf{c}_i\mathbf{c}_i^\intercal\right\| \le \frac{1}{\overline{q}^2}\left\|\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\frac{1}{\overline{p}_i}\tau_{h,i}\mathbf{c}_i\mathbf{c}_i^\intercal\right\| \le \frac{1}{\overline{q}^2}\left\|\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\rho\frac{\tau_{h,i}}{\tau_{h,i}}\mathbf{c}_i\mathbf{c}_i^\intercal\right\| \\
&\le \frac{\rho}{\overline{q}}\left\|\sum_{s=1}^{\nu}\mathbf{c}_s\mathbf{c}_s^\intercal\right\| = \frac{\rho}{\overline{q}}\left\|\mathbf{C}_t\mathbf{C}_t^\intercal\right\| = \frac{\rho}{\overline{q}}\left\|\mathbf{P}_t\right\| \le \frac{\rho}{\overline{q}} := \sigma^2.
\end{aligned}$$

Where we used Lemma 3.17 to bound $1/\overline{p}_i \le \rho/\tau_{h,i}$. ■

In a nutshell, this bound show us that if we can bound the chain $\sum_{s=1}^{h}\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}}\left(\frac{1}{\overline{p}_{s,i,j}}-\frac{1}{\overline{p}_{s-1,i,j}}\right)$ with $1/\tau_{h,i}$, then we can find a good bound on the variance.
Note that we could unroll and simplify $\sum_{s=1}^{h}\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}}\left(\frac{1}{\overline{p}_{s,i,j}}-\frac{1}{\overline{p}_{s-1,i,j}}\right)$ only because KORS does not change the probabilities $\overline{p}_{s,i,j}$, and $\overline{z}_{s,i,j}$ is constant after $s=i$. In SQUEAK these probabilities continue to grow, and if the variable $\overline{z}_{h,i,j}$ is still non-zero at the end of the chain, after unrolling and simplifying we are left with

$$\left(\frac{1}{\overline{p}_{h,i,j}\overline{p}_{h-1,i,j}}-\frac{1}{\overline{p}^2_{1,i,j}}\right)=\left(\frac{1}{\overline{p}_{h,i,j}\overline{p}_{h-1,i,j}}-1\right)\le\frac{1}{\overline{p}^2_{h,i,j}},$$

which is usually much larger than the $1/\overline{p}_{i,i,j}$ that we obtained for KORS, since $\tau_{h,i}\le\tau_{i,i}$. Therefore, in the worst case we have a variance that scales with $1/\overline{p}^2_{h,i,j}\le\rho^2/\tau^2_{h,i}$, making it impossible to prove a deterministic bound of the order of $1/\tau_{h,i}$.
To tame the larger variance of SQUEAK we continue our derivation looking for a more sophisticated bound that holds not uniformly but in high probability.

Step 2 (a preliminary bound). We can now upper-bound $\mathbf{W}_h$ as

$$\begin{aligned}
\mathbf{W}_h &\preceq \frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\sum_{s=1}^{h}\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}}\left(\frac{1}{\overline{p}_{s,i,j}}-\frac{1}{\overline{p}_{s-1,i,j}}\right)\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal \\
&= \frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\left(\frac{\overline{z}_{h,i,j}}{\overline{p}_{h,i,j}^2}-\frac{\overline{z}_{h,i,j}}{\overline{p}_{h,i,j}^2}+\sum_{s=1}^{h}\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s-1,i,j}}\left(\frac{1}{\overline{p}_{s,i,j}}-\frac{1}{\overline{p}_{s-1,i,j}}\right)\right)\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal \\
&= \frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\left(\frac{\overline{z}_{h,i,j}}{\overline{p}_{h,i,j}^2}+\left(\sum_{s=1}^{h}-\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}^2}+\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s,i,j}\overline{p}_{s-1,i,j}}\right)-\frac{\overline{z}_{0,i,j}}{\overline{p}_{0,i,j}^2}\right)\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal \\
&\preceq \frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\left(\frac{\overline{z}_{h,i,j}}{\overline{p}_{h,i,j}^2}+\left(\sum_{s=1}^{h}\frac{\overline{z}_{s-1,i,j}}{\overline{p}_{s,i,j}\overline{p}_{s-1,i,j}}-\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}\overline{p}_{s-1,i,j}}\right)\right)\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal \\
&= \frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\left(\frac{\overline{z}_{h,i,j}}{\overline{p}_{h,i,j}^2}+\sum_{s=1}^{h}\frac{\overline{z}_{s-1,i,j}(1-\overline{z}_{s,i,j})}{\overline{p}_{s,i,j}\overline{p}_{s-1,i,j}}\right)\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal,
\end{aligned}$$

where in the inequality we use the fact $\overline{p}_{s,i,j} \leq \overline{p}_{s-1,i,j}$. From the definition of $\overline{p}_{s,i,j}$, we know that when $\overline{z}_{s,i,j} = 0$, $\overline{p}_{s,i,j} = \overline{p}_{s-1,i,j}$. Therefore $\frac{\overline{z}_{s-1,i,j}(1-\overline{z}_{s,i,j})}{\overline{p}_{s,i,j}\overline{p}_{s-1,i,j}} = \frac{\overline{z}_{s-1,i,j}(1-\overline{z}_{s,i,j})}{\overline{p}_{s-1,i,j}^2}$, since the term is non-zero only when $\overline{z}_{s,i,j} = 0$, i.e., dropped copies are not reweighted. Finally, we see that only one of the $\overline{z}_{s-1,i,j}(1-\overline{z}_{s,i,j})$ terms can be active for $s \in [h]$ and thus

$$\begin{aligned}
\mathbf{W}_h &\preceq \frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\left(\frac{\overline{z}_{h,i,j}}{\overline{p}_{h,i,j}^2}+\sum_{s=1}^{h}\frac{\overline{z}_{s-1,i,j}(1-\overline{z}_{s,i,j})}{\overline{p}_{s-1,i,j}^2}\right)\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal \\
&= \frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\left(\max\left\{\max_{s=1,\dots,h}\left\{\frac{\overline{z}_{s-1,i,j}(1-\overline{z}_{s,i,j})}{\overline{p}_{s-1,i,j}^2}\right\};\frac{\overline{z}_{h,i,j}}{\overline{p}_{h,i,j}^2}\right\}\right)\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal \\
&= \frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal\left(\max_{s=0,\dots,h}\left\{\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}^2}\right\}\right). \qquad 3.11
\end{aligned}$$

Step 3 (introduction of a stochastically dominant process). We want to study $\max_{s=0,\dots,h}\left\{\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}^2}\right\}$. To simplify notation, we will consider $\max_{s=0,\dots,h}\left\{\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}}\right\}$, where we removed the square, which will be re-added in the end. We know trivially that this quantity is larger or equal than, 1 because $\overline{z}_{0,i,j}/\overline{p}_{0,i,j} = 1$, but upper-bounding this quantity is not trivial as the evolution of the various $\overline{p}_{s,i,j}$ depends in a complex way on the interaction between the random variables $\overline{z}_{s,i,j}$. Nonetheless, whenever $\overline{p}_{s,i,j}$ is significantly smaller than $\overline{p}_{s-1,i,j}$, the probability of keeping a copy of point i at iteration s (i.e., $\overline{z}_{s,i,j} = 1$) is also very small. As a result, we expect the ratio $\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}}$ to be still small with high probability.

Unfortunately, due to the dependence between different copies of the point at different iterations, it seems difficult to exploit this intuition directly to provide an overall high-probability bound on $\mathbf{W}_h$. For this reason, we simplify the analysis by replacing each of the (potentially dependent) chains $\{\overline{z}_{s,i,j}/\overline{p}_{s,i,j}\}_{s=0}^{h}$ with a set of (independent) random variables $w_{0,i,j}$ that will stochastically dominate them.

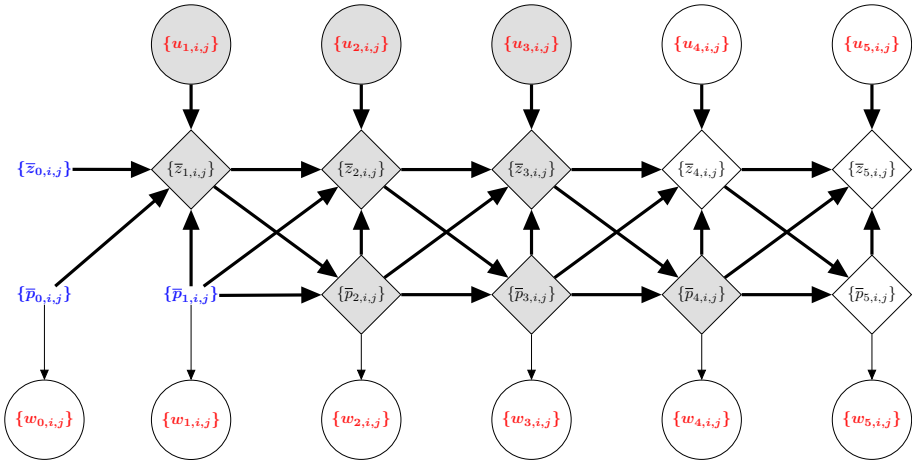


Figure 3.5: The dependency graph of the considered variables. Red variables are random. Black variables are deterministically computed using their input (a function of their input), with bold lines indicating the deterministic (functional) relation. Blue variables are constants. A grey filling indicates that a random variable is observed or a function of observed variables.

We define the random variable $w_{s,i,j}$ using the following conditional distribution,[7]

$$\mathbb{P}\left(\frac{1}{w_{s,i,j}} \leq a \,\middle|\, \mathcal{F}_s\right) = \begin{cases} 0 & \text{for} \quad a < 1/\overline{p}_{s,i,j} \\ 1 - \frac{1}{\overline{p}_{s,i,j}\, a} & \text{for} \quad 1/\overline{p}_{s,i,j} \leq a < \rho/p_{h,i} \\ 1 & \text{for} \quad \rho/p_{h,i} \leq a \end{cases}.$$

To show that this distribution is well defined, we use Lemma 3.17 to guarantee that $1/\overline{p}_{s,i,j} \leq a < \rho/p_{h,i}$. Note that the distribution of $\frac{1}{w_{s,i,j}}$ conditioned on $\mathcal{F}_s$ is determined by only $\overline{p}_{s,i,j}$, $p_{h,i}$, and ρ, where $p_{h,i}$ and ρ are fixed. Remembering that $\overline{p}_{s,i,j}$ is a function of $\mathcal{F}_{s-1}$ (computed using the previous iteration), we have that

$$\mathbb{P}\left(\frac{1}{w_{s,i,j}} \leq a \,\middle|\, \mathcal{F}_s\right) = \mathbb{P}\left(\frac{1}{w_{s,i,j}} \leq a \,\middle|\, \mathcal{F}_{s-1}\right).$$

Notice that in the definition of $w_{s,i,j}$, none of the other $w_{s',i',j'}$ (for any different s', i', or j') appears and $\overline{p}_{s,i,j}$ is a function of $\mathcal{F}_{s-1}$. It follows that given $\mathcal{F}_{s-1}$, $w_{s,i,j}$ is independent from all other $w_{s',i',j'}$ (for any different s', i', or j'). This is easier to see in the probabilistic graphical model reported in Fig. 3.5, which illustrates the dependence between the various variables.

Finally for the special case $w_{0,i,j}$ the definition above reduces to

$$\mathbb{P}\left(\frac{1}{w_{0,i,j}} \leq a\right) = \begin{cases} 0 & \text{for} \quad a < 1 \\ 1 - \frac{1}{a} & \text{for} \quad 1 \leq a < \rho/p_{h,i} \\ 1 & \text{for} \quad \rho/p_{h,i} \leq a \end{cases}, \tag{3.12}$$

[7]Notice that unlike $\overline{z}_{s,i,j}$, $w_{s,i,j}$ is no longer $\mathcal{F}_s$-measurable but it is $\mathcal{F}'_s$-measurable, where

$$\mathcal{F}'_{\{s,i,j\}} = \left\{u_{s',i',j'},\ \forall\{s',i',j'\} \leq \{s,i,j\}\right\} \cup \{w_{s,i,j}\} = \mathcal{F}_{\{s,i,j\}} \cup \{w_{s,i,j}\}.$$

since $\overline{p}_{0,i,j} = 1$ by definition. Due to its definition, $1/w_{0,i,j}$ is a truncated Pareto r.v., with scale 1 and shape 1, that gets truncated at $\rho/p_{h,i}$.
We can also see from this definition that $w_{0,i,j}$ and $w_{0,i',j'}$ are all independent, and this will allow us to use stronger concentration inequalities for independent random variables.

Step 4 Proving the dominance. We remind the reader that a random variable A stochastically dominates random variable B, if for all values a the two equivalent conditions are verified,

$$\mathbb{P}(A \geq a) \geq \mathbb{P}(B \geq a) \Leftrightarrow \mathbb{P}(A \leq a) \leq \mathbb{P}(B \leq a).$$

As a consequence, if A dominates B, the following implication holds,

$$\mathbb{P}(A \geq a) \geq \mathbb{P}(B \geq a) \implies \mathbb{E}[A] \geq \mathbb{E}[B],$$

while the reverse (A dominates B, if $\mathbb{E}[A] \geq \mathbb{E}[B]$) is not true in general. Following this definition of stochastic dominance, our goal is to prove

$$\mathbb{P}\left(\max_{s=0,\dots,h} \frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}} \leq a\right) \geq \mathbb{P}\left(\frac{1}{w_{0,i,j}} \leq a\right).$$

We prove this inequality by proceeding backwards with a sequence of conditional probabilities. We first study the distribution of the maximum conditional to the state of the algorithm at the end of iteration h, i.e., $\mathcal{F}_h$. From the definition of $w_{h,i,j}$, we know that, w.p. 1, $1/\overline{p}_{h,i} \leq 1/w_{h,i,j}$. Therefore,

$$\mathbb{P}\left(\max_{s=0,\dots,h} \frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}} \leq a\right) \geq \mathbb{P}\left(\max\left\{\max_{s=0,\dots,h-1} \frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}}; \frac{\overline{z}_{h,i,j}}{w_{h,i,j}}\right\} \leq a\right).$$

Now focus on an arbitrary intermediate step $1 \leq k \leq h$, where we fix $\mathcal{F}_{k-1}$. Since $u_{k,i,j}$ and $w_{k,i,j}$ are independent given $\mathcal{F}_{k-1}$, we have

$$\begin{aligned}
&\mathbb{P}\left(\frac{\overline{z}_{k,i,j}}{w_{k,i,j}} \leq a \,\middle|\, \mathcal{F}_{k-1}\right) = \mathbb{P}\left(\mathbb{I}\left\{u_{k,i,j} \leq \frac{\overline{p}_{k,i,j}}{\overline{p}_{k-1,i,j}}\right\} \frac{1}{w_{k,i,j}} \leq a \,\middle|\, \mathcal{F}_{k-1}\right) \\
&= \begin{cases} 0 & \text{for} \quad a \leq 0 \\ 1 - \frac{\overline{p}_{k,i,j}}{\overline{p}_{k-1,i,j}} & \text{for} \quad 0 \leq a < 1/\overline{p}_{k,i,j} \\ 1 - \frac{\overline{p}_{k,i,j}}{\overline{p}_{k-1,i,j}} + \frac{\overline{p}_{k,i,j}}{\overline{p}_{k-1,i,j}}\left(1 - \frac{1}{\overline{p}_{k,i,j}a}\right) = 1 - \frac{1}{\overline{p}_{k-1,i,j}a} & \text{for} \quad 1/\overline{p}_{k,i,j} \leq a < \rho/p_{h,i} \\ 1 & \text{for} \quad \rho/p_{h,i} \leq a \end{cases} \\
&\geq \begin{cases} 0 & \text{for} \quad a < 1/\overline{p}_{k-1,i,j} \\ 1 - \frac{1}{\overline{p}_{k-1,i,j}a} & \text{for} \quad 1/\overline{p}_{k-1,i,j} \leq a < 1/\overline{p}_{k,i,j} \\ 1 - \frac{1}{\overline{p}_{k-1,i,j}a} & \text{for} \quad 1/\overline{p}_{k,i,j} \leq a < \rho/p_{h,i} \\ 1 & \text{for} \quad \rho/p_{h,i} \leq a \end{cases} \\
&= \mathbb{P}\left(\frac{1}{w_{k-1,i,j}} \leq a \,\middle|\, \mathcal{F}_{k-2}\right) = \mathbb{P}\left(\frac{1}{w_{k-1,i,j}} \leq a \,\middle|\, \mathcal{F}_{k-1}\right),
\end{aligned} \tag{3.13}$$

where the inequality is also represented in Fig. 3.6. We now proceed by peeling off layers from the end of the chain one by one, taking advantage of the dominance we just proved. Fig. 3.6 visualizes one step of the peeling when $\overline{z}_{k-1,i,j} = 1$ (note that the peeling is trivially

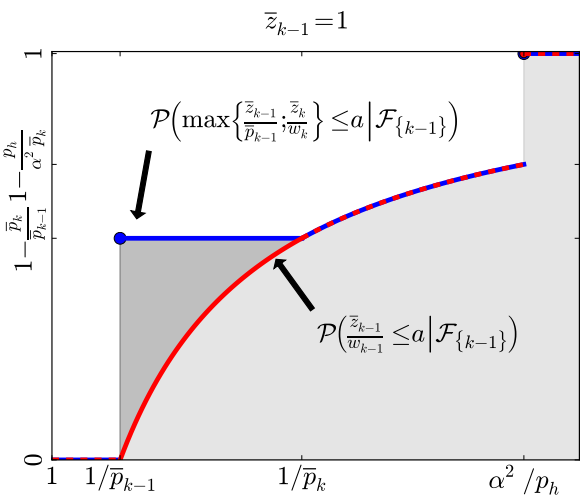


Figure 3.6: C.d.f. of $\max\left\{\overline{z}_{k-1,i,j}/\overline{p}_{t,i,j};\overline{z}_{k,i,j}/\overline{p}_{k,i,j}\right\}$ and $\overline{z}_{k-1,i,j}/w_{k-1,i,j}$ conditioned on $\mathcal{F}_{\{k-1\}}$. For conciseness, we omit the i,j indices.

true when $\overline{z}_{k-1,i,j}=0$ since the whole chain terminated at step $\overline{z}_{k-1,i,j}$). We show how to move from an iteration $k\leq h$ to $k-1$.

$$
\begin{aligned}
&\mathbb{P}\left(\max\left\{\max_{s=0\ldots k-1}\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}};\frac{\overline{z}_{k,i,j}}{w_{k,i,j}}\right\}\leq a\right)\\
&=\mathop{\mathbb{E}}_{\mathcal{F}_{k-1}}\left[\mathbb{P}\left(\max\left\{\max_{s=0\ldots k-1}\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}};\frac{\overline{z}_{k,i,j}}{w_{k,i,j}}\right\}\leq a\;\middle|\;\mathcal{F}_{k-1}\right)\right]\\
&\overset{(a)}{\geq}\mathop{\mathbb{E}}_{\mathcal{F}_{k-1}}\left[\mathbb{P}\left(\max\left\{\max_{s=0\ldots k-1}\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}};\frac{\overline{z}_{k-1,i,j}}{w_{k-1,i,j}}\right\}\leq a\;\middle|\;\mathcal{F}_{k-1}\right)\right]\\
&=\mathop{\mathbb{E}}_{\mathcal{F}_{k-1}}\left[\mathbb{P}\left(\max\left\{\max_{s=0\ldots k-2}\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}};\frac{\overline{z}_{k-1,i,j}}{\overline{p}_{k-1,i,j}};\frac{\overline{z}_{k-1,i,j}}{w_{k-1,i,j}}\right\}\leq a\;\middle|\;\mathcal{F}_{k-1}\right)\right]\\
&=\mathop{\mathbb{E}}_{\mathcal{F}_{k-1}}\left[\mathbb{P}\left(\max\left\{\max_{s=0\ldots k-2}\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}};\overline{z}_{k-1,i,j}\max\left\{\frac{1}{\overline{p}_{k-1,i,j}};\frac{1}{w_{k-1,i,j}}\right\}\right\}\leq a\;\middle|\;\mathcal{F}_{k-1}\right)\right]\\
&\overset{(b)}{=}\mathop{\mathbb{E}}_{\mathcal{F}_{k-1}}\left[\mathbb{P}\left(\max\left\{\max_{s=0\ldots k-2}\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}};\frac{\overline{z}_{k-1,i,j}}{w_{k-1,i,j}}\right\}\leq a\;\middle|\;\mathcal{F}_{k-1}\right)\right]\\
&=\mathbb{P}\left(\max\left\{\max_{s=0\ldots k-2}\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}};\frac{\overline{z}_{k-1,i,j}}{w_{k-1,i,j}}\right\}\leq a\right),
\end{aligned}
$$

where in (a), given $\mathcal{F}_{k-1}$, everything is fixed except $u_{k,i,j}$ and $w_{k,i,j}$ and we can use the stochastic dominance in Eq. 3.13, and in (b) we use the fact that the inner maximum is always attained by $1/w_{k,i,j}$ since by definition $1/w_{k-1,i,j}$ is lower-bounded by $1/\overline{p}_{k-1,i,j}$. Applying the inequality recursively from $k=h$ to $k=1$ removes all $\overline{z}_{s,i,j}$ from the maximum and we are finally left with only $w_{0,i,j}$ as we wanted,

$$
\mathbb{P}\left(\max_{s=0,\ldots,h}\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}}\leq a\right)\geq\mathbb{P}\left(\max\left\{\frac{\overline{z}_{0,i,j}}{\overline{p}_{0,i,j}};\frac{\overline{z}_{0,i,j}}{w_{0,i,j}}\right\}\leq a\right)\geq\mathbb{P}\left(\frac{1}{w_{0,i,j}}\leq a\right),
$$

where in the last inequality we used that $\overline{z}_{0,i,j}=1$ from the definition of the algorithm and $\overline{p}_{0,i,j}=1$ while $w_{0,i,j}\leq 1$ by Eq. 3.12.

Step 5 (stochastic dominance on $\mathbf{W}_h$). Now that we proved the stochastic dominance of $1/w_{0,i,j}$, we plug this result in the definition of $\mathbf{W}_h$. For the sake of notation, we introduce

the term $\overline{p}^{\max}_{h',i,j}$ to indicate the maximum over the first h' step of copy i,j such that

$$\max_{s=0,\dots,h'} \frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}} = \frac{1}{\overline{p}^{\max}_{h',i,j}}.$$

We first notice that while $\overline{\mathbf{Y}}_h$ is not necessarily PSD, $\mathbf{W}_h$ is a sum of PSD matrices. Introducing the function $\Lambda(\{1/\overline{p}^{\max}_{h,i,j}\}_{i,j})$ we can restate Eq. 3.11 as

$$\|\mathbf{W}_h\| = \lambda_{\max}(\mathbf{W}_h) \leq \Lambda(\{1/\overline{p}^{\max}_{h,i,j}\}_{i,j}) \stackrel{\text{def}}{=} \lambda_{\max}\left(\frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\left(\frac{1}{\overline{p}^{\max}_{h,i,j}}\right)^2 \mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal\right).$$

In Step 4, we showed that $1/\overline{p}^{\max}_{h,i,j}$ is stochastically dominated by $1/w_{0,i,j}$ for every i and j. In order to bound $\Lambda(\{1/\overline{p}^{\max}_{h,i,j}\}_{i,j})$, we need to show that this dominance also applies to the summation over all columns inside the matrix norm. We can reformulate $\Lambda(\{1/\overline{p}^{\max}_{h,i,j}\}_{i,j})$ as

$$\begin{aligned}\lambda_{\max}\left(\frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\left(\frac{1}{\overline{p}^{\max}_{h,i,j}}\right)^2 \mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal\right) &= \max_{\mathbf{x}:\|\mathbf{x}\|=1}\mathbf{x}^\intercal\left(\frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\left(\frac{1}{\overline{p}^{\max}_{h,i,j}}\right)^2 \mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal\right)\mathbf{x}\\ = \max_{\mathbf{x}:\|\mathbf{x}\|=1}\frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\left(\frac{1}{\overline{p}^{\max}_{h,i,j}}\right)^2 \|\mathbf{c}_i\|_2^2\mathbf{x}^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{x} &= \max_{\mathbf{x}:\|\mathbf{x}\|=1}\frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\left(\frac{1}{\overline{p}^{\max}_{h,i,j}}\right)^2 (\|\mathbf{c}_i\|_2\mathbf{c}_i^\intercal\mathbf{x})^2.\end{aligned}$$

From this reformulation, it is easy to see that, because $1/\overline{p}^{\max}_{h,i,j}$ is strictly positive, the function $\Lambda(\{1/\overline{p}^{\max}_{h,i,j}\}_{i,j})$ is monotonically increasing w.r.t. the individual $1/\overline{p}^{\max}_{h,i,j}$, or in other words that increasing an $1/\overline{p}^{\max}_{h,i,j}$ without decreasing the others can only increase the maximum. Introducing $\Lambda(\{1/w_{0,i,j}\}_{i,j})$ as

$$\Lambda(\{1/w_{0,i,j}\}_{i,j}) \stackrel{\text{def}}{=} \max_{\mathbf{x}:\|\mathbf{x}\|=1}\frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\left(\frac{1}{w_{0,i,j}}\right)^2 (\|\mathbf{c}_i\|_2\mathbf{c}_i^\intercal\mathbf{x})^2,$$

we now need to prove the stochastic dominance of $\Lambda(\{1/w_{0,i,j}\}_{i,j})$ over $\Lambda(\{1/\overline{p}^{\max}_{h,i,j}\}_{i,j})$. Using the definition of $1/\overline{p}^{\max}_{h,i,j}$, $w_{h,i,j}$, and the monotonicity of Λ we have

$$\begin{aligned}\mathbb{P}\left(\Lambda\left(\left\{\frac{1}{\overline{p}^{\max}_{h,i,j}}\right\}_{i,j}\right) \leq a\right) &= \mathbb{P}\left(\Lambda\left(\left\{\max\left\{\max_{s=0,\dots,h-1}\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}};\frac{\overline{z}_{h,i,j}}{\overline{p}_{h,i,j}}\right\}\right\}_{i,j}\right) \leq a\right)\\ &\geq \mathbb{P}\left(\Lambda\left(\left\{\max\left\{\max_{s=0,\dots,h-1}\frac{\overline{z}_{s,i,j}}{\overline{p}_{s,i,j}};\frac{\overline{z}_{h,i,j}}{w_{h,i,j}}\right\}\right\}_{i,j}\right) \leq a\right).\end{aligned}$$

Now pick $1 \leq k \leq h$, for a fixed $\mathcal{F}_{k-1}$, $\frac{1}{\overline{p}^{\max}_{k-1,i,j}}$ is a constant and $\max\left\{\frac{1}{\overline{p}^{\max}_{k,i,j}};x\right\}$ is a monotonically increasing function in x, making $\Lambda\left(\max\left\{\frac{1}{\overline{p}^{\max}_{k,i,j}};x\right\}\right)$ also an increasing

function. Therefore, we have

$$
\begin{aligned}
&\mathbb{P}\left(\Lambda\left(\left\{\max\left\{\frac{1}{\overline{p}_{k-1,i,j}^{\max}};\frac{\overline{z}_{k,e,j}}{w_{k,i,j}}\right\}\right\}_{i,j}\right)\leq a\right)\\
&=\mathop{\mathbb{E}}_{\mathcal{F}_{k-1}}\left[\mathbb{P}\left(\Lambda\left(\left\{\max\left\{\frac{1}{\overline{p}_{k-1,i,j}^{\max}};\frac{\overline{z}_{k,e,j}}{w_{k,i,j}}\right\}\right\}_{i,j}\right)\leq a\,\middle|\,\mathcal{F}_{k-1}\right)\right]\\
&\overset{(a)}{\geq}\mathop{\mathbb{E}}_{\mathcal{F}_{k-1}}\left[\mathbb{P}\left(\Lambda\left(\left\{\max\left\{\frac{1}{\overline{p}_{k-1,i,j}^{\max}};\frac{\overline{z}_{k-1,i,j}}{w_{k-1,i,j}}\right\}\right\}_{i,j}\right)\leq a\,\middle|\,\mathcal{F}_{k-1}\right)\right]\\
&\overset{(b)}{=}\mathop{\mathbb{E}}_{\mathcal{F}_{k-1}}\left[\mathbb{P}\left(\Lambda\left(\left\{\max\left\{\frac{1}{\overline{p}_{k-2,i,j}^{\max}};\frac{\overline{z}_{k-1,i,j}}{w_{k-1,i,j}}\right\}\right\}_{i,j}\right)\leq a\,\middle|\,\mathcal{F}_{k-1}\right)\right],
\end{aligned}
$$

where inequality (a) follows from the fact that stochastic dominance is preserved by monotonically increasing functions (Levy, 2015), such as Λ, combined with the fact that for a fixed $\mathcal{F}_{k-1}$ the variables $\overline{z}_{k,i,j}$ and $w_{k,i,j}$ are all independent and (b) from the definition of $1/\overline{p}_{k-1,i,j}^{\max}$ and the fact that by definition $1/w_{k-1,i,j}$ is lower-bounded by $1/\overline{p}_{k-1,i,j}$. We can iterate this inequality to obtain the desired result

$$
\begin{aligned}
\mathbb{P}(\|\mathbf{W}_h\|\geq\sigma^2)&\leq\mathbb{P}\left(\Lambda\left(\left\{\frac{1}{\overline{p}_{h,i,j}^{\max}}\right\}_{i,j}\right)\geq\sigma^2\right)\\
&\leq\mathbb{P}\left(\lambda_{\max}\left(\frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\left(\frac{1}{w_{0,i,j}}\right)^2\mathbf{c}_i\mathbf{c}_i^{\mathsf{T}}\mathbf{c}_i\mathbf{c}_i^{\mathsf{T}}\right)\geq\sigma^2\right).
\end{aligned}
$$

Step 6 (concentration inequality). Since all $w_{0,i,j}$ are (unconditionally) independent from each other, we can apply the following theorem.

Proposition 3.21 — Tropp, (2015), Theorem 5.1.1. Consider a finite sequence $\{\mathbf{X}_k : k = 1,2,3,\dots\}$ whose values are independent, random, PSD Hermitian matrices with dimension d. Assume that each term in the sequence is uniformly bounded in the sense that

$$\lambda_{\max}(\mathbf{X}_k)\leq L\quad\text{almost surely}\quad\text{for } k=1,2,3,\dots.$$

Introduce the random matrix $\mathbf{V}\overset{\text{def}}{=}\sum_k\mathbf{X}_k$, and the maximum eigenvalue of its expectation

$$\mu_{\max}\overset{\text{def}}{=}\lambda_{\max}(\mathbb{E}[\mathbf{V}])=\lambda_{\max}\left(\sum_k\mathbb{E}[\mathbf{X}_k]\right).$$

Then, for all $h \geq 0$,

$$\mathbb{P}\left(\lambda_{\max}(\mathbf{V}) \geq (1+h)\mu_{\max}\right) \leq d \cdot \left[\frac{e^h}{(1+h)^{1+h}}\right]^{\frac{\mu_{\max}}{L}}$$
$$\leq d \cdot \exp\left\{-\frac{\mu_{\max}}{L}((h+1)\log(h+1)-h)\right\}.$$

In our case, we have

$$\mathbf{X}_{\{i,j\}} = \frac{1}{\overline{q}^2}\frac{1}{w_{0,i,j}^2}\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal \preceq \frac{1}{\overline{q}^2}\frac{\rho^2}{p_{h,i}^2}\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal \preceq \frac{1}{\overline{q}^2}\frac{\rho^2}{p_{h,i}^2}\|\mathbf{c}_i\mathbf{c}_i^\intercal\|^2\mathbf{I} \preceq \frac{\rho^2}{\overline{q}^2}\mathbf{I},$$

where the first inequality follows from the definition of $w_{0,i,j}$ in Eq. 3.12, the second from the PSD ordering, and the third from the definition of $\|\mathbf{c}_i\mathbf{c}_i^\intercal\|$.

Therefore, we can use $L \stackrel{\text{def}}{=} \rho^2/\overline{q}^2$ for the purpose of Proposition 3.21. We need now to compute $\mathbb{E}\left[\mathbf{X}_k\right]$, that we can use in turn to compute $\mu_{\max}$. We begin by computing the expected value of $1/w_{0,i,j}$. Let us denote the c.d.f. of $1/w_{0,i,j}^2$ as

$$F_{1/w_{0,i,j}^2}(a) = \mathbb{P}\left(\frac{1}{w_{0,i,j}^2} \leq a\right) = \mathbb{P}\left(\frac{1}{w_{0,i,j}} \leq \sqrt{a}\right) = \begin{cases} 0 & \text{for} \quad a < 1 \\ 1 - \frac{1}{\sqrt{a}} & \text{for} \quad 1 \leq a < \rho^2/p_{h,i}^2 \\ 1 & \text{for} \quad \rho^2/p_{h,i}^2 \leq a \end{cases}.$$

Since $\mathbb{P}\left(1/w_{0,i,j}^2 \geq 0\right) = 1$, we have that

$$\mathbb{E}\left[\frac{1}{w_{0,i,j}^2}\right] = \int_{a=0}^{\infty}\left[1 - F_{1/w_{0,i,j}^2}(a)\right]\mathrm{d}a$$
$$= \int_{a=0}^{1}\left(1 - F_{1/w_{0,i,j}^2}(a)\right)\mathrm{d}a + \int_{a=1}^{\frac{\rho^2}{p_{h,i}^2}}\left(1 - F_{1/w_{0,i,j}^2}(a)\right)\mathrm{d}a + \int_{a=\frac{\rho^2}{p_{h,i}^2}}^{\infty}\left(1 - F_{1/w_{0,i,j}^2}(a)\right)\mathrm{d}a$$
$$= \int_{a=0}^{1}(1-0)\,\mathrm{d}a + \int_{a=1}^{\rho^2/p_{h,i}^2}\left(1 - \left(1 - \frac{1}{\sqrt{a}}\right)\right)\mathrm{d}a + \int_{a=\rho^2/p_{h,i}^2}^{\infty}(1-1)\,\mathrm{d}a$$
$$= \int_{a=0}^{1}\mathrm{d}a + \int_{a=1}^{\rho^2/p_{h,i}^2}\frac{1}{\sqrt{a}}\,\mathrm{d}a = 1 + \left[2\sqrt{a}\right]_1^{\rho^2/p_{h,i}^2} = 2\rho/p_{h,i} - 1.$$

Therefore,

$$\mu_{\max} = \lambda_{\max}(\mathbb{E}\left[\mathbf{V}\right]) = \lambda_{\max}\Big(\sum_{\{i,j\}}\mathbb{E}\left[\mathbf{X}_{\{i,j\}}\right]\Big) = \lambda_{\max}\left(\frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\mathbb{E}\left[\frac{1}{w_{0,i,j}^2}\right]\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal\right)$$
$$= \lambda_{\max}\left(\frac{1}{\overline{q}}\sum_{i=1}^{\nu}\left(\frac{2\rho}{p_{h,i}} - 1\right)p_{h,i}\mathbf{c}_i\mathbf{c}_i^\intercal\right) \leq \lambda_{\max}\left(\frac{2\rho}{\overline{q}}\sum_{i=1}^{\nu}\mathbf{c}_i\mathbf{c}_i^\intercal\right) = \frac{2\rho}{\overline{q}}\lambda_{\max}\left(\mathbf{P}\right) \leq \frac{2\rho}{\overline{q}} \stackrel{\text{def}}{=} L.$$

Therefore, selecting $h = 2$, $\sigma^2 = 6\rho/\overline{q}$ and applying Proposition 3.21 we have

$$\mathbb{P}\left(\|\mathbf{W}_h\| \geq \sigma^2\right) \leq \mathbb{P}\left(\lambda_{\max}\left(\frac{1}{\overline{q}^2}\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu}\frac{1}{w_{0,i,j}^2}\mathbf{c}_i\mathbf{c}_i^\intercal\mathbf{c}_i\mathbf{c}_i^\intercal\right) \geq (1+2)\frac{2\rho}{\overline{q}}\right)$$
$$\leq \nu \cdot \exp\left\{-\frac{2\rho}{\overline{q}}\frac{\overline{q}^2}{\rho^2}(3\log(3) - 2)\right\} \leq n \cdot \exp\left\{-\frac{2\overline{q}}{\rho}\right\}.$$

3.4.5 Space complexity bound

Denote with A the event $A = \left\{\forall h' \in \{1,\dots,h\} : \|\mathbf{P}^{h'} - \widetilde{\mathbf{P}}^{h'}\|_2 \le \varepsilon\right\}$, and again $\nu = |\mathcal{D}_{\{h,l\}}|$. Letting $q = |\mathcal{I}_{\{h,l\}}| = \sum_{i=1}^{\nu} q_{h,i} = \sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu} z_{h,i,j}$ be the random number of points in $\mathcal{I}_{\{h,l\}}$, we reformulate

$$\begin{aligned}
&\mathbb{P}\left(|\mathcal{I}_{\{h,l\}}| \ge 3\overline{q}d_{\text{eff}}^{\{h,l\}}(\gamma) \cap \left\{\forall h' \in \{1,\dots,h\} : \left(\|\mathbf{P}^{h'} - \widetilde{\mathbf{P}}^{h'}\|_2 \le \varepsilon\right) \le \varepsilon\right\}\right)\\
&= \mathbb{P}\left(|\mathcal{I}_{\{h,l\}}| \ge 3\overline{q}d_{\text{eff}}^{\{h,l\}}(\gamma) \cap A\right) = \mathbb{P}\left(\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu} z_{h,i,j} \ge 3\overline{q}d_{\text{eff}}^{\{h,l\}}(\gamma) \cap A\right)\\
&= \mathbb{P}\left(\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu} z_{h,i,j} \ge 3\overline{q}d_{\text{eff}}^{\{h,l\}}(\gamma) \,\middle|\, A\right)\mathbb{P}(A).
\end{aligned}$$

While we do know that the $z_{h,i,j}$ are Bernoulli random variables (since they are either 0 or 1), it is not easy to compute the success probability of each $z_{h,i,j}$, and in addition there could be dependencies between $z_{h,i,j}$ and $z_{h,i',j'}$. Similarly to Lem. 3.20, we are going to find a stochastic variable to dominate $z_{h,i,j}$. Denoting with $u'_{s,i,j} \sim \mathcal{U}(0,1)$ a uniform random variable, we will define $w'_{s,i,j}$ as

$$w'_{s,i,j}|\mathcal{F}_{\{s,i',j'\}} = w'_{s,i,j}|\mathcal{F}_{s-2} \overset{\text{def}}{=} \mathbb{I}\left\{u'_{s,i,j} \le \frac{p_{h,i}}{\widetilde{p}_{s-1,i}}\right\} \sim Ber\left(\frac{p_{h,i}}{\widetilde{p}_{s-1,i}}\right)$$

for any i' and j' such that $\{s,1,1\} \le \{s,i',j'\} < \{s,i,j\}$. Note that $w'_{s,i,j}$, unlike $z_{s,i,j}$, does not have a recursive definition, and its only dependence on any other variable comes from $\widetilde{p}_{s-1,i}$. First, we peel off the last step

$$\begin{aligned}
&\mathbb{P}\left(\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu} z_{h,i,j} \ge g \,\middle|\, A\right)\\
&= \mathop{\mathbb{E}}_{\mathcal{F}_{t-1}|A}\left[\mathbb{P}\left(\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu} \mathbb{I}\left\{u_{h,i,j} \le \frac{\widetilde{p}_{h,i}}{\widetilde{p}_{t-1,i}}\right\} z_{h-1,i,j} \ge g \,\middle|\, \mathcal{F}_{t-1} \cap A\right)\right]\\
&\le \mathop{\mathbb{E}}_{\mathcal{F}_{t-1}|A}\left[\mathbb{P}\left(\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu} \mathbb{I}\left\{u'_{h,i,j} \le \frac{p_{h,i}}{\widetilde{p}_{t-1,i}}\right\} z_{h-1,i,j} \ge g \,\middle|\, \mathcal{F}_{t-1} \cap A\right)\right]\\
&= \mathbb{P}\left(\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu} w'_{h,i,j} z_{h-1,i,j} \ge g \,\middle|\, A\right),
\end{aligned}$$

where we used the fact that conditioned on A, $\mathcal{I}_{\{h,l\}}$ is accurate w.r.t. $\mathcal{D}_{\{h,l\}}$, which guarantees that $\widetilde{p}_{h,i} \le p_{h,i}$. Plugging this in the previous bound,

$$\begin{aligned}
\mathbb{P}\left(\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu} z_{h,i,j} \ge g \,\middle|\, A\right)\mathbb{P}(A) &\le \mathbb{P}\left(\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu} w'_{h,i,j} z_{h-1,i,j} \ge g \cap A\right)\\
&\le \mathbb{P}\left(\sum_{j=1}^{\overline{q}}\sum_{i=1}^{\nu} w'_{h,i,j} z_{h-1,i,j} \ge g\right).
\end{aligned}$$

We now proceed by peeling off layers from the end of the chain one by one. We show how to move from an iteration $s \leq h$ to $s-1$.

$$\mathbb{P}\left(\sum_{j=1}^{\bar{q}}\sum_{i=1}^{\nu} w'_{s,i,j} z_{s-1,i,j} \geq g\right) = \mathop{\mathbb{E}}_{\mathcal{F}_{s-2}}\left[\mathbb{P}\left(\sum_{j=1}^{\bar{q}}\sum_{i=1}^{\nu} \mathbb{I}\left\{u'_{s,i,j} \leq \frac{p_{h,i}}{\widetilde{p}_{s-1,i}}\right\} z_{s-1,i,j} \geq g \,\middle|\, \mathcal{F}_{s-2}\right)\right]$$
$$= \mathop{\mathbb{E}}_{\mathcal{F}_{s-2}}\left[\mathbb{P}\left(\sum_{j=1}^{\bar{q}}\sum_{i=1}^{\nu} \mathbb{I}\left\{u'_{s,i,j} \leq \frac{p_{h,i}}{\widetilde{p}_{s-1,i}}\right\} \mathbb{I}\left\{u_{s-1,i,j} \leq \frac{\widetilde{p}_{s-1,i}}{\widetilde{p}_{s-2,i}}\right\} z_{s-2,i,j} \geq g \,\middle|\, \mathcal{F}_{s-2}\right)\right]$$
$$= \mathop{\mathbb{E}}_{\mathcal{F}_{s-2}}\left[\mathbb{P}\left(\sum_{j=1}^{\bar{q}}\sum_{i=1}^{\nu} \mathbb{I}\left\{u'_{s-1,i,j} \leq \frac{p_{h,i}}{\widetilde{p}_{s-2,i}}\right\} z_{s-2,i,j} \geq g \,\middle|\, \mathcal{F}_{s-2}\right)\right]$$
$$= \mathbb{P}\left(\sum_{j=1}^{\bar{q}}\sum_{i=1}^{\nu} w'_{s-1,i,j} z_{s-2,i,j} \geq g\right)$$

Applying this repeatedly from $s = h$ to $s = 2$ we have,

$$\mathbb{P}\left(\sum_{j=1}^{\bar{q}}\sum_{i=1}^{\nu} w'_{h,i,j} z_{h-1,i,j} \geq g\right) = \mathbb{P}\left(\sum_{j=1}^{\bar{q}}\sum_{i=1}^{\nu} w'_{1,i,j} z_{0,i,j} \geq g\right) = \mathbb{P}\left(\sum_{j=1}^{\bar{q}}\sum_{i=1}^{\nu} w'_{1,i,j} \geq g\right).$$

Now, all the $w'_{1,i,j}$ are independent Bernoulli random variables, and we can bound their sum with Proposition 2.13.

3.5 Extended proofs

Proof of Lemma 3.8 and Lemma 3.13. It is easy to show that $\tau_{t,i} \leq \tau_{t-1,i}$ for all $t > i$ ($\tau_{t,i}$ is not defined for $t < i$). Since $\Phi_t\Phi_t^\intercal + \mathbf{\Pi}_t \preceq \Phi_{t-1}\Phi_{t-1}^\intercal + \mathbf{\Pi}_{t-1}$ and $\phi_i = \mathbf{\Pi}_t\phi_i = \mathbf{\Pi}_{t-1}\phi_i$ we have

$$\tau_{t,i} = \phi_i^\intercal(\Phi_t\Phi_t^\intercal + \gamma\mathbf{\Pi}_t)^{-1}\phi_i \leq \phi_i^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_{t-1})^{-1}\phi_i = \tau_{t-1,i}.$$

We will now prove the lower bound $\tau_{t,i} \geq \tau_{t-1,i}/(\tau_{t-1,1}+1)$.

Note that since both $\mathbf{\Pi}_{t-1}$ and $\mathbf{\Pi}_t$ include ϕ_i, we have $\tau_{t-1,i} = \phi_i^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal+\gamma\mathbf{\Pi}_{t-1})^{-1}\phi_i = \phi_i^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_t)^{-1}\phi_i$. Considering the definition of $\tau_{t,i}$ in terms of ϕ_i and Φ_t, and applying the Sherman-Morrison formula we obtain

$$\begin{aligned}\tau_{t,i} &= \phi_i^\intercal(\Phi_t\Phi_t^\intercal + \gamma_t\mathbf{\Pi})^{-1}\phi_i = \phi_i^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal + \phi_t\phi_t^\intercal + \gamma_t\mathbf{\Pi}_t)^{-1}\phi_i \\ &= \phi_i^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_t)^{-1}\phi_i - \frac{\phi_i^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_t)^{-1}\phi_t\phi_t^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_t)^{-1}\phi_i}{1 + \phi_t^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_t)^{-1}\phi_t} \\ &= \tau_{t-1,i} - \frac{\phi_i^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_t)^{-1}\phi_t\phi_t^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_t)^{-1}\phi_i}{1 + \phi_t^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_t)^{-1}\phi_t}.\end{aligned}$$

Let

$$\mathbf{x} = (\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_t)^{-1/2}\phi_i \quad \text{and} \quad \mathbf{y} = (\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_t)^{-1/2}\phi_t.$$

Then $\tau_{t,i}/\tau_{t-1,i}$ is equal to

$$\begin{aligned}\frac{\tau_{t,i}}{\tau_{t-1,i}} &= 1 - \frac{(\phi_t^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_t)^{-1}\phi_i)^2}{(1+\phi_t^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_t)^{-1}\phi_t)\phi_i^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_t)^{-1}\phi_i} \\ &= 1 - \frac{(\mathbf{y}^\intercal\mathbf{x})^2}{(1+\mathbf{y}^\intercal\mathbf{y})\,\mathbf{x}^\intercal\mathbf{x}}.\end{aligned}$$

Defining the cosine between $\mathbf{y}$ and $\mathbf{x}$ as $\cos(\mathbf{y},\mathbf{x}) = \mathbf{y}^\intercal\mathbf{x}/(\|\mathbf{x}\|\|\mathbf{y}\|)$, we have that

$$1 - \frac{(\mathbf{y}^\intercal\mathbf{x})^2}{(1+\mathbf{y}^\intercal\mathbf{y})\,\mathbf{x}^\intercal\mathbf{x}} = 1 - \frac{\mathbf{y}^\intercal\mathbf{y}\mathbf{x}^\intercal\mathbf{x}\cos(\mathbf{y},\mathbf{x})^2}{(1+\mathbf{y}^\intercal\mathbf{y})\,\mathbf{x}^\intercal\mathbf{x}} = 1 - \frac{\|\mathbf{y}\|^2}{1+\|\mathbf{y}\|^2}\cos(\mathbf{y},\mathbf{x})^2,$$

where $\frac{\|\mathbf{y}\|^2}{1+\|\mathbf{y}\|^2}$ depends only on the norm of $\mathbf{y}$ and not its direction, and $\cos(\mathbf{y},\mathbf{x})$ depends only on the direction of $\mathbf{y}$ and is maximized when $\mathbf{y} = \mathbf{x}$. Therefore,

$$\begin{aligned}\frac{\tau_{t+1,i}}{\tau_{t,i}} &= 1 - \frac{(\mathbf{y}^\intercal\mathbf{x})^2}{(1+\mathbf{y}^\intercal\mathbf{y})\,\mathbf{x}^\intercal\mathbf{x}} = 1 - \frac{\|\mathbf{y}\|^2}{1+\|\mathbf{y}\|^2}\cos(\mathbf{y},\mathbf{x})^2 \\ &\geq 1 - \frac{\|\mathbf{x}\|^2}{1+\|\mathbf{x}\|^2} = \frac{1}{1+\|\mathbf{x}\|^2} = \frac{1}{1+\tau_{t,i}}.\end{aligned}$$

For the effective dimension, we can simply use the fact (Alaoui and Mahoney, 2015) that the map

$$\Phi\Phi^\intercal \to \mathrm{Tr}(\Phi\Phi^\intercal(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})),$$

is monotone in $\Phi\Phi^\intercal$ to see that

$$d_{\mathrm{eff}}^{t-1}(\gamma) = \mathrm{Tr}(\Phi_{t-1}\Phi_{t-1}^\intercal(\Phi_{t-1}\Phi_{t-1}^\intercal + \gamma\mathbf{\Pi}_{t-1})) \leq \mathrm{Tr}(\Phi_t\Phi_t^\intercal(\Phi_t\Phi_t^\intercal + \gamma\mathbf{\Pi}_t)) = d_{\mathrm{eff}}^t(\gamma).$$

For the more general Lemma 3.13, the first point $\tau_{\mathcal{D},i} \geq \tau_{\mathcal{D}\cup\mathcal{D}',i}$ can be easily proven by fixing the first $\mathcal{D}$, and then invoke Lemma 3.8 as we add one sample at a time from $\mathcal{D}'$. Also as easily for $d_{\mathrm{eff}}^{\mathcal{D}}(\gamma) + d_{\mathrm{eff}}^{\mathcal{D}'}(\gamma)$ we have

$$d_{\mathrm{eff}}^{\mathcal{D}}(\gamma) + d_{\mathrm{eff}}^{\mathcal{D}'}(\gamma) \leq 2\max\{d_{\mathrm{eff}}^{\mathcal{D}}(\gamma); d_{\mathrm{eff}}^{\mathcal{D}'}(\gamma)\} \leq 2\max\{d_{\mathrm{eff}}(\gamma)_{\mathcal{D}\cup\mathcal{D}'}; d_{\mathrm{eff}}(\gamma)_{\mathcal{D}\cup\mathcal{D}'}\} = 2d_{\mathrm{eff}}(\gamma)_{\mathcal{D}\cup\mathcal{D}'}.$$

Finally, we prove the other side of the inequality for $d_{\mathrm{eff}}^{\mathcal{D}}(\gamma) + d_{\mathrm{eff}}^{\mathcal{D}}(\gamma)$. Let $\Phi_{\mathcal{D}}, \Phi_{\mathcal{D}'}$ be the maps constructed using the samples in $\mathcal{D}$ and $\mathcal{D}'$ respectively. Then,

$$\begin{aligned}&d_{\mathrm{eff}}^{\mathcal{D}}(\gamma) + d_{\mathrm{eff}}^{\mathcal{D}'}(\gamma) = \sum_{i\in\mathcal{D}}\tau_{\mathcal{D},i} + \sum_{i\in\mathcal{D}'}\tau_{\mathcal{D}',i} \\ &= \sum_{i\in\mathcal{D}}\phi_i^\intercal(\Phi_{\mathcal{D}}\Phi_{\mathcal{D}}^\intercal + \gamma\mathbf{\Pi}_{\mathcal{D}})^{-1}\phi_i + \sum_{i\in\mathcal{D}'}\phi_i^\intercal(\Phi_{\mathcal{D}'}\Phi_{\mathcal{D}'}^\intercal + \gamma\mathbf{\Pi}_{\mathcal{D}'})^{-1}\phi_i \\ &\geq \sum_{i\in\mathcal{D}}\phi_i^\intercal(\Phi_{\mathcal{D}\cup\mathcal{D}'}\Phi_{\mathcal{D}\cup\mathcal{D}'}^\intercal + \gamma\mathbf{\Pi}_{\mathcal{D}\cup\mathcal{D}'})^{-1}\phi_i + \sum_{i\in\mathcal{D}'}\phi_i^\intercal(\Phi_{\mathcal{D}\cup\mathcal{D}'}\Phi_{\mathcal{D}\cup\mathcal{D}'}^\intercal + \gamma\mathbf{\Pi}_{\mathcal{D}\cup\mathcal{D}'})^{-1}\phi_i \\ &= \sum_{i\in\mathcal{D}\cup\mathcal{D}'}\phi_i^\intercal(\Phi_{\mathcal{D}\cup\mathcal{D}'}\Phi_{\mathcal{D}\cup\mathcal{D}'}^\intercal + \gamma\mathbf{\Pi}_{\mathcal{D}\cup\mathcal{D}'})^{-1}\phi_i = \sum_{i\in\mathcal{D}\cup\mathcal{D}'}\tau_{\mathcal{D}\cup\mathcal{D}',i} = d_{\mathrm{eff}}^{\mathcal{D}\cup\mathcal{D}'}(\gamma).\end{aligned}$$

■

Proof of Lemma 3.3 and Lemma 3.14. We give a proof for a generalized estimator that instead of using a single dictionary $\mathcal{I}$ uses k ε-accurate dictionaries $\mathcal{I}_k$. Given the dataset $\mathcal{D}$ and its associated Φ and $\mathbf{\Pi}$ we split it in disjoint datasets $\{\mathcal{D}_i\}_{i=1}^k$ with associated maps Φ_i. From each dataset, construct an (ε, γ)-accurate dictionary $\mathcal{I}_i$, with its associated selection matrix $\mathbf{S}_i$.

To estimate the RLS τ_i of point i w.r.t. the whole dataset $\mathcal{D} = \cup_{j=1}^k \mathcal{D}_j$, and corresponding feature matrix Φ, we set the estimator to be

$$\widetilde{\tau}_i = (1-\varepsilon)\phi_i^\intercal \left(\sum_{j=1}^k \Phi_j \mathbf{S}_j \mathbf{S}_j^\intercal \Phi_j^\intercal + (1+(k-1)\varepsilon)\gamma \mathbf{\Pi} \right)^{-1} \phi_i.$$

Part 1: accuracy of the RLS estimator $\widetilde{\tau}_i$. Since each of the dictionaries $\mathcal{I}_i$ used to generate $\mathbf{S}_i$ is (ε, γ)-accurate, we have from Definition 2.6 and Lemma 2.7 that

$$(1-\varepsilon)\Phi_i\Phi_i^\intercal - \varepsilon\gamma\mathbf{\Pi} \preceq \Phi_i \mathbf{S}_i \mathbf{S}_i^\intercal \Phi_i^\intercal \preceq (1+\varepsilon)\Phi_i\Phi_i^\intercal + \varepsilon\gamma\mathbf{\Pi}.$$

Therefore, we have

$$\begin{aligned}
\widetilde{\tau}_i &= (1-\varepsilon)\phi_i^\intercal \left(\sum_{j=1}^k \Phi_j \mathbf{S}_j \mathbf{S}_j^\intercal \Phi_j^\intercal + (1+(k-1)\varepsilon)\gamma \mathbf{\Pi} \right)^{-1} \phi_i \\
&\leq (1-\varepsilon)\phi_i^\intercal \left(\sum_{j=1}^k \left((1-\varepsilon)\Phi_j\Phi_j^\intercal - \varepsilon\gamma\mathbf{\Pi}\right) + (1+(k-1)\varepsilon)\gamma \mathbf{\Pi} \right)^{-1} \phi_i \\
&\leq (1-\varepsilon)\phi_i^\intercal \left((1-\varepsilon)\Phi\Phi^\intercal - k\varepsilon\gamma\mathbf{\Pi} + (1+(k-1)\varepsilon)\gamma\mathbf{\Pi}\right)^{-1} \phi_i \\
&= (1-\varepsilon)\phi_i^\intercal \left((1-\varepsilon)(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})\right)^{-1} \phi_i = \frac{(1-\varepsilon)}{(1-\varepsilon)}\phi_i^\intercal \left(\Phi\Phi^\intercal + \gamma\mathbf{\Pi}\right)^{-1} \phi_i = \tau_i,
\end{aligned}$$

and

$$\begin{aligned}
\widetilde{\tau}_i &= (1-\varepsilon)\phi_i^\intercal \left(\sum_{j=1}^k \Phi_j \mathbf{S}_j \mathbf{S}_j^\intercal \Phi_j^\intercal + (1+(k-1)\varepsilon)\gamma \mathbf{\Pi} \right)^{-1} \phi_i \\
&\geq (1-\varepsilon)\phi_i^\intercal \left(\sum_{j=1}^k \left((1+\varepsilon)\Phi_j\Phi_j^\intercal + \varepsilon\gamma\mathbf{\Pi}\right) + (1+(k-1)\varepsilon)\gamma \mathbf{\Pi} \right)^{-1} \phi_i \\
&= (1-\varepsilon)\phi_i^\intercal \left((1+\varepsilon)\Phi\Phi^\intercal + k\varepsilon\gamma\mathbf{\Pi} + (1+(k-1)\varepsilon)\gamma\mathbf{\Pi}\right)^{-1} \phi_i \\
&= (1-\varepsilon)\phi_i^\intercal \left((1+\varepsilon)\Phi\Phi^\intercal + (1+(2k-1)\varepsilon)\gamma\mathbf{\Pi}\right)^{-1} \phi_i \\
&\geq (1-\varepsilon)\phi_i^\intercal \left((1+(2k-1)\varepsilon)\Phi\Phi^\intercal + (1+(2k-1)\varepsilon)\gamma\mathbf{\Pi}\right)^{-1} \phi_i \\
&= \frac{(1-\varepsilon)}{(1+(2k-1)\varepsilon)}\phi_i^\intercal \left(\Phi\Phi^\intercal - \gamma\mathbf{\Pi}\right)^{-1} \phi_i = \frac{(1-\varepsilon)}{(1+(2k-1)\varepsilon)}\tau_i.
\end{aligned}$$

Then, we can instantiate this result with $k = 1$ to prove the accuracy claim in Lem. 3.3, and with $k = 2$ to prove the accuracy claim in Lem. 3.14.

Part 2: accuracy of $\min\{\widetilde{\tau}_t, \widetilde{\tau}_{t-1}\}$. To simplify the notation, for this part of the proof we indicate with $\tau_t \leq \tau_{t-1}$ that for each $i \in \{1, \dots, t-1\}$ we have $\tau_{t,i} \leq \tau_{t-1,i}$. From Lem. 3.8, we know that $\tau_{t-1} \geq \tau_t$. Given α-accurate $\widetilde{\tau}_t$ and $\widetilde{\tau}_{t-1}$ we have the upper bound

$$\min\{\widetilde{\tau}_t,\ \widetilde{\tau}_{t-1}\} \leq \min\{\tau_t,\ \tau_{t-1}\} = \tau_t,$$

and the lower bound,

$$\min\{\widetilde{\tau}_t,\ \widetilde{\tau}_{t-1}\} \geq \frac{1}{\alpha}\min\{\tau_t,\ \tau_{t-1}\} = \frac{1}{\alpha}\tau_t,$$

which combined gives us $\frac{1}{\alpha}\tau_t \leq \min\{\widetilde{\tau}_t,\ \widetilde{\tau}_{t-1}\} \leq \tau_t$ as required by the definition of α-accuracy. ■

Proof of Lemma 3.5. We will prove this result for a generic $k \geq 1$ and $(\varepsilon, k\gamma)$-accurate dictionary $\mathcal{I}$, with $(\varepsilon, 2\gamma)$-accuracy as a special case when $k = 2$. Starting from the proof of Lemma 3.3 and Lemma 3.14 we can adapt some passages. In particular we have only one single $(\varepsilon, k\gamma)$-accurate dictionary $\mathcal{I}$ and a single dataset $\mathcal{D}$, and we want to show that

$$\widetilde{\tau}_i = (1-\varepsilon)\phi_i^\intercal\left(\Phi\mathbf{S}\mathbf{S}^\intercal\Phi^\intercal + (1+(k-1)\varepsilon)\gamma\mathbf{\Pi}\right)^{-1}\phi_i.$$

is α-accurate with $\alpha = \frac{1+2(k-1)\varepsilon}{1-\varepsilon}$.

Since $\mathcal{I}$ used to generate $\mathbf{S}$ is $(\varepsilon, k\gamma)$-accurate, The identically to the proof of Lemma 3.3 we have

$$\begin{aligned}
\widetilde{\tau}_i &= (1-\varepsilon)\phi_i^\intercal\left(\Phi\mathbf{S}\mathbf{S}^\intercal\Phi^\intercal + (1+(k-1)\varepsilon)\gamma\mathbf{\Pi}\right)^{-1}\phi_i \\
&\leq (1-\varepsilon)\phi_i^\intercal\left(((1-\varepsilon)\Phi\Phi^\intercal - \varepsilon\gamma\mathbf{\Pi}) + (1+(k-1)\varepsilon)\gamma\mathbf{\Pi}\right)^{-1}\phi_i \\
&\leq (1-\varepsilon)\phi_i^\intercal\left((1-\varepsilon)\Phi\Phi^\intercal - k\varepsilon\gamma\mathbf{\Pi} + (1+(k-1)\varepsilon)\gamma\mathbf{\Pi}\right)^{-1}\phi_i \\
&= (1-\varepsilon)\phi_i^\intercal\left((1-\varepsilon)(\Phi\Phi^\intercal + \gamma\mathbf{\Pi})\right)^{-1}\phi_i = \frac{(1-\varepsilon)}{(1-\varepsilon)}\phi_i^\intercal\left(\Phi\Phi^\intercal + \gamma\mathbf{\Pi}\right)^{-1}\phi_i = \tau_i,
\end{aligned}$$

and

$$\begin{aligned}
\widetilde{\tau}_i &= (1-\varepsilon)\phi_i^\intercal\left(\Phi\mathbf{S}\mathbf{S}^\intercal\Phi^\intercal + (1+(k-1)\varepsilon)\gamma\mathbf{\Pi}\right)^{-1}\phi_i \\
&\geq (1-\varepsilon)\phi_i^\intercal\left((1+\varepsilon)\Phi\Phi^\intercal + k\varepsilon\gamma\mathbf{\Pi} + (1+(k-1)\varepsilon)\gamma\mathbf{\Pi}\right)^{-1}\phi_i \\
&= (1-\varepsilon)\phi_i^\intercal\left((1+\varepsilon)\Phi\Phi^\intercal + (1+(2k-1)\varepsilon)\gamma\mathbf{\Pi})\right)^{-1}\phi_i \\
&\geq (1-\varepsilon)\phi_i^\intercal\left((1+(2k-1)\varepsilon)\Phi\Phi^\intercal + (1+(2k-1)\varepsilon)\gamma\mathbf{\Pi})\right)^{-1}\phi_i \\
&= \frac{(1-\varepsilon)}{(1+(2k-1)\varepsilon)}\phi_i^\intercal\left(\Phi\Phi^\intercal - \gamma\mathbf{\Pi}\right)^{-1}\phi_i = \frac{(1-\varepsilon)}{(1+(2k-1)\varepsilon)}\tau_i.
\end{aligned}$$

Then, we can instantiate this result with $k = 2$ to prove the accuracy claim in Lem. 3.5. ■

Proof of Lem. 3.9. From the definition of $\tau_{t,t}$ we have

$$\begin{aligned}
\sum_{t=1}^{T}\tau_{t,t} &= \sum_{t=1}^{T}\phi_t^\intercal(\Phi_t\Phi_t^\intercal + \alpha\mathbf{\Pi}_T)^{-1}\phi_t \\
&= \sum_{t=1}^{T}(\phi_t^\intercal/\sqrt{\alpha})\left(\Phi_t\Phi_t^\intercal/\alpha + \mathbf{\Pi}_T\right)^{-1}(\phi_t/\sqrt{\alpha}) \leq \log(\mathrm{Det}(\Phi_T\Phi_T^\intercal/\alpha + \mathbf{\Pi}_T)),
\end{aligned}$$

where the last passage is proved by Hazan et al., (2006). Using Sylvester's determinant identity,

$$\mathrm{Det}(\Phi_T\Phi_T^\intercal/\alpha + \mathbf{\Pi}_T) = \mathrm{Det}(\Phi_T^\intercal\Phi_T/\alpha + \mathbf{I}_T) = \prod_{t=1}^{T}(\lambda_t/\alpha + 1),$$

where λ_t are the eigenvalues of $\Phi_T^\intercal \Phi_T$. Then,

$$\sum_{t=1}^{T} \tau_{t,t} \leq \log\left(\prod\nolimits_{t=1}^{T} (\lambda_t/\alpha + 1)\right) = \sum\nolimits_{t=1}^{T} \log(\lambda_t/\alpha + 1).$$

We can decompose this as

$$\begin{aligned}
\sum_{t=1}^{T} \log(\lambda_t/\alpha + 1) &= \sum_{t=1}^{T} \log(\lambda_t/\alpha + 1)\left(\frac{\lambda_t/\alpha + 1}{\lambda_t/\alpha + 1}\right) \\
&= \sum_{t=1}^{T} \log(\lambda_t/\alpha + 1)\frac{\lambda_t/\alpha}{\lambda_t/\alpha + 1} + \sum_{t=1}^{T} \frac{\log(\lambda_t/\alpha + 1)}{\lambda_t/\alpha + 1} \\
&\leq \log(\|\Phi_T^\intercal \Phi_T\|/\alpha + 1) \sum_{t=1}^{T} \frac{\lambda_t}{\lambda_t + \alpha} + \sum_{t=1}^{T} \frac{\log(\lambda_t/\alpha + 1)}{\lambda_t/\alpha + 1} \\
&\leq \log(\|\Phi_T^\intercal \Phi_T\|/\alpha + 1) d_{\text{eff}}^T(\alpha) + \sum_{t=1}^{T} \frac{(\lambda_t/\alpha + 1) - 1}{\lambda_t/\alpha + 1} \\
&= \log(\|\Phi_T^\intercal \Phi_T\|/\alpha + 1) d_{\text{eff}}^T(\alpha) + d_{\text{eff}}^T(\alpha),
\end{aligned}$$

where the first inequality is due to $\|\Phi_T^\intercal \Phi_T\| \geq \lambda_t$ for all t and the monotonicity of $\log(\cdot)$, and the second inequality is due to $\log(x) \leq x - 1$. ■

4. Sequential RLS Sampling in the Batch Setting

In this section, we show how ε-accurate and (ε, γ)-accurate dictionaries can be used for important downstream tasks, and how to instantiate our sequential RLS sampling algorithms to each setting. In particular, we focus on low-rank kernel matrix approximation and spectral graph sparsification.

4.1 Sequential RLS sampling for kernel matrix approximation

One of the major limits of kernel ridge regression (KRR), kernel PCA (Schölkopf et al., 1999), and other kernel methods is that for n samples storing and manipulating the *empirical kernel matrix* $\mathbf{K}_n$ requires $\mathcal{O}(n^2)$ space, which becomes rapidly infeasible for even a relatively small n. A popular approach (Alaoui and Mahoney, 2015; Rudi et al., 2015; Williams and Seeger, 2001) is to construct low-rank approximations of the kernel matrix by randomly selecting a subset (dictionary) of columns from $\mathbf{K}_n$, i.e., dictionary learning where the atoms are the columns of $\mathbf{K}_n$. Since they are also based on Nyström sampling, these methods are often referred to as *Nyström approximations*.
It is also easy to see that the dictionary of columns can be replaced with a dictionary of atoms. For example, the standard Nyström approximation $\widetilde{\mathbf{K}}_n = \mathbf{K}_n\mathbf{S}(\mathbf{S}^\intercal\mathbf{K}_n\mathbf{S})^+\mathbf{S}^\intercal\mathbf{K}_n$ can be rewritten as $\widetilde{\mathbf{K}}_n = \Phi^\intercal\Phi\mathbf{S}(\mathbf{S}^\intercal\mathbf{K}\mathbf{S}^\intercal)^+\mathbf{S}^\intercal\Phi^\intercal\Phi$ using the relationship $\mathbf{K}_n = \Phi_n^\intercal\Phi_n$, or in other words it is equivalent to projecting the map Φ_n on the subspace $\mathbf{\Pi}_{\mathcal{I}_n}$ spanned by the dictionary $\mathcal{I}_n$. Therefore, we can use KORS or SQUEAK to find a good dictionary $\mathcal{I}_n$ to approximate $\mathbf{K}_n$.

More formally, we have again that our atoms ϕ_i are point in an $\mathcal{H}$ and our goal is to approximate the empirical kernel matrix $\mathbf{K} = \Phi^\intercal\Phi$. Note that this is a different goal than approximating the covariance operator $\Phi\Phi^\intercal$, although the equivalence of Lemma 2.10 showed us how the two objects are closely related.
We will not consider the simpler case where $\mathcal{H} = \mathbb{R}^d$ is a euclidean space (i.e., we will stick to the $d \gg n$ setting), since in this case the atoms $\phi_i = \mathbf{x}_i$ can be explicitly stored and

manipulated, and there exists a plethora of different efficient methods that can be applied. For example, we saw already that in this special case KORS is equivalent to a method proposed by Cohen et al., (2016). For any curiosity on to the topic of matrix approximation in Euclidean spaces, we refer the reader to the excellent survey by Woodruff, (2014) on randomized linear algebra.

We begin by introducing additional notation. In particular, using the kernel trick (Aizerman et al., 1964) we can introduce the kernel function $\mathcal{K}$ so that the inner product between two points in $\mathcal{H}$ can be expressed as $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \langle \mathcal{K}(\mathbf{x}, \cdot), \mathcal{K}(\mathbf{x}', \cdot) \rangle_{\mathcal{H}} = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{H}} = \varphi(\mathbf{x})^\intercal \varphi(\mathbf{x}')$. Given a dataset of samples $\mathcal{D} = \{\mathbf{x}_t\}_{t=1}^n$, we define the (empirical) kernel matrix $\mathbf{K}_t \in \mathbb{R}^{t \times t}$ as the application of the kernel function on all pairs of input values (i.e., $[\mathbf{K}_t]_{ij} = k_{i,j} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$ for any $i, j \in [t]$), with $\mathbf{k}_{t,i} = \mathbf{K}_t \mathbf{e}_{t,i}$ as its i-th column.

4.1.1 Kernel matrix reconstruction

We begin by showing how to get an accurate approximation $\widetilde{\mathbf{K}}$ from an (ε, γ)-accurate dictionary. For this task Alaoui and Mahoney, (2015) shows the following reconstruction guarantees

Proposition 4.1 — (Alaoui and Mahoney, 2015). Given an (ε, γ)-accurate dictionary $\mathcal{I}$ of map $\Phi : \mathbb{R}^n \to \mathcal{H}$, and its selection matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ the regularized Nyström approximation $\widetilde{\mathbf{K}}$ of $\mathbf{K}$

$$\widetilde{\mathbf{K}} = \mathbf{K}\mathbf{S}(\mathbf{S}^\intercal \mathbf{K}\mathbf{S} + \gamma \mathbf{I}_{\mathcal{I}})^{-1} \mathbf{S}^\intercal \mathbf{K} = \Phi^\intercal \Phi \mathbf{S}(\mathbf{S}^\intercal \mathbf{K}\mathbf{S}^\intercal + \gamma \mathbf{I}_{\mathcal{I}})^{+} \mathbf{S}^\intercal \Phi^\intercal \Phi,$$

satisfies

$$\mathbf{0} \preceq \mathbf{K} - \widetilde{\mathbf{K}} \preceq \frac{\gamma}{1-\varepsilon} \mathbf{K}(\mathbf{K} + \gamma \mathbf{I}_n)^{-1} \preceq \frac{\gamma}{1-\varepsilon} \mathbf{I}_n.$$

Since $\widetilde{\mathbf{K}}$ is the result of passing Φ trough a regularized projection on the atoms in the dictionary $\mathcal{I}$ $\mathbf{\Gamma}_{\mathcal{I}} = \Phi \mathbf{S}(\mathbf{S}^\intercal \mathbf{K}\mathbf{S}^\intercal + \gamma \mathbf{I}_{\mathcal{I}})^{+} \mathbf{S}^\intercal \Phi^\intercal$, it is easy to see that $\widetilde{\mathbf{K}} \preceq \mathbf{K}$ no matter the dictionary. Proposition 4.1 gives us a also an upper bound, showing that if the dictionary is (ε, γ)-accurate the projection preserves the matrix $\mathbf{K}$ up to a small additive error $\frac{\gamma}{1-\varepsilon} \mathbf{K}(\mathbf{K} + \gamma \mathbf{I}_n)^{-1}$, which can be further upper bounded by $\frac{\gamma}{1-\varepsilon} \mathbf{I}_n$.
Musco and Musco, (2017) show a slightly tighter bound for the unregularized projection matrix construed using the dictionary.

Proposition 4.2 — (Musco and Musco, 2017). Given an (ε, γ)-accurate dictionary $\mathcal{I}$ of map $\Phi : \mathbb{R}^n \to \mathcal{H}$, and its selection matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ the unregularized Nyström approximation $\widetilde{\mathbf{K}}$ of $\mathbf{K}$ satisfies

$$\widetilde{\mathbf{K}} = \mathbf{K}\mathbf{S}(\mathbf{S}^\intercal \mathbf{K}\mathbf{S})^{+} \mathbf{S}^\intercal \mathbf{K} = \Phi^\intercal \Phi \mathbf{S}(\mathbf{S}^\intercal \mathbf{K}\mathbf{S}^\intercal)^{+} \mathbf{S}^\intercal \Phi^\intercal \Phi, \qquad \mathbf{0} \preceq \mathbf{K} - \widetilde{\mathbf{K}} \preceq \frac{\varepsilon\gamma}{1-\varepsilon} \mathbf{I}_n.$$

Note that this time the we do not have a bound in terms of $\mathbf{K}(\mathbf{K} + \gamma \mathbf{I}_n)^{-1}$, but directly in terms of $\mathbf{I}_n$. We could have obtained a similar result, but with a larger $\gamma/(1-\varepsilon)$ bound, simply by observing that Alaoui and Mahoney, (2015)'s bound for the regularized approximation implies

$$\mathbf{K} \preceq \mathbf{K}\mathbf{S}(\mathbf{S}^\intercal \mathbf{K}\mathbf{S} + \gamma \mathbf{I}_{\mathcal{I}})^{-1} \mathbf{S}^\intercal \mathbf{K} + \frac{\gamma}{1-\varepsilon} \mathbf{I}_n \preceq \mathbf{K}\mathbf{S}(\mathbf{S}^\intercal \mathbf{K}\mathbf{S})^{+} \mathbf{S}^\intercal \mathbf{K} + \frac{\gamma}{1-\varepsilon} \mathbf{I}_n.$$

Removing the regularization γ has several advantages. Computationally, we do not need anymore to reweight the atoms using $\mathbf{S}$ when constructing the approximation, which in practice saves us time and space when computing $\widetilde{\mathbf{K}}$ and improves numerical stability. Moreover the operator $\mathbf{\Pi}_{\mathcal{I}} = \mathbf{\Phi S}(\mathbf{S}^\intercal \mathbf{K} \mathbf{S}^\intercal)^+ \mathbf{S}^\intercal$ is the projection on $\mathrm{Im}(\mathbf{\Phi S})$, and this gives us an intuition that the subspaces spanned by the dataset and the dictionary are close. Finally, since $\mathbf{\Pi}_{\mathcal{I}}$ is a projection, it defines an accurate finite-rank subspace $\mathcal{H}_{\mathcal{I}}$, which will be useful in Chapter 5.
We can simplify both of the original proofs, and slightly improve the bound of Alaoui and Mahoney, (2015), by showing that they are special cases of the following result.

Lemma 4.3 Given a dictionary $\mathcal{I}$ that is (ε, γ)-accurate w.r.t. map $\Phi : \mathbb{R}^n \to \mathcal{H}$, and its selection matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, for any $\alpha \geq \varepsilon\gamma$ we have

$$\mathbf{\Pi} \preceq \mathbf{\Phi S}(\mathbf{S}^\intercal \mathbf{\Phi}^\intercal \mathbf{\Phi S} + \alpha \mathbf{I}_{\mathcal{I}})^{-1} \mathbf{S}^\intercal \mathbf{\Phi}^\intercal + \frac{\alpha}{1-\varepsilon} \left(\mathbf{\Phi \Phi}^\intercal + \frac{\alpha - \varepsilon\gamma}{1-\varepsilon} \mathbf{\Pi} \right)^+ . \tag{4.1}$$

Therefore

$$\begin{aligned} \mathbf{K} &\preceq \mathbf{KS}(\mathbf{S}^\intercal \mathbf{K} \mathbf{S} + \alpha \mathbf{I}_{\mathcal{I}})^{-1} \mathbf{S}^\intercal \mathbf{K} + \frac{\alpha}{1-\varepsilon} \mathbf{K} \left(\mathbf{K} + \frac{\alpha - \varepsilon\gamma}{1-\varepsilon} \mathbf{I}_n \right)^+ \\ &\preceq \mathbf{KS}(\mathbf{S}^\intercal \mathbf{K} \mathbf{S})^+ \mathbf{S}^\intercal \mathbf{K} + \frac{\alpha}{1-\varepsilon} \mathbf{K} \left(\mathbf{K} \right)^+ . \end{aligned}$$

Proof of Lemma 4.3. Using $\mathbf{\Pi}$'s definition

$$\begin{aligned} \mathbf{\Pi} &= (\mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal + \alpha \mathbf{\Pi})^{-1/2} (\mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal + \alpha \mathbf{\Pi}) (\mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal + \alpha \mathbf{\Pi})^{-1/2} \\ &= (\mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal + \alpha \mathbf{\Pi})^{-1/2} \mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal (\mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal + \alpha \mathbf{\Pi})^{-1/2} \\ &\qquad + \alpha (\mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal + \alpha \mathbf{\Pi})^{-1/2} \mathbf{\Pi} (\mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal + \alpha \mathbf{\Pi})^{-1/2} \\ &= (\mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal + \alpha \mathbf{\Pi})^{-1/2} \mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal (\mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal + \alpha \mathbf{\Pi})^{-1/2} + \alpha (\mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal + \alpha \mathbf{\Pi})^{-1}. \end{aligned}$$

Using the (ε, γ)-accuracy

$$\begin{aligned} (\mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal + \gamma \mathbf{\Pi})^{-1} &\preceq ((1-\varepsilon) \mathbf{\Phi \Phi}^\intercal + (\alpha - \varepsilon\gamma) \mathbf{\Pi})^{-1} \\ &= \frac{1}{1-\varepsilon} \left(\mathbf{\Phi \Phi}^\intercal + \frac{\alpha - \varepsilon\gamma}{1-\varepsilon} \mathbf{I}_n \right)^{-1} . \end{aligned}$$

We finish by reformulating

$$\mathbf{\Phi S}(\mathbf{S}^\intercal \mathbf{\Phi}^\intercal \mathbf{\Phi S} + \alpha \mathbf{I}_{\mathcal{I}})^{-1} \mathbf{S}^\intercal \mathbf{\Phi}^\intercal = (\mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal + \alpha \mathbf{\Pi})^{-1/2} \mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal (\mathbf{\Phi S S}^\intercal \mathbf{\Phi}^\intercal + \alpha \mathbf{\Pi})^{-1/2}.$$

■

Therefore, setting $\alpha = \gamma$ gives us Alaoui and Mahoney, (2015)'s bound, and setting $\alpha = \varepsilon\gamma$ gives us

$$\mathbf{0} \preceq \mathbf{K} - \mathbf{KS}(\mathbf{S}^\intercal \mathbf{K} \mathbf{S} + \varepsilon\gamma \mathbf{I}_{\mathcal{I}})^{-1} \mathbf{S}^\intercal \mathbf{K} \preceq \frac{\varepsilon\gamma}{1-\varepsilon} \mathbf{K}\mathbf{K}^+ \preceq \frac{\varepsilon\gamma}{1-\varepsilon} \mathbf{I}_n.$$

Noting

$$\mathbf{K} \preceq \mathbf{KS}(\mathbf{S}^\intercal \mathbf{K} \mathbf{S} + \varepsilon\gamma \mathbf{I}_{\mathcal{I}})^{-1} \mathbf{S}^\intercal \mathbf{K} + \frac{\varepsilon\gamma}{1-\varepsilon} \mathbf{I}_n \preceq \mathbf{KS}(\mathbf{S}^\intercal \mathbf{K} \mathbf{S})^+ \mathbf{S}^\intercal \mathbf{K} + \frac{\varepsilon\gamma}{1-\varepsilon} \mathbf{I}_n$$

recovers Musco and Musco, (2017)'s bound.

4.2 Efficient learning with kernels

In this section, we show how (ε, γ)-accuracy and the guarantees of Lemma 4.3 translate into accuracy guarantees for typical kernel methods. As a consequence, the time and space guarantees of KORS or SQUEAK can be used to find provably accurate approximate solution faster than other existing methods.

In particular, we focus on *regularized* statistical problems: kernel ridge regression and k-rank kernel PCA. As we will see, the additive error we introduce using our kernel matrix approximation will not disturb the solution of our regularized statistical problem if the error threshold is comparable to the regularization.

For more results on kernel K-means, kernel CCA, and alternative proofs on how an (ε, γ)-accurate dictionary can be used to compute a provably accurate approximate solution we refer the reader to Musco and Musco, (2017).

4.2.1 Kernel ridge regression

Consider now a regression dataset $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$, with input $\mathbf{x}_t \in \mathcal{X}$ and output $y_t \in \mathbb{R}$. The goal of *sequential* kernel ridge regression is, for any time $t \in [n]$, to find the vector $\widehat{\boldsymbol{\omega}}_t \in \mathbb{R}^t$ that minimizes the regularized quadratic loss

$$\widehat{\boldsymbol{\omega}}_t = \operatorname*{arg\,min}_{\boldsymbol{\omega} \in \mathbb{R}^t} \|\mathbf{y}_t - \mathbf{K}_t \boldsymbol{\omega}\|^2 + \lambda \|\boldsymbol{\omega}\|^2, \tag{4.2}$$

where $\lambda > 0$ is a regularization parameter. This objective admits the closed form solution

$$\widehat{\boldsymbol{\omega}}_t = (\mathbf{K}_t + \lambda \mathbf{I}_t)^{-1} \mathbf{y}_t. \tag{4.3}$$

In batch regression, $\widehat{\boldsymbol{\omega}}_n$ is computed only once when all the samples of $\mathcal{D}$ are available, solving the linear system in Eq. 4.3 with $\mathbf{K}_n$, but in practice we might be collecting our dataset incrementally, and want to periodically construct an intermediate batch solution using the samples seen so far.

In the *fixed-design* kernel regression problem, the accuracy of the solution $\widehat{\boldsymbol{\omega}}_n$ is measured by the prediction error on the input set $\mathcal{D}$ itself. More precisely, the prediction of the estimator $\widehat{\boldsymbol{\omega}}_n$ in each point is obtained as $[\mathbf{K}_n \widehat{\boldsymbol{\omega}}_n]_i$, while the outputs y_i in the dataset are assumed to be a noisy observation of an unknown target function $f^* : \mathcal{X} \to \mathbb{R}$, evaluated in $\mathbf{x}_i$ i.e., for any $i \in [n]$,

$$y_i = f^*(\mathbf{x}_i) + \epsilon_i = \varphi(\mathbf{x}_i)^\intercal \mathbf{w}^* + \epsilon_i,$$

where ϵ_i is a zero-mean i.i.d. noise with bounded variance σ^2. Let $\mathbf{f}^* \in \mathbb{R}^n$ be the vector with components $f^*(x_i)$, then the empirical risk of $\widehat{\boldsymbol{\omega}}_n$ on dataset $\mathcal{D}$ is measured as

$$\mathcal{R}_{\mathcal{D}}(\widehat{\boldsymbol{\omega}}_n) = \mathbb{E}_\epsilon\big[||\mathbf{f}^* - \mathbf{K}_n \widehat{\boldsymbol{\omega}}_n||^2\big]. \tag{4.4}$$

Given a dictionary $\mathcal{I}_n$ that is (ε, γ)-accurate w.r.t. Φ_n, we can use Lemma 4.3 to compute an approximate $\widetilde{\mathbf{K}}_n$ to replace $\mathbf{K}_n$ in Eq. 4.3. Taking for example the regularized version of $\widetilde{\mathbf{K}}$ with $\alpha = \gamma$, and exploiting the Woodbury formula we compute the regression weights as

$$\begin{aligned}\widetilde{\boldsymbol{\omega}}_n =& (\widetilde{\mathbf{K}}_n + \lambda \mathbf{I}_n)^{-1} \mathbf{y}_n = (\mathbf{K}_n \mathbf{S}_n (\mathbf{S}_n^\intercal \mathbf{K}_n \mathbf{S}_n + \gamma \mathbf{I}_{\mathcal{I}_n})^{-1} \mathbf{S}_n^\intercal \mathbf{K}_n + \lambda \mathbf{I}_n)^{-1} \mathbf{y}_n \\ =& \frac{1}{\lambda} \left(\mathbf{y}_n - \mathbf{K}_n \mathbf{S}_n (\mathbf{S}_n^\intercal \mathbf{K}_n \mathbf{S}_n + \lambda (\mathbf{S}_n^\intercal \mathbf{K}_n \mathbf{S}_n + \gamma \mathbf{I}_{\mathcal{I}_n}))^{-1} \mathbf{S}_n^\intercal \mathbf{K}_n \mathbf{y}_n \right).\end{aligned} \tag{4.5}$$

Computing $(\mathbf{S}_n^\intercal \mathbf{K}_n \mathbf{S}_n + \lambda(\mathbf{S}_n^\intercal \mathbf{K}_n \mathbf{S}_n + \gamma \mathbf{I}_{\mathcal{I}_n}))^{-1}$ takes $\mathcal{O}(n|\mathcal{I}_n|^2)$ time to construct the matrix and $\mathcal{O}(|\mathcal{I}_n|^3)$ to invert it, while the other matrix-matrix multiplication take at most $\mathcal{O}(n|\mathcal{I}_n|^2)$ time. Overall, these operations require to store at most an $n \times |\mathcal{I}_n|$ matrix. Therefore the final complexity of computing a KRR using the dictionary is reduced from $\mathcal{O}(n^3)$ to a linear $\mathcal{O}(n|\mathcal{I}_n|^2 + |\mathcal{I}_n|^3)$ time, and from $\mathcal{O}(n^2)$ to again linear $\mathcal{O}(n|\mathcal{I}_n|)$ space. The following corollary provides guarantees for the empirical risk of the solution $\widetilde{\boldsymbol{\omega}}_n$ in a fixed design setting.

Corollary 4.4 — (Alaoui and Mahoney, 2015, Thm. 3). For an arbitrary dataset $\mathcal{D}$, let $\mathbf{K}$ be the kernel matrix constructed on $\mathcal{D}$. Given an (ε, γ)-accurate dictionary $\mathcal{I}$ of $\mathcal{D}$, the solution $\widetilde{\boldsymbol{\omega}}$ computed using the regularized Nyström approximation $\widetilde{\mathbf{K}}$ satisfies

$$\mathcal{R}_{\mathcal{D}}(\widetilde{\boldsymbol{\omega}}) \leq \left(1 + \frac{\gamma}{\lambda}\frac{\varepsilon}{1-\varepsilon}\right)^2 \mathcal{R}_{\mathcal{D}}(\widehat{\boldsymbol{\omega}}),$$

where λ is the regularization of kernel ridge regression problem.

Therefore, if we choose $\lambda = \gamma$, i.e., if the additive error that we introduce is of the same order as the regularization, the approximate solution that can be computed in $\widetilde{\mathcal{O}}(nd_{\text{eff}}^n(\gamma)^2)$ time (including computing the dictionary with KORS or SQUEAK), and will have an empirical risk that is only a constant factor $1/(1-\varepsilon)^2$ larger. In other words, using RLS sampling we can avoid expending computational power to compute a solution along directions that are not relevant to the problem, since all directions with an influence smaller than λ will be suppressed by the regularization.

If we also assume that the samples $\mathbf{x}_i$ are being generated according to a distribution $\mu(\mathbf{x}_i)$ we can move from bounding the empirical risk $\mathcal{R}_{\mathcal{D}}$ to bounding the *expected* risk $\mathcal{R}_\mu$. To do so, we will adapt a result by Rudi et al., (2015). We now consider the case where we are given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i$ sampled according to μ and where the target variables $y_i = \phi_i^\intercal \mathbf{w}^* + \epsilon_i$ are computed using an arbitrary function $\mathbf{w}^*$ in $\mathcal{H}$ with bounded norm.
Given $\mathcal{D}$ the usual estimator used in KRR, i.e., Eq. 4.3 is $\widehat{\mathbf{w}} = \Phi_n \boldsymbol{\omega}_n$. Rudi et al., (2015) propose instead to use the estimator $\widetilde{\mathbf{w}}_n = \Phi_{\mathcal{I}} \widetilde{\boldsymbol{\omega}}_{\mathcal{I}}$ where $\Phi_{\mathcal{I}}$ are the atoms contained in an (ε, γ)-accurate dictionary $\mathcal{I}$, and the weights $\widetilde{\boldsymbol{\omega}}_{\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}|}$ are computed as

$$\widetilde{\boldsymbol{\omega}}_{\mathcal{I}} = (\Phi_{\mathcal{I}}^\intercal \Phi_n \Phi_n^\intercal \Phi_{\mathcal{I}} + \lambda \Phi_{\mathcal{I}}^\intercal \Phi_{\mathcal{I}})^+ \Phi_{\mathcal{I}}^\intercal \Phi_n \mathbf{y}_n. \tag{4.6}$$

Note that since $(\Phi_{\mathcal{I}}^\intercal \Phi_n \Phi_n^\intercal \Phi_{\mathcal{I}} + \lambda \Phi_{\mathcal{I}}^\intercal \Phi_{\mathcal{I}})^+$ is again a $|\mathcal{I}| \times |\mathcal{I}|$ matrix, computing $\widetilde{\boldsymbol{\omega}}_{\mathcal{I}}$ takes only $\mathcal{O}(n|\mathcal{I}_n|^2 + |\mathcal{I}_n|^3)$ time. If the (ε, γ)-accurate dictionary is constructed using SQUEAK or KORS, this means that overall we can compute a solution using only $\widetilde{\mathcal{O}}(nd_{\text{eff}}^n(\gamma)^3)$ time, and in a single pass over the dataset, without ever constructing or storing the full matrix $\mathbf{K}_n$. Another advantage of the estimator $\widetilde{\boldsymbol{\omega}}_{\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}|}$ compared to $\widehat{\boldsymbol{\omega}} \in \mathbb{R}^n$ is that at evaluation time we will only require $\widetilde{\mathcal{O}}(d_{\text{eff}}^n(\gamma))$ kernel evaluations, regardless of n.

We can now define the expected risk of the estimator $\widetilde{\boldsymbol{\omega}}_{\mathcal{I}}$ as

$$\mathcal{R}_\mu(\Phi_{\mathcal{I}} \widetilde{\boldsymbol{\omega}}_{\mathcal{I}}) = \mathbb{E}_{\mathbf{x}\sim\mu;\epsilon}\big[(\varphi(\mathbf{x})^\intercal \mathbf{w}^* + \epsilon - \varphi(\mathbf{x})^\intercal \Phi_{\mathcal{I}} \widetilde{\boldsymbol{\omega}}_{\mathcal{I}})^2\big].$$

We can also define an analogous of the empirical effective dimension $d_{\text{eff}}^n(\gamma)$ based on the distribution μ rather than a dataset.

Definition 4.1 — (Rudi et al., 2015). Given a feature map $\varphi(\cdot) : \mathcal{X} \to \mathcal{H}$ and a distribution μ defined over $\mathcal{X}$ the effective dimension of $\mathcal{H}$ w.r.t. μ is

$$d_{\text{eff}}^{\mu}(\gamma) = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mu}\left[\varphi(\mathbf{x})^{\intercal}\left(\mathop{\mathbb{E}}_{\mathbf{x}'\sim\mu}[\varphi(\mathbf{x}')\varphi(\mathbf{x}')^{\intercal}] + \gamma\mathbf{\Pi}\right)^{-1}\varphi(\mathbf{x})\right].$$

To obtain their expected risk bound, Rudi et al., (2015) make several technical assumption. We report the main one, in the statement of Proposition 4.5 and refer the reader to the full paper for details.

Proposition 4.5 — (Rudi et al., 2015). Given a dataset $\mathcal{D}$ sampled according to distribution μ, and a dictionary $\mathcal{I}$ (ε, γ)-accurate w.r.t. $\mathcal{D}$, let $\widetilde{\boldsymbol{\omega}}_{\mathcal{I}}$ be computed according to Eq. 4.6 with $\lambda = n^{-1/(1+q)}$.

If the eigenvalues of the covariance operator $\mathbb{E}_{\mathbf{x}'\sim\mu}[\varphi(\mathbf{x}')\varphi(\mathbf{x}')^{\intercal}]$ decay at least polynomially, i.e., $\lambda_i(\mathbb{E}_{\mathbf{x}'\sim\mu}[\varphi(\mathbf{x}')\varphi(\mathbf{x}')^{\intercal}]) \leq i^{-q}$, with polynomial degree $q \geq 1$ then w.h.p. there exist some constants R and R' such that

$$d_{\text{eff}}^{\mu}(\lambda) \leq R\lambda^{-1/q} = Rn^{1/(1+q)}, \qquad \mathcal{R}_{\mu}(\Phi_{\mathcal{I}}\widetilde{\boldsymbol{\omega}}_{\mathcal{I}}) - \mathcal{R}_{\mu}(\mathbf{w}^*) \leq R'n^{-1/(1+q)}$$

As Rudi et al., (2015) remarks, this holds for $q = 1$ whenever the covariance operator is trace class, which includes our bounded norm ϕ_i case. Therefore, running KORS or SQUEAK to construct the dictionary, we can efficiently compute a solution with expected risk guarantees that achieve the optimal (Caponnetto and De Vito, 2005) non-parametric rate $n^{-1/(1+q)}$, and falls back to the standard non-parametric rate $n^{-1/2}$ in the worst case. Moreover, Rudi et al., (2015) show that w.h.p. $\widetilde{\mathcal{O}}(nd_{\text{eff}}^{n}(\lambda)^3) \leq \widetilde{\mathcal{O}}(nd_{\text{eff}}^{\mu}(\lambda))$ and therefore our computational complexity will be smaller than $\widetilde{\mathcal{O}}(nd_{\text{eff}}^{n}(\lambda)^3) \leq \widetilde{\mathcal{O}}(nd_{\text{eff}}^{\mu}(\lambda)^2) \leq \widetilde{\mathcal{O}}(n^{3/(q+1)})$, which is always a strict improvement over the $\mathcal{O}(n^3)$ of exact KRR, and even over the $\mathcal{O}(n^2)$ time required to construct the matrix $\mathbf{K}_n$.
Computationally, we also outperform simpler strategies like uniform sampling (Bach, 2013), and if we take into account that SQUEAK can be run in less than linear time using a parallel merge tree, we outperform other state of the art RLS sampling methods such as Recursive-RLS (Musco and Musco, 2017), or even distributed averaging approaches such as the one in (Zhang et al., 2015).

4.2.2 Kernel principal component analysis

Starting again from Lemma 4.3 we can provide guarantees for Kernel principal component analysis (K-PCA). In K-PCA, given a dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$ we are interested in finding a k-rank projection $\mathbf{\Pi}_k : \mathcal{H} \to \mathcal{H}$ that minimizes

$$\mathbf{\Pi}_k = \mathop{\arg\min}_{\mathbf{\Pi}:\text{Rank}(\mathbf{\Pi})=k} \|\Phi_n - \mathbf{\Pi}\Phi_n\|_F^2 = \mathop{\arg\min}_{\mathbf{\Pi}:\text{Rank}(\mathbf{\Pi})=k} \text{Tr}(\Phi_n^{\intercal}\Phi_n - \Phi_n^{\intercal}\mathbf{\Pi}\Phi_n),$$

where $\|\cdot\|_F$ is the Frobenius norm. Given an (ε, γ)-accurate dictionary $\mathcal{I}$, using Lemma 4.3 with $\alpha = \gamma$ it easy to see that the regularized projection on the dictionary $\mathcal{I}$ satisfies

$$\mathbf{\Pi}_n - \frac{\gamma}{1-\varepsilon}(\Phi\Phi^{\intercal} + \gamma\mathbf{\Pi}_n)^{-1} \preceq \Phi\mathbf{S}(\mathbf{S}^{\intercal}\Phi^{\intercal}\Phi\mathbf{S} + \gamma\mathbf{I}_{\mathcal{I}})^{-1}\mathbf{S}^{\intercal}\Phi^{\intercal} \preceq \Phi\mathbf{S}(\mathbf{S}^{\intercal}\Phi^{\intercal}\Phi\mathbf{S})^{+}\mathbf{S}^{\intercal}\Phi^{\intercal}.$$

and therefore using the columns in the dictionary as a projection

$$\begin{aligned}&\mathrm{Tr}(\Phi_n^\intercal\Phi_n - \Phi_n^\intercal\Phi\mathbf{S}(\mathbf{S}^\intercal\Phi^\intercal\Phi\mathbf{S})^+\mathbf{S}^\intercal\Phi^\intercal\Phi_n)\\ &\leq \mathrm{Tr}\left(\Phi_n^\intercal\Phi_n - \Phi_n^\intercal\mathbf{\Pi}_n\Phi_n + \frac{\gamma}{1-\varepsilon}\Phi_n^\intercal\left(\Phi\Phi^\intercal + \gamma\mathbf{\Pi}_n\right)^{-1}\Phi_n\right) = \frac{\gamma}{1-\varepsilon}d_{\mathrm{eff}}^n(\gamma).\end{aligned}$$

All that is left is to see how far away the optimal $\mathbf{\Pi}_k$ is, in terms of Frobenius norm, from $d_{\mathrm{eff}}^n(\gamma)$, where we will fix k and optimize over γ.

Let again $\Phi_n = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^\intercal$ be the SVD decomposition of Φ_n. We know that the K-PCA objective is minimized by $\mathbf{V}_k$, the map containing the first k columns $\mathbf{v}_i$ of $\mathbf{V}$ (Schölkopf et al., 1999), and therefore $\mathbf{\Pi}_k = \mathbf{V}_k\mathbf{V}_k^\intercal$. Similarly, let $\mathbf{\Sigma}_k$ be the diagonal matrix with only the first k singular value on the diagonal. The cost of the optimal solution is

$$\mathrm{Tr}(\Phi_n^\intercal\Phi_n - \Phi_n^\intercal\mathbf{\Pi}_k\Phi_n) = \mathrm{Tr}(\Phi_n^\intercal\Phi_n - \Phi_n^\intercal\mathbf{V}_k\mathbf{V}_k^\intercal\Phi_n) = \mathrm{Tr}(\mathbf{\Sigma}^\intercal\mathbf{\Sigma} - \mathbf{\Sigma}_k^\intercal\mathbf{\Sigma}_k) = \sum_{i=k+1}^n \sigma_i^2.$$

Setting $\gamma = \frac{1}{k}\sum_{i=k+1}^n \sigma_i^2$ we have

$$\begin{aligned}d_{\mathrm{eff}}^n\left(\frac{1}{k}\sum_{i=k+1}^n \sigma_i^2\right) &= \sum_{j=1}^n \frac{\sigma_j^2}{\sigma_j^2 + \frac{1}{k}\sum_{i=k+1}^n \sigma_i^2}\\ &= \sum_{j=1}^k \frac{\sigma_j^2}{\sigma_j^2 + \frac{1}{k}\sum_{i=k+1}^n \sigma_i^2} + \sum_{j=k+1}^n \frac{\sigma_j^2}{\sigma_j^2 + \frac{1}{k}\sum_{i=k+1}^n \sigma_i^2}\\ &\leq \sum_{j=1}^k 1 + \sum_{j=k+1}^n \frac{\sigma_j^2}{\frac{1}{k}\sum_{i=k+1}^n \sigma_i^2} = k + k\sum_{j=k+1}^n \frac{\sigma_j^2}{\sum_{i=k+1}^n \sigma_i^2} = 2k.\end{aligned}$$

Putting it all together we have the final result.

Theorem 4.6 Given a dictionary $\mathcal{I}$ that is (ε,γ)-accurate w.r.t. map $\Phi : \mathbb{R}^n \to \mathcal{H}$, and its selection matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$ the projection $\mathbf{\Pi}_{\mathcal{I}} = \Phi_n^\intercal\Phi\mathbf{S}(\mathbf{S}^\intercal\Phi^\intercal\Phi\mathbf{S})^+\mathbf{S}^\intercal\Phi^\intercal$ on the atoms contained in the dictionary satisfies

$$\|\Phi_n - \mathbf{\Pi}_{\mathcal{I}}\Phi_n\|_F^2 \leq \frac{\gamma}{(1-\varepsilon)}d_{\mathrm{eff}}^n(\gamma).$$

If $\gamma = \frac{1}{k}\sum_{i=k+1}^n \sigma_i^2 = \|\Phi_n - \mathbf{\Pi}_k\Phi_n\|_F^2/k$ we have

$$\|\Phi_n - \mathbf{\Pi}_{\mathcal{I}}\Phi_n\|_F^2 \leq \frac{2}{1-\varepsilon}\|\Phi_n - \mathbf{\Pi}_k\Phi_n\|_F^2.$$

While in general it is not easy to compute the right γ that satisfies the condition, it is possible to invert the relationship, i.e., express k in function of γ and the spectrum of the matrix, and use additional information on Φ_n (e.g., bounded trace) to know how many accurate eigenfunctions we can extract.

Computationally, using SQUEAK with a sequential merge tree to compute the (ε,γ)-accurate dictionary matches the near-linear time and $\mathcal{O}(d_{\mathrm{eff}}^n(\gamma))$ space complexity of other state of the art batch approximate K-PCA methods (Musco and Musco, 2017), or outperforms them using a parallel merge tree that requires less than linear runtime.

For streaming approximate K-PCA methods, existing methods that provide guarantees, such as the one proposed in Ghashami et al., (2016b), all require at least linear runtime,

and parallelizing SQUEAK gives us again an improvement. This despite the fact that many of these method require the input space to be Euclidean, and SQUEAK makes no such assumption.

4.3 Sequential RLS sampling for graph spectral sparsification

We will now look at the special case when the atoms that we want to select for our dictionary are edges in a graph $\mathcal{G}$.

Graphs are a useful tool in machine learning to study the relationship between heterogeneous entities. In many settings, the graph structure arises naturally from the data (e.g., social or collaboration networks, influence graphs), and we have only access to the nodes and edges without knowing in detail the process that generated them. Even if we have access to a flat vectorial representation for the entities (nodes), it is common to define a similarity function (e.g., exponential kernel), and construct a similarity graph out of the vectorial data.
Regardless of the origin of the graph $\mathcal{G}$ (*natural* or *constructed*), many important inference problems are defined starting from the *Laplacian* representation of the graph, for example, graph semi-supervised learning (SSL) (Chapelle et al., 2010; Zhu et al., 2003), graph regularized least squares (Belkin et al., 2006), laplacian embedding (Belkin and Niyogi, 2001) and spectral clustering (Von Luxburg, 2007).

The Laplacian of a graph can be expressed as the covariance matrix of the edge-vertex incidence vectors. Therefore, we can apply KORS and SQUEAK to approximate it. Unlike in the dictionary learning for $\mathcal{H}$ approximation problem, where KORS and SQUEAK greatly improve on existing methods, there are many efficient algorithms to approximate Laplacian matrices. In particular, the whole field of *graph spectral sparsification* is focused on this problem.
Similarly to the case of RLS sampling in Euclidean spaces, these methods can exploit particular properties of the graph setting, and are comparable or more efficient than the methods we proposed. For example, if we specialize KORS to this setting we recover an equivalent algorithm from Cohen et al., (2016), and both Kelner and Levin, (2013) and Kyng et al., (2017) proposed algorithms similar to SQUEAK with a fully sequential merge tree.
Nonetheless, looking in detail at the similarities between graph sparsification and learning an ε-accurate dictionary, will prepare us for the next section, where we show how graph sparsifiers can be applied to efficiently solve machine learning problems, and introduce new results for semi-supervised learning on graphs.

4.3.1 Graph spectral sparsification in the semi-streaming setting

We denote with $\mathcal{G} = (\mathcal{X}, \mathcal{E})$ an undirected weighted graph with n vertices $\mathcal{X}$ and m edges $\mathcal{E}$. Associated with each edge $e_{i,j} \in \mathcal{E}$ there is a weight $a_{e_{i,j}}$ (shortened a_e) measuring the "distance" between vertex i and vertex j.[1] Given two graphs $\mathcal{G}$ and $\mathcal{G}'$ over the same set of nodes $\mathcal{X}$, we denote by $\mathcal{G} + \mathcal{G}'$ the graph obtained by summing the weights of the edges of $\mathcal{G}'$ and $\mathcal{G}$.

Given the weighted adjacency matrix $\mathbf{A}_{\mathcal{G}}$ and the degree matrix $\mathbf{D}_{\mathcal{G}}$, the Laplacian of $\mathcal{G}$ is the PSD matrix defined as $\mathbf{L}_{\mathcal{G}} = \mathbf{D}_{\mathcal{G}} - \mathbf{A}_{\mathcal{G}}$. Furthermore, we assume that $\mathcal{G}$ is connected

[1] The graph $\mathcal{G}$ can be either constructed from raw data (e.g., building a k-nn graph with an exponential kernel) or it can be provided directly as input (e.g., in social networks).

and thus has only one eigenvalue equal to 0 and $\mathrm{Ker}(\mathbf{L}_\mathcal{G}) = \mathbf{1}$. Let $\mathbf{L}_\mathcal{G}^+$ be the pseudoinverse of $\mathbf{L}_\mathcal{G}$, and $\mathbf{L}_\mathcal{G}^{-1/2} = (\mathbf{L}_\mathcal{G}^+)^{1/2}$. For any node $i = 1, \dots, n$, we denote with $\chi_i \in \mathbb{R}^n$ the indicator vector so that $\mathbf{b}_e = \chi_i - \chi_j$ is the "edge" vector. If we denote with $\mathbf{B}_\mathcal{G}$ the $m \times n$ signed edge-vertex incidence matrix, then the Laplacian matrix can be written as $\mathbf{L}_\mathcal{G} = \sum_e a_e \mathbf{b}_e \mathbf{b}_e^\intercal = \mathbf{B}_\mathcal{G}^\intercal \mathbf{E}_\mathcal{G} \mathbf{B}_\mathcal{G}$, where $\mathbf{E}_\mathcal{G}$ is the $m \times m$ diagonal matrix with $\mathbf{E}_\mathcal{G}(e,e) = a_e$.

We indicate with $\mathbf{\Pi} = \mathbf{L}_\mathcal{G}\mathbf{L}_\mathcal{G}^+$ the matrix of the orthogonal projection on the $n-1$ dimensional space orthogonal to the all one vector $\mathbf{1}$. Since the Laplacian of any connected graph $\mathcal{G}$ has a null space equal to $\mathbf{1}$, then $\mathbf{\Pi}$ is invariant w.r.t. the specific graph $\mathcal{G}$ on n vertices used to defined it. Alternatively, the projection matrix $\mathbf{\Pi}$ can be obtained as $\mathbf{\Pi} = \mathbf{L}_\mathcal{G}^+\mathbf{L}_\mathcal{G} = \mathbf{L}_\mathcal{G}^{-1/2}\mathbf{L}_\mathcal{G}\mathbf{L}_\mathcal{G}^{-1/2}$. Finally, let $\boldsymbol{\pi}_e = \sqrt{a_e}\mathbf{L}_\mathcal{G}^{-1/2}\mathbf{b}_e$, then we have $\mathbf{\Pi} = \sum_e \boldsymbol{\pi}_e \boldsymbol{\pi}_e^\intercal$.

A graph $\mathcal{H}$ is a spectral sparsifier of $\mathcal{G}$ if the whole spectrum of the original graph is well approximated by using only a small portion of its edges. More formally,

Definition 4.2 A $1 \pm \varepsilon$ spectral sparsifier of $\mathcal{G}$ is a graph $\mathcal{H} \subseteq \mathcal{G}$ such that for all $\mathbf{x} \in \mathbb{R}^n$

$$(1-\varepsilon)\mathbf{x}^\intercal \mathbf{L}_\mathcal{G}\mathbf{x} \leq \mathbf{x}^\intercal \mathbf{L}_\mathcal{H}\mathbf{x} \leq (1+\varepsilon)\mathbf{x}^\intercal \mathbf{L}_\mathcal{G}\mathbf{x}.$$

It is easy to see that this is equivalent to the definition of an ε-accurate dictionary $\mathcal{I}_\mathcal{H}$ w.r.t. the signed edge-vertex incidence matrix $\mathbf{B}_\mathcal{G}^\intercal \mathbf{E}_\mathcal{G}^{1/2} \in \mathbb{R}^{n \times m}$. Then definition Definition 4.2 is just equivalent to Lemma 2.7 applied to $\mathbf{L}_\mathcal{H} = \mathbf{B}_\mathcal{G}^\intercal \mathbf{E}_\mathcal{G}^{1/2} \mathbf{S}\mathbf{S}^\intercal \mathbf{E}_\mathcal{G}^{1/2} \mathbf{B}_\mathcal{G}$.
In the context of Laplacian sparsification, the leverage scores $\tau_e = \|\pi_e\|^2$ are usually replaced by the effective resistances.

Definition 4.3 The effective resistance of an edge e in graph $\mathcal{G}$ is defined as $r_e = \mathbf{b}_e^\intercal \mathbf{L}_\mathcal{G}^+ \mathbf{b}_e$. The total weighted sum of effective resistances in a graph is the same for all graphs, and is equal to $\sum_e a_e r_e = \mathrm{Tr}(\mathbf{E}_\mathcal{G}\mathbf{B}_\mathcal{G}\mathbf{L}_\mathcal{G}^+\mathbf{B}_\mathcal{G}^\intercal) = \mathrm{Tr}(\mathbf{L}_\mathcal{G}\mathbf{L}_\mathcal{G}^+) = n-1$.

The effective resistance $r_e = \tau_e / a_e$ is defined as the leverage scores of edge e dived by its weight. It also has particular interpretations that are specific to the graph setting.
Intuitively, the effective resistance encodes the importance of an edge in preserving the minimum distance between two nodes. If an edge is the only connection between two parts of the graph, its r_e is large. On the other hand, if there are multiple parallel paths across many edges to connect two nodes, the effective resistance of an edge between the two nodes will be small, similarly to actual resistances in parallel in an electrical network.
An important consequence of this intuition is that it makes it easy to see that adding edges to a graph can only reduce the effective resistance of other edges, because it can only introduce new alternative (parallel) paths in the graph. This is yet another reformulation of Lemma 3.8 on the evolution of LS and RLS.
Also as expected, the weighted sum of the effective resistance is the rank of the Laplacian, and differently to our previous settings, it is always $n-1$.

Similarly to what Drineas et al., (2008) did for LS sampling (i.e., Corollary 2.6), Spielman and Srivastava, (2011) could prove that sampling the edges of $\mathcal{G}$ with replacement using a distribution proportional to their effective resistance (i.e., oracle Nyström sampling with LS) produces a spectral sparsifier $\mathcal{H}$ of size $\mathcal{O}(n \log(n)/\varepsilon^2)$ with high probability, regardless of the number of edges in the original graph.
As in Chapter 2 and Chapter 3, the main issue of this approach is that we want to compute a sparsifier to avoid storing and processing the whole Laplacian, but we need to store and (pseudo-)invert the Laplacian to compute exact effective resistances to construct the

sparsifier. But, as in the case of Nyström sampling in Euclidean spaces, there have been many approximate methods proposed over the years to approximate effective resistances. In particular, it is possible to estimate them using approaches unique to graphs, which fall mostly in two broad categories

- combinatorial methods, that exploit the interpretation of effective resistances as distances over parallel paths in the graphs and try to approximate this quantity, for example building ensembles of low-stretch trees (Kapralov and Panigrahy, 2012; Koutis and Xu, 2016), or recursive update rules (Kelner et al., 2013).
- solver based methods, that exploit the fact that effective resistances can be expressed as the solution of a linear system, i.e., computing $\min_{\mathbf{B}_{\mathcal{G}}^{\mathsf{T}}\mathbf{E}^{1/2}\mathbf{x}=\mathbf{b}_e} \|\mathbf{x}\|^2$ (Koutis et al., 2012),

We will focus on solver based methods, which are based on fast, graph-specific algorithms to solve Laplacian linear systems, such as the one proposed in Koutis et al., (2012). Note that this fast solvers also allowed Cohen et al., (2016) to directly adapt efficient leverage score estimation techniques for Euclidean spaces to the graph setting and obtain efficient methods.
The details of these fast solvers are beyond the scope of this thesis, and we refer the reader to Spielman, (2017) for further reference. For our purposes, we just assume we have access to a fast method that can estimate the effective resistance of a graph. For the rest of the thesis we will focus on the one described by Kelner and Levin, (2013) that internally uses the Laplacian solver introduced by Koutis et al., (2012).
Kelner and Levin, (2013) show that given as input a graph with m edges, we can compute an ρ-accurate estimate of the effective resistance of all in edges in the graph in $\mathcal{O}(m\log(n))$ time and $\mathcal{O}(m)$ space, or in other words in near-linear time and space in m.

As Spielman and Srivastava, (2011) note, computing all effective resistances in near-linear time is already optimal, since we need anyway $\mathcal{O}(m)$ time to sample or even to just look at all edges. Conversely, the $\mathcal{O}(m)$ space requirement is much larger than the optimal $\mathcal{O}(n\log(n))$ sparsifier size, and is not feasible when m is large.

4.3.2 SQUEAK for graph Laplacian approximation

To reduce the space complexity of graph sparsification, Kelner and Levin, (2013) introduced a simple single-pass approach to generate a spectral sparsifier of a graph in the semi-streaming setting, where edges are received one at a time. They store only an intermediate, approximate sparsifier and every time a new edge arrives, it is added to it. Whenever the sparsifier gets too large, they apply a resparsification algorithm to reduce its size, without compromising its spectral guarantees.

Although the algorithm is intuitive and simple to implement, the original proof presented in their paper is incomplete, as originally pointed out in Cohen et al., (2016). In particular, Kelner and Levin, (2013) relies on a concentration inequality for independent random variables, while in the sparsification algorithm the probability of keeping edge e in the sparsifier at step s does depend on whether other edges e' have been included in the sparsifier at previous iterations. This structure introduces subtle statistical dependencies through different iterations of the algorithm, and a more careful analysis is necessary. In this section, we will show that the analysis of SQUEAK can be directly applied to complete the analysis of Kelner and Levin, (2013).

In addition to pointing out the problems with the original proof in Kelner and Levin, (2013),

Algorithm 11 Kelner and Levin (Kelner and Levin, 2013) stream sparsification algorithm.

Input: Graph $\mathcal{G}$, weights a_e for all edges in $\mathcal{G}$.
Output: $\mathcal{H}_\tau$, a $1 \pm \varepsilon$ sparsifier of $\mathcal{G}$

Set space budget $\overline{q} = 40\rho \log(3m/\delta)/\varepsilon^2$
Partition the graph (dataset) $\mathcal{G}$ into $k = \lceil m/\overline{q} \rceil$ edge-disjoint sub-graphs (sub-datasets) $\{\mathcal{G}_s\}_{s=0}^k$ such that $\mathcal{G} = \sum_{s=0}^k \mathcal{G}_s$
Initialize the sparsifier (dictionary) $\mathcal{H}_0 = \emptyset$
for all $s \in [1, \dots, k]$ **do**
  Receive $\mathcal{G}_s$
  Set $\mathcal{H}_* = \mathcal{H}_{s-1} + \mathcal{G}_s$
  Compute ρ-accurate estimates of all effective resistances $\widetilde{r}_{s,e}$ in $\mathcal{H}_*$ using a fast SDD solver (Kelner and Levin, 2013, Theorem 3)
  Compute new probabilities $\widetilde{p}_{s,e} = a_e \widetilde{r}_{s,e}$
  for all edges $e \in \mathcal{H}_{s-1}$ **do**
    $\widetilde{p}_{s,e} \leftarrow \min\{\widetilde{p}_{s-1,e}, \widetilde{p}_{s,e}\}$
  end for
  Initialize $\mathcal{H}_s = \emptyset$
  for all edges $e \in \mathcal{H}_{s-1}$ **do**
    Set $q_{s,e} \sim \mathcal{B}(\widetilde{p}_{s,e}/\widetilde{p}_{s-1,e}, q_{s-1,e})$
    If $q_{s,e} \neq 0$ add edge e to $\mathcal{H}_s$ with weight $\frac{a_e}{\widetilde{p}_{s,e}} \frac{q_{s,e}}{\overline{q}}$
    If $q_{s,e} = 0$ discard edge from memory{▷ SHRINK }
  end for
  for all edges $e \in \mathcal{G}_s$ **do**
    Set $q_{s,e} \sim \mathcal{B}(\widetilde{p}_{s,e}, \overline{q})$
    If $q_{s,e} \neq 0$ add edge e to $\mathcal{H}_s$ with weight $\frac{a_e}{\widetilde{p}_{s,e}} \frac{q_{s,e}}{\overline{q}}$ {▷ EXPAND }
  end for
  Return $\mathcal{H}_k$
end for

Cohen et al., (2016) introduces a new algorithm to construct a sparsifier in a semi-streaming setting but, differently from the original algorithm in Kelner and Levin, (2013), interactions between iterations are avoided because the algorithm proposed in Cohen et al., (2016) never drops an edge once it is introduced in the sparsifier. In particular, Cohen et al., (2016) is equivalent to specializing KORS to the problem of graph sparsification, and therefore only able to generate a suboptimal $\mathcal{O}(n \log^2(n))$ number of edges (Theorem 3.11).

If we specialize SQUEAK to the specific setting of effective resistance sampling and graph sparsification, and use a fully sequential tree, we recover exactly the original algorithm of Kelner and Levin, (2013), reported in Algorithm 11. Using the analysis of SQUEAK, we can also prove that w.h.p. it generates an ε-accurate dictionary, or in other words a graph sparsifier.

Theorem 4.7 Given parameters $0 < \varepsilon \leq 1$, $0 < \delta < 1$, an arbitrary graph $\mathcal{G}$, and a fully sequential merge tree of height $\lceil m/\overline{q} \rceil$, let $\rho = \frac{1+3\varepsilon}{1-\varepsilon}$ and run Algorithm 11 with **budget** $\overline{q} \geq 26\rho \log(3m/\delta)/\varepsilon^2$. Then w.p. $1-\delta$,

Accuracy: each graph sparsifier $\mathcal{H}_s$ is ε-accurate w.r.t. $\sum_{t=1}^s \mathcal{G}_t$, and each $\widetilde{r}_{s,e}$ is ρ-accurate w.r.t. $r_{s,e}$.

Space: the size of each sparsifier is bounded by $|\mathcal{H}_s| \leq 3\overline{q} d_{\text{eff}}^s(0) \leq 3\overline{q}n$.

Time: the algorithm runs in $\mathcal{O}(n\log^2(n))$ per-merge time, and $\mathcal{O}(m\log(n))$ overall time.
Passes: the algorithm requires a single pass over $\mathcal{G}$.

Unlike in the analysis of SQUEAK, here we have fixed the merge tree (fully sequential) and we have a closed form solution for the effective dimension $d_{\text{eff}}(0) = \text{Rank}(\mathbf{L}_{\mathcal{G}}) = n-1$, and therefore a $\mathcal{O}(n\log(n))$ bound on the size of the dictionary and the memory requirement of the algorithm.
To compute the time complexity, we combine the runtime analysis of SQUEAK, the number of merges $\lceil m/\overline{q}\rceil$, the $\mathcal{O}(n\log(n))$ sparsifier (dictionary) size bound, and the $\mathcal{O}(|\mathcal{I}_s|\log(n)) \leq \mathcal{O}(n\log^2(n))$ cost of estimating the effective resistances (LS) using a near-linear time solver. The final result is a closed-form $\mathcal{O}(m\log(n))$ bound on the runtime of the algorithm.
Compared to Kelner and Levin, (2013) this is a $\log(n)$ factor improvement in space and runtime, not due to any change in the algorithm, but simply because of a tighter analysis. We can also compare our result with the independent analysis of Kyng et al., (2017) of sequential graph resparsification, which also resolves the problems in Kelner and Levin, (2013)'s analysis. Kyng et al., (2017) can be seen as a special case of our sequential RLS sampling with removal (i.e., SQUEAK), and both works take a similar analysis approach based on the result of Pachocki, (2016). As the main tool for the proof, Kyng et al., (2017) uses an adversarial game argument to bound the variance of the process $\mathbf{W}_t$ (see Section 3.4.4). For SQUEAK's proof instead we derived an argument based on stochastic dominance that essentially tells us how the adversary of Kyng et al., (2017) can be constructed.
Moreover, since they focus on the graph setting and effective resistances (LS), they do not generalize their result to RLS, or have the need to derive an estimator that can work both in Euclidean and RKH spaces (i.e., Lemma 3.14), relying instead on fast solvers.
Conversely, when Kyng et al., (2017) move from sequential to parallel graph sparsification, they take an approach completely different from the one we used for SQUEAK. Although the resulting algorithm is only applicable to graphs, since it is based on combinatorial arguments, it can extract more parallelism than SQUEAK and construct a graph sparsifier in a smaller time.

4.4 Efficient graph semi-supervised learning

We can now show how using graph sparsification, in particular fully sequential SQUEAK, will allow us to greatly reduce the computational cost of solving semi supervised learning problems.

In many classification and regression tasks, obtaining labels for large datasets is expensive. When the number of labeled samples is too small, traditional supervised learning algorithms fail in learning accurate predictors. *Semi-supervised learning* (SSL) (Chapelle et al., 2010; Zhu, 2008) effectively deals with this problem by integrating the labeled samples with an additional set of unlabeled samples, which are abundant and readily available in many applications (e.g., set of images collected on web sites (Fergus et al., 2009)).
The intuition behind SSL is that unlabeled data may reveal the underlying structure of the problem (e.g., a manifold) that could be exploited to compensate for the limited number of labels and improve the prediction accuracy. Among different SSL settings, in this thesis we focus on the case where data are embedded in a *graph*.
The graph is expected to effectively represent the geometry of data and graph-based

SSL (Belkin et al., 2006; Subramanya and Talukdar, 2014; Zhu et al., 2003) methods leverage the intuition that nodes that are similar according to the graph are more likely to be labeled similarly.
A popular approach is the *harmonic function solution* (HFS) (Belkin et al., 2004; Fergus et al., 2009; Zhu et al., 2003), whose objective is to find a solution where each node's value is the weighted average of its neighbors. Computing the HFS solution requires solving a Laplacian regularized least-squares problem. While the resulting solution is both empirically effective (Fergus et al., 2009) and enjoys strong performance guarantees (Belkin et al., 2006; Cortes et al., 2008), solving *exactly* the least-squares problem on a graph with n nodes amounts to $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space complexity.
Using a general iterative solver on a graph with m edges to obtain a comparable solution requires only $\mathcal{O}(mn)$ time and $\mathcal{O}(m)$ space, but this is still practically unfeasible in many applications of interest. In fact, many graphs have naturally a large number of edges, so that even if the n nodes could fit in memory, storing m edges largely exceeds the memory capacity. For instance, Facebook's graph of relationships (Ching et al., 2015) counts about $n = 1.39e9$ users connected by a trillion ($m = 1e12$) edges. While n is still in the order of a normal memory capacity, the edges cannot be stored in a single computer.
A similar issue is faced when the graph is built starting from a dataset, for instance using a k-nn graph. In this case $m = kn$ edges are created, and sometimes a large k is necessary to obtain good performance (Saluja et al., 2014), or artificially adding neighbours can improve the stability of the method (Gleich and Mahoney, 2015).
In such problems, a direct application of HFS is not possible and thus some form of approximation or graph sketching is required. A straightforward approach is to distribute the graph over multiple machines in a cluster and resort to an iterative solver for the solution of the least-squares problem (Ching et al., 2015). Distributed algorithms require infrastructure and careful engineering in order to deal with communication issues (Cai et al., 2014).
But even assuming that these problems are satisfactorily dealt with, all known iterative solvers that have provably fast convergence do not have known distributed implementations, as they assume random, constant time access to all the edges in the graph. Thus one would have to resort to distributed implementations of simpler but much slower methods, in effect trading-off space for a significant reduction in overall efficiency.

More principled methods try to address the memory bottleneck by directly manipulating the structure of the graph to reduce its size. These include *subsampling* the nodes of the original graph, *quantization*, approaches related to *manifold learning*, and various *approximation* strategies.
The most straightforward way to reduce the complexity in graph-based method is to subsample the nodes to create a smaller, *backbone graph* of representative vertices, or *landmarks* (Talwalkar et al., 2008). Nyström sampling methods (Kumar et al., 2012) randomly select s nodes from the original graph and compute q eigenvectors of the smaller graph, which can be later used to solve the HFS regularized problem. It can be shown (Kumar et al., 2012) that the reconstructed Laplacian is accurate in ℓ_2-norm and thus only its largest eigenvalue is preserved. Unfortunately, the HFS solution does not depend only on the largest eigenvalues, both because the largest eigenvectors are the ones most penalized by HFS's regularizer ((Belkin et al., 2006)) and because theoretical analysis shows that preserving the smallest eigenvalue is important for generalization bounds ((Belkin et al., 2004)).
As a result, subsampling methods can completely fail when the sampled nodes compromise the spectral structure of the graph (Fergus et al., 2009). Although alternative techniques

have been developed over years (see e.g., (Garcke and Griebel, 2005; Jebara et al., 2009; Liu et al., 2010; Tsang and Kwok, 2006; Yu and Yu, 2005; Zhu and Lafferty, 2005)), this drawback is common to all backbone graph methods.
Motivated by this observation, other approaches focus on computing a more accurate approximation of the spectrum of the Laplacian. Fergus *et al.* (Fergus et al., 2009) build on the observation that when the number of unlabeled samples n tends to infinity, the eigenvectors of the Laplacian tend to the eigenfunctions of the sampling distribution $\mathcal{P}$. Thus instead of approximating eigenvectors in $\mathbb{R}^n$, they first compute empirical eigenfunctions of the estimated sampling distribution (defined on the d-dimensional feature space) obtained by assuming that $\mathcal{P}$ is factorized and by using a histogram estimation over b bins over each dimension separately. While the method scales to the order of million nodes, it still requires d and b to be small to be efficient. Furthermore, no theoretical analysis is available, and the method may return poor approximations whenever the sampling distribution is not factorized.
Motivated by the empirical success of (Fergus et al., 2009), Ji *et al.* (Ji et al., 2012) proposed a similar algorithm, Simple-HFS, for which they provide theoretical guarantees. However, in order to prove bounds on the generalization error, they need to assume several strong and hard to verify assumptions, such as a sufficiently large eigengap. On the contrary, the guarantees for our method work for any graph.

Our contribution

In this section, we focus on reducing the space complexity of graph-based SSL while matching the smallest possible computational complexity of $\Omega(m)$ up to logarithmic factors[2] and providing strong guarantees about the quality of the solution. In particular, we introduce a novel approach which employs efficient spectral graph sparsification techniques to incrementally process the original graph. This method, coupled with dedicated solvers for symmetric diagonally dominant (SDD) systems (Koutis et al., 2011), allows to find an approximate HFS solution without storing the whole graph in memory and to control the computational complexity as n grows.
In fact, we show that our proposed method, called Sparse-HFS, requires only fixed $\mathcal{O}(n\log(n))$ space to run, and allows to compute solutions to large HFS problems in memory. For example, in the experimental section we show that the sparsifier can achieve an accuracy comparable to the full graph, using one order of magnitude less edges. With a careful choice of the frequency of resparsification, the proposed method does not increase significantly the running time. Given a minimum amortized cost of $\Omega(1)$ per edge, necessary to examine each edge at least once, our algorithm only increases this cost to $\mathcal{O}(\log(n))$. Furthermore, using the approximation properties of spectral sparsifiers and results from algorithmic stability theory (Bousquet and Elisseeff, 2002; Cortes et al., 2008) we provide theoretical guarantees for the generalization error for Sparse-HFS, showing that the performance is asymptotically the same as the exact solution of HFS.
Finally, we report empirical results on both synthetic and real data showing that Sparse-HFS is competitive with subsampling and the EigFun method in (Fergus et al., 2009).

[2]While the computational complexity of exact HFS is $\mathcal{O}(mn)$, many approximated methods can significantly reduce it. Nonetheless, any method that requires reading all the edges once has at least $\Omega(m)$ time complexity.

4.4.1 Semi-supervised learning with graph sparsifiers

Notation. We consider the problem of regression in the semi-supervised setting, where a large set of n points $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \subset \mathbb{R}^d$ is drawn from a distribution $\mathcal{P}$ and labels $\{y_i\}_{i=1}^l$ are provided only for a small (random) subset $\mathcal{S} \subset \mathcal{X}$ of l points. Graph-based SSL builds on the observation that $\mathcal{P}$ is often far from being uniform and it may display a specific structure that could be exploited to "propagate" the labels to similar unlabeled points. Building on this intuition, graph-based SSL algorithms consider the case when the points in $\mathcal{X}$ are embedded into an undirected weighted graph $\mathcal{G} = (\mathcal{X}, \mathcal{E})$ with $|\mathcal{E}| = m$ edges. Associated with each edge $e_{i,j} \in \mathcal{E}$ there is a weight $a_{e_{i,j}}$ measuring the "distance" between $\mathbf{x}_i$ and $\mathbf{x}_j$.

A graph-based SSL algorithm receives as input $\mathcal{G}$ and the labels of the nodes in $\mathcal{S}$ and it returns a function $\mathbf{f} : \mathcal{X} \to \mathbb{R}$ that predicts the label for all nodes in $\mathcal{X}$. The objective is to minimize the prediction error over the set $\mathcal{T}$ of $u = n - l$ unlabeled nodes. In the following we denote by $\mathbf{y} \in \mathbb{R}^n$ the full vector of labels.

Stable-HFS. HFS directly exploits the structure embedded in $\mathcal{G}$ to learn functions that are smooth over the graph, thus predicting similar labels for similar nodes. We assume that $\mathcal{G}$ is connected and thus has only one eigenvalue at 0. Let $\mathbf{L}_\mathcal{G}^+$ be the pseudoinverse of $\mathbf{L}_\mathcal{G}$, and $\mathbf{L}_\mathcal{G}^{-1/2} = (\mathbf{L}_\mathcal{G}^+)^{1/2}$. The HFS method (Zhu et al., 2003) can be formulated as the Laplacian-regularized least-squares problem

$$\widehat{\mathbf{f}} = \operatorname*{arg\,min}_{\mathbf{f} \in \mathbb{R}^n} \tfrac{1}{l}(\mathbf{f} - \mathbf{y})^\intercal \mathbf{I}_\mathcal{S}(\mathbf{f} - \mathbf{y}) + \eta \mathbf{f}^\intercal \mathbf{L}_\mathcal{G} \mathbf{f}, \tag{4.7}$$

where $\mathbf{I}_\mathcal{S} \in \mathbb{R}^{n \times n}$ is the identity matrix with zeros corresponding to the nodes not in $\mathcal{S}$ and η is a regularization parameter. The solution can be computed in closed form as $\widehat{\mathbf{f}} = (\eta l \mathbf{L}_\mathcal{G} + \mathbf{I}_\mathcal{S})^+ \mathbf{y}_\mathcal{S}$, where $\mathbf{y}_\mathcal{S} = \mathbf{I}_\mathcal{S}\mathbf{y} \in \mathbb{R}^n$.
The singularity of the Laplacian may lead to unstable behavior with drastically different results for small perturbations to the dataset. For this reason, we focus on the Stable-HFS algorithm proposed in (Belkin et al., 2004) where an additional regularization term is introduced to restrict the space of admissible hypotheses to the space $\mathcal{F} = \{\mathbf{f} : \langle \mathbf{f}, \mathbf{1} \rangle = 0\}$ of functions orthogonal to null space of $\mathbf{L}_\mathcal{G}$ (i.e., centered functions). This restriction can be easily enforced by introducing an additional regularization term $\frac{\mu}{l}\mathbf{f}^\top \mathbf{1}$ in Eq. 4.7.
As shown in (Belkin et al., 2004), in order to guarantee that the resulting $\widehat{\mathbf{f}}$ actually belongs to $\mathcal{F}$, it is sufficient to set $\mu = ((\eta l \mathbf{L}_\mathcal{G} + \mathbf{I}_\mathcal{S})^+ \mathbf{y}_S)^\intercal \mathbf{1} / ((\eta l \mathbf{L}_\mathcal{G} + \mathbf{I}_\mathcal{S})^+ \mathbf{1})^\intercal \mathbf{1}$, and compute the solution as $\widehat{\mathbf{f}} = (\eta l \mathbf{L}_\mathcal{G} + \mathbf{I}_\mathcal{S})^+ (\mathbf{y}_\mathcal{S} - \mu \mathbf{1})$.
Furthermore, it can be shown that if we center the vector of labels $\widetilde{\mathbf{y}}_\mathcal{S} = \mathbf{y}_\mathcal{S} - \overline{\mathbf{y}}_\mathcal{S}$, with $\overline{\mathbf{y}} = \frac{1}{l}\mathbf{y}_\mathcal{S}^\intercal \mathbf{1}$, then the solution of Stable-HFS can be rewritten as projected solution $\widehat{\mathbf{f}} = (\eta l \mathbf{L}_\mathcal{G} + \mathbf{I}_\mathcal{S})^+ (\widetilde{\mathbf{y}}_\mathcal{S} - \mu \mathbf{1}) = \big(\mathbf{\Pi}_\mathcal{F}(\eta l \mathbf{L}_\mathcal{G} + \mathbf{I}_\mathcal{S})\big)^+ \widetilde{\mathbf{y}}_\mathcal{S}$, where $\mathbf{\Pi}_\mathcal{F} = \mathbf{L}_\mathcal{G}\mathbf{L}_\mathcal{G}^+$ is the projection matrix on the $n-1$ dimensional space $\mathcal{F}$. Indeed, since the Laplacian of any graph $\mathcal{G}$ has a null space equal to the one vector $\mathbf{1}$, then $\mathbf{\Pi}_\mathcal{F}$ is invariant w.r.t. the specific graph $\mathcal{G}$ used to defined it. While Stable-HFS is more stable and thus more suited for theoretical analysis, its time and space requirements remain $\mathcal{O}(mn)$ and $\mathcal{O}(m)$, and cannot be applied to graph with a large number of edges.

To accelerate Stable-HFS, we introduce a novel variant of HFS, called Sparse-HFS, where the graph $\mathcal{G}$ is pre-processed using spectral graph sparsification techniques, and the Stable-HFS solution is computed efficiently on a smaller subgraph $\mathcal{H}$, drastically reducing the time and memory requirements without compromising the resulting accuracy.

Sparse-HFS. Replacing the graph $\mathcal{G}$ with the sparsifier produced by Algorithm 11 (i.e., fully sequential SQUEAK), we can compute an *approximate* Stable-HFS solution as $\widetilde{\mathbf{f}} = (\eta l \mathbf{L}_{\mathcal{H}} + \mathbf{I}_{\mathcal{S}})^{+}(\mathbf{y}_{\mathcal{S}} - \mu \mathbf{1})$, with $\mu = ((\eta l \mathbf{L}_{\mathcal{H}} + \mathbf{I}_{\mathcal{S}})^{+}\mathbf{y}_{S})^{\intercal}\mathbf{1}/((\eta l \mathbf{L}_{\mathcal{H}} + \mathbf{I}_{\mathcal{S}})^{+}\mathbf{1})^{\intercal}\mathbf{1}$. Since both $\mathcal{G}$ and $\mathcal{H}$ are connected graphs, they induce the same projection matrix $\mathbf{\Pi}_{\mathcal{F}}$, and the Sparse-HFS solution can be rewritten as $\widetilde{\mathbf{f}} = \left(\mathbf{\Pi}_{\mathcal{F}}(\eta l \mathbf{L}_{\mathcal{H}} + \mathbf{I}_{\mathcal{S}})\right)^{+}\widetilde{\mathbf{y}}_{\mathcal{S}}$.
Using the runtime guarantees of SQUEAK and of the near-linear solver by Koutis et al., (2011) it is easy to see how the pre-processing sparsification step takes only $\mathcal{O}(m \log(n))$ time. Computing the final Sparse-HFS solution using $\mathcal{H}$ and the near-linear time solver only adds $\mathcal{O}(n \log^2(n))$ time. Since $m \geq n$ because the graph is connected, we have that the overall runtime of the algorithm is $\mathcal{O}(m \log(n))$.
Therefore Sparse-HFS gives us drastic improvements over Stable-HFS's $O(m)$ space and $O(mn)$ time, both in terms of time and space complexity. This allows us to scale Stable-HFS to graphs orders of magnitude bigger.
These computational improvements have only a limited impact on the spectrum of $\mathcal{G}$ and all its eigenvalues are approximated up to a $(1 \pm \varepsilon)$ factor. Moreover, all of the sparsification guarantees hold w.h.p. for any graph, regardless of how it is generated, its original spectrum, and more importantly regardless of the exact order in which the edges are assigned to the blocks. Finally, we notice that the choice of the number of blocks as $m/\overline{q}$ is crucial to guarantee a logarithmic amortized time, since each iteration takes $O(n\overline{q} \log(n))$ time. This property allows to directly apply Sparse-HFS in online learning settings where edges arrive in a stream and intermediate solutions have to be computed incrementally.

4.4.2 Bounding the stability

In the following, we show that, unlike other heuristics, the space complexity improvements obtained with sparsification come with guarantees and do not degrade the actual learning performance of HFS. The analysis of SSL algorithms is built around the algorithm stability theory (Bousquet and Elisseeff, 2002), which has been extensively used to analyse transductive learning algorithms (Cortes et al., 2008; El-Yaniv and Pechyony, 2006). We first remind the definition of algorithmic stability.

Definition 4.4 Let $\mathcal{L}$ be a transductive learning algorithm. We denote by $\mathbf{f}$ and $\mathbf{f}'$ the functions obtained by running $\mathcal{L}$ on datasets $\mathcal{X} = (\mathcal{S}, \mathcal{T})$ and $\mathcal{X} = (\mathcal{S}', \mathcal{T}')$ respectively. $\mathcal{L}$ is uniformly β-stable w.r.t. the squared loss if there exists $\beta \geq 0$ such that for any two partitions $(\mathcal{S}, \mathcal{T})$ and $(\mathcal{S}', \mathcal{T}')$ that differ by exactly one training (and test) point and for all $\mathbf{x} \in \mathcal{X}$,

$$|(\mathbf{f}(\mathbf{x}) - \mathbf{y}(\mathbf{x}))^2 - (\mathbf{f}'(\mathbf{x}) - \mathbf{y}(\mathbf{x}))^2| \leq \beta.$$

Define the empirical error as $\widehat{R}(\mathbf{f}) = \frac{1}{l}\sum_{i=1}^{l}(\mathbf{f}(x_i) - \mathbf{y}(x_i))^2$ and the generalization error as $R(\mathbf{f}) = \frac{1}{u}\sum_{i=1}^{u}(\mathbf{f}(x_i) - \mathbf{y}(x_i))^2$.

Theorem 4.8 — (Calandriello et al., 2015). Let $\mathcal{G}$ be a fixed (connected) graph with n nodes $\mathcal{X}$ and m edges $\mathcal{E}$ and eigenvalues $0 = \lambda_1(\mathcal{G}) < \lambda_2(\mathcal{G}) \leq \ldots \leq \lambda_n(\mathcal{G})$. Let $\mathbf{y} \in \mathbb{R}^n$ be the labels of the nodes in $\mathcal{G}$ with $|\mathbf{y}(x)| \leq M$ and $\mathcal{F}$ be the set of centered functions such that $|\mathbf{f}(x) - \mathbf{y}(x)| \leq c$. Let $\mathcal{S} \subset \mathcal{X}$ be a random subset of labeled nodes. If the corresponding labels $\widetilde{\mathbf{y}}_S$ are centered and Sparse-HFS is run with parameter ε, then w.p. at least $1 - \delta$ (w.r.t. the random generation of the sparsifier $\mathcal{H}$ and the random

subset of labeled points $\mathcal{S}$) the resulting function $\widetilde{\mathbf{f}}$ satisfies,

$$R(\widetilde{\mathbf{f}}) \leq \widehat{R}(\widehat{\mathbf{f}}) + \beta + \left(2\beta + \frac{c^2(l+u)}{lu}\right)\sqrt{\frac{\pi(l,u)\ln\frac{1}{\delta}}{2}} + \left(\frac{\varepsilon l\eta\lambda_2(\mathcal{G})M}{((1-\varepsilon)l\eta\lambda_2(\mathcal{G})-1)^2}\right)^2, \quad 4.8$$

where $\widehat{\mathbf{f}}$ is the solution of exact Stable-HFS on $\mathcal{G}$,

$$\pi(l,u) = \frac{lu}{l+u-0.5}\frac{2\max\{l,u\}}{2\max\{l,u\}-1},$$

and

$$\beta \leq \frac{1.5M\sqrt{l}}{((1-\varepsilon)l\eta\lambda_2(\mathcal{G})-1)^2} + \frac{4M}{(1-\varepsilon)l\eta\lambda_2(\mathcal{G})-1}.$$

Theorem 4.8 shows how approximating $\mathcal{G}$ with $\mathcal{H}$ impacts the generalization error as the number of labeled samples l increases. If we set $\varepsilon = 0$, we recover the exact case bound by Cortes et al., (2008), which depends only on $\widehat{R}(\widehat{\mathbf{f}})$ and β.
When $\varepsilon > 0$, we see from Eq. 4.8 that the two terms already present in the exact case are either unchanged ($\widehat{R}(\widehat{\mathbf{f}})$) or increase only by a constant factor (β). Because of the approximation, a new error term (the last one in Eq. 4.8) is added to the bound, but we can see that it is negligible compared to β. In fact, it converges to zero as $O(\varepsilon^2/l^2(1-\varepsilon)^4)$ as l grows and it is dominated by β for any constant value of ε.
This means that increasing ε corresponds to a constant increase in the bound, regardless of the size of the problem. Consequently, ε can be freely chosen to trade off accuracy and space complexity (Theorem 4.7) depending on the problem constraints. Furthermore running time does not depend on this trade-off, because a larger block size is balanced by less frequent resparsifications.
Finally, because the eigenvalues present in the bound are the ones of the original graph, any external knowledge on the spectral properties of the input graph can be easily included in the analysis. Therefore it is straightforward to provide stronger guarantees for Sparse-HFS when combined with assumptions on the graph generating model. We also remark the level of generality of this result that holds for the integration between HFS and any ε-accurate spectral sparsification method.

Proof of Theorem 4.8. **Step 1 (generalization of stable algorithms).** Let β be the stability of Sparse-HFS, then using the result in (Cortes et al., 2008), we have that with probability at least $1-\delta$ (w.r.t. the randomness of the labeled set $\mathcal{S}$) the solution $\widetilde{\mathbf{f}}$ returned by the Sparse-HFS satisfies

$$R(\widetilde{\mathbf{f}}) \leq \widehat{R}(\widetilde{\mathbf{f}}) + \beta + \left(2\beta + \frac{c^2(l+u)}{lu}\right)\sqrt{\frac{\pi(l,u)\log(1/\delta)}{2}}.$$

In order to obtain the final result we first derive an upper bound on the stability of Sparse-HFS and then relate its empirical error to the one of Stable-HFS.

Step 2 (stability). The bound on the stability follows similar steps as in the analysis of Stable-HFS in Belkin et al., (2004) integrated with the properties of streaming spectral sparsifiers reported in Definition 4.2 and Proposition 2.1.
Let $\mathcal{S}$ and $\mathcal{S}'$ be two labeled sets only differing by one element and $\widetilde{\mathbf{f}}$ and $\widetilde{\mathbf{f}}'$ be the solutions obtained by running Sparse-HFS on $\mathcal{S}$ and $\mathcal{S}'$ respectively. Without loss of generality,

we assume that $\mathbf{I}_{\mathcal{S}}(l,l)=1$ and $\mathbf{I}_{\mathcal{S}}(l+1,l+1)=0$, and the opposite for $\mathbf{I}_{\mathcal{S}'}$. The original proof in (Cortes et al., 2008) showed that the stability β can be bounded as $\beta \leq \|\widetilde{\mathbf{f}}-\widetilde{\mathbf{f}}'\|$. In the following we show that the difference between the solutions $\widetilde{\mathbf{f}}$ and $\widetilde{\mathbf{f}}'$, and thus the stability of the algorithm, is strictly related to eigenvalues of the sparse graph $\mathcal{H}$. Let $\mathbf{A}=\mathbf{\Pi}_{\mathcal{F}}(l\eta\mathbf{L}_{\mathcal{H}}+\mathbf{I}_{\mathcal{S}})$ and $\mathbf{B}=\mathbf{\Pi}_{\mathcal{F}}(l\eta\mathbf{L}_{\mathcal{H}}+\mathbf{I}_{\mathcal{S}'})$, we remind that if the labels are centered, the solutions of Sparse-HFS can be conveniently written as $\widetilde{\mathbf{f}}=\mathbf{A}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}}$ and $\widetilde{\mathbf{f}}'=\mathbf{B}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}'}$. As a result, the difference between the solutions can be written as

$$\|\widetilde{\mathbf{f}}-\widetilde{\mathbf{f}}'\| = \|\mathbf{A}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}}-\mathbf{B}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}'}\| \leq \|\mathbf{A}^{-1}(\widetilde{\mathbf{y}}_{\mathcal{S}}-\widetilde{\mathbf{y}}_{\mathcal{S}'})\| + \|\mathbf{A}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}'}-\mathbf{B}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}'}\|. \tag{4.9}$$

Let us consider any vector $\mathbf{f}\in\mathcal{F}$, since the null space of a Laplacian $\mathbf{L}_{\mathcal{H}}$ is the one vector $\mathbf{1}$ and $\mathbf{\Pi}_{\mathcal{F}}=\mathbf{L}_{\mathcal{H}}\mathbf{L}_{\mathcal{H}}^{+}$, then $\mathbf{\Pi}_{\mathcal{F}}\mathbf{f}=\mathbf{f}$. Thus we have

$$\|\mathbf{\Pi}_{\mathcal{F}}(l\eta\mathbf{L}_{\mathcal{H}}+\mathbf{I}_{\mathcal{S}})\mathbf{f}\| \overset{(1)}{\geq} \|\mathbf{\Pi}_{\mathcal{F}}l\eta\mathbf{L}_{\mathcal{H}}\mathbf{f}\|-\|\mathbf{\Pi}_{\mathcal{F}}\mathbf{I}_{\mathcal{S}}\mathbf{f}\| \overset{(2)}{\geq} \|\mathbf{\Pi}_{\mathcal{F}}l\eta\mathbf{L}_{\mathcal{H}}\mathbf{f}\|-\|\mathbf{f}\| \overset{(3)}{\geq} (l\eta\lambda_2(\mathcal{H})-1)\|\mathbf{f}\| \tag{4.10}$$

where (1) follows from the triangle inequality and (2) follows from the fact that $\|\mathbf{\Pi}_{\mathcal{F}}\mathbf{I}_{\mathcal{S}}\mathbf{f}\|\leq\|\mathbf{f}\|$ since the largest eigenvalue of the project matrix $\mathbf{\Pi}_{\mathcal{F}}$ is one and the norm of $\mathbf{f}$ restricted on $\mathcal{S}$ is smaller than the norm of $\mathbf{f}$. Finally (3) follows from the fact that $\|\mathbf{\Pi}_{\mathcal{F}}\mathbf{L}_{\mathcal{H}}\mathbf{f}\|=\|\mathbf{L}_{\mathcal{H}}\mathbf{L}_{\mathcal{H}}^{+}\mathbf{L}_{\mathcal{H}}\mathbf{f}\|=\|\mathbf{L}_{\mathcal{H}}\mathbf{f}\|$ and since $\mathbf{f}$ is orthogonal to the null space of $\mathbf{L}_{\mathcal{H}}$ then $\|\mathbf{L}_{\mathcal{H}}\mathbf{f}\|\geq\lambda_2(\mathcal{H})\|\mathbf{f}\|$, where $\lambda_2(\mathcal{H})$ is the smallest non-zero eigenvalue of $\mathbf{L}_{\mathcal{H}}$. At this point we can exploit the spectral guarantees of the sparsified graph $\mathbf{L}_{\mathcal{H}}$ and from the definition of graph sparsifier, we have that $\lambda_2(\mathcal{H})\geq(1-\varepsilon)\lambda_2(\mathcal{G})$. As a result, we have an upper-bound on the spectral radius of the inverse operator $(\mathbf{\Pi}_{\mathcal{F}}(l\eta\mathbf{L}_{\mathcal{H}}+\mathbf{I}_{\mathcal{S}}))^{-1}$ and thus

$$\|\mathbf{A}^{-1}(\mathbf{y}_{\mathcal{S}}-\mathbf{y}_{\mathcal{S}'})\| \leq \frac{4M}{l\eta(1-\varepsilon)\lambda_2(\mathcal{G})-1},$$

where the first step follows from Eq. 4.10 since both $\widetilde{\mathbf{y}}_{\mathcal{S}}$ and $\widetilde{\mathbf{y}}_{\mathcal{S}'}$ are centered and thus $(\mathbf{y}_{\mathcal{S}}-\mathbf{y}_{\mathcal{S}'})\in\mathcal{F}$, and the second step is obtained by bounding $\|\widetilde{\mathbf{y}}_{\mathcal{S}}-\widetilde{\mathbf{y}}_{\mathcal{S}'}\|\leq\|\mathbf{y}_{\mathcal{S}}-\mathbf{y}_{\mathcal{S}'}\|+\|\overline{\mathbf{y}}_{\mathcal{S}}-\overline{\mathbf{y}}_{\mathcal{S}'}\|\leq 4M$. The second term in Eq. 4.9 can be bounded as

$$\begin{aligned}
&\|\mathbf{A}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}'}-\mathbf{B}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}'}\| = \|\mathbf{B}^{-1}(\mathbf{B}-\mathbf{A})\mathbf{A}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}'}\| \\
&= \|\mathbf{B}^{-1}\mathbf{\Pi}_{\mathcal{F}}(\mathbf{I}_{\mathcal{S}}-\mathbf{I}_{\mathcal{S}'})\mathbf{A}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}'}\| \leq \frac{1.5M\sqrt{l}}{(l\eta(1-\varepsilon)\lambda_2(\mathcal{G})-1)^2},
\end{aligned}$$

where we used $\|\widetilde{\mathbf{y}}_{\mathcal{S}'}\|\leq\|\mathbf{y}_{\mathcal{S}'}\|+\|\overline{\mathbf{y}}_{\mathcal{S}'}\|\leq 2M\sqrt{l}$, $\|\mathbf{\Pi}_{\mathcal{F}}(\mathbf{I}_{\mathcal{S}}-\mathbf{I}_{\mathcal{S}'})\|\leq\sqrt{2}<1.5$ and we applied Eq. 4.10 twice. Putting it all together we obtain the stated bound.

Step 3 (empirical error). The other element effected by the sparsification is the empirical error $\widehat{R}(\widetilde{\mathbf{f}})$. We first recall that $\mathbf{\Pi}_{\mathcal{F}}=\mathbf{L}_{\mathcal{G}}^{+}\mathbf{L}_{\mathcal{G}}=\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{L}_{\mathcal{G}}\mathbf{L}_{\mathcal{G}}^{-1/2}$ (and equivalently with $\mathcal{G}$ replaced by $\mathcal{H}$) and we introduce $\widetilde{\mathbf{\Pi}}_{\mathcal{F}}=\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{L}_{\mathcal{H}}\mathbf{L}_{\mathcal{G}}^{-1/2}$ Let $\widetilde{\mathbf{A}}=\mathbf{\Pi}_{\mathcal{F}}(l\eta\mathbf{L}_{\mathcal{H}}+\mathbf{I}_{\mathcal{S}})$, $\widehat{\mathbf{A}}=\mathbf{\Pi}_{\mathcal{F}}(l\eta\mathbf{L}_{\mathcal{G}}+\mathbf{I}_{\mathcal{S}})$, then rewrite the empirical error as

$$\begin{aligned}
\widehat{R}(\widetilde{\mathbf{f}}) &= \frac{1}{l}\|\mathbf{I}_{\mathcal{S}}\widetilde{\mathbf{f}}-\mathbf{I}_{\mathcal{S}}\widehat{\mathbf{f}}+\mathbf{I}_{\mathcal{S}}\widehat{\mathbf{f}}-\widetilde{\mathbf{y}}_{\mathcal{S}}\|^2 \leq \tfrac{1}{l}\|\mathbf{I}_{\mathcal{S}}\widehat{\mathbf{f}}-\widetilde{\mathbf{y}}_{\mathcal{S}}\|^2+\tfrac{1}{l}\|\mathbf{I}_{\mathcal{S}}\widetilde{\mathbf{f}}-\mathbf{I}_{\mathcal{S}}\widehat{\mathbf{f}}\|^2 \\
&\leq \widehat{R}(\widehat{\mathbf{f}})+\tfrac{1}{l}\|\mathbf{I}_{\mathcal{S}}(\widetilde{\mathbf{A}}^{-1}-\widehat{\mathbf{A}}^{-1})\widetilde{\mathbf{y}}_{\mathcal{S}}\|^2 \leq \widehat{R}(\widehat{\mathbf{f}})+\tfrac{1}{l}\|\widehat{\mathbf{A}}^{-1}(\widehat{\mathbf{A}}-\widetilde{\mathbf{A}})\widetilde{\mathbf{A}}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}}\|^2 \\
&= \widehat{R}(\widehat{\mathbf{f}})+\tfrac{l^2\eta^2}{l}\|\widehat{\mathbf{A}}^{-1}(\mathbf{\Pi}_{\mathcal{F}}(\mathbf{L}_{\mathcal{G}}-\mathbf{L}_{\mathcal{H}}))\widetilde{\mathbf{A}}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}}\|^2 \\
&= \widehat{R}(\widehat{\mathbf{f}})+l\eta^2\|\widehat{\mathbf{A}}^{-1}(\mathbf{\Pi}_{\mathcal{F}}(\mathbf{L}_{\mathcal{G}}-\mathbf{L}_{\mathcal{H}})\mathbf{\Pi}_{\mathcal{F}})\widetilde{\mathbf{A}}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}}\|^2
\end{aligned}$$

Where in the last passage we use $\mathbf{L}_{\mathcal{G}}\mathbf{\Pi}_{\mathcal{F}} = \mathbf{L}_{\mathcal{G}}$ and $\mathbf{L}_{\mathcal{H}}\mathbf{\Pi}_{\mathcal{F}} = \mathbf{L}_{\mathcal{H}}$. To bound the second term we derive

$$
\begin{aligned}
&\|\widehat{\mathbf{A}}^{-1}\mathbf{\Pi}_{\mathcal{F}}(\mathbf{L}_{\mathcal{G}} - \mathbf{L}_{\mathcal{H}})\mathbf{\Pi}_{\mathcal{F}}\widetilde{\mathbf{A}}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}}\|^2 \\
&\overset{(1)}{=} \|\widehat{\mathbf{A}}^{-1}\mathbf{L}_{\mathcal{G}}^{1/2}\mathbf{L}_{\mathcal{G}}^{-1/2}(\mathbf{L}_{\mathcal{G}} - \mathbf{L}_{\mathcal{H}})\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{L}_{\mathcal{G}}^{1/2}\widetilde{\mathbf{A}}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}}\|^2 \\
&\overset{(2)}{=} \|\widehat{\mathbf{A}}^{-1}\mathbf{L}_{\mathcal{G}}^{1/2}\mathbf{\Pi}_{\mathcal{F}}\mathbf{L}_{\mathcal{G}}^{-1/2}(\mathbf{L}_{\mathcal{G}} - \mathbf{L}_{\mathcal{H}})\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{\Pi}_{\mathcal{F}}\mathbf{L}_{\mathcal{G}}^{1/2}\widetilde{\mathbf{A}}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}}\|^2 \\
&\overset{(3)}{=} \|\widehat{\mathbf{A}}^{-1}\mathbf{L}_{\mathcal{G}}^{1/2}\mathbf{\Pi}_{\mathcal{F}}(\mathbf{\Pi}_{\mathcal{F}} - \widetilde{\mathbf{\Pi}}_{\mathcal{F}})\mathbf{\Pi}_{\mathcal{F}}\mathbf{L}_{\mathcal{G}}^{1/2}\widetilde{\mathbf{A}}^{-1}\widetilde{\mathbf{y}}_{\mathcal{S}}\|^2 \\
&\leq \|\widehat{\mathbf{A}}^{-1}\mathbf{L}_{\mathcal{G}}^{1/2}\mathbf{\Pi}_{\mathcal{F}}\|^2\|\mathbf{\Pi}_{\mathcal{F}} - \widetilde{\mathbf{\Pi}}_{\mathcal{F}}\|^2\|\mathbf{\Pi}_{\mathcal{F}}\mathbf{L}_{\mathcal{G}}^{1/2}\widetilde{\mathbf{A}}^{-1}\|^2\|\widetilde{\mathbf{y}}_{\mathcal{S}}\|^2 \\
&\overset{(4)}{\leq} lM^2\varepsilon^2\|\widehat{\mathbf{A}}^{-1}\mathbf{L}_{\mathcal{G}}^{1/2}\mathbf{\Pi}_{\mathcal{F}}\|^2\|\widetilde{\mathbf{A}}^{-1}\mathbf{L}_{\mathcal{G}}^{1/2}\mathbf{\Pi}_{\mathcal{F}}\|^2
\end{aligned}
$$

where in (1) and (2) we use the definition of $\mathbf{\Pi}_{\mathcal{F}}$, in (3) we use the definition of $\widetilde{\mathbf{\Pi}}_{\mathcal{F}}$, while in (4) we use the fact that Definition 4.2 implies that $(1-\varepsilon)\mathbf{\Pi}_{\mathcal{F}} \preceq \widetilde{\mathbf{\Pi}}_{\mathcal{F}} \preceq (1+\varepsilon)\mathbf{\Pi}_{\mathcal{F}}$ and thus the largest eigenvalue of $\mathbf{\Pi}_{\mathcal{F}} - \widetilde{\mathbf{\Pi}}_{\mathcal{F}}$ is ε^2. We need now to bound $\|\widetilde{\mathbf{A}}^{-1}\mathbf{L}_{\mathcal{G}}^{1/2}\mathbf{\Pi}_{\mathcal{F}}\|^2$. Similarly to Eq. 4.10 finding a lower bound on $\|\widetilde{\mathbf{A}}\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{\Pi}_{\mathcal{F}}\mathbf{x}\|$ for all $\mathbf{x}$ is equivalent to find a lower bound for all $\mathbf{f} \in \mathcal{F}$ to

$$
\begin{aligned}
&\|\mathbf{\Pi}_{\mathcal{F}}(l\eta\mathbf{L}_{\mathcal{H}} + \mathbf{I}_{\mathcal{S}})\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{f}\| \geq \|\mathbf{\Pi}_{\mathcal{F}}l\eta\mathbf{L}_{\mathcal{H}}\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{f}\| - \|\mathbf{\Pi}_{\mathcal{F}}\mathbf{I}_{\mathcal{S}}\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{f}\| \\
&\geq \|\mathbf{\Pi}_{\mathcal{F}}l\eta\mathbf{L}_{\mathcal{H}}\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{f}\| - \|\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{f}\| \geq \left(l\eta\frac{\lambda_2(\mathcal{H})}{\sqrt{\lambda_2(\mathcal{G})}} - \frac{1}{\sqrt{\lambda_2(\mathcal{G})}}\right)\|\mathbf{f}\| \\
&\geq \frac{1}{\sqrt{\lambda_2(\mathcal{G})}}\left(l\eta\lambda_2(\mathcal{H}) - 1\right)\|\mathbf{f}\| \geq \frac{1}{\sqrt{\lambda_2(\mathcal{G})}}\left(l\eta(1-\varepsilon)\lambda_2(\mathcal{G}) - 1\right)\|\mathbf{f}\|
\end{aligned}
$$

Similarly, we can show that

$$
\begin{aligned}
&\|\mathbf{\Pi}_{\mathcal{F}}(l\eta\mathbf{L}_{\mathcal{G}} + \mathbf{I}_{\mathcal{S}})\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{f}\| \geq \|\mathbf{\Pi}_{\mathcal{F}}l\eta\mathbf{L}_{\mathcal{G}}\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{f}\| - \|\mathbf{\Pi}_{\mathcal{F}}\mathbf{I}_{\mathcal{S}}\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{f}\| \\
&\geq \|\mathbf{\Pi}_{\mathcal{F}}l\eta\mathbf{L}_{\mathcal{G}}\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{f}\| - \|\mathbf{L}_{\mathcal{G}}^{-1/2}\mathbf{f}\| \geq \left(l\eta\frac{\lambda_2(\mathcal{G})}{\sqrt{\lambda_2(\mathcal{G})}} - \frac{1}{\sqrt{\lambda_2(\mathcal{G})}}\right)\|\mathbf{f}\| \\
&\geq \frac{1}{\sqrt{\lambda_2(\mathcal{G})}}\left(l\eta\lambda_2(\mathcal{G}) - 1\right)\|\mathbf{f}\| \geq \frac{1}{\sqrt{\lambda_2(\mathcal{G})}}\left(l\eta(1-\varepsilon)\lambda_2(\mathcal{G}) - 1\right)\|\mathbf{f}\|
\end{aligned}
$$

Taking this and putting all together shows

$$
\widehat{R}(\widetilde{\mathbf{f}}) \leq \widehat{R}(\widehat{\mathbf{f}}) + \frac{l^2\eta^2\lambda_2(\mathcal{G})^2\varepsilon^2M^2}{(l\eta(1-\varepsilon)\lambda_2(\mathcal{G}) - 1)^4}
$$

Combining the three steps above concludes the proof. ■

4.4.3 Experiments on spam detection

In this section we evaluate the empirical accuracy of Sparse-HFS compared to other baselines for large-scale SSL on both synthetic and real datasets.

Synthetic data. The objective of this first experiment is to show that the sparsification method is effective in reducing the number of edges in the graph and that preserving the full spectrum of $\mathcal{G}$ retains the accuracy of the exact HFS solution. We evaluate the algorithms on the $\mathbb{R}^2$ data distributed as in Fig. 4.1(a), which is designed so that a large number of

	Guarant.	Space	Preprocessing Time	Solving Time
SPARSE-HFS	✓	$N = \mathcal{O}(n\log(n))$	$\mathcal{O}(m\log^2(n))$	$\mathcal{O}(n\log^2(n))$
STABLE-HFS	✓	$\boldsymbol{\mathcal{O}(m)}$	$\mathcal{O}(m)$	$\mathcal{O}(mn)$
SIMPLE-HFS	○	$\boldsymbol{\mathcal{O}(m)}$	$\mathcal{O}(mq)$	$\mathcal{O}(q^4)$
EIGFUN	✗	$\mathcal{O}(nd + nq + b^2)$	$\mathcal{O}(qb^3 + db^3)$	$\mathcal{O}(q^3 + nq)$
SUBSAMPLING	✗	$\mathcal{O}(sk)$	$\mathcal{O}(m)$	$\mathcal{O}(s^2k + n)$

Table 4.1: **Guarantees and Computational complexities.** Bold text indicates unfeasible time or space complexity. ✗ Guarantees unavailable. ○ SIMPLE-HFS's guarantees require assumptions on the graph $\mathcal{G}$.

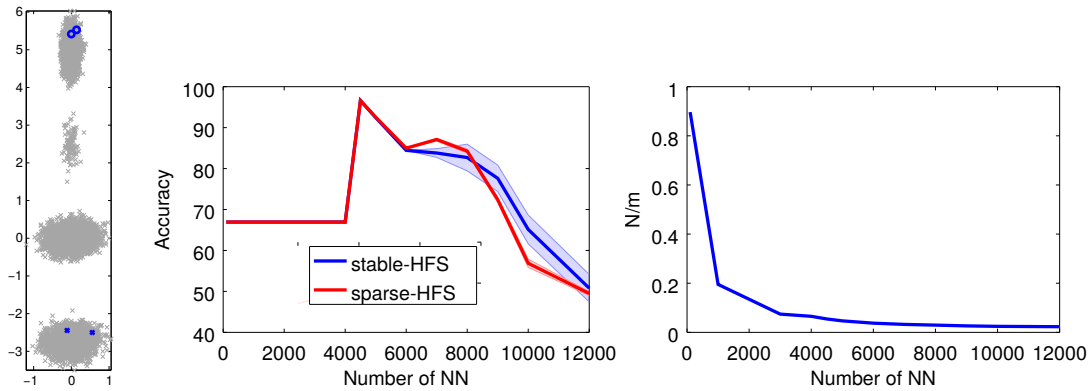


Figure 4.1: (a) The dataset of the synthetic experiment, (b) Accuracy of STABLE-HFS and SPARSE-HFS, (c) ratio of the number of edges $|\mathcal{H}|/|\mathcal{G}|$.

neighbours is needed to achieve a good accuracy. The dataset is composed of $n = 12100$ points, where the two upper clusters belong to one class and the two lower to the other. We build an unweighted, k-nn graph $\mathcal{G}$ for $k = 100, \ldots, 12000$. After constructing the graph, we randomly select two points from the uppermost and two from lowermost cluster as our labeled set $\mathcal{S}$. We then run SPARSE-HFS with $\varepsilon = 0.8$ to compute $\mathcal{H}$ and $\widetilde{\mathbf{f}}$, and run (exact) STABLE-HFS on $\mathcal{G}$ to compute $\widehat{\mathbf{f}}$, both with $\gamma = 1$. Fig. 4.1(b) reports the accuracy of the two algorithms. Both algorithms fail to recover a good solution until $k \approx 4000$. This is due to the fact that until a certain threshold, each cluster remains separated and the labels cannot propagate. Beyond this threshold, STABLE-HFS is very accurate, while, as k increases again, the graph becomes almost full, masking the actual structure of the data and thus loosing performance again. We notice that the accuracy of STABLE-HFS and SPARSE-HFS is never significantly different, and, quite importantly, they match around the value of $k = 4500$ that provides the best performance. This is in line with the theoretical analysis that shows that the contribution due to the sparsification error has the same order of magnitude as the other elements in the bound. Furthermore, in Fig. 4.1(c) we report the ratio of the number of edges in the sparsifier $\mathcal{H}$ and $\mathcal{G}$. This quantity is always smaller than one and it constantly decreases since the number of edges in $\mathcal{H}$ is constant, while the size $\mathcal{G}$ increases linearly with the number of neighbors (i.e., $|\mathcal{H}|/|\mathcal{G}| = \mathcal{O}(1/k)$). We notice that for the optimal k the sparsifier contains less than 10% of the edges of the original graph but it achieves almost the same accuracy.

Spam-filtering dataset. We now evaluate the performance of our algorithm on the TREC 2007 Public Spam Corpus[3], that contains $n = 75419$ raw emails labeled as either *SPAM* or *HAM*. The emails are provided as raw text and we applied standard NLP techniques

[3] http://plg.uwaterloo.ca/~gvcormac/treccorpus07/

to extract features vectors from it. In particular, we computed TF-IDF scores for each of the emails, with some additional cleaning in the form of a stop word list, simple stemming and dropping the 1% most common and most rare words. We ended up with $d = 68697$ features, each representing a word present in some of the emails. From these features we proceeded to build a graph $\mathcal{G}$ where given two emails $\mathbf{x}_i, \mathbf{x}_j$, the weight is computed as $a_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|/2\sigma^2)$, with $\sigma^2 = 3$. We consider the transductive setting, where the graph is fixed and known, but only a small random subset of $l = \{20, 100, 1000\}$ labels is revealed to the algorithm. As a performance measure, we consider the prediction accuracy over the whole dataset. We compare our method to several baselines. The most basic supervised baseline is 1-NN, which connects each node to the closest labeled node. The SubSampling algorithm selects uniformly s nodes out of n, computes the HFS solution on the induced subgraph of $\mathcal{G}$ and assigns to each node outside of the subset the same label as the closest node in the subset. SubSampling's complexity depends on the size s of the subgraph and the number k of neighbors retained when building the k-nn subgraph. The eigenfunction (EigFun) algorithm (Fergus et al., 2009) tries to sidestep the computational complexity of finding an HFS solution on $\mathcal{G}$, by directly approximating the distribution that created the graph. Starting directly from the samples, each of the d feature's density is separately approximated using histograms with b bins. From the histograms, q empirical eigenfunctions (vectors in $\mathbb{R}^n$) are extracted and used to compute the final solution. We did not include Stable-HFS and Simple-HFS in the comparison because their $\mathcal{O}(m)$ space complexity made them unfeasible for this dataset. In Fig. 4.2, we report the accuracy of each method against the time and space complexity, where each separate point corresponds to a different choice in metaparameters (e.g. k, q, s). For EigFun, we use the same $b = 50$ as in the original implementation, but we varied q from 10 to 2000. For SubSampling, $s = 15000$ and k varies from 100 to 10000. We run Sparse-HFS on $\mathcal{G}$ setting $\varepsilon = 0.9$, and changing the size m of the input graph by changing the number of neighbours k from 1000 to 7500. Since the actual running time and memory occupation are highly dependent on the implementation (e.g., EigFun is implemented in Matlab, while Sparse-HFS is Matlab/C), the complexities are computed using their theoretical form (e.g., $\mathcal{O}(m \log(n))$ for Sparse-HFS) with the values actually used in the experiment (e.g., $m = nk$ for a k-nn graph). All the complexities are reported in Table 4.1. The only exception is the number of edges in the sparsifier $|\mathcal{H}|$ used in the space complexity of Sparse-HFS. Since this is a random quantity that holds only w.h.p. and that is independent from implementation details, we measured it empirically and used it for the complexities. For all methods we notice that the performance increases as the space complexity gets larger, until a peak is reached, while additional space induces the algorithms to overfit and reduces accuracy. For EigFun this means that a large number of eigenfunctions is necessary to accurately model the high dimensional distribution. And as theory predicts, SubSampling's uniform sampling is not efficient to approximate the graph spectra, and a large subset of the nodes is required for good performance. Sparse-HFS's accuracy also increases as the input graph gets richer, but unlike the other methods the space complexity does not change much. This is because the sparsifier is oblivious to the structure of the graph, and even if Sparse-HFS reaches its optimum performance for $k = 3000$, the sparsifier contains roughly the same number of edges present as $k = 1000$, and only 5% of the edges present in the input graph. Although preliminary, this experiment shows that the theoretical properties of Sparse-HFS translate into an effective practical algorithm which is competitive with state-of-the-art methods for large-scale SSL.

4.4.4 Recap and open questions: dynamic sparsifiers

We introduced Sparse-HFS, an algorithm that combines sparsification methods and efficient solvers for SDD systems to find approximate HFS solutions using only $\mathcal{O}(n \log^2(n))$ space instead of $\mathcal{O}(m)$. Furthermore, we show that the $\mathcal{O}(m \log(n))$ time complexity of the methods is only a polylog term away from the smallest possible complexity $\Omega(m)$. Finally, we provide a bound on the generalization error that shows that the sparsification does not affect the asymptotic convergence rate of HFS. As such, the accuracy parameter ε can be freely chosen to meet the desired trade-off between accuracy and space complexity.

An interesting feature of Sparse-HFS is that it could be easily employed in online learning problems where edges arrive in a stream and intermediate solutions have to be computed over time. Since Sparse-HFS has a $\mathcal{O}(\log(n))$ amortized time per edge, it could compute intermediate solutions every $\overline{q}$ edges without compromising its overall time complexity.

The fully dynamic setting, where edges can be both inserted and removed, is an important extension where our approach could be further investigated, especially because it has been observed in several domains that graphs become denser as they evolve over time (Leskovec et al., 2005). While sparsifiers have been developed for this setting (see e.g., (Kapralov et al., 2014)), current solutions would require $\mathcal{O}(n^2 \operatorname{polylog}(n))$ time to construct the sparsifier, thus making it unfeasible to repeat this computation many times over the stream. Extending sparsification techniques to the fully dynamic setting in a computationally efficient manner is currently an open problem.

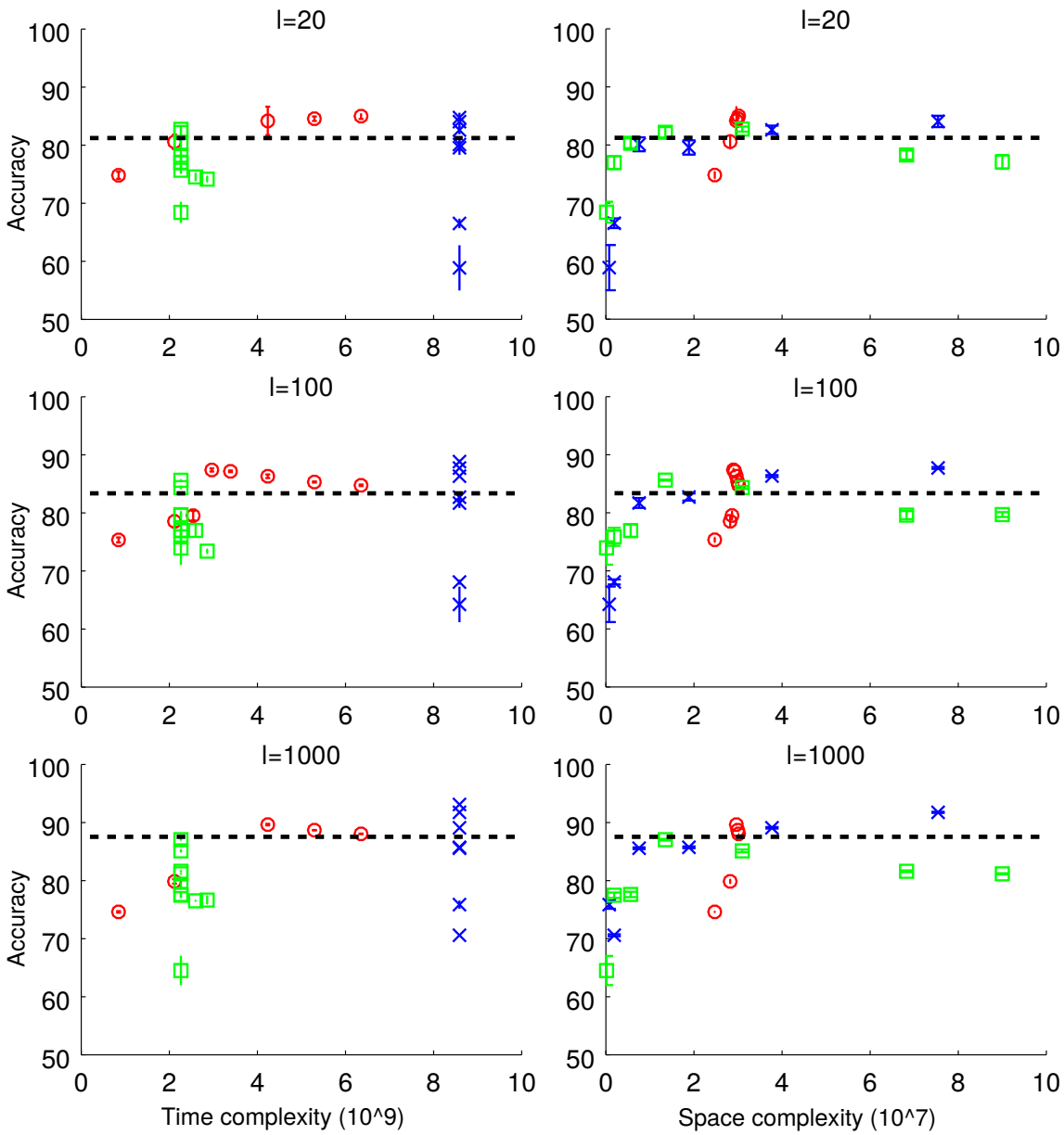
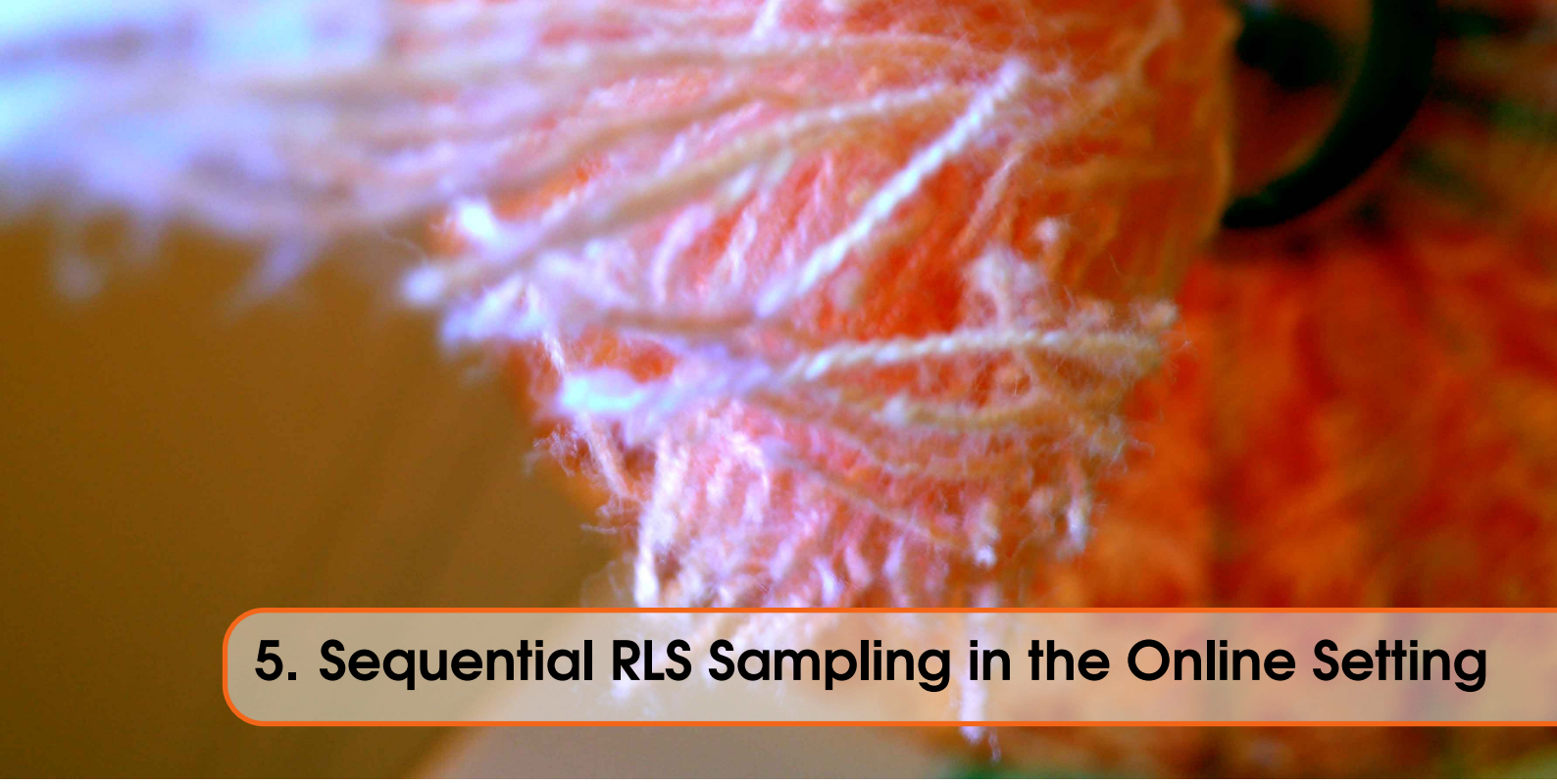


Figure 4.2: Accuracy vs complexity on the TREC 2007 SPAM Corpus for different number of labels. *Legend:* ○ SPARSE-HFS, × EIGFUN, □ SUBSAMPLING, - - - 1-NN



5. Sequential RLS Sampling in the Online Setting

Online learning (OL) represents a family of efficient and scalable learning algorithms for building a predictive model incrementally from a sequence of T data points.
A popular online learning approach, introduced in (Zinkevich, 2003), is to learn a linear predictor using *gradient descent* (GD) in the input space $\mathbb{R}^d$. Since we can explicitly store and update the d weights of the linear predictor, the total runtime of this algorithm is $\mathcal{O}(Td)$, allowing it to scale to large problems. Unfortunately, it is sometimes the case that no good predictor can be constructed starting from only the linear combination of the input features.
For this reason, *online kernel learning* (OKL) (Kivinen et al., 2004a) first maps the points into a high-dimensional *reproducing kernel Hilbert space* (RKHS) using a non-linear feature map φ, and then runs GD on the projected points, which is often referred to as functional GD (FGD) (Kivinen et al., 2004b). With the kernel approach, each gradient step does not update a fixed set of weights, but instead introduces the feature-mapped point in the predictor as a *support vector* (SV). The resulting kernel-based predictor is flexible and data adaptive, but the number of parameters, and therefore the per-step space and time cost, now scales with $\mathcal{O}(t)$, the number of SVs included after t steps of GD. This *curse of kernelization*, results in an $\mathcal{O}(T^2)$ total runtime, and prevents standard OKL methods from scaling to large problems.

When judging an online learning algorithm, we are not only interested in its computational complexity, but also in its prediction accuracy. Given a function space containing functions with very small prediction loss, the objective of an online learning algorithm is to approach over time the performance of the best predictor in the space and thus minimize the regret, that is the difference in cumulative loss between the OL algorithm and the best predictor *in hindsight*.
Over time, many methods have been proposed to trade-off minimizing the regret of the online learning algorithm, and the computational cost required to compute the predictions.

Gradient descent. For OL, Zinkevich, (2003) showed that simple *gradient descent* (GD),

combined with a smart choice for the stepsize η_t of the gradient updates, achieves a $\mathcal{O}(\sqrt{dT})$ regret with a $\mathcal{O}(d)$ space and time cost per iteration. When the only assumption on the losses is simple convexity, this upper bound matches a corresponding lower bound (Luo et al., 2016), thus making first-order methods (e.g., GD) essentially unimprovable in a minimax sense.
However, if we know that the losses are *strongly convex*, setting a more aggressive step-size achieves a smaller $\mathcal{O}(\log(T))$ regret (Hazan et al., 2006; Zhu and Xu, 2015). Unfortunately, most common losses, such as the *squared loss*, are not strongly convex when evaluated for a single point $\mathbf{x}_t$.
Nonetheless, they possess a certain *directional curvature* and Hazan et al., (2006) show that *online Newton step* (ONS), an adaptive method that exploits second-order (second derivative) information on the losses, can achieve a *logarithmic* regret $\mathcal{O}(d \log T)$ without strong convexity along all directions.
The downside of this adaptive method is the larger $\mathcal{O}(d^2)$ space and per-step time complexity, since second-order updates require to construct, store, and invert $\mathbf{H}_t$, a preconditioner matrix related to the Hessian of the losses used to *correct* the first-order updates.

Kernel gradient descent. OKL is fundamentally harder than OL, due to 1) the fact that an infinite parametrization makes regret bounds scaling with the dimension d meaningless and 2) the size of the model, and therefore time and space complexities, scales with t itself, making these methods even less performant than OL algorithms.
Kernel extensions of first-order OL algorithms have been proposed for OKL, such as functional GD (e.g., NORMA, (Kivinen et al., 2004a)) which achieves a $\mathcal{O}(\sqrt{T})$ regret with a $\mathcal{O}(t)$ space and time cost per iteration.
For second-order methods, the Second-Order Perceptron (Cesa-Bianchi et al., 2005) or NAROW (Orabona and Crammer, 2010) for generic curved losses and Recursive Kernel Least Squares (Zhdanov and Kalnishkan, 2010) or Kernel AAR (Gammerman et al., 2004) for the specific case of ℓ_2 losses provide bounds that scale with the log-determinant of the kernel-matrix.
As we showed in Lemma 3.9 this quantity can be upper bounded using the effective dimension $d_{\text{eff}}^T(\gamma)$ of the of the points $\mathbf{x}_t$, and scales as $\mathcal{O}(d_{\text{eff}}^T(\gamma)\log(T))$, playing a similar role as the $\mathcal{O}(d \log T)$ bound from OL.

Approximate GD. To trade off between computational complexity $\big($smaller than $\mathcal{O}(d^2)\big)$ and improved regret (close to $\mathcal{O}(d \log T)$), several OL methods try approximate second-order updates, replacing $\mathbf{H}_t$ with an approximate $\widetilde{\mathbf{H}}_t$ that can be efficiently stored and inverted. AdaGrad (Duchi et al., 2011) and ADAM (Kingma and Ba, 2015) reweight the gradient updates on a per-coordinate basis using a diagonal $\widetilde{\mathbf{H}}_t$, but these methods ultimately only improve the regret dependency on d and leave the $\sqrt{T}$ component unchanged.
Sketched-ONS, by Luo et al., (2016), uses matrix sketching to approximate $\mathbf{H}_t$ with a r-rank sketch $\widetilde{\mathbf{H}}_t$, that can be efficiently stored and updated in $\mathcal{O}(dr^2)$ time and space, close to the $\mathcal{O}(d)$ complexity of diagonal approximations. More importantly, Sketched-ONS achieves a much smaller regret compared to diagonal approximations: When the true $\mathbf{H}_t$ is of low-rank r, it recovers a $\mathcal{O}(r \log T)$ regret bound logarithmic in T.
Unfortunately, when the true $\mathbf{H}_t$ is not low-rank, a new term appears in the bound, due to the sketch approximation, that scales with the spectrum of $\mathbf{H}_t$ and in some cases can grow much larger than $\mathcal{O}(\log T)$.

Approximate kernel GD. Many approximation methods have been proposed to make first-order OKL methods scale to large datasets. They all try to reduce the $\mathcal{O}(t)$ per-step complexity to a constant independent from t. Approximate methods usually take one of

two approaches:

- budgeted methods perform *approximate gradient* updates in the *true RKHS*, where instead of updating the SVs' weights in our predictor using the true gradient direction we use an update that keeps most weight at zero,
- functional approximation methods perform *exact gradient* updates in an *approximate RKHS*, where the points are embedded in a finite-dimensional space and the curse of kernelization does not apply, allowing us to follow the gradient without increasing the number of SV.

Overall, the goal is to never exceed a budget of SVs in order to maintain a fixed per-step cost.
Among budgeted methods, how to modify the gradient update to maintain zero weights (i.e., weight degradation (Wang et al., 2012)) can be approached in many different ways, such as removal (Cavallanti et al., 2007; Dekel et al., 2008) or more expensive projection (Orabona et al., 2008) and merging strategies. Nonetheless, we will see that as long as the size of the budget is fixed, the adversary can exploit this to increase the regret of the algorithm, and oblivious inclusion strategies such as uniform sampling (He and Kwok, 2014) fail.
As for functional approximation methods, the most common approach is to replace the exact feature-map φ with a finite-rank approximate feature map $\widetilde{\varphi}$ which allows to explicitly represent the mapped points in an Euclidean space, and run OL on this embedding (Lu et al., 2016; Yang et al., 2012).
When the embedding is oblivious to data, the method is known as random-feature expansion(Le et al., 2013), while a common data-dependent embedding mapping is the Nyström method (Williams and Seeger, 2001). Again, if the embedding is fixed or with a limit in size, the adversary can exploit it. In addition, analyzing a change in embedding during the gradient descent is an open problem, since the underlying RKHS changes with it.
Unfortunately, to guarantee $\mathcal{O}(\sqrt{T})$ regret using less than $\mathcal{O}(t)$ space and time per round w.h.p., all of these methods require additional assumptions, such as points $\mathbf{x}_t$ coming from a distribution or strong convexity on the losses. Moreover, as approximate first-order methods, they can at most hope to match the $\mathcal{O}(\sqrt{T})$ regret of exact GD, and among second-order kernel methods, no approximation scheme has been proposed that can provably maintain the same $\mathcal{O}(\log T)$ regret as exact second-order GD.
In addition, approximating $\mathbf{H}_t$ is harder for OKL. Since we cannot directly access the matrix representation of $\mathbf{H}_t$ in the feature-space, diagonal approximation is impossible, and low-rank sketching harder.

Contributions: In this chapter, we provide two second-order OKL algorithms: Sketched-KONS takes the approach of approximate gradient updates, and PROS-N-KONS takes the approach of approximating the feature map. Both are based on an exact second-order method OKL algorithm called Kernelized-ONS (KONS), which is a generalization of Hazan et al., (2006)'s online Newton step (ONS) from OL to OKL.

With Sketched-KONS (Calandriello et al., 2017c) our goal is to show that it is possible to approximate the second-order updates of an exact second-order OKL method (i.e., approximate the preconditioner $\mathbf{H}_t$) using only a small budget of SV, and without losing the logarithmic regret rate. In particular, Sketched-KONS chooses which SV to include in the approximate preconditioner $\widetilde{\mathbf{H}}_t$ based on their RLS, using KORS to estimate them. The result is the first approximate second-order OKL algorithm that w.h.p. achieves logarithmic regret, and offers a favorable regret-performance trade-off. For a given factor

$\beta \leq 1$, we can increase the regret by a linear $1/\beta$ factor to $\mathcal{O}(d^t_{\text{eff}}(\gamma)\log(T)/\beta)$, while obtaining a quadratic β^2 improvement in runtime compared to exact methods, and achieving a $\mathcal{O}(t^2\beta^2)$ space and time cost per iteration.
Unfortunately, to construct a truly scalable OKL algorithm, we need to achieve a near-linear $\widetilde{\mathcal{O}}(Td^T_{\text{eff}}(\gamma))$ runtime. In Section 5.2.1 we show a counterexample where any SV inclusion rule will fail to preserve both logarithmic regret and $\mathcal{O}(d^T_{\text{eff}}(\gamma))$ runtime, highlighting the limits of our approximate gradient update rule.

To resolve the failure mode proposed by the counterexample, we introduce PROS-N-KONS (Calandriello et al., 2017b), an improved second-order OKL method that for the first time guarantees both logarithmic regret and avoids the curse of kernelization, taking only a near-constant $\widetilde{\mathcal{O}}(d^T_{\text{eff}}(\gamma)^2)$ per-step time and space cost.
To achieve this, instead of replacing the update rule for the preconditioner $\mathbf{H}_t$, we replace the exact feature map φ with an approximate $\widetilde{\varphi}$ and $\widetilde{\mathcal{H}}$ constructed using a Nyström dictionary approximation, where the dictionary is computed using KORS. While previous methods (Le et al., 2013; Lu et al., 2016) used *fixed* embeddings, we adaptively *update* our dictionary and embedding $\widetilde{\mathcal{H}}$, so that it cannot be indefinitely exploited by the adversary. For a dictionary of size j, this non-linearly embeds the points in $\mathbb{R}^j$, where we can efficiently perform exact second-order updates in constant $\mathcal{O}(j^2)$ per-step time, and achieve the desired $\mathcal{O}(\log(T))$ regret, with respect to the best competitor in $\widetilde{\mathcal{H}}$.
Using KORS' (ε, γ)-accuracy guarantees to show that $\mathcal{H}$ and $\widetilde{\mathcal{H}}$ are close, we prove that we never get stuck performing GD in an embedded space that is too distant from the true $\mathcal{H}$, but at the same time the size of the embedding j never grows larger than the effective dimension of the problem. Combined with an adaptive restart strategy, this allows us to bound our regret w.r.t. to the best solution in $\mathcal{H}$.

5.1 Online kernel learning and the kernelized online Newton step

We focus on the online kernel learning setting, where an adversary chooses an arbitrary sequence of points $\{\mathbf{x}_t\}_{t=1}^T$ and convex differentiable losses $\{\ell_t\}_{t=1}^T$. The learning protocol is the following. At each round $t \in [T]$

(1) the adversary reveals the new point $\mathbf{x}_t$,
(2) the learner chooses a function $f_{\mathbf{w}_t}$ and predicts $f_{\mathbf{w}_t}(\mathbf{x}_t) = \varphi(\mathbf{x}_t)^\intercal \mathbf{w}_t$,
(3) the adversary reveals the loss ℓ_t,
(4) the learner suffers $\ell_t(\varphi(\mathbf{x}_t)^\intercal \mathbf{w}_t)$ and observes the associated gradient $\mathbf{g}_t$.

The adversary can choose any function $f \in \mathcal{H}$ that can be represented as a (potentially infinite) set of weights $\mathbf{w}$ such that $f_{\mathbf{w}}(\mathbf{x}) = \varphi(\mathbf{x})^\intercal \mathbf{w}$ and the norm $\|\mathbf{w}\|$ is bounded.
After t timesteps, we indicate with $\mathcal{D}_t = \{\mathbf{x}_i\}_{i=1}^t$, the dataset containing the points observed so far. In the rest of the thesis we consider the problem of OKL where $\mathcal{H}$ is arbitrary and potentially infinite dimensional (non-parametric setting). We refer to the special case $\mathcal{H} = \mathbb{R}^d$ and $\phi_t = \mathbf{x}_t$ as OL (parametric setting).

We are interested in bounding the cumulative regret between the learner and a fixed function $\mathbf{w}$ defined as $R_T(\mathbf{w}) = \sum_{t=1}^T \ell(\phi_t \mathbf{w}_t) - \ell(\phi_t \mathbf{w})$. Since $\mathcal{H}$ is potentially a very large space, we also need to restrict the class of comparators $\mathbf{w}$.
As Luo et al., (2016), we define the feasible set as $\mathcal{S}_t = \{\mathbf{w} : |\phi_t^\intercal \mathbf{w}| \leq C\}$ and $\mathcal{S} = \cap_{t=1}^T \mathcal{S}_t$. This comparison class contains all functions $f_{\mathbf{w}}$ whose output is contained (clipped) in the interval $[-C, C]$ on all points $\phi_1, \ldots, \phi_T$. Unlike the often used constraint on $\|\mathbf{w}\|$ (Hazan et al., 2006; Zhu and Xu, 2015), comparing against clipped functions (Gammerman et al.,

Algorithm 12 Kernelized online Newton step (KONS)

Input: Feasible parameter C, stepsizes η_t, regulariz. α

1: Initialize $\mathbf{w}_0 = \mathbf{0}, \mathbf{g}_0 = \mathbf{0}, b_0 = 0$, $\mathbf{A}_0 = \alpha \mathbf{\Pi}_T$
2: **for** $t = \{1, \dots, T\}$ **do**
3: receive $\mathbf{x}_t$
4: compute b_s as in Lem. 5.2
5: compute $\mathbf{u}_t = \mathbf{A}_{t-1}^{-1}(\sum_{s=0}^{t-1} b_s \mathbf{g}_s)$
6: compute $\overline{y}_t = \varphi(\mathbf{x}_t)^\mathsf{T} \mathbf{u}_t$
7: predict $\widehat{y}_t = \varphi(\mathbf{x}_t)^\mathsf{T} \mathbf{w}_t = \overline{y}_t - h(\overline{y}_t)$
8: observe $\mathbf{g}_t$, update $\mathbf{A}_t = \mathbf{A}_{t-1} + \eta_t \mathbf{g}_t \mathbf{g}_t^\mathsf{T}$
9: **end for**

2004; Luo et al., 2016; Zhdanov and Kalnishkan, 2010) has a clear interpretation even when passing from $\mathbb{R}^d$ to $\mathcal{H}$. Moreover, $\mathcal{S}$ is invariant to linear transformations of $\mathcal{H}$ and suitable for practical problems where it is often easier to choose a reasonable interval for the predictions $\widehat{y}_t$ rather than a bound on the norm of a (possibly non-interpretable and infinite) parametrization $\mathbf{w}$.

We make the following assumptions on the losses

Assumption 1 — Scalar Lipschitz. The loss functions ℓ_t satisfy $|\ell_t'(z)|$ whenever $|z| \leq C$.

Assumption 2 — Curvature. There exists $\sigma_t \geq \sigma > 0$ such that for all $\mathbf{u}, \mathbf{w} \in \mathcal{S}$ and for all $t \in [T]$,

$$\ell_t(\phi_t^\mathsf{T} \mathbf{w}) := l_t(\mathbf{w}) \geq l_t(\mathbf{u}) + \nabla l_t(\mathbf{u})^\mathsf{T}(\mathbf{w} - \mathbf{u}) + \frac{\sigma_t}{2} \left(\nabla l_t(\mathbf{u})^\mathsf{T}(\mathbf{w} - \mathbf{u})\right)^2.$$

This assumption is *weaker than strong convexity* as it only requires the losses to be strongly convex in the direction of the gradient. It is satisfied by squared loss, *squared* hinge loss, logistic loss, and in general all exp-concave losses (Hazan et al., 2006). Under this weaker requirement, exact second-order learning methods (Calandriello et al., 2017c; Hazan et al., 2006), can achieve logarithmic regret at the cost of a higher computational complexity w.r.t. first-order methods.

5.1.1 Kernelized online newton step

We now present an simple second-order OKL method, that will serve as a basis for Sketched-KONS and PROS-N-KONS. In particular, we generalize to the OKL setting the online Newton step algorithm, originally introduced by Hazan et al., (2006) for OL. ONS is a projected gradient descent that uses the following update rules

$$\mathbf{u}_t = \mathbf{w}_{t-1} - \mathbf{A}_{t-1}^{-1} \mathbf{g}_{t-1}, \qquad \mathbf{w}_t = \Pi_{\mathcal{S}_t}^{\mathbf{A}_{t-1}}(\mathbf{u}_t), \tag{5.1}$$

where $\Pi_{\mathcal{S}_t}^{\mathbf{A}_{t-1}}(u_t) = \arg\min_{\mathbf{w} \in \mathcal{S}_t} \|\mathbf{u}_t - \mathbf{w}\|_{\mathbf{A}_{t-1}}$ is an oblique projection on a set $\mathcal{S}_t$ with matrix $\mathbf{A}_{t-1}$. If $\mathcal{S}_t$ is the set of vectors with bounded prediction in $[-C, C]$, then as shown by Luo et al., (2016) the projection reduces to

$$\mathbf{w}_t = \Pi_{\mathcal{S}_t}^{\mathbf{A}_{t-1}}(\mathbf{u}_t) = \mathbf{u}_t - \frac{h(\phi_t^\mathsf{T} \mathbf{u}_t)}{\phi_t^\mathsf{T} \mathbf{A}_{t-1}^{-1} \phi_t} \mathbf{A}_{t-1}^{-1} \phi_t,$$

where $h(z) = \operatorname{sign}(z) \max\{|z| - C,\ 0\}$ computes how much z is above or below the interval $[-C, C]$. When $\mathbf{A}_t = \mathbf{\Pi}_T / \eta_t$, ONS is equivalent to vanilla projected gradient descent, which

in OL achieves $\mathcal{O}(\sqrt{dT})$ regret (Zinkevich, 2003). In the same setting, Hazan et al., (2006) shows that choosing $\mathbf{A}_t = \sum_{s=1}^{t} \eta_s \mathbf{g}_s \mathbf{g}_s^\mathsf{T} + \alpha \mathbf{\Pi}_T$ makes ONS an efficient reformulation of *follow the approximate leader* (FTAL). While traditional follow-the-leader algorithms play the set of weights $\mathbf{w}_t = \arg\min_{\mathbf{w}\in\mathcal{S}_t} \sum_{s=1}^{t-1} l_t(\mathbf{w})$, FTAL replaces the loss l_t with a convex approximation using Asm. 2, and plays the minimizer of the surrogate function. As a result, under Asm. 1-2 and when $\sigma_t \geq \sigma > 0$, FTAL achieves a logarithmic $\mathcal{O}(d \log T)$ regret. FTAL's solution path can be computed in $\mathcal{O}(d^2)$ time using ONS updates, and further speedups were proposed by Luo et al., (2016) using matrix sketching.

Unfortunately, in OKL, vectors ϕ_t and weights $\mathbf{w}_t$ cannot be explicitly represented, and most of the quantities used in vanilla ONS (Eq. 5.1) cannot be directly computed. Instead, we derive a closed form alternative (Alg. 12) that can be computed in practice. Using a rescaled variant of our feature vectors ϕ_t, $\overline{\phi}_t = \dot{g}_t \sqrt{\eta_t} \phi_t = \sqrt{\eta_t} \mathbf{g}_t$ and $\overline{\Phi}_t = [\overline{\phi}_1, \ldots, \overline{\phi}_t]$, we can rewrite $\mathbf{A}_t = \overline{\Phi}_t \overline{\Phi}_t^\mathsf{T} + \alpha \mathbf{\Pi}_T$ and $\overline{\Phi}_t^\mathsf{T} \overline{\Phi}_t = \overline{\mathbf{K}}_t$, where the empirical kernel matrix $\overline{\mathbf{K}}_t$ is computed using the rescaled kernel $\overline{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j) = \dot{g}_i \sqrt{\eta_i} \dot{g}_j \sqrt{\eta_j} \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$ instead of the original $\mathcal{K}$, or equivalently $\overline{\mathbf{K}}_t = \mathbf{D}_t \mathbf{K}_t \mathbf{D}_t$ with $\mathbf{D}_t = \mathrm{Diag}(\{\dot{g}_i \sqrt{\eta_i}\}_{i=1}^t)$ the rescaling diagonal matrix. We begin by noting that

$$\widehat{y}_t = \phi_t^\mathsf{T} \mathbf{w}_t = \phi_t^\mathsf{T} \left(\mathbf{u}_t - \frac{h(\phi_t^\mathsf{T} \mathbf{u}_t)}{\phi_t^\mathsf{T} \mathbf{A}_{t-1}^{-1} \phi_t} \mathbf{A}_{t-1}^{-1} \phi_t \right) = \phi_t^\mathsf{T} \mathbf{u}_t - h(\phi_t^\mathsf{T} \mathbf{u}_t) \frac{\phi_t^\mathsf{T} \mathbf{A}_{t-1}^{-1} \phi_t}{\phi_t^\mathsf{T} \mathbf{A}_{t-1}^{-1} \phi_t} = \overline{y}_t - h(\overline{y}_t).$$

As a consequence, if we can find a way to compute $\overline{y}_t$, then we can obtain $\widehat{y}_t$ without explicitly computing $\mathbf{w}_t$. Before that, we first derive a non-recursive formulation of $\mathbf{u}_t$.

Lemma 5.1 — (Calandriello et al., 2017c). In Alg. 12 we introduce

$$b_i = [\mathbf{b}_t]_i = \dot{g}_i \sqrt{\eta_i} \left(\widehat{y}_i - \frac{h(\overline{y}_i)}{\overline{\phi}_i^\mathsf{T} \mathbf{A}_{i-1}^{-1} \overline{\phi}_i} \right) - \frac{1}{\sqrt{\eta_i}}$$

and compute $\mathbf{u}_t$ as

$$\mathbf{u}_t = \mathbf{A}_{t-1}^{-1} \overline{\Phi}_{t-1} \mathbf{b}_{t-1}.$$

Then, $\mathbf{u}_t$ is equal to the same quantity in Eq. 5.1 and the sequence of predictions $\widehat{y}_t$ is the same in both algorithms.

While the definition of $\mathbf{b}_t$ and $\mathbf{u}_t$ still requires performing operations in the (possibly infinitely dimensional) feature space, in the following we show that $\mathbf{b}_t$ and the prediction $\overline{y}_t$ can be conveniently computed using only inner products.

Lemma 5.2 — (Calandriello et al., 2017c). All the components $b_i = [\mathbf{b}_t]_i$ of the vector introduced in Lem. 5.1 can be computed as

$$\dot{g}_i \sqrt{\eta_i} \left(\widehat{y}_i - \frac{\alpha h(\overline{y}_i)}{k_{i,i} - \overline{\mathbf{k}}_{[i-1],i}^\mathsf{T} (\overline{\mathbf{K}}_{i-1} + \alpha \mathbf{I}_{i-1})^{-1} \overline{\mathbf{k}}_{[i-1],i}} - \frac{1}{\eta_i} \right).$$

Then, we can compute

$$\overline{y}_t = \frac{1}{\alpha} \mathbf{k}_{[t-1],t}^\mathsf{T} \mathbf{D}_{t-1} (\mathbf{b}_{t-1} - (\overline{\mathbf{K}}_{t-1} + \alpha \mathbf{I}_{i-1})^{-1} \overline{\mathbf{K}}_{t-1} \mathbf{b}_{t-1}).$$

Since Alg. 12 is equivalent to ONS (Eq. 5.1), existing regret bounds for ONS directly applies to its kernelized version.

Proposition 5.3 — (Luo et al., 2016). For any sequence of losses ℓ_t satisfying Asm. 1-2, the regret R_T of Alg. 12 is bounded by $R_T \leq \alpha\|\mathbf{w}^*\|^2 + R_G + R_D$ with

$$R_G := \sum_{t=1}^{T} \mathbf{g}_t^\intercal \mathbf{A}_t^{-1}\mathbf{g}_t = \sum_{t=1}^{T} \overline{\phi}_t^\intercal (\overline{\Phi}_t\overline{\Phi}_t^\intercal + \alpha\mathbf{\Pi}_T)^{-1}\overline{\phi}_t/\eta_t$$

$$R_D := \sum_{t=1}^{T} (\mathbf{w}_t - \mathbf{w}^*)^\intercal (\mathbf{A}_t - \mathbf{A}_{t-1} - \sigma_t \mathbf{g}_t\mathbf{g}_t^\intercal)(\mathbf{w}_t - \mathbf{w}^*)$$

$$= \sum_{t=1}^{T} (\eta_t - \sigma_t)\dot{g}_t^2 (\phi_t^\intercal(\mathbf{w}_t - \mathbf{w}^*))^2.$$

In the d-dimensional OL, choosing a decreasing step-size $\eta_t = \sqrt{d/(C^2L^2t)}$ allows ONS to achieve a $\mathcal{O}(CL\sqrt{dT})$ regret for the cases where $\sigma_t = 0$. When $\sigma_t \geq \sigma > 0$ (e.g., when the functions are exp-concave) we can set $\eta_t = \sigma_t$ and improve the regret to $\mathcal{O}(d\log(T))$. Unfortunately, these quantities hold little meaning for OKL, where d is not guaranteed to be finite-dimensional, and a $\mathcal{O}(\sqrt{d})$ regret can be very large or even infinite. Instead, we expect the regret of KONS to depend on quantities that are more strictly related to the kernel $\overline{\mathbf{K}}_t$ and its complexity.

By focusing the first regret term in the decomposition of Prop. 5.3, we notice that

$$R_G = \sum_{t=1}^{T} \overline{\phi}_t^\intercal (\overline{\Phi}_t\overline{\Phi}_t^\intercal + \alpha\mathbf{\Pi}_T)^{-1}\overline{\phi}_t/\eta_t = \sum_{t=1}^{T} \overline{\tau}_{t,t}/\eta_t,$$

where $\overline{\tau}_{t,t}$ it the RLS of sample t computed using the rescaled $\overline{\mathcal{K}}$ rather than $\mathcal{K}$ (i.e., using $\overline{\phi}_t$ and w.r.t. $\overline{\Phi}_t$). This reveals a deep connection between the regret of KONS and the cumulative sum of the RLS. In other words, the RLS capture how much the adversary can increase the regret by picking orthogonal directions that have not been seen before.
While in OL, this can happen at most d times (hence the dependency on d in the final regret, which is mitigated by a suitable choice of η_t), in OKL, R_G can grow linearly with time, since large $\mathcal{H}$ can have infinite near-orthogonal directions. Nonetheless, the actual growth rate is now directly related to the complexity of the sequence of points chosen by the adversary and the kernel function $\mathcal{K}$. While the sum $\sum_i \overline{\tau}_{i,i}$ scales with the online effective dimension of the points, we can use Lemma 3.9 to bound it as $d_{\text{eff}}^t(\alpha)\log(t)$. This inequality shows again that in general the complexity of online learning is only a factor $\log(T)$ (in the worst case) away from the complexity of batch learning. Moreover, notice that the log-determinant of the covariance matrix $\mathbf{A}_T$ appears in many other online learning problems (Cesa-Bianchi et al., 2005; Srinivas et al., 2010).

At this point, we can generalize the regret bounds of ONS for OL to KONS and OKL.

Theorem 5.4 — (Calandriello et al., 2017c). For any sequence of losses ℓ_t satisfying Asm. 1-2, let $\sigma = \min_t \sigma_t$. If $\eta_t \geq \sigma \geq 0$ for all t and $\alpha \leq \sqrt{T}$, the regret of Alg. 12 is

Algorithm 13 SKETCHED-KONS

Input: Feasible parameter C, stepsizes η_t, regulariz. α

1: Initialize $\mathbf{w}_0 = \mathbf{0}, \mathbf{g}_0 = \mathbf{0}, b_0 = 0$, $\widetilde{\mathbf{A}}_0 = \alpha \mathbf{\Pi}_T$
2: Initialize independent run of KORS
3: **for** $t = \{1, \dots, T\}$ **do**
4: receive $\mathbf{x}_t$
5: compute $\widetilde{\mathbf{u}}_t = \widetilde{\mathbf{A}}_{t-1}^{-1}(\sum_{s=0}^{t-1} \widetilde{b}_s \mathbf{g}_s)$
6: compute $\breve{y}_t = \varphi(\mathbf{x}_t)^\intercal \widetilde{\mathbf{u}}_t$
7: predict $\widetilde{y}_t = \varphi(\mathbf{x}_t)^\intercal \widetilde{\mathbf{w}}_t = \breve{y}_t - h(\breve{y}_t)$, observe $\mathbf{g}_t$
8: compute $\widetilde{\tau}_{t,t}$ using KORS (i.e., Algorithm 8 and Eq. 3.4)
9: compute $\widetilde{p}_t = \max\{\widetilde{\tau}_{t,t}, \beta\}$
10: draw $q_t \sim \mathcal{B}(\widetilde{p}_t, \overline{q})$
11: update $\widetilde{\mathbf{A}}_t = \widetilde{\mathbf{A}}_{t-1} + \eta_t \frac{q_t}{\overline{q}} \mathbf{g}_t \mathbf{g}_t^\intercal$
12: **end for**

upper-bounded as

$$R_T \leq \alpha \|\mathbf{w}^*\|^2 + d_{\text{onl}}^T(\alpha)/\eta_T + 4C^2L^2 \sum_{t=1}^{T} (\eta_t - \sigma).$$

In particular, if for all t we have $\sigma_t \geq \sigma > 0$, setting $\eta_t = \sigma$ we obtain

$$R_T \leq \alpha \|\mathbf{w}^*\|^2 + 2d_{\text{eff}}^T\left(\alpha/(\sigma L^2)\right) \frac{\log(2\sigma L^2 T)}{\sigma},$$

otherwise, $\sigma = 0$ and setting $\eta_t = 1/(LC\sqrt{t})$ we obtain

$$R_T \leq \alpha \|\mathbf{w}^*\|^2 + 4LC\sqrt{T} d_{\text{eff}}^T(\alpha/L^2) \log(2L^2 T).$$

Comparison to OL algorithms. We first notice that the effective dimension $d_{\text{eff}}^T(\alpha)$ can be seen as a soft rank for $\overline{\mathbf{K}}_T$ and that it is smaller than the rank r for any α.[1] For exp-concave functions (i.e., $\sigma > 0$), we slightly improve over the bound of Luo et al., (2016) from $\mathcal{O}(d \log T)$ down to $\mathcal{O}(d_{\text{eff}}^T(\alpha) \log T) \leq \mathcal{O}(r \log T)$, where r is the (unknown) rank of the dataset. Furthermore, when $\sigma = 0$, setting $\eta_t = \sqrt{1/(L^2C^2t)}$ gives us a regret $\mathcal{O}(d_{\text{eff}}^T(\alpha)\sqrt{T})$, which is potentially much smaller than $\mathcal{O}(r\sqrt{T})$ and $\mathcal{O}(\sqrt{Td})$. Furthermore, if an oracle provided us in advance with $d_{\text{eff}}^T(\alpha)$, setting $\eta_t = \sqrt{d_{\text{eff}}^T(\alpha)/(L^2C^2t)}$ gives a regret $\mathcal{O}(\sqrt{d_{\text{eff}}^T(\alpha)T}) \leq \mathcal{O}(\sqrt{rT})$.

Comparison to OKL algorithms. Simple functional gradient descent, e.g., NORMA (Kivinen et al., 2004a), achieves a $\mathcal{O}(\sqrt{T})$ regret when properly tuned (Zhu and Xu, 2015), regardless of the loss function. For the special case of squared loss, Zhdanov and Kalnishkan, (2010) show that kernel recursive least squares achieves the same $\mathcal{O}(\log(\text{Det}(\overline{\mathbf{K}}_T/\alpha + \mathbf{I}_T)))$ regret as achieved by KONS for general exp-concave losses.

5.2 Sketched KONS

[1] This can be easily seen as $d_{\text{eff}}^T(\alpha) = \sum_t \lambda_t/(\lambda_t + \alpha)$, where λ_t are the eigenvalues of $\overline{\mathbf{K}}_T$.

Building on KORS, we now introduce a sketched variant of KONS that can efficiently trade off between computational performance and regret. Sketched-KONS (Algorithm 13) runs KORS (Algorithm 8) as a black-box estimating RLS $\widetilde{\tau}_t$, that are then used to sketch the original matrix $\mathbf{A}_t$ with a matrix $\widetilde{\mathbf{A}}_t = \sum_{s=1}^t \eta_t(q_s/\overline{q})\mathbf{g}_t\mathbf{g}_t^\intercal + \mathbf{\Pi}_T$, where at each step we add the current gradient $\mathbf{g}_t\mathbf{g}_t^\intercal$ only if the one of the coin flips in q_t succeeded (i.e., $q_t \neq 0$). Unlike KORS, the elements added to $\widetilde{\mathbf{A}}_t$ are not weighted, and the probabilities $\widetilde{p}_t$ used for the individual coins $z_{t,j}$ are chosen as the maximum between $\widetilde{\tau}_{t,t}$, and a parameter $0 \leq \beta \leq 1$. Let $\mathbf{R}_t$ be the unweighted counterpart of $\mathbf{S}_t$, that is $[\mathbf{R}_t]_{i,j} = 0$ if $[\mathbf{S}_t]_{i,j} = 0$ and $[\mathbf{R}_t]_{i,j} = 1$ if $[\mathbf{S}_t]_{i,j} \neq 0$. Then we can efficiently compute the coefficients $\widetilde{b}_t$ and predictions $\widetilde{y}_t$ as follows.

Lemma 5.5 — (Calandriello et al., 2017c). Let $\mathbf{E}_t = \mathbf{R}_t^\intercal\overline{\mathbf{K}}_t\mathbf{R}_t + \alpha\mathbf{I}_t$ be an auxiliary matrix, then all the components $\widetilde{b}_i = [\widetilde{\mathbf{b}}_t]_i$ used in Alg. 13 can be computed as

$$\dot{g}_i\sqrt{\eta_i}\left(\widetilde{y}_i - \frac{\alpha h(\breve{y}_i)}{k_{i,i} - \overline{\mathbf{k}}_{[i-1],i}^\intercal\mathbf{R}_{i-1}\mathbf{E}_{i-1}^{-1}\mathbf{R}_{i-1}\overline{\mathbf{k}}_{[i-1],i}} - \frac{1}{\eta_i}\right).$$

Then we can compute

$$\breve{y}_t = \frac{1}{\alpha}\left(\mathbf{k}_{[t-1],t}^\intercal\mathbf{D}_{t-1}\mathbf{b}_{t-1} - \mathbf{k}_{[t-1],t}^\intercal\mathbf{D}_{t-1}\mathbf{R}_{t-1}\mathbf{E}_{t-1}^{-1}\mathbf{R}_{t-1}\overline{\mathbf{K}}_{t-1}\mathbf{b}_{t-1}\right).$$

Note that since the columns in $\mathbf{R}_t$ are selected without weights, $(\mathbf{R}_t^\intercal\overline{\mathbf{K}}_t\mathbf{R}_t + \alpha\mathbf{I}_t)^{-1}$ can be updated efficiently using block inverse updates, and only when $\widetilde{\mathbf{A}}_t$ changes. While the specific reason for choosing the unweighted sketch $\widetilde{\mathbf{A}}_t$ instead of the weighted version $\mathbf{A}_t^{\mathcal{I}_t}$ used in KORS is discussed further in the discussion of Theorem 5.7, the following corollary shows that $\widetilde{\mathbf{A}}_t$ is as accurate as $\mathbf{A}_t^{\mathcal{I}_t}$ in approximating $\mathbf{A}_t$ up to the smallest sampling probability $\widetilde{p}_t^\beta$.

Corollary 5.6 — Theorem 3.11. Let $\widetilde{p}_{\min}^\beta = \min_{t=1}^T \widetilde{p}_t^\beta$. Then w.h.p., we have

$$(1-\varepsilon)\widetilde{p}_{\min}\mathbf{A}_t \preceq \widetilde{p}_{\min}\mathbf{A}_t^{\mathcal{I}_t} \preceq \widetilde{\mathbf{A}}_t.$$

We can now state the main result of this section. Since for Sketched-KONS we are interested not only in regret minimization, but also in space and time complexity, we do not consider the case $\sigma = 0$, because when the function does not have any curvature, standard GD already achieves the optimal regret of $\mathcal{O}(\sqrt{T})$ (Zhu and Xu, 2015) while requiring only $\mathcal{O}(t)$ space and time per iteration.

Theorem 5.7 — (Calandriello et al., 2017c). For any sequence of losses ℓ_t satisfying Asm. 1-2, let $\sigma = \min_t \sigma_t$ and $\overline{\tau}_{\min} = \min_{t=1}^T \overline{\tau}_{t,t}$. When $\eta_t \geq \sigma > 0$ for all t, $\alpha \leq \sqrt{T}$, $\overline{q} \geq 3\log(T/\delta)/\varepsilon^2$, if we set $\eta_t = \sigma$ then w.p. $1-\delta$ the regret of Alg. 13 satisfies

$$\widetilde{R}_T \leq \alpha\|\mathbf{w}^*\|^2 + 2\frac{d_{\text{eff}}^T\left(\alpha/(\sigma L^2)\right)\log(2\sigma L^2 T)}{\sigma\max\{\beta, \overline{\tau}_{\min}\}}, \tag{5.2}$$

and the algorithm runs in $\widetilde{\mathcal{O}}(d_{\text{eff}}^t(\alpha)^2 + t^2\beta^2)$ time and $\widetilde{\mathcal{O}}(d_{\text{eff}}^t(\alpha)^2 + t^2\beta^2)$ space complexity for each iteration t.

Proof sketch: Given these guarantees, we need to bound R_G and R_D. Bounding R_D

is straightforward, since by construction SKETCHED-KONS adds at most $\eta_t \mathbf{g}_t \mathbf{g}_t^\mathsf{T}$ to $\widetilde{\mathbf{A}}_t$ at each step. To bound R_G instead, we must take into account that an unweighted $\widetilde{\mathbf{A}}_t = \overline{\Phi}_t \mathbf{R}_t \mathbf{R}_t^\mathsf{T} \overline{\Phi}_t^\mathsf{T} + \alpha \mathbf{\Pi}_T$ can be up to $\widetilde{p}_{\min}$ distant from the weighted $\overline{\Phi}_t \mathbf{S}_t \mathbf{S}_t^\mathsf{T} \overline{\Phi}_t^\mathsf{T}$ for which we have guarantees. Hence the $\max\{\beta, \overline{\tau}_{\min}\}$ term appearing at the denominator. The full proof is available in Section 5.4.

Regret guarantees. From Eq. 5.2 we can see that when $\overline{\tau}_{\min}$ is not too small, setting $\beta = 0$ we recover the guarantees of exact KONS. Since usually we do not know $\overline{\tau}_{\min}$, we can choose to set $\beta > 0$, and as long as $\beta \geq 1/\operatorname{polylog} T$, we preserve a (poly)-logarithmic regret.

Computational speedup. The time required to compute $\mathbf{k}_{[t-1],t}$, $k_{t,t}$, and $\mathbf{k}_{[t-1],t}^\mathsf{T} \mathbf{D}_{t-1} \mathbf{b}_{t-1}$ gives a minimum $\mathcal{O}(t)$ per-step complexity. Note that $\overline{\mathbf{K}}_{t-1} \mathbf{b}_{t-1}$ can also be computed incrementally in $\mathcal{O}(t)$ time. Denoting the size of the dictionary at time t as $B_t = \widetilde{\mathcal{O}}(d_{\text{eff}}(\alpha)_t + t\beta)$, computing $[\widetilde{\mathbf{b}}_t]_i$ and $\mathbf{k}_{[t-1],t}^\mathsf{T} \mathbf{D}_{t-1} \mathbf{R}_{t-1} \mathbf{E}_{t-1}^{-1} \mathbf{R}_{t-1} \overline{\mathbf{K}}_{t-1} \mathbf{b}_{t-1}$ requires an additional $\mathcal{O}(B_t^2)$ time. When $\beta \leq d_{\text{eff}}^t(\alpha)/t$, each iteration takes $\widetilde{\mathcal{O}}(d_{\text{eff}}^t(\alpha)^2)$ to compute $\widetilde{\tau}_{t,t}$ incrementally using KORS, $\widetilde{\mathcal{O}}(d_{\text{eff}}^t(\alpha)^2)$ time to update $\widetilde{\mathbf{A}}_t^{-1}$ and $\widetilde{\mathcal{O}}(d_{\text{eff}}^t(\alpha)^2)$ time to compute $[\mathbf{b}_t]_t$. When $\beta > d_{\text{eff}}^t(\alpha)/t$, each iteration still takes $\widetilde{\mathcal{O}}(d_{\text{eff}}^t(\alpha)^2)$ to compute $\widetilde{\tau}_{t,t}$ using KORS and $\mathcal{O}(t^2\beta^2)$ time to update the inverse and compute $[\mathbf{b}_t]_t$.
Therefore, in the case when $\overline{\tau}_{\min}$ is not too small, our runtime is of the order $\widetilde{\mathcal{O}}(d_{\text{eff}}^t(\alpha)^2 + t)$, which is almost as small as the $\mathcal{O}(t)$ runtime of GD but with the advantage of a second-order method logarithmic regret. Moreover, when $\overline{\tau}_{\min}$ is small and we set a large β, we can trade off a $1/\beta$ increase in regret for a β^2 decrease in space and time complexity when compared to exact KONS (e.g., setting $\beta = 1/10$ would correspond to a tenfold increase in regret, but a hundred-fold reduction in computational complexity).

Asymptotic behavior. Notice however, that space and time complexity, grow roughly with a term $\Omega(t \min_{s=1}^t \widetilde{p}_s) \sim \Omega(t \max\{\beta, \overline{\tau}_{\min}\})$, so if this quantity does not decrease over time, the computational cost of SKETCHED-KONS will remain large and close to exact KONS.
This is to be expected, since SKETCHED-KONS must always keep an accurate sketch in order to guarantee a logarithmic regret bound. Note that Luo et al., (2016) took an opposite approach for OL, where they keep a fixed-size sketch but possibly pay in regret, if this fixed size happens to be too small. Since a non-logarithmic regret is achievable simply running vanilla GD, we rather opted for an adaptive sketch at the cost of space and time complexity. In batch optimization, where ℓ_t does not change over time, another possibility is to stop updating the solution once $\overline{\tau}_{\min}$ becomes too small. When $\mathbf{H}_s$ is the Hessian of ℓ w.r.t. $\mathbf{w}_s$, then the quantity $\mathbf{g}_t^\mathsf{T} \mathbf{H}_t^{-1} \mathbf{g}_t$, in the context of Newton's method, is called *Newton decrement* and it corresponds up to constant factors to $\overline{\tau}_{\min}$. Since a stopping condition based on Newton's decrement is directly related to the near-optimality of the current $\mathbf{w}_t$ (Nesterov and Nemirovskii, 1994), stopping when $\overline{\tau}_{\min}$ is small also provides guarantees about the quality of the solution.

Sampling distribution. Note that although $\beta > 0$ means that all columns have a small uniform chance of being selected for inclusion in $\widetilde{\mathbf{A}}_t$, this is *not* equivalent to uniformly sampling columns. It is rather a combination of a RLS-based sampling to ensure that columns important to reconstruct $\mathbf{A}_t$ are selected and a threshold on the probabilities to avoid too much variance in the estimator.

Biased estimator and results in expectation. The random approximation $\widetilde{\mathbf{A}}_t$ is biased, since $\mathbb{E}[\overline{\Phi}_t \mathbf{R}_t \mathbf{R}_t^\mathsf{T} \overline{\Phi}_t^\mathsf{T}] = \overline{\Phi}_t \operatorname{Diag}(\{\overline{\tau}_{t,t}\}) \overline{\Phi}_t^\mathsf{T} \neq \overline{\Phi}_t \overline{\Phi}_t^\mathsf{T}$. Another option would be to use

a weighted and unbiased approximation $\widetilde{\mathbf{A}}'_t = \sum_{s=1}^t \eta_s q_s/(\overline{q}\widetilde{p}_s)\mathbf{g}_s\mathbf{g}_s^\intercal$ used in KORS and a common choice in matrix approximation methods, see e.g., (Alaoui and Mahoney, 2015). Due to its unbiasedness, this variant would automatically achieve the same logarithmic regret as exact KONS *in expectation*, similar to the result obtained by Luo et al., (2016), using Gaussian random projection in OL. While any unbiased estimator, e.g., uniform sampling of $\mathbf{g}_t$, would achieve this result, RLS-based sampling already provides strong reconstruction guarantees sufficient to bound R_G. Unfortunately, and differently from the batch case, the adaptive weights $1/\widetilde{p}_s$ may cause large variations in $\widetilde{\mathbf{A}}_t$ over consecutive steps, thus leading to a large regret R_D in high probability.

5.2.1 Limitations of budgeted OKL algorithms

From the discussion above, it appears that a weighted, unbiased dictionary may not achieve high-probability logarithmic guarantee because of the high variance coming from sampling. Therefore, if we want to recover the logarithmic regret guarantee, we may have to pay for it with a large dictionary. This may actually be due to the analysis, the setting, or the algorithm.

- The regret analysis of is tight for the OL setting without any curvature assumption, since Luo et al., (2016) shows a lower bound for this case. As OL is a special case of OKL, it does not seem easy to find a tighter bound, although possible data-adaptive bounds remain an interesting open question.
- Regarding the difficulty of approximation in the online setting, notice that in the batch setting (Chapter 4), the subsampling does not cause any issue and we can have strong learning guarantees in high probability with a small dictionary. Differently from the batch setting, in OKL we have to balance two opposing goals: we want our solution to change quickly, in order to minimize R_G, but we also want our solution to not change too much, to minimize R_D.
- Algorithmically, an important property of the dictionary learning approach used in KORS is that it can only include atoms coming from the input, re-weight them, and cannot remove an atom from the dictionary after insertion. If we set the weights to have an unbiased estimate, we achieve an accurate R_G but suffer a huge regret in R_D. On the other hand, we can store unweighted atoms to have small R_D but large R_G.

To see how this sampling without removal approach could fail, consider the following simple scenario. The adversary always presents to the learner the same point $\mathbf{x}$ (with associated ϕ), but for the loss it alternates between $\ell_{2t}(\mathbf{w}_t) = (C - \phi^\intercal\mathbf{w}_t)^2$ on even steps and $\ell_{2t+1}(\mathbf{w}_t) = (-C - \phi^\intercal\mathbf{w}_t)^2$ on odd steps. Then, $\sigma_t = \sigma = 1/(8C^2)$, and we have a gradient that always points in the same ϕ direction, but switches sign at each step. The optimal solution in hindsight is asymptotically $\mathbf{w} = \mathbf{0}$ and let this be also our starting point $\mathbf{w}_0$. We also set $\eta_t = \sigma$, since this is what KONS would do, and $\alpha = 1$ for simplicity.

For this scenario, we can compute R_G and R_D in closed form,

$$R_G \leq \sum_{t=1}^T \frac{\dot{g}_t^2}{\sum_{s=1}^t \dot{g}_s^2\sigma + \alpha} \leq \sum_{t=1}^T \frac{C^2}{C^2\sigma t + \alpha} \leq \mathcal{O}(\log T), \qquad R_D = \sum\nolimits_{s=1}^t (\eta_t - \sigma)(\mathbf{w}_t^\intercal\mathbf{g}_t)^2 = 0.$$

Note that although the matrix $\mathbf{A}_t$ is rank 1 at each time step, vanilla ONS does not take advantage of this easy data, and would store it all with a $\mathcal{O}(t^2)$ space in OKL.

Notice that in this all-identical samples example, the losses ℓ_t are effectively strongly convex,

and even basic gradient descent with a stepsize $\eta_t = 1/t$ would achieve logarithmic regret (Zhu and Xu, 2015) with even smaller space. On the other hand, we show how sampling without removal has difficulties in minimizing the regret bound from Proposition 5.3 in our simple scenario.
In particular, consider an arbitrary (possibly randomized) algorithm that is allowed only to reweight atoms in the dictionary and not to remove them. In our example, this translates to choosing a schedule of weights w_s and set $\widetilde{\mathbf{A}}_t = \sum_{s=1}^t w_s \overline{\phi}_s \overline{\phi}_s = W_t \overline{\phi}\overline{\phi}$ with total weight $W = W_T = \sum_{s=1}^T w_s$ and space complexity equal to the number of non-zero weights $B = |\{w_s \neq 0\}|$. We can show that there is no schedule for this specific class of algorithms with good performance due to the following three conflicting goals.

(1) To maintain R_G small, $\sum_{s=1}^t w_s$ should be as large as possible, as early as possible.
(2) To maintain R_D small, we should choose weights $w_t > 1$ as few times as possible, since we accumulate $\max\{w_t - 1, 0\}$ regret every time.
(3) To maintain the space complexity small, we should choose only a few $w_t \neq 0$.

To enforce goal (3), we must choose a schedule with no more than B non-zero entries. Given the budget B, to satisfy goal (2) we should use all the B budget in order to exploit as much as possible the $\max\{w_t - 1, 0\}$ in R_D, or in other words we should use exactly B non-zero weights, and none of these should be smaller than 1. Finally, to minimize R_G we should raise the sum $\sum_{s=1}^t w_s$ as quickly as possible, settling on a schedule where $w_1 = W - B$ and $w_s = 1$ for all the other B weights. It easy to see that if we want logarithmic R_G, W needs to grow as T, but doing so with a logarithmic B would make $R_D = T - B = \Omega(T)$. Similarly, keeping $W = B$ in order to reduce R_D would increase R_G. In particular notice, that the issue does not go away even if we know the RLS perfectly, because the same reasoning applies. This simple example suggests that dictionary-based sketching methods without removal, which are very successful in batch scenarios, may actually fail in achieving logarithmic regret in online optimization.

Algorithmically, this could be potentially fixed if we knew how to *remove* less important columns from dictionary to gain some slack in R_D, or how to *create new atoms* that can be used to represent multiple inputs, such as Frequent Directions (FD, Ghashami et al., 2016a) do in the Euclidean case.

Removing an atom present in the dictionary reduces our space complexity, but at the same time increases R_G, since the dampening effect of $\widetilde{\mathbf{A}}_t^{-1}$ decreases. Moreover, let $\mathbf{g}_t$ be the last computed gradient, and let $\mathbf{g}_s$ with $s < t$ be the gradient we want to remove from $\widetilde{\mathbf{A}}_t$ and our dictionary. Unless $\mathbf{g}_t = \mathbf{g}_s$ (i.e., for any arbitrarily small perturbation) the operator $\widetilde{\mathbf{A}}_t - \widetilde{\mathbf{A}}_{t-1} = \mathbf{g}_t\mathbf{g}_t^\intercal - \mathbf{g}_s\mathbf{g}_s^\intercal$ has both a positive and negative eigenvalue, and the adversary can make the corresponding $(\mathbf{w}_t - \mathbf{w}^*)^\intercal((1-\sigma_t)\mathbf{g}_t\mathbf{g}_t^\intercal - \mathbf{g}_s\mathbf{g}_s^\intercal)(\mathbf{w}_t - \mathbf{w}^*)$ term in R_D arbitrarily large. Therefore, removing a gradient from the dictionary will decrease space, but it increases R_G, and might cause an arbitrarily large increase in R_D due to the adversary. Similar arguments can be derived for schemes that keep reweighting atoms after insertion.

Finally, as our counterexample in the simple scenario hints, creating new atoms (either through projection or merging) allows for better adaptivity. For example, if we have a single gradient $\mathbf{g}$ each time slightly perturbed, we can represent it using just the principal components of these perturbed copies without storing each copy separately. This is essentially the approach taken by Luo et al., (2016) in the OL setting using FD as a sketching algorithm.
However, in OKL we cannot explicitly represent new atoms, and therefore the kernelization

of FD does not appear to be straightforward. The most recent step in this direction is proposed again by Ghashami et al., (2016b) for batch kernel PCA. It first embeds the points into a finite-dimensional space using a fixed set of random feature expansions and afterwards uses FD to reduce the computational complexity. Unfortunately, their method only works for an Euclidean input space $\mathcal{X}$, and does not have online or any-time guarantees.
Moreover, note that our example can be easily generalized to defeat any approximation strategy that uses a *fixed* embedding technique (e.g., fixed amount of random features or a fixed dictionary) or a fixed budget of SV. It suffice to replace the single ϕ vector with a set of repeating orthogonal vectors that exceed the budget.

Nonetheless, this single sample problem is intrinsically simple: its effective dimension $d_{\text{eff}}^{T}(\alpha) \simeq 1$ is small, and its induced RKHS $\mathcal{H} = \phi$ is a singleton. Therefore, following an *adaptive embedding* approach, we can reduce it to a 1 dimensional parametric problem, and solve it efficiently in this space using exact ONS updates. We can see this approach as constructing an approximate feature map $\widetilde{\varphi}$ that after one step will exactly coincide with the exact feature map φ, but allows us to run exact KONS updates efficiently replacing $\mathcal{K}$ with $\widetilde{\mathcal{K}}$.
In other words, since correctly approximating second-order updates in the exact RKHS $\mathcal{H}$ is hard due to the adversary, we instead resort to make exact second order updates, that the adversary cannot exploit, but in an approximate and smaller RKHS $\widetilde{\mathcal{H}}$.
Building on this intuition, we propose PROS-N-KONS, a new second-order FGD method that continuously searches for the best embedding space $\mathcal{H}_{\mathcal{I}_t}$ and, at the same time, exploits the small embedding space $\mathcal{H}_{\mathcal{I}_t}$ to efficiently perform exact second-order updates.

5.3 The PROS-N-KONS algorithm

A common approach to alleviate the computational cost of OKL is to replace the high-dimensional feature map φ with a finite-dimensional approximate feature map $\widetilde{\varphi}$.
Let $\mathcal{I} = \{\phi_i\}_{i=1}^{j}$ be a dictionary of j points chosen from those revealed by the adversary, and let and $\Phi_{\mathcal{I}}$ be the associated feature matrix with $\varphi(\mathbf{x}_i)$ as columns. We define the embedding $\widetilde{\varphi}(\mathbf{x}) = \mathbf{\Sigma}^{-1}\mathbf{U}^{\intercal}\Phi_{\mathcal{I}}^{\intercal}\varphi(\mathbf{x}) \in \mathbb{R}^j$, where $\Phi_{\mathcal{I}} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^{\intercal}$ is the singular value decomposition of the feature matrix. While in general $\Phi_{\mathcal{I}}$ is infinite dimensional and cannot be directly decomposed, we exploit the fact that $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\intercal}\mathbf{V}\mathbf{\Sigma}\mathbf{U}^{\intercal} = \Phi_{\mathcal{I}}^{\intercal}\Phi_{\mathcal{I}} = \mathbf{K}_{\mathcal{I}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\intercal}$ and that $\mathbf{K}_{\mathcal{I}}$ is a (finite-dimensional) PSD matrix. Therefore it is sufficient to compute the eigenvectors $\mathbf{U}$ and eigenvalues $\mathbf{\Lambda}$ of $\mathbf{K}_{\mathcal{I}}$ and take the square root $\mathbf{\Lambda}^{1/2} = \mathbf{\Sigma}$.
Note that with this definition we are effectively replacing the kernel $\mathcal{K}$ and $\mathcal{H}$ with an approximate $\mathcal{K}_{\mathcal{I}}$ and $\mathcal{H}_{\mathcal{I}}$, such that $\mathcal{K}_{\mathcal{I}}(\mathbf{x}, \mathbf{x}') = \widetilde{\varphi}(\mathbf{x})^{\intercal}\widetilde{\varphi}(\mathbf{x}') = \varphi(\mathbf{x})^{\intercal}\Phi_{\mathcal{I}}\mathbf{U}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}^{-1}\mathbf{U}^{\intercal}\Phi_{\mathcal{I}}^{\intercal}\varphi(\mathbf{x}') = \varphi(\mathbf{x})^{\intercal}\mathbf{\Pi}_{\mathcal{I}}\varphi(\mathbf{x}')$ where $\mathbf{\Pi}_{\mathcal{I}} = \Phi_{\mathcal{I}}(\Phi_{\mathcal{I}}^{\intercal}\Phi_{\mathcal{I}})^{-1}\Phi_{\mathcal{I}}^{\intercal}$ is the projection matrix on the column span of $\Phi_{\mathcal{I}}$. Since $\widetilde{\varphi}$ returns vectors in $\mathbb{R}^j$, this transformation effectively reduces the computation complexity of kernel operations from t down to the size of the dictionary j. The accuracy of $\widetilde{\varphi}$ is directly related to the accuracy of the projection $\mathbf{\Pi}_{\mathcal{I}}$ in approximating the projection $\mathbf{\Pi}_t = \Phi_t(\Phi_t^{\intercal}\Phi_t)^{-1}\Phi_t^{\intercal}$, so that for all $s, s' \in [t]$, $\widetilde{\varphi}(\mathbf{x}_s)^{\intercal}\widetilde{\varphi}(\mathbf{x}_{s'})$ is close to $\varphi(\mathbf{x}_s)^{\intercal}\mathbf{\Pi}_t\varphi(\mathbf{x}_{s'}) = \varphi(\mathbf{x}_s)^{\intercal}\varphi(\mathbf{x}_{s'})$.

All that is left is to find an efficient algorithm to choose a good dictionary $\mathcal{I}$ to minimize the error $\mathbf{\Pi}_{\mathcal{I}} - \mathbf{\Pi}_t$. Among dictionary-selection methods, we focus on KORS (Section 3.2). While SQUEAK can generate smaller dictionaries, as we will see the frequent insertion and removal of atoms from the dictionary can change the subspace $\mathbf{\Pi}_{\mathcal{I}}$ too quickly over time. While in the batch setting this was not a problem, in the online setting the adversary can exploit this large variance and negatively affect our performance.
Therefore we prefer the slightly suboptimal but more stable KORS, where the total number

Algorithm 14 PROS-N-KONS

Input: Feasible parameter C, step-sizes η_t, regularizer α

Initialize $j = 0$, $\widetilde{\mathbf{w}}_0 = \mathbf{0}, \widetilde{\mathbf{g}}_0 = \mathbf{0}$, $\widetilde{\mathbf{\Pi}}_0 = \mathbf{0}$, $\widetilde{\mathbf{A}}_0 = \alpha \mathbf{I}_0$,
Start a KORS instance with an empty dictionary $\mathcal{I}_0$.
for $t = \{1, \dots, T\}$ **do**
  Receive $\mathbf{x}_t$, feed ϕ_t to KORS.
  Receive q_t from KORS (point added to dictionary or not)
  if $q_{t-1} \neq 0$ **then** { ▷ Dictionary changed, reset.}
    $j = j + 1$
    Build $\mathbf{K}_j$ from $\mathcal{I}_j$ and decompose it in $\mathbf{U}_j \mathbf{\Sigma}_j \mathbf{\Sigma}_j^\intercal \mathbf{U}_j^\intercal$
    Set $\widetilde{\mathbf{A}}_{t-1} = \alpha \mathbf{I}_j \in \mathbb{R}^{j \times j}$.
    $\widetilde{\boldsymbol{\omega}}_t = \mathbf{0} \in \mathbb{R}^j$
  else {▷ Execute a gradient-descent step.}
    Compute map ϕ_t and approximate map $\widetilde{\phi}_t = \mathbf{\Sigma}_j^{-1} \mathbf{U}_j^\intercal \Phi_j^\intercal \phi_t \in \mathbb{R}^j$.
    Compute $\widetilde{\boldsymbol{\upsilon}}_t = \widetilde{\boldsymbol{\omega}}_{t-1} - \widetilde{\mathbf{A}}_{t-1}^{-1} \widetilde{\mathbf{g}}_{t-1}$.
    Compute $\widetilde{\boldsymbol{\omega}}_t = \widetilde{\boldsymbol{\upsilon}}_t - \frac{h(\widetilde{\phi}_t^\intercal \widetilde{\boldsymbol{\upsilon}}_t)}{\widetilde{\phi}_t^\intercal \widetilde{\mathbf{A}}_{t-1}^{-1} \widetilde{\phi}_t} \widetilde{\mathbf{A}}_{t-1}^{-1} \widetilde{\phi}_t$ where $h(z) = \operatorname{sign}(z) \max\{|z| - C,\ 0\}$
  end if
  Predict $\widetilde{y}_t = \widetilde{\phi}_t^\intercal \widetilde{\boldsymbol{\omega}}_t$.
  Observe $\widetilde{\mathbf{g}}_t = \nabla_{\widetilde{\boldsymbol{\omega}}_t} \ell_t(\widetilde{\phi}_t^\intercal \widetilde{\boldsymbol{\omega}}_t) = \ell_t'(\widetilde{y}_t) \widetilde{\phi}_t$.
  Update $\widetilde{\mathbf{A}}_t = \widetilde{\mathbf{A}}_{t-1} + \frac{\sigma_t}{2} \widetilde{\mathbf{g}}_t \widetilde{\mathbf{g}}_t^\intercal$.
end for

of dictionary changes equals the final size of the dictionary, and we can prove that w.h.p. this number of changes is small.

We start from an empty dictionary $\mathcal{I}_0$ and a null predictor $\mathbf{w}_0 = \mathbf{0}$. At each round, PROS-N-KONS (Algorithm 14) receives a new point $\mathbf{x}_t$ and invokes KORS to decide whether it should be included in the current dictionary or not.
Let t_j with $j \in [J]$ be the *random* step when KORS introduces ϕ_{t_j} in the dictionary. We analyze PROS-N-KONS as an epoch-based algorithm using these milestones t_j. Note that the length $h_j = t_{j+1} - t_j$ and total number of epochs J is random, and is decided in a data-adaptive way by KORS based on the difficulty of the problem.
During epoch j, we have a fixed dictionary $\mathcal{I}_j$ that induces a feature matrix $\Phi_{\mathcal{I}_j}$ containing samples $\phi_i \in \mathcal{I}_j$, an embedding $\widetilde{\varphi}(\mathbf{x}) : \mathcal{X} \to \mathbb{R}^j = \mathbf{\Sigma}_j^{-1} \mathbf{U}_j^\intercal \Phi_j^\intercal \varphi(\mathbf{x})$ based on the singular values $\mathbf{\Sigma}_j$ and singular vectors $\mathbf{U}_j$ of Φ_j, with its associated approximate kernel function $\widetilde{\mathcal{K}}$ and induced RKHS $\mathcal{H}_j$. At each round $t_j < t < t_{j+1}$, we perform an *exact* KONS update using the *approximate* map $\widetilde{\varphi}$. This can be computed *explicitly* since $\widetilde{\phi}_t$ is in $\mathbb{R}^j$ and can be easily stored in memory. The update rules are

$$\widetilde{\boldsymbol{\upsilon}}_t = \widetilde{\boldsymbol{\omega}}_{t-1} - \widetilde{\mathbf{A}}_{t-1}^{-1} \widetilde{\mathbf{g}}_{t-1}, \qquad \widetilde{\boldsymbol{\omega}}_t = \Pi_{\mathcal{S}_t}^{\mathbf{A}_{t-1}}(\boldsymbol{\upsilon}_t) = \widetilde{\boldsymbol{\upsilon}}_t - \frac{h(\widetilde{\phi}_t^\intercal \widetilde{\boldsymbol{\upsilon}}_t)}{\widetilde{\phi}_t^\intercal \widetilde{\mathbf{A}}_{t-1}^{-1} \widetilde{\phi}_t} \widetilde{\mathbf{A}}_{t-1}^{-1} \widetilde{\phi}_t,$$

$$\widetilde{\mathbf{A}}_t = \widetilde{\mathbf{A}}_{t-1} + \frac{\sigma_t}{2} \widetilde{\mathbf{g}}_t \widetilde{\mathbf{g}}_t^\intercal,$$

where we compute the oblique projection $\Pi_{\mathcal{S}_t}^{\mathbf{A}_{t-1}}$ using Luo et al., (2016)'s closed-form solution.
When $t = t_j$ and a new epoch begins, we perform a *reset step* before taking the first gradient step in the new embedded space. We update the feature-map $\widetilde{\varphi}$, but we reset $\widetilde{\mathbf{A}}_{t_j}$ and $\widetilde{\boldsymbol{\omega}}_{t_j}$ to

zero. While this may seem a poor choice, as information learned over time is lost, it leaves intact the dictionary. As long as *(a)* the dictionary, and therefore the embedded space where we perform our GD, keeps improving and *(b)* we do not needlessly reset too often, we can count on the fast second-order updates to quickly catch up to the best function in the current $\mathcal{H}_j$. The motivating reason to reset the descent procedure when we switch subspace is to guarantee that our starting point in the descent cannot be influenced by the adversary, and therefore allow us to provide regret guarantees for the overall process (Section 5.3.1).

Computational complexity. PROS-N-KONS's computation complexity is dominated by $\widetilde{\mathbf{A}}_t^{-1}$ inversion required to compute the projection and the gradient update and by the query to KORS, that internally also inverts a $j \times j$ matrix. Therefore, a naïve implementation requires $\mathcal{O}(j^3)$ per-step time, and has a space $\mathcal{O}(j^2)$ space complexity necessary to store $\widetilde{\mathbf{A}}_t$. Notice that taking advantage of the fact that KORS only adds SV to the dictionary and never removes them, and that similarly, the $\widetilde{\mathbf{A}}_t$ matrix is constructed using rank-one updates, a careful implementation reduces the per-step cost to $\mathcal{O}(j^2)$.
Overall the total runtime of PROS-N-KONS is then $\mathcal{O}(TJ^2)$, which using the bound on J provided by Theorem 3.11 and neglecting logarithmic terms reduces to $\widetilde{\mathcal{O}}(Td_{\text{eff}}^T(\gamma)^2)$.
Compared to other exact second-order FGD methods, such as KONS or RKLS, PROS-N-KONS dramatically improves the time and space complexity from *polynomial to linear*.
Against other approximate second-order methods, PROS-N-KONS does not add a new SV at each step. This way it removes T^2 from the $\widetilde{\mathcal{O}}(T^2\beta^2 + Td_{\text{eff}}^T(\gamma)^3)$ time complexity of Sketched-KONS (Calandriello et al., 2017c). Moreover, when $\min_t \tau_{t,t}$ is small, Sketched-KONS needs to compensate by adding a constant probability of adding a SV to the dictionary, resulting in a larger runtime complexity, while PROS-N-KONS has *no dependency on the value of the RLS*.
Even compared to first-order methods, which incur a larger regret, PROS-N-KONS performs favorably, improving on the $\mathcal{O}(T^2)$ runtime of exact first-order FGD. Compared to other approximate methods, the variant using rank-one updates matches the $\mathcal{O}(J^2)$ per-step cost of the more accurate first-order methods such as the Budgeted Perceptron (Cavallanti et al., 2007), Projectron (Orabona et al., 2008), and Nyström-GD (Lu et al., 2016), while improving on their regret. PROS-N-KONS also closely matches faster but less accurate $\mathcal{O}(J)$ methods such as the Forgetron (Dekel et al., 2008) and Budgeted-GD (Zhao et al., 2012).

5.3.1 Regret guarantees

In this section, we study the regret performance of PROS-N-KONS.

Theorem 5.8 — (Calandriello et al., 2017b). For any sequence of losses ℓ_t satisfying Assumption 2 with Lipschitz constant L, let $\sigma = \min_t \sigma_t$. If $\eta_t \geq \sigma$ for all t, $\alpha \leq \sqrt{T}$, $\gamma \leq \alpha$, and predictions are bounded by C, then the regret of PROS-N-KONS over T steps is bounded w.p. $1-\delta$ as

$$R_T(\mathbf{w}) \leq J\Big(\alpha\|\mathbf{w}\|^2 + \frac{4}{\sigma} d_{\text{eff}}^T\Big(\frac{\alpha}{\sigma L^2}\Big) \log\big(2\sigma L^2 T/\alpha\big)\Big) + \frac{L^2}{\alpha}\left(\frac{T\gamma\varepsilon}{4(1-\varepsilon)} + 1\right) + 2JC, \quad 5.3$$

where $J \leq 3\overline{q} d_{\text{eff}}^T(\gamma)\log(2T)$ is the number of epochs. If $\gamma = \alpha/T$ the previous bound reduces to

$$R_T(\mathbf{w}) = \mathcal{O}\Big(\alpha\|\mathbf{w}\|^2 d_{\text{eff}}^T(\alpha/T)\log(T) + d_{\text{eff}}^T(\alpha/T)\, d_{\text{eff}}^T(\alpha)\log^2(T)\Big). \quad 5.4$$

Remark (bound). The bound in Eq. 5.3 is composed of three terms.
At each epoch of PROS-N-KONS, an instance of KONS is run on the embedded feature space $\mathcal{H}_j$ obtained by using the dictionary $\mathcal{I}_j$ constructed up to the previous epoch. As a result, we directly use the bound on the regret of KONS (Theorem 5.4) for each of the J epochs, thus leading to the first term in the regret. Since a new epoch is started whenever a new SV is added to the dictionary, the number of epochs J is at most the size of the dictionary returned by KORS up to step T, which w.h.p. is $\widetilde{\mathcal{O}}(d_{\text{eff}}^T(\gamma))$, making the first term scale as $\widetilde{\mathcal{O}}(d_{\text{eff}}^T(\gamma)d_{\text{eff}}^T(\alpha))$ overall.
Nonetheless, the comparator used in the per-epoch regret of KONS is constrained to the RKHS $\mathcal{H}_j$ induced by the embedding used in epoch j. The second term accounts for the difference in performance between the best solutions in the RKHS in epoch j and in the original RKHS $\mathcal{H}$. This error is directly controlled by the (ε, γ)-accuracy of the dictionary constructed by KORS, where the RLS regularization γ and the parameter ε contribute to the factor $\gamma\varepsilon/(1-\varepsilon)$.
Since the true RKHS is never fully reconstructed, i.e., ε is constant and does not go to zero, the approximation impact on the regret is amplified by the length of the process, thus leading to an overall linear term that needs to be regularized.
Finally, the last term summarizes the regret suffered every time a new epoch is started and the default prediction $\widehat{y} = 0$ is returned. Since the values y_t and $\hat{y}_t$ are constrained in $\mathcal{S}$, this results in a regret of $2JC$.

Remark (regret comparison). Tuning the RLS regularization as $\gamma = \alpha/T$ leads to the bound in Eq. 5.4. Unlike Sketched-KONS, the bound finally displays only an explicit logarithmic dependency on T, but this comes at the cost of increasing the effective dimension, which now depends on the regularization α/T.
While in general this could possibly compromise the overall regret, if the sequence of points $\phi_1, \dots, \phi_T$ induces a kernel matrix with a rapidly decaying spectrum, the resulting regret is still competitive. For instance, if the eigenvalues of $\mathbf{K}_T$ decrease as $\lambda_t = at^{-q}$ with constants $a > 0$ and $q > 1$, then $d_{\text{eff}}^T(\alpha/T) \leq aqT^{1/q}/(q-1)$. This shows that for any $q > 2$ we obtain a regret[2] $o(\sqrt{T}\log^2(T))$ showing that KONS still improves over first-order methods.
Furthermore, if the kernel is low rank or the eigenvalues decrease exponentially, the final regret is poly-logarithmic, thus preserving the full advantage of the second-order approach. Notice that this scenario is always verified when $\mathcal{H} = \mathbb{R}^d$, and is also verified when the adversary draws samples from a stationary distribution and, e.g., we use the Gaussian kernel (Sun et al., 2012; Wathen and Zhu, 2015; Yang et al., 2017).
This result is particularly remarkable when compared to Sketched-KONS, whose regret scales as $\mathcal{O}\big(\alpha\|\mathbf{w}\|^2 + d_{\text{eff}}^T(\alpha)\log(T)/\beta\big)$, where β is the fraction of samples which is forced into the dictionary (when $\beta = 1$ we recover the bound for KONS). Even when the effective dimension is small (e.g., exponentially decaying eigenvalues), Sketched-KONS requires setting β to T^{-p} for a constant $p > 0$ to get a sub-quadratic space complexity, at the cost of increasing the regret to $\mathcal{O}(T^p\log(T))$. On the other hand, PROS-N-KONS achieves a poly-logarithmic regret with linear space complexity up to poly-log factors (i.e., $(Td_{\text{eff}}^T(\gamma)^2)$, thus greatly improving both the learning and computational performance w.r.t. Sketched-KONS.
Finally, notice that while $\gamma = \alpha/T$ is the best choice agnostic to the kernel, better bounds can be obtained optimizing Eq. 5.3 for γ depending on $d_{\text{eff}}^T(\gamma)$. For instance, let $\gamma = \alpha/T^s$, then the optimal value of s for q-polynomially decaying spectrum is $s = q/(1+q)$, leading

[2] Here we ignore the term $d_{\text{eff}}^T(\alpha)$ which is a constant w.r.t. T for any constant α.

to a regret bound $\widetilde{\mathcal{O}}(T^{q/(1+q)})$, which is always $o(\sqrt{T})$ for any $q > 1$.

Remark (comparison in the Euclidean case). In the special case $\mathcal{H} = \mathbb{R}^d$, we can make a comparison with existing approximate methods for OL. In particular, the closest algorithm is Sketched-ONS by Luo et al., (2016). Unlike PROS-N-KONS, and similarly to Sketched-KONS, they take the approach of directly approximating $\mathbf{A}_t$ in the exact $\mathcal{H} = \mathbb{R}^d$ using FD (Ghashami et al., 2015) to construct a k-rank approximation of $\mathbf{A}_t$ for a *fixed* k.
The resulting algorithm achieve a regret that is bounded by $k\log(T) + k\sum_{i=k+1}^{T}\sigma_i^2$, where the sum $\sum_{i=k+1}^{T}\sigma_i^2$ is equal to the sum of all the smallest $d-k$ eigenvalues of the covariance matrix. This quantity can vary from 0, when the data lies in a subspace of rank $r \leq k$, to $T\frac{d-k}{d}$ when the sample lie orthogonally and in equal number along all d directions available in $\mathbb{R}^d$. Computationally, the algorithm requires $\mathcal{O}(Tdk)$ time and $\mathcal{O}(dk)$ space.
Conversely, PROS-N-KONS automatically adapt its time and space complexity to the effective dimension of the algorithm $d_{\text{eff}}^T(\alpha/T)$ which is smaller than the rank for any α. As a consequence, it requires only $\widetilde{\mathcal{O}}(Tr^2)$ time and $\widetilde{\mathcal{O}}(r^2)$ space, achieving a $\mathcal{O}(r^2\log(T))$ regret independently from the spectrum of the covariance matrix.
Computationally, all of these complexities are smaller than the ones of Sketched-ONS in the regime $r < k$, which is the only one where Sketched-ONS can guarantee logarithmic (or even sublinear) regret, and where the regrets of the two algorithms are close.
Overall, while Sketched-ONS implicitly relies on the $r < k$ assumption, but continues to operate in a d dimensional space and suffers large regret if $r > k$, PROS-N-KONS will adaptively convert the d dimensional problem into a simpler one with the appropriate rank, fully reaping the computational and regret benefits.

The bound in Theorem 5.8 can be refined in the specific case of squared loss as follows.

Theorem 5.9 — (Calandriello et al., 2017b). For any sequence of squared losses $\ell_t = (y_t - \hat{y}_t)^2$, $L{=}4C$ and $\sigma{=}1/(8C^2)$, if $\eta_t \geq \sigma$ for all t, $\alpha \leq \sqrt{T}$ and $\gamma \leq \alpha$, the regret of PROS-N-KONS over T steps is bounded w.p. $1-\delta$ as

$$R_T(\mathbf{w}) \leq \sum_{j=1}^{J}\left(\frac{4}{\sigma}d_{\text{eff}}^j\left(\frac{\alpha}{\sigma L^2}\right)\log\left(2\sigma\frac{L^2}{\alpha}\operatorname{Tr}(\mathbf{K}_j)\right)+\varepsilon'\mathcal{L}_j^*\right)+J\left(L\left(C+\frac{L}{\alpha}\right)+\varepsilon'\alpha\|\mathbf{w}\|_2^2\right), \quad 5.5$$

where $\varepsilon' = \alpha\left(\alpha - \frac{\gamma\varepsilon}{1-\varepsilon}\right)^{-1} - 1$ and $\mathcal{L}_j^* = \min_{\mathbf{w}\in\mathcal{H}}\left(\sum_{t=t_j}^{t_{j+1}-1}\left(\phi_t^\intercal\mathbf{w} - y_t\right)^2 + \alpha\|\mathbf{w}\|_2^2\right)$ is the best regularized cumulative loss in $\mathcal{H}$ within epoch j.

Let $\mathcal{L}_T^*$ be the best regularized cumulative loss over all T steps, then $\mathcal{L}_j^* \leq \mathcal{L}_T^*$. Furthermore, we have that $d_{\text{eff}}^j \leq d_{\text{eff}}^T$ and thus regret in Eq. 5.5 can be (loosely) bounded as

$$R_T(\mathbf{w}) = \mathcal{O}\left(J\left(d_{\text{eff}}^T(\alpha)\log(T) + +\varepsilon'\mathcal{L}_j^* + \varepsilon'\alpha\|\mathbf{w}\|_2^2\right)\right).$$

The major difference w.r.t. the general bound in Eq. 5.3 is that we directly relate the regret of PROS-N-KONS to the performance of the best predictor in $\mathcal{H}$ in hindsight, which replaces the linear term $\gamma T/\alpha$. As a result, we can set $\gamma = \alpha$ (for which $\varepsilon' = \varepsilon/(1-2\varepsilon)$) and avoid increasing the effective dimension of the problem.
Furthermore, since $\mathcal{L}_T^*$ is the regularized loss of the optimal batch solution, we expect it to be small whenever the $\mathcal{H}$ is well designed for the prediction task at hand. For instance, if $\mathcal{L}_T^*$ scales as $\mathcal{O}(\log(T))$ for a given regularization α (e.g., in the realizable case $\mathcal{L}_T^*$ is actually just $\alpha\|\mathbf{w}\|$), then the regret of PROS-N-KONS is directly comparable with

KONS up to a multiplicative factor depending on the number of epochs J and with a much smaller time and space complexity that adapt to the effective dimension of the problem (see Theorem 3.11).

5.3.2 Experiments on online regression and classification

We empirically validate PROS-N-KONS on several regression and binary classification problems, showing that it is competitive with state-of-the-art methods. We focused on verifying 1) the advantage of second-order vs. first-order updates, 2) the effectiveness of data-adaptive embedding w.r.t. the oblivious one, and 3) the effective dimension in real datasets. Note that our guarantees hold for more challenging (possibly adversarial) settings than what we test empirically.

Algorithms. Beside PROS-N-KONS, we introduce two heuristic variants. Con-KONS follows the same update rules as PROS-N-KONS during the descent steps, but at reset steps it does not reset the solution and instead computes $\widetilde{\mathbf{w}}_{t-1} = \Phi_{j-1}\mathbf{U}_{j-1}\boldsymbol{\Sigma}_{j-1}^{-1}\widetilde{\boldsymbol{\omega}}_{t-1}$ starting from $\widetilde{\boldsymbol{\omega}}_{t-1}$ and sets $\widetilde{\boldsymbol{\omega}}_t = \boldsymbol{\Sigma}_j^{-1}\mathbf{U}_j^\intercal\Phi_j^\intercal\widetilde{\mathbf{w}}_{t-1}$. A similar update rule is used to map $\widetilde{\mathbf{A}}_{t-1}$ into the new embedded space without resetting it. B-KONS is a budgeted version of PROS-N-KONS that stops updating the dictionary at a maximum budget $J_{\max}$ and then it continues learning on the last space for the rest of the run. Finally, we also include the best BATCH solution in the final space $\mathcal{H}_J$ returned by KORS as a best-in-hindsight comparator. We also compare to two state-of-the-art embedding-based first-order methods from (Lu et al., 2016). NOGD selects the first J points and uses them to construct an embedding and then perform exact GD in the embedded space. FOGD uses random feature expansion to construct an embedding, and then runs first-order GD in the embedded space. While oblivious embedding methods are cheaper than data-adaptive Nyström, they are usually less accurate. Finally, Dual-SGD also performs a random feature expansion embedding, but in the dual space. We do not run Sketched-KONS because the T^2 runtime is prohibitive in large datasets.

Experimental setup. We replicate the experimental setting in (Lu et al., 2016) with 9 datasets for regression and 3 datasets for binary classification. We preprocess the points $\mathbf{x}_t$ by rescaling each feature in $[0, 1]$. For regression the target variable y_t is rescaled in $[0, 1]$, while in binary classification the labels are $\{-1, 1\}$. For all datasets, we set $\beta = 1$, $\varepsilon = 0.5$ for all PROS-N-KONS variants and $J_{\max} = 100$ for B-KONS. We use the same Gaussian kernel with bandwidth set to 8. For each algorithm and dataset, we report average and standard deviation of the losses, the number of SVs stored in the predictor (as a proxy for space complexity) and runtime in seconds. The scores for the competitor baselines are reported as provided in the original papers (Le et al., 2016; Lu et al., 2016). We only report scores for NOGD, FOGD and Dual-SGD, since they have been shown to outperform other baselines such as Budgeted Perceptron (Cavallanti et al., 2007), Projectron (Orabona et al., 2008), Forgetron (Dekel et al., 2008) and Budgeted-GD (Zhao et al., 2012). All experiments are run on a single machine with 2 Xeon E5-2630 CPUs for a total of 10 cores, and are averaged over 15 runs.

Effective dimension and runtime. We use size of the dictionary returned by KORS as a proxy for the effective dimension of the datasets. As expected, larger datasets and datasets with a larger input dimension have a larger effective dimension. Furthermore, $d_{\text{eff}}^T(\gamma)$ increasing (sublinearly) when γ is reduced from 1 to 0.01 in the *ijcnn1* dataset. More importantly, $d_{\text{eff}}^T(\gamma)$ remains empirically small even for datasets with hundreds of thousands samples, such as *year*, *ijcnn1* and *cod-rna*. On the other hand, in the *slice*

Algorithm	parkinson $n = 5,875$, $d = 20$			cpusmall $n = 8,192$, $d = 12$		
	Avg. Squared Loss	#SV	Time	Avg. Squared Loss	#SV	Time
FOGD	**0.04909** ± 0.00020	30	—	0.02577 ± 0.00050	30	—
NOGD	**0.04896** ± 0.00068	30	—	0.02559 ± 0.00024	30	—
PROS-N-KONS	0.05798 ± 0.00136	18	5.16	0.02494 ± 0.00141	20	7.28
Con-KONS	0.05696 ± 0.00129	18	5.21	0.02269 ± 0.00164	20	7.40
B-KONS	0.05795 ± 0.00172	18	5.35	0.02496 ± 0.00177	20	7.37
BATCH	0.04535 ± 0.00002	—	—	0.01090 ± 0.00082	—	

Algorithm	cadata $n = 20,640$, $d = 8$			casp $n = 45,730$, $d = 9$		
	Avg. Squared Loss	#SV	Time	Avg. Squared Loss	#SV	Time
FOGD	0.04097 ± 0.00015	30	—	0.08021 ± 0.00031	30	—
NOGD	0.03983 ± 0.00018	30	—	0.07844 ± 0.00008	30	—
PROS-N-KONS	0.03095 ± 0.00110	20	18.59	**0.06773** ± 0.00105	21	40.73
Con-KONS	**0.02850** ± 0.00174	19	18.45	**0.06832** ± 0.00315	20	40.91
B-KONS	0.03095 ± 0.00118	19	18.65	**0.06775** ± 0.00067	21	41.13
BATCH	0.02202 ± 0.00002	—	—	0.06100 ± 0.00003	—	—

Algorithm	slice $n = 53,500$, $d = 385$			year $n = 463,715$, $d = 90$		
	Avg. Squared Loss	#SV	Time	Avg. Squared Loss	#SV	Time
FOGD	0.00726 ± 0.00019	30	—	0.01427 ± 0.00004	30	—
NOGD	0.02636 ± 0.00460	30	—	0.01427 ± 0.00004	30	—
Dual-SGD	—	—	—	0.01440 ± 0.00000	100	—
PROS-N-KONS	did not complete	—	—	0.01450 ± 0.00014	149	884.82
Con-KONS	did not complete	—	—	0.01444 ± 0.00017	147	889.42
B-KONS	**0.00913** ± 0.00045	100	60	**0.01302** ± 0.00006	100	505.36
BATCH	0.00212 ± 0.00001	—	—	0.01147 ± 0.00001	—	—

Table 5.1: Regression datasets

dataset, the effective dimension is too large for PROS-N-KONS to complete and only results for B-KONS are provided. Overall, the proposed algorithm can process hundreds of thousands of points in a matter of minutes, and show that it can practically scale to large datasets.

Regression. All algorithms are trained and evaluated using the squared loss. Notice that whenever the budget $J_{\max}$ is not exceeded, B-KONS and PROS-N-KONS are the same algorithm and obtain the same result. On regression datasets (Tab. 5.1) we set $\alpha = 1$ and $\gamma = 1$, which satisfies the requirements of Thm. 5.9. On smaller datasets such as *parkinson* and *cpusmall*, where frequent restarts greatly interfere with the gradient descent, and even a small non-adaptive embedding can capture the geometry of the data, PROS-N-KONS is outperformed by simpler first-order methods. As soon as T reaches the order of tens of thousands (*cadata*, *casp*) second-order updates and data adaptivity becomes relevant and PROS-N-KONS outperform its competitors, both in the number of SVs and in the average loss. In this intermediate regime, Con-KONS outperforms PROS-N-KONS and B-KONS since it is not as much affected by restarts. Finally, when the number of samples raises to hundreds of thousands, the intrinsic effective dimension of the dataset starts playing a larger role. On *slice*, where the effective dimension is too large to run, B-KONS still outperforms NOGD with a comparable budget of SVs, showing the advantage of second-order updates.

$\alpha = 1$, $\gamma = 1$						
Algorithm	ijcnn1 $n = 141,691$, $d = 22$			cod-rna $n = 271,617$, $d = 8$		
	Accuracy	#SV	Time	Accuracy	#SV	Time
FOGD	9.06 ± 0.05	400	—	10.30 ± 0.10	400	—
NOGD	9.55 ± 0.01	100	—	13.80 ± 2.10	100	—
DUAL-SGD	**8.35** ± 0.20	100	—	**4.83** ± 0.21	100	—
PROS-N-KONS	9.70 ± 0.01	100	211.91	13.95 ± 1.19	38	270.81
CON-KONS	9.64 ± 0.01	101	215.71	18.99 ± 9.47	38	271.85
B-KONS	9.70 ± 0.01	98	206.53	13.99 ± 1.16	38	274.94
BATCH	8.33 ± 0.03	—	—	3.781 ± 0.01	—	—

$\alpha = 0.01$, $\gamma = 0.01$						
Algorithm	ijcnn1 $n = 141,691$, $d = 22$			cod-rna $n = 271,617$, $d = 8$		
	Accuracy	#SV	Time	Accuracy	#SV	Time
FOGD	9.06 ± 0.05	400	—	10.30 ± 0.10	400	—
NOGD	9.55 ± 0.01	100	—	13.80 ± 2.10	100	—
DUAL-SGD	8.35 ± 0.20	100	—	4.83 ± 0.21	100	—
PROS-N-KONS	10.73 ± 0.12	436	1003.82	4.91 ± 0.04	111	459.28
CON-KONS	6.23 ± 0.18	432	987.33	5.81 ± 1.96	111	458.90
B-KONS	**4.85** ± 0.08	100	147.22	**4.57** ± 0.05	100	333.57
BATCH	5.61 ± 0.01	—	—	3.61 ± 0.01	—	—

Table 5.2: Binary classification datasets

Binary classification All algorithms are trained using the hinge loss and they are evaluated using the average online error rate. Results are reported in Table 5.2. While for regression an arbitrary value of $\gamma = \alpha = 1$ is sufficient to obtain good results, it fails for binary classification. Decreasing the two parameters to 0.01 resulted in a 3-fold increase in the number of SVs included and runtime, but almost a 2-fold decrease in error rate, placing PROS-N-KONS and B-KONS on par or ahead of competitors without the need of any further parameter tuning.

5.3.3 Recap and open questions: do we need restarts?

We introduced SKETCHED-KONS and PROS-N-KONS, two approximate second-order OKL algorithms.
SKETCHED-KONS is the first *approximate* second-order algorithm that can provably achieve logarithmic regret with high probability, matching exact methods. Unfortunately, it only provides constant speed-ups compared to the exact method, and remains affected by the curse of kernelization, requiring $\mathcal{O}(T^2)$ overall runtime.

PROS-N-KONS breaks this quadratic barrier, and is the first second-order OKL algorithm that guarantees both logarithmic regret and avoids the curse of kernelization, taking only a near-constant $\widetilde{\mathcal{O}}(d_{\text{eff}}^T(\gamma)^2)$ per-step time and space cost.

PROS-N-KONS can also be seen as an adaptive *doubling* strategy. In online learning, doubling strategies are common (Cesa-Bianchi and Lugosi, 2006) when we need to deal with parameters unknown in advance. Informally, the doubling trick corresponds to breaking up our learning process in phases, each twice as long as the previous.
In PROS-N-KONS case, we do not know which is the right RKHS $\mathcal{H}_j$ that we should

consider for our descent (using the exact $\mathcal{H}$ is too expensive and we do not consider it). Therefore, we break down our learning process in phases dictated by KORS.
If the adversary keeps playing points that are already in our dictionary, this phase can be indefinitely long, but our approximation is accurate and we pay logarithmic regret. If instead the adversary plays multiple points along a new direction, KORS will quickly (i.e., in logarithmic time) notice and include it in the dictionary, starting a new phase. Overall, the adversary cannot keep us playing in the wrong $\mathcal{H}_j$ for more than a logarithmic number of rounds in each phase.
Ultimately, either the effective dimension of the problem is small, and we will quickly enter a final phase in which $\mathcal{H}_j$ never changes, or the effective dimension is large and even exact second-order methods would have a large regret.

Nonetheless, doubling tricks are often a last resort in online learning, as they are wasteful. And since we know that $\mathcal{H}_j$ and $\mathcal{H}_{j+1}$ are close (they differ in a single atom), it remains an important open question if we can transfer the solution of a phase into the next one without paying a large regret. Empirically, we see that Con-KONS and B-KONS often outperform PROS-N-KONS, which suggest that this might be the case, at least in the simpler, non-adversarial setting of our experiments.

5.4 Extended proofs

Proof of Lemma 5.1. We begin by applying the definition of $\mathbf{u}_{t+1}$ and collecting $\mathbf{A}_t^{-1}$, which can always be done since, for $\alpha > 0$, $\mathbf{A}_t$ is invertible,

$$\mathbf{u}_{t+1} = \mathbf{w}_t - \mathbf{A}_t^{-1}\mathbf{g}_t = \mathbf{A}_t^{-1}(\mathbf{A}_t\mathbf{w}_t - \mathbf{g}_t).$$

We focus now on the last term and use the definition of $\mathbf{A}_t$,

$$\begin{aligned}\mathbf{A}_t\mathbf{w}_t - \mathbf{g}_t &= \mathbf{A}_{t-1}\mathbf{w}_t + \eta_t\mathbf{g}_t\mathbf{g}_t^\intercal\mathbf{w}_t - \mathbf{g}_t \\ &= \mathbf{A}_{t-1}\mathbf{u}_t - \mathbf{A}_{t-1}\mathbf{r}_t + (\sqrt{\eta_t}\mathbf{g}_t^\intercal\mathbf{w}_t - 1/\sqrt{\eta_t})\overline{\phi}_t.\end{aligned}$$

Looking at $\mathbf{A}_{t-1}\mathbf{r}_t$ and using the assumption $\dot{g}_t \neq 0$,

$$\mathbf{A}_{t-1}\mathbf{r}_t = \frac{h(\phi_t^\intercal\mathbf{u}_t)}{\phi_t^\intercal\mathbf{A}_{t-1}^{-1}\phi_t}\mathbf{A}_{t-1}\mathbf{A}_{t-1}^{-1}\phi_t = \frac{h(\phi_t^\intercal\mathbf{u}_t)}{\phi_t^\intercal\mathbf{A}_{t-1}^{-1}\phi_t}\frac{\dot{g}_t^2\eta_t}{\dot{g}_t^2\eta_t}\phi_t = \frac{\dot{g}_t\sqrt{\eta_t}h(\phi_t^\intercal\mathbf{u}_t)}{\overline{\phi}_t^\intercal\mathbf{A}_{t-1}^{-1}\overline{\phi}_t}\overline{\phi}_t.$$

Putting together all three terms, and using the fact that $\mathbf{g}_t^\intercal\mathbf{w}_t = \dot{g}_t\phi_t\mathbf{w}_t = \dot{g}_t\widehat{y}_t$ and denoting $b_t = [\mathbf{b}_t]_t$ we have

$$\begin{aligned}\mathbf{u}_{t+1} &= \mathbf{A}_t^{-1}(\mathbf{A}_t\mathbf{w}_t - \mathbf{g}_t) \\ &= \mathbf{A}_t^{-1}(\mathbf{A}_{t-1}\mathbf{u}_t + b_t\overline{\phi}_t) \\ &= \mathbf{A}_t^{-1}(\mathbf{A}_{t-1}(\mathbf{w}_{t-1} - \mathbf{A}_{t-1}^{-1}\mathbf{g}_{t-1}) + b_t\overline{\phi}_t) \\ &= \mathbf{A}_t^{-1}(\mathbf{A}_{t-1}\mathbf{w}_{t-1} - \mathbf{g}_{t-1} + b_t\overline{\phi}_t) \\ &= \mathbf{A}_t^{-1}(\mathbf{A}_{t-2}\mathbf{w}_{t-2} - \mathbf{g}_{t-2} + b_{t-1}\overline{\phi}_{t-1} + b_t\overline{\phi}_t) \\ &= \mathbf{A}_t^{-1}(\mathbf{A}_0\mathbf{w}_0 + \sum\nolimits_{s=1}^{t} b_s\overline{\phi}_s).\end{aligned}$$

■

Proof of Lemma 5.2. Throughout this proof, we make use of the linear algebra identity from Proposition 2.12. We begin with the reformulation of $[\mathbf{b}_t]_t$. In particular, the only term that we need to reformulate is

$$\begin{aligned}
&\overline{\phi}_t \mathbf{A}_{t-1}^{-1} \overline{\phi}_t = \overline{\phi}_t (\overline{\Phi}_{t-1} \overline{\Phi}_{t-1}^\intercal + \alpha \mathbf{\Pi}_T)^{-1} \overline{\phi}_t \\
&= \frac{1}{\alpha} \overline{\phi}_t (\mathbf{\Pi}_T - \overline{\Phi}_{t-1} (\overline{\Phi}_{t-1}^\intercal \overline{\Phi}_{t-1} + \alpha \mathbf{\Pi}_T)^{-1} \overline{\Phi}_{t-1}^\intercal) \overline{\phi}_t \\
&= \frac{1}{\alpha} (\overline{\phi}_t^\intercal \overline{\phi}_t - \overline{\phi}_t^\intercal \overline{\Phi}_{t-1} (\overline{\Phi}_{t-1}^\intercal \overline{\Phi}_{t-1} + \alpha \mathbf{I}_{t-1})^{-1} \overline{\Phi}_{t-1}^\intercal \overline{\phi}_t) \\
&= \frac{1}{\alpha} (k_{t,t} - \overline{\mathbf{k}}_{[t-1],t}^\intercal (\overline{\mathbf{K}}_{t-1} + \alpha \mathbf{I}_{t-1})^{-1} \overline{\mathbf{k}}_{[t-1],t}).
\end{aligned}$$

For $\overline{y}_t$, we have

$$\begin{aligned}
\overline{y}_t &= \phi_t^\intercal \mathbf{u}_t = \phi_t^\intercal \mathbf{A}_{t-1}^{-1} \overline{\Phi}_{t-1} \mathbf{b}_{t-1} \\
&= \phi_t^\intercal (\overline{\Phi}_{t-1} \overline{\Phi}_{t-1}^\intercal + \alpha \mathbf{\Pi}_T)^{-1} \overline{\Phi}_{t-1} \mathbf{b}_{t-1} \\
&= \frac{1}{\alpha} \phi_t^\intercal (\mathbf{\Pi}_T - \overline{\Phi}_{t-1} (\overline{\Phi}_{t-1}^\intercal \overline{\Phi}_{t-1} + \alpha \mathbf{\Pi}_T)^{-1} \overline{\Phi}_{t-1}^\intercal) \overline{\Phi}_{t-1} \mathbf{b}_{t-1} \\
&= \frac{1}{\alpha} \phi_t^\intercal \Phi_{t-1} \mathbf{D}_{t-1} (\mathbf{b}_{t-1} - (\overline{\mathbf{K}}_{t-1} + \alpha \mathbf{I}_{t-1})^{-1} \overline{\mathbf{K}}_{t-1} \mathbf{b}_{t-1}) \\
&= \frac{1}{\alpha} \mathbf{k}_{[t-1],t}^\intercal \mathbf{D}_{t-1} (\mathbf{b}_{t-1} - (\overline{\mathbf{K}}_{t-1} + \alpha \mathbf{I}_{t-1})^{-1} \overline{\mathbf{K}}_{t-1} \mathbf{b}_{t-1}).
\end{aligned}$$

■

Proof of Theorem 5.4. We need to bound $R_T(\mathbf{w}^*)$, and we use Proposition 5.3. For R_D nothing changes from the parametric case, and we use Assumption 1 and the definition of the set $\mathcal{S}$ to bound

$$\begin{aligned}
R_D &= \sum_{t=1}^T (\eta_t - \sigma_t) \dot{g}_t^2 (\phi_t^\intercal (\mathbf{w}_t - \mathbf{w}))^2 \\
&\leq \sum_{t=1}^T (\eta_t - \sigma) L^2 (|\phi_t^\intercal \mathbf{w}_t| + |\phi_t^\intercal \mathbf{w}|)^2 \leq 4L^2 C^2 \sum_{t=1}^T (\eta_t - \sigma).
\end{aligned}$$

For R_G, we reformulate

$$\begin{aligned}
\sum_{t=1}^T \mathbf{g}_t^\intercal \mathbf{A}_t^{-1} \mathbf{g}_t &= \sum_{t=1}^T \frac{\eta_t}{\eta_t} \mathbf{g}_t^\intercal \mathbf{A}_t^{-1} \mathbf{g}_t = \sum_{t=1}^T \frac{1}{\eta_t} \overline{\phi}_t^\intercal \mathbf{A}_t^{-1} \overline{\phi}_t \leq \frac{1}{\eta_T} \sum_{t=1}^T \overline{\phi}_t^\intercal \mathbf{A}_t^{-1} \overline{\phi}_t \\
&= \frac{1}{\eta_T} \sum_{t=1}^T \overline{\tau}_{t,t} = \overline{d}_{\text{onl}}(\alpha)/\eta_T \leq \frac{\overline{d}_{\text{eff}}^T(\alpha)}{\eta_T} (1 + \log(\|\overline{\mathbf{K}}_T\|/\alpha + 1)),
\end{aligned}$$

where $\overline{d}_{\text{eff}}^T(\alpha)$ and $\overline{\mathbf{K}}_T$ are computed using the rescaled kernel $\overline{\mathcal{K}}$.

Let us remind ourselves the definition $\mathbf{D} = \text{Diag}\left(\{\dot{g}_t \sqrt{\eta_t}\}_{t=1}^T\right)$. Since $\eta_t \neq 0$ and $\dot{g}_t \neq 0$ for all t, $\mathbf{D}$ is invertible and we have $\lambda_{\min}(\mathbf{D}^{-2}) = \min_{t=1}^T 1/(\dot{g}_t^2 \eta_t) \geq 1/(L^2 \eta_1)$. For simplicity, we assume $\eta_1 = \sigma$, leaving the case $\eta_1 = 1/1 = 1$ as a special case. We derive

$$\begin{aligned}
\overline{d}_{\text{eff}}^T(\alpha) &= \text{Tr}(\overline{\mathbf{K}}_T (\overline{\mathbf{K}}_T + \alpha \mathbf{I}_T)^{-1}) = \text{Tr}(\mathbf{D}\mathbf{K}_T\mathbf{D}(\mathbf{D}\mathbf{K}_T\mathbf{D} + \alpha \mathbf{D}\mathbf{D}^{-2}\mathbf{D})^{-1}) \\
&= \text{Tr}(\mathbf{D}\mathbf{K}_T\mathbf{D}\mathbf{D}^{-1}(\mathbf{K}_T + \alpha \mathbf{D}^{-2})^{-1}\mathbf{D}^{-1}) = \text{Tr}(\mathbf{K}_T \mathbf{I}_T (\mathbf{K}_T + \alpha \mathbf{D}^{-2})^{-1} \mathbf{D}^{-1} \mathbf{D}) \\
&= \text{Tr}(\mathbf{K}_T (\mathbf{K}_T + \alpha \mathbf{D}^{-2})^{-1}) \leq \text{Tr}(\mathbf{K}_T (\mathbf{K}_T + \alpha \lambda_{\min}(\mathbf{D}^{-2}) \mathbf{I}_T)^{-1}) \\
&\leq \text{Tr}\left(\mathbf{K}_T \left(\mathbf{K}_T + \frac{\alpha}{\sigma L^2} \mathbf{I}_T\right)^{-1}\right) = d_{\text{eff}}^T\left(\alpha/(\sigma L^2)\right).
\end{aligned}$$

Similarly,

$$\begin{aligned}\log(\|\overline{\mathbf{K}}_T\|/\alpha+1) &\leq \log(\operatorname{Tr}(\overline{\mathbf{K}}_T)/\alpha+1) \leq \log(\sigma L^2 \operatorname{Tr}(\mathbf{K}_t)/\alpha+1)\\ &\leq \log(\sigma L^2 T/\alpha+1) \leq \log(2\sigma L^2 T/\alpha),\end{aligned}$$

since $\operatorname{Tr}(\mathbf{K}_t) = \sum_{t=1}^T k_{t,t} = \sum_{t=1}^T \phi_t^\intercal \phi_t \leq \sum_{t=1}^T 1 = T$. ■

Proof of Lemma 5.5. Through this proof, we make use of the linear algebra identity from Proposition 2.12. We begin with the reformulation of $\widetilde{b}_i = [\widetilde{\mathbf{b}}_t]_i$. In particular, the only term that we need to reformulate is

$$\begin{aligned}\overline{\phi}_t \widetilde{\mathbf{A}}_{t-1}^{-1} \overline{\phi}_t &= \overline{\phi}_t (\overline{\Phi}_{t-1}\mathbf{R}_{t-1}\mathbf{R}_{t-1}^\intercal \overline{\Phi}_{t-1}^\intercal + \alpha \mathbf{\Pi}_T)^{-1} \overline{\phi}_t\\ &= \frac{1}{\alpha}\overline{\phi}_t(\mathbf{\Pi}_T - \overline{\Phi}_{t-1}\mathbf{R}_{t-1}(\mathbf{R}_{t-1}^\intercal \overline{\Phi}_{t-1}^\intercal \overline{\Phi}_{t-1}\mathbf{R}_{t-1} + \alpha\mathbf{\Pi}_T)^{-1}\mathbf{R}_{t-1}^\intercal \overline{\Phi}_{t-1}^\intercal)\overline{\phi}_t\\ &= \frac{1}{\alpha}(\overline{\phi}_t^\intercal \overline{\phi}_t - \overline{\phi}_t^\intercal \overline{\Phi}_{t-1}\mathbf{R}_{t-1}(\mathbf{R}_{t-1}^\intercal \overline{\Phi}_{t-1}^\intercal \overline{\Phi}_{t-1}\mathbf{R}_{t-1} + \alpha\mathbf{I}_{t-1})^{-1}\mathbf{R}_{t-1}^\intercal \overline{\Phi}_{t-1}^\intercal \overline{\phi}_t)\\ &= \frac{1}{\alpha}(k_{t,t} - \overline{\mathbf{k}}_{[t-1],t}^\intercal \mathbf{R}_{t-1}(\mathbf{R}_{t-1}^\intercal \overline{\mathbf{K}}_{t-1}\mathbf{R}_{t-1} + \alpha\mathbf{I}_{t-1})^{-1}\mathbf{R}_{t-1}^\intercal \overline{\mathbf{k}}_{[t-1],t}).\end{aligned}$$

For $\breve{y}_t$, we have

$$\begin{aligned}\breve{y}_t &= \phi_t^\intercal \widetilde{\mathbf{u}}_t = \phi_t^\intercal \widetilde{\mathbf{A}}_{t-1}^{-1}\overline{\Phi}_{t-1}\widetilde{\mathbf{b}}_{t-1}\\ &= \phi_t^\intercal (\overline{\Phi}_{t-1}\mathbf{R}_{t-1}\mathbf{R}_{t-1}^\intercal \overline{\Phi}_{t-1}^\intercal + \alpha\mathbf{\Pi}_T)^{-1}\overline{\Phi}_{t-1}\widetilde{\mathbf{b}}_{t-1}\\ &= \frac{1}{\alpha}\phi_t^\intercal \left(\mathbf{\Pi}_T - \overline{\Phi}_{t-1}\mathbf{R}_{t-1}(\mathbf{R}_{t-1}^\intercal \overline{\Phi}_{t-1}^\intercal \overline{\Phi}_{t-1}\mathbf{R}_{t-1} + \alpha\mathbf{\Pi}_T)^{-1}\mathbf{R}_{t-1}^\intercal \overline{\Phi}_{t-1}^\intercal\right)\overline{\Phi}_{t-1}\widetilde{\mathbf{b}}_{t-1}\\ &= \frac{1}{\alpha}\phi_t^\intercal \Phi_{t-1}\mathbf{D}_{t-1}\left(\widetilde{\mathbf{b}}_{t-1} - \mathbf{R}_{t-1}(\mathbf{R}_{t-1}^\intercal \overline{\mathbf{K}}_{t-1}\mathbf{R}_{t-1} + \alpha\mathbf{I}_{t-1})^{-1}\mathbf{R}_{t-1}^\intercal \overline{\mathbf{K}}_{t-1}\widetilde{\mathbf{b}}_{t-1}\right)\\ &= \frac{1}{\alpha}\mathbf{k}_{[t-1],t}^\intercal \mathbf{D}_{t-1}\left(\widetilde{\mathbf{b}}_{t-1} - \mathbf{R}_{t-1}(\mathbf{R}_{t-1}^\intercal \overline{\mathbf{K}}_{t-1}\mathbf{R}_{t-1} + \alpha\mathbf{I}_{t-1})^{-1}\mathbf{R}_{t-1}^\intercal \overline{\mathbf{K}}_{t-1}\widetilde{\mathbf{b}}_{t-1}\right).\end{aligned}$$

■

Proof of Theorem 5.7. Since the only thing that changed is the formulation of the $\mathbf{A}_t$ matrix, the bound from Proposition 5.3 still applies. In particular, we have that the regret $\widetilde{R}_T$ of Algorithm 13 is bounded as

$$\widetilde{R}(\mathbf{w}) \leq \alpha\|\mathbf{w}\|_{\mathbf{A}_0}^2 + \sum_{t=1}^T \mathbf{g}_t^\intercal \widetilde{\mathbf{A}}_t^{-1}\mathbf{g}_t + \sum_{t=1}^T (\mathbf{w}_t - \mathbf{w})^\intercal (\widetilde{\mathbf{A}}_t - \widetilde{\mathbf{A}}_{t-1} - \sigma_t \mathbf{g}_t\mathbf{g}_t^\intercal)(\mathbf{w}_t - \mathbf{w}).$$

From Theorem 3.11, we have that KORS succeeds with high probability. In particular, using the guarantees of the (ε,γ)-accuracy (1), we can bound for the case $\eta_t = \sigma$ as

$$\begin{aligned}\mathbf{g}_t^\intercal \widetilde{\mathbf{A}}_t^{-1}\mathbf{g}_t &= \frac{\eta_t}{\eta_t}\mathbf{g}_t^\intercal (\overline{\Phi}_t\mathbf{R}_t\mathbf{R}_t^\intercal \overline{\Phi}_t^\intercal + \alpha\mathbf{\Pi}_T)^{-1}\mathbf{g}_t = \frac{1}{\eta_t}\overline{\phi}_t^\intercal (\overline{\Phi}_t\mathbf{R}_t\mathbf{R}_t^\intercal \overline{\Phi}_t^\intercal + \alpha\mathbf{\Pi}_T)^{-1}\overline{\phi}_t\\ &= \frac{1}{\eta_t}\frac{\widetilde{p}_{\min}}{\widetilde{p}_{\min}}\overline{\phi}_t^\intercal (\overline{\Phi}_t\mathbf{R}_t\mathbf{R}_t^\intercal \overline{\Phi}_t^\intercal + \alpha\mathbf{\Pi}_T)^{-1}\overline{\phi}_t = \frac{1}{\eta_t}\frac{1}{\widetilde{p}_{\min}}\overline{\phi}_t^\intercal \left(\frac{1}{\widetilde{p}_{\min}}\overline{\Phi}_t\mathbf{R}_t\mathbf{R}_t^\intercal \overline{\Phi}_t^\intercal + \alpha\mathbf{\Pi}_T\right)^{-1}\overline{\phi}_t\\ &\leq \frac{1}{\eta_t}\frac{1}{\widetilde{p}_{\min}}\overline{\phi}_t^\intercal \left(\overline{\Phi}_t\mathbf{S}_t\mathbf{S}_t^\intercal \overline{\Phi}_t^\intercal + \alpha\mathbf{\Pi}_T\right)^{-1}\overline{\phi}_t\\ &\leq \frac{1}{\widetilde{p}_{\min}\eta_t}\overline{\phi}_t^\intercal ((1-\varepsilon)\overline{\Phi}_t\overline{\Phi}_t^\intercal - \varepsilon\alpha\mathbf{\Pi}_T + \alpha\mathbf{\Pi}_T)^{-1}\overline{\phi}_t\\ &= \frac{1}{(1-\varepsilon)\sigma\widetilde{p}_{\min}}\overline{\phi}_t^\intercal (\overline{\Phi}_t\overline{\Phi}_t^\intercal + \alpha\mathbf{\Pi}_T)^{-1}\overline{\phi}_t = \frac{\overline{\tau}_{t,t}}{(1-\varepsilon)\sigma\widetilde{p}_{\min}},\end{aligned}$$

where in the first inequality we used the fact that the weight matrix $\mathbf{S}_t$ contains weights such that $1/\sqrt{\widetilde{p}_{\min}} \geq 1/\sqrt{\widetilde{p}_t}$, in the second inequality we used the (ε, γ)-accuracy, and finally, we used $\eta_t = \sigma$ and the definition of $\overline{\tau}_{t,t}$. Therefore,

$$R_G = \sum_{t=1}^{T} \mathbf{g}_t^\top \widetilde{\mathbf{A}}_t^{-1} \mathbf{g}_t \leq \frac{1}{(1-\varepsilon)\sigma \widetilde{p}_{\min}} \sum_{t=1}^{T} \overline{\tau}_{t,t} \leq \frac{\overline{d}_{\mathrm{onl}}(\alpha)}{(1-\varepsilon)\sigma \max\{\overline{\tau}_{\min}, \gamma\}}.$$

To bound R_D, we have

$$\begin{aligned}
&\sum_{t=1}^{T} (\mathbf{w}_t - \mathbf{w})^\top (\widetilde{\mathbf{A}}_t - \widetilde{\mathbf{A}}_{t-1} - \sigma_t \mathbf{g}_t \mathbf{g}_t^\top)(\mathbf{w}_t - \mathbf{w}) \\
&= \sum_{t=1}^{T} (\mathbf{w}_t - \mathbf{w})^\top \left(\eta_t \frac{q_s}{\overline{q}} \mathbf{g}_t \mathbf{g}_t^\top - \sigma_t \mathbf{g}_t \mathbf{g}_t^\top \right) (\mathbf{w}_t - \mathbf{w}) \\
&\leq \sum_{t=1}^{T} (\sigma - \sigma_t)(\mathbf{g}_t^\top (\mathbf{w}_t - \mathbf{w}))^2 \leq 0.
\end{aligned}$$

■

Proof of Theorem 5.8. PROS-N-KONS predicts $\widetilde{y}_t$ in round t. We want to bound the cumulative regret of the loss of $\widetilde{y}_t$ with respect to an arbitrary fixed vector $\mathbf{w} \in \mathcal{H}$ on the (mapped) points ϕ_t. From the definition of $\widetilde{y}_t$ in Algorithm 14, we have

$$\begin{aligned}
\sum_{t=1}^{T} \ell_t(\widetilde{y}_t) - \ell_t(\phi_t^\top \mathbf{w}) &= \sum_{t=1}^{T} \ell_t\left(\widetilde{\phi}_t^\top \widetilde{\boldsymbol{\omega}}_t\right) - \ell_t(\phi_t^\top \mathbf{w}) \\
&= \sum_{t=1}^{T} \ell_t\left(\phi_t^\top \Phi_j \mathbf{U}_j \boldsymbol{\Sigma}_j^{-1} \widetilde{\boldsymbol{\omega}}_t\right) - \ell_t(\phi_t^\top \mathbf{w}) \\
&= \sum_{t=1}^{T} \ell_t(\phi_t^\top \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t^\top \mathbf{w}),
\end{aligned}$$

with $\widetilde{\mathbf{w}}_t = \Phi_j \mathbf{U}_j \boldsymbol{\Sigma}_j^{-1} \widetilde{\boldsymbol{\omega}}_t$. From Assumption 2 on the losses, we know that the losses $\ell_t(z)$ satisfy

$$\ell_t(\phi^\top \mathbf{w}) \geq \ell_t(\phi^\top \mathbf{u}) + \nabla \ell_t(\phi^\top \mathbf{u})^\top (\mathbf{w} - \mathbf{u}) + \frac{\sigma_t}{2} \left(\nabla \ell_t(\phi^\top \mathbf{u})^\top (\mathbf{w} - \mathbf{u})\right)^2$$

and therefore

$$\begin{aligned}
\ell_t(\phi^\top \mathbf{u}) - \ell_t(\phi^\top \mathbf{w}) &\leq \nabla \ell_t(\phi^\top \mathbf{u})^\top (\mathbf{u} - \mathbf{w}) - \frac{\sigma_t}{2} \left(\nabla \ell_t(\phi^\top \mathbf{u})^\top (\mathbf{w} - \mathbf{u})\right)^2 \\
&= \ell_t'(\phi^\top \mathbf{u})\, \phi_t^\top (\mathbf{u} - \mathbf{w}) - \frac{\sigma_t}{2} \left(\ell_t'(\phi^\top \mathbf{u})\, \phi_t^\top (\mathbf{w} - \mathbf{u})\right)^2.
\end{aligned}$$

Definition of the epochs

We define the epochs $j \in [J]$ that are separated by $q_t \neq 0$, which indicate the dictionary change. Formally $q_{t_j} \neq 0$, and $q_{t'} = 0$ for all $t_j < t' < t_j + h_j$ where $h_j = t_{j+1} - t_j$ is the length of the epoch. We also use $\mathcal{I}_j$ for the dictionary in the j-th phase, $\Phi_{\mathcal{I}_j}$ for the feature matrix containing samples $\phi_i \in \mathcal{I}_j$, and $\boldsymbol{\Pi}_{\mathcal{I}_j} = \Phi_{\mathcal{I}_j}(\Phi_{\mathcal{I}_j}^\top \Phi_{\mathcal{I}_j})^+ \Phi_{\mathcal{I}_j}^\top$ for the projection matrix on the column span of $\Phi_{\mathcal{I}_j}$. Similarly, given $\mathcal{I}_j$, we define the embedding $\widetilde{\varphi}_j(\cdot)$ such that

$$\widetilde{\varphi}_j(\mathbf{x}_t) = \widetilde{\phi}_t = \boldsymbol{\Sigma}_j^{-1} \mathbf{U}_j^\top \Phi_{\mathcal{I}_j}^\top \phi_t \in \mathbb{R}^j$$

and $\widetilde{\phi}_t^\top \widetilde{\phi}_{t'} = \phi_t \boldsymbol{\Pi}_{\mathcal{I}_j} \phi_t'$. Given the embedding $\widetilde{\varphi}_j$, we introduce the *restricted* RKHS $\mathcal{H}_j$ and the approximate kernel function $\widetilde{\mathcal{K}}_j(\cdot,\cdot)$. Note that although the mapping $\widetilde{\varphi}_j$ changes across different epochs, it is *unique* for a fixed round t, therefore we simply write $\widetilde{\phi}_t$ instead of the more explicit $\widetilde{\phi}_t^j$.

Regret decomposition

We introduce two intermediate comparators: $\overline{\mathbf{w}}_t$ and $\widehat{\mathbf{w}}_t$. First, $\widehat{\mathbf{w}}_j$ is the best fixed solution within the epoch for the dictionary of the epoch and the related $\mathcal{H}_j$. Second, $\widehat{\mathbf{w}}_j$ is the best fixed solution within the epoch over the whole space $\mathcal{H}$. Notice, neither $\widehat{\mathbf{w}}_j$ nor $\widehat{\mathbf{w}}_j$ change *during the same epoch*. Formally, for all t' such that $t_j < t' \leq t_j + h_j$ we have $\overline{\mathbf{w}}_{t'} = \overline{\mathbf{w}}_j$ and $\widehat{\mathbf{w}}_{t'} = \widehat{\mathbf{w}}_j$, defined as

$$\overline{\mathbf{w}}_j = \operatorname*{arg\,min}_{\mathbf{w}\in\mathcal{H}_j} \sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\intercal \mathbf{w}) + \alpha\|\mathbf{w}\|^2, \quad \text{and} \quad \widehat{\mathbf{w}}_j = \operatorname*{arg\,min}_{\mathbf{w}\in\mathcal{H}} \sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\intercal \mathbf{w}) + \alpha\|\mathbf{w}\|^2.$$

Using the above intermediate comparators now split the regret into three parts, one per epoch,

$$\begin{aligned}
&\sum_{t=1}^{T} \ell_t(\phi_t^\intercal \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t^\intercal \mathbf{w}) \\
&\quad = \sum_{t=1}^{T} \ell_t(\phi_t^\intercal \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t^\intercal \overline{\mathbf{w}}_t) + \ell_t(\phi_t^\intercal \overline{\mathbf{w}}_t) - \ell_t(\phi_t^\intercal \widehat{\mathbf{w}}_t) + \ell_t(\phi_t^\intercal \widehat{\mathbf{w}}_t) - \ell_t(\phi_t^\intercal \mathbf{w}) \\
&= \sum_{j=1}^{J} \sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\intercal \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t^\intercal \overline{\mathbf{w}}_t) + \ell_t(\phi_t^\intercal \overline{\mathbf{w}}_t) - \ell_t(\phi_t^\intercal \widehat{\mathbf{w}}_t) + \ell_t(\phi_t^\intercal \widehat{\mathbf{w}}_t) - \ell_t(\phi_t^\intercal \mathbf{w}) \\
&= \sum_{j=1}^{J} \sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\intercal \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t^\intercal \overline{\mathbf{w}}_j) + \ell_t(\phi_t^\intercal \overline{\mathbf{w}}_j) - \ell_t(\phi_t^\intercal \widehat{\mathbf{w}}_j) + \ell_t(\phi_t^\intercal \widehat{\mathbf{w}}_j) - \ell_t(\phi_t^\intercal \mathbf{w}) \\
&= \sum_{j=1}^{J} \underbrace{\left(\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\intercal \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t^\intercal \overline{\mathbf{w}}_j) \right)}_{A_j} + \underbrace{\left(\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\intercal \overline{\mathbf{w}}_j) - \ell_t(\phi_t^\intercal \widehat{\mathbf{w}}_j) \right)}_{B_j} + \underbrace{\left(\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\intercal \widehat{\mathbf{w}}_j) - \ell_t(\phi_t^\intercal \mathbf{w}) \right)}_{C_j}.
\end{aligned}$$

Fix an epoch j. We now bound A_j, B_j, and C_j separately.

Bounding A_j

Since $\overline{\mathbf{w}}_j \in \mathcal{H}_j$, projecting $\overline{\mathbf{w}}_j$ to $\mathcal{H}_j$ won't change it, i.e., $\overline{\mathbf{w}}_j = \mathbf{\Pi}_{\mathcal{I}_j} \overline{\mathbf{w}}_j$ and

$$\phi_t^\intercal \overline{\mathbf{w}}_j = \phi_t^\intercal \mathbf{\Pi}_{\mathcal{I}_j} \overline{\mathbf{w}}_j = \phi_t^\intercal \Phi_{\mathcal{I}_j} \mathbf{U}_j \mathbf{\Sigma}_j^{-1} \mathbf{\Sigma}_j^{-1} \mathbf{U}_j^\intercal \Phi_{\mathcal{I}_j}^\intercal \overline{\mathbf{w}}_j = \widetilde{\phi}_t^\intercal \overline{\boldsymbol{\omega}}_j$$

with $\overline{\boldsymbol{\omega}}_j = \mathbf{\Sigma}_j^{-1} \mathbf{U}_j^\intercal \Phi_{\mathcal{I}_j}^\intercal \overline{\mathbf{w}}_j$. Remembering that $\widetilde{\mathbf{w}}_t = \Phi_{\mathcal{I}_j} \mathbf{U}_j \mathbf{\Sigma}_j^{-1} \widetilde{\boldsymbol{\omega}}_t$ for all t in the epoch

$$\ell_t\left(\phi_t^\intercal \widetilde{\mathbf{w}}_t\right) - \ell_t\left(\phi_t^\intercal \overline{\mathbf{w}}_j\right) = \ell_t\left(\phi_t^\intercal \Phi_{\mathcal{I}_j} \mathbf{U}_j \mathbf{\Sigma}_j^{-1} \widetilde{\boldsymbol{\omega}}_t\right) - \ell_t\left(\phi_t^\intercal \mathbf{\Pi}_{\mathcal{I}_j} \overline{\mathbf{w}}_j\right) = \ell_t\left(\widetilde{\phi}_t^\intercal \widetilde{\boldsymbol{\omega}}_t\right) - \ell_t\left(\widetilde{\phi}_t^\intercal \overline{\boldsymbol{\omega}}_j\right).$$

Now, using Assumption 2, we get

$$\ell_t(\widetilde{\phi}_t^\intercal \widetilde{\boldsymbol{\omega}}_t) - \ell_t(\widetilde{\phi}_t^\intercal \overline{\boldsymbol{\omega}}_j) \leq \widetilde{\mathbf{g}}_t^\intercal (\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j) - \frac{\sigma_t}{2} (\widetilde{\mathbf{g}}_t^\intercal (\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j))^2 \tag{5.6}$$

and due to the update rules and the contracting property of projections (Hazan et al., 2006),

$$\begin{aligned}
\|\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j\|_{\widetilde{\mathbf{A}}_{t-1}}^2 &\leq \|\widetilde{\boldsymbol{\omega}}_{t-1} - \widetilde{\mathbf{A}}_{t-1}^{-1} \widetilde{\mathbf{g}}_{t-1} - \overline{\boldsymbol{\omega}}_j\|_{\widetilde{\mathbf{A}}_{t-1}}^2 \\
&= \|\widetilde{\boldsymbol{\omega}}_{t-1} - \overline{\boldsymbol{\omega}}_j\|_{\widetilde{\mathbf{A}}_{t-1}}^2 - 2\widetilde{\mathbf{g}}_{t-1}^\intercal \widetilde{\mathbf{A}}_{t-1}^{-1} \widetilde{\mathbf{A}}_{t-1} (\widetilde{\boldsymbol{\omega}}_{t-1} - \overline{\boldsymbol{\omega}}_j) + \|\widetilde{\mathbf{A}}_{t-1}^{-1} \widetilde{\mathbf{g}}_{t-1}\|_{\widetilde{\mathbf{A}}_{t-1}}^2 \\
&= \|\widetilde{\boldsymbol{\omega}}_{t-1} - \overline{\boldsymbol{\omega}}_j\|_{\widetilde{\mathbf{A}}_{t-1}}^2 - 2\widetilde{\mathbf{g}}_{t-1}^\intercal (\widetilde{\boldsymbol{\omega}}_{t-1} - \overline{\boldsymbol{\omega}}_j) + \widetilde{\mathbf{g}}_{t-1}^\intercal \widetilde{\mathbf{A}}_{t-1}^{-1} \mathbf{g}_{t-1}
\end{aligned}$$

except for last round of epoch j, i.e., round $t_{j+1}-1$, where due to the reset we cannot use the update rule, and we need to treat separately. Therefore, for any round $t = t_j, t_j+1, \ldots, t_{j+1}-2$ of epoch j, we have that

$$2\widetilde{\mathbf{g}}_t^\intercal(\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j) \leq \|\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_t} - \|\widetilde{\boldsymbol{\omega}}_{t+1} - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_t} + \widetilde{\mathbf{g}}_t^\intercal \widetilde{\mathbf{A}}_t^{-1}\mathbf{g}_t. \tag{5.7}$$

Using the upper bound on the loss difference 5.6, we get

$$\begin{aligned}
&\sum_{t=t_j}^{t_{j+1}-1} \ell_t\left(\boldsymbol{\phi}_t^\intercal \widetilde{\mathbf{w}}_t\right) - \ell_t\left(\boldsymbol{\phi}_t^\intercal \overline{\mathbf{w}}_t\right) \\
&= \ell_{t_{j+1}-1}\left(\boldsymbol{\phi}_{t_{j+1}-1}^\intercal \widetilde{\mathbf{w}}_{t_{j+1}-1}\right) - \ell_{t_{j+1}-1}\left(\boldsymbol{\phi}_{t_{j+1}-1}^\intercal \overline{\mathbf{w}}_{t_{j+1}-1}\right) + \sum_{t=t_j}^{t_{j+1}-2} \ell_t\left(\boldsymbol{\phi}_t^\intercal \widetilde{\mathbf{w}}_t\right) - \ell_t\left(\boldsymbol{\phi}_t^\intercal \overline{\mathbf{w}}_t\right) \\
&= R_j + \sum_{t=t_j}^{t_{j+1}-2} \ell_t\left(\widetilde{\boldsymbol{\phi}}_t^\intercal \widetilde{\boldsymbol{\omega}}_t\right) - \ell_t\left(\widetilde{\boldsymbol{\phi}}_t^\intercal \overline{\boldsymbol{\omega}}_j\right) \leq R_j + \sum_{t=t_j}^{t_{j+1}-2} \widetilde{\mathbf{g}}_t^\intercal\left(\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j\right) - \frac{\sigma_t}{2}\left(\widetilde{\mathbf{g}}_t^\intercal\left(\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j\right)\right)^2,
\end{aligned}$$

with $R_j \stackrel{\text{def}}{=} \ell_{t_{j+1}-1}\left(\boldsymbol{\phi}_{t_{j+1}-1}^\intercal \widetilde{\mathbf{w}}_{t_{j+1}-1}\right) - \ell_{t_{j+1}-1}\left(\boldsymbol{\phi}_{t_{j+1}-1}^\intercal \overline{\mathbf{w}}_{t_{j+1}-1}\right)$ that corresponds to the regret of the last round of the epoch that we need to treat separately. Now for the all other rounds,

$$\begin{aligned}
&\sum_{t=t_j}^{t_{j+1}-2} \widetilde{\mathbf{g}}_t^\intercal\left(\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j\right) - \frac{\sigma_t}{2}\left(\widetilde{\mathbf{g}}_t^\intercal\left(\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j\right)\right)^2 \\
&\quad\leq \sum_{t=t_j}^{t_{j+1}-2} \|\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_t} - \|\widetilde{\boldsymbol{\omega}}_{t+1} - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_t} + \widetilde{\mathbf{g}}_t^\intercal \widetilde{\mathbf{A}}_t^{-1}\mathbf{g}_t - \|\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j\|^2_{\frac{\sigma_t}{2}\widetilde{\mathbf{g}}_t\widetilde{\mathbf{g}}_t^\intercal} \\
&\quad= -\|\widetilde{\boldsymbol{\omega}}_{t_{j+1}-1} - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_{t_{j+1}-1}} + \|\widetilde{\boldsymbol{\omega}}_{t_j} - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_{t_j}} + \widetilde{\mathbf{g}}_{t_j}^\intercal \widetilde{\mathbf{A}}_{t_j}^{-1}\mathbf{g}_{t_j} - \|\widetilde{\boldsymbol{\omega}}_{t_j} - \overline{\boldsymbol{\omega}}_j\|^2_{\frac{\sigma_{t_j}}{2}\widetilde{\mathbf{g}}_{t_j}\widetilde{\mathbf{g}}_{t_j}^\intercal} \\
&\qquad+ \sum_{t=t_j+1}^{t_{j+1}-2} \|\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_t} - \|\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_{t-1}} + \widetilde{\mathbf{g}}_t^\intercal \widetilde{\mathbf{A}}_t^{-1}\mathbf{g}_t - \|\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j\|^2_{\frac{\sigma_t}{2}\widetilde{\mathbf{g}}_t\widetilde{\mathbf{g}}_t^\intercal}
\end{aligned}$$

Now we treat the terms above equation separately. First, since $\widetilde{\mathbf{A}}_t = \widetilde{\mathbf{A}}_{t-1} + \frac{\sigma_t}{2}\widetilde{\mathbf{g}}_t\widetilde{\mathbf{g}}_t^\intercal$, we have $\|\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_{t-1}} + \|\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j\|^2_{\frac{\sigma_t}{2}\widetilde{\mathbf{g}}_t\widetilde{\mathbf{g}}_t^\intercal} = \|\widetilde{\boldsymbol{\omega}}_t - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_t}$. Second, by Algorithm 14, we know that at the beginning of each epoch, $\widetilde{\mathbf{A}}_{t_j} = \alpha\mathbf{I} + \widetilde{\mathbf{g}}_{t_j}\widetilde{\mathbf{g}}_{t_j}^\intercal$. This also helps us to bound the term $\widetilde{\mathbf{g}}_{t_j}^\intercal \widetilde{\mathbf{A}}_{t_j}^{-1}\mathbf{g}_{t_j}$ as

$$\begin{aligned}
\widetilde{\mathbf{g}}_{t_j}^\intercal \widetilde{\mathbf{A}}_{t_j}^{-1}\mathbf{g}_{t_j} &= \widetilde{\mathbf{g}}_{t_j}^\intercal\left(\alpha\mathbf{I} + \tfrac{\sigma_{t_j}}{2}\widetilde{\mathbf{g}}_{t_j}\widetilde{\mathbf{g}}_{t_j}^\intercal\right)^{-1}\mathbf{g}_{t_j} = \frac{\widetilde{\mathbf{g}}_{t_j}^\intercal\widetilde{\mathbf{g}}_{t_j}}{\alpha + \frac{\sigma_{t_j}}{2}\widetilde{\mathbf{g}}_{t_j}^\intercal\widetilde{\mathbf{g}}_{t_j}} \leq \frac{\widetilde{\mathbf{g}}_{t_j}^\intercal\widetilde{\mathbf{g}}_{t_j}}{\alpha} = \frac{\left(\ell'_{t_j}\left(\widetilde{y}_{t_j}\right)\right)^2 \widetilde{\boldsymbol{\phi}}_{t_j}^\intercal\widetilde{\boldsymbol{\phi}}_{t_j}}{\alpha} \\
&\leq \frac{L^2\widetilde{\boldsymbol{\phi}}_{t_j}^\intercal\widetilde{\boldsymbol{\phi}}_{t_j}}{\alpha} = \frac{L^2\boldsymbol{\phi}_{t_j}^\intercal\boldsymbol{\Pi}_{\mathcal{I}_j}\boldsymbol{\phi}_{t_j}}{\alpha} \leq \frac{L^2\boldsymbol{\phi}_{t_j}^\intercal\boldsymbol{\phi}_{t_j}}{\alpha} \leq \frac{L^2}{\alpha}.
\end{aligned}$$

By Algorithm 14, we also know that at the beginning of each epoch $\widetilde{\boldsymbol{\omega}}_{t_j} = \mathbf{0}$ which helps us to bound the two terms outside of the summation as

$$\begin{aligned}
&\|\widetilde{\boldsymbol{\omega}}_{t_j} - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_{t_j}} - \|\widetilde{\boldsymbol{\omega}}_{t_j} - \overline{\boldsymbol{\omega}}_j\|^2_{\frac{\sigma_{t_j}}{2}\widetilde{\mathbf{g}}_{t_j}\widetilde{\mathbf{g}}_{t_j}^\intercal} \\
&\qquad= \|\widetilde{\boldsymbol{\omega}}_{t_j} - \overline{\boldsymbol{\omega}}_j\|^2_{\alpha\mathbf{I}} + \|\widetilde{\boldsymbol{\omega}}_{t_j} - \overline{\boldsymbol{\omega}}_j\|^2_{\frac{\sigma_{t_j}}{2}\widetilde{\mathbf{g}}_{t_j}\widetilde{\mathbf{g}}_{t_j}^\intercal} - \|\widetilde{\boldsymbol{\omega}}_{t_j} - \overline{\boldsymbol{\omega}}_j\|^2_{\frac{\sigma_{t_j}}{2}\widetilde{\mathbf{g}}_{t_j}\widetilde{\mathbf{g}}_{t_j}^\intercal} \\
&\qquad= \alpha\|\widetilde{\boldsymbol{\omega}}_{t_j} - \overline{\boldsymbol{\omega}}_j\|_2^2 = \alpha\|\overline{\boldsymbol{\omega}}_j\|_2^2.
\end{aligned}$$

Altogether, we combine the upper bounds of the terms to

$$\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\intercal \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t^\intercal \overline{\mathbf{w}}_t)$$
$$\leq \alpha\|\overline{\boldsymbol{\omega}}_j\|^2 + R_j + \frac{L^2}{\alpha} - \|\widetilde{\boldsymbol{\omega}}_{t_{j+1}-1} - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_{t_{j+1}-1}} + \sum_{t=t_j+1}^{t_{j+1}-2} \widetilde{\mathbf{g}}_t^\intercal \widetilde{\mathbf{A}}_t^{-1} \mathbf{g}_t.$$

Using the result of (Hazan et al., 2006) we can upper bound the sum of the quadratic forms as

$$\begin{aligned}
\sum_{t=t_j}^{t_{j+1}-2} \widetilde{\mathbf{g}}_t^\intercal \widetilde{\mathbf{A}}_t^{-1} \widetilde{\mathbf{g}}_t &= \sum_{t=t_j}^{t_{j+1}-2} \widetilde{\mathbf{g}}_t^\intercal \left(\sum_{s=t_j}^{t} \tfrac{\sigma_s}{2} \widetilde{\mathbf{g}}_s \widetilde{\mathbf{g}}_s^\intercal + \alpha \mathbf{I} \right)^{-1} \widetilde{\mathbf{g}}_t \\
&= \sum_{t=t_j}^{t_{j+1}-2} \frac{2}{\sigma_t} \left(\sqrt{\sigma_t/2\alpha} \cdot \widetilde{\mathbf{g}}_t^\intercal \right) \left(\sum_{s=t_j}^{t} \tfrac{\sigma_s}{2} \widetilde{\mathbf{g}}_s \widetilde{\mathbf{g}}_s^\intercal / \alpha + \mathbf{I} \right)^{-1} \left(\sqrt{\sigma_t/2\alpha} \cdot \widetilde{\mathbf{g}}_t \right) \\
&\leq \frac{2}{\sigma_{\min}} \log \left(\operatorname{Det} \left(\widetilde{\mathbf{G}}_j \widetilde{\mathbf{G}}_j^\intercal / \alpha + \mathbf{I} \right) \right).
\end{aligned}$$

where $\widetilde{\mathbf{G}}_j$ is the $j \times h_j$ matrix with $\sqrt{\sigma_t/2} \cdot \widetilde{\mathbf{g}}_t$ columns. Let $\widetilde{\mathbf{D}}_j$ be the $h_j \times h_j$ diagonal matrix with $\dot{g}_t \sqrt{\sigma_t/2}$ on the diagonal and Φ_j (resp., $\widetilde{\Phi}_j$) the matrix with ϕ_t (resp., $\widetilde{\phi}_t$) as columns for $t_j \leq t < t_{j+1}$ We can rewrite $\widetilde{\mathbf{G}}_j = \widetilde{\Phi}_j \widetilde{\mathbf{D}}_j = \boldsymbol{\Sigma}_j^{-1} \mathbf{U}_j^\intercal \Phi_{\mathcal{I}_j}^\intercal \Phi_j \widetilde{\mathbf{D}}_j$. We also have

$$\widetilde{\mathbf{G}}_j^\intercal \widetilde{\mathbf{G}}_j = \widetilde{\mathbf{D}}_j \Phi_j^\intercal \Phi_{\mathcal{I}_j} \mathbf{U}_j^\intercal \boldsymbol{\Sigma}_j^{-1} \boldsymbol{\Sigma}_j^{-1} \mathbf{U}_j^\intercal \Phi_{\mathcal{I}_j}^\intercal \Phi_j \widetilde{\mathbf{D}}_j = \widetilde{\mathbf{D}}_j \Phi_j^\intercal \boldsymbol{\Pi}_{\mathcal{I}_j} \Phi_j \widetilde{\mathbf{D}}_j \preceq \widetilde{\mathbf{D}}_j \Phi_{\mathcal{I}_j}^\intercal \Phi_{\mathcal{I}_j} \widetilde{\mathbf{D}}_j,$$

since $\|\boldsymbol{\Pi}_{\mathcal{I}_j}\| \leq 1$ because $\boldsymbol{\Pi}_{\mathcal{I}_j}$ is a projection matrix. Knowing that $\operatorname{Det}(\mathbf{A}) \leq \operatorname{Det}(\mathbf{B})$ whenever $\mathbf{A} \preceq \mathbf{B}$, together with Sylvester's determinant identity, we get that

$$\operatorname{Det}(\widetilde{\mathbf{G}}_j \widetilde{\mathbf{G}}_j^\intercal / \alpha + \mathbf{I}) \leq \operatorname{Det}(\widetilde{\mathbf{D}}_j \Phi_j^\intercal \Phi_j \widetilde{\mathbf{D}}_j / \alpha + \mathbf{I}) = \prod_{t=1}^{h_j} (\lambda_t/\alpha + 1),$$

where λ_t are the eigenvalues of $\widetilde{\mathbf{D}}_j \Phi_{\mathcal{I}_j}^\intercal \Phi_{\mathcal{I}_j} \widetilde{\mathbf{D}}_j = \widetilde{\mathbf{D}}_j \mathbf{K}_j \widetilde{\mathbf{D}}_j = \overline{\mathbf{K}}_j$ and $\mathbf{K}_j$ is the kernel matrix between the samples in epoch j. Using Lemma 3.9 we can further bound the expression above as

$$\begin{aligned}
\log \left(\prod_{t=1}^{h_j} \lambda_t/\alpha + 1 \right) &\leq 2 d_{\text{eff}}^j \left(\frac{\alpha}{\sigma_{\min} L^2} \right) \log \left(2\sigma_{\min} L^2 \operatorname{Tr}(\mathbf{K}_j)/\alpha \right) \\
&\leq 2 d_{\text{eff}}^T \left(\frac{\alpha}{\sigma_{\min} L^2} \right) \log \left(2\sigma_{\min} L^2 \operatorname{Tr}(\mathbf{K}_t)/\alpha \right).
\end{aligned}$$

Putting it all together, and using $\|\overline{\boldsymbol{\omega}}_j\|_2^2 = \|\boldsymbol{\Sigma}_j^{-1} \mathbf{U}_j \Phi_{\mathcal{I}_j}^\intercal \overline{\mathbf{w}}_j\|_2^2 = \overline{\mathbf{w}}_j^\intercal \boldsymbol{\Pi}_{\mathcal{I}_j} \overline{\mathbf{w}}_j = \|\overline{\mathbf{w}}_j\|_2^2$ we get

$$\begin{aligned}
A_j \leq \frac{4}{\sigma_{\min}} d_{\text{eff}}^T \left(\frac{\alpha}{\sigma_{\min} L^2} \right) \log \left(2\sigma_{\min} L^2 \operatorname{Tr}(\mathbf{K}_j)/\alpha \right) + \alpha \|\overline{\mathbf{w}}_j\|_2^2 \\
+ R_j + \frac{L^2}{\alpha} - \|\widetilde{\boldsymbol{\omega}}_{t_{j+1}-1} - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_{t_{j+1}-1}}.
\end{aligned}$$

Bounding B_j

We begin by adding and subtracting $\alpha\|\overline{\mathbf{w}}_j\|^2$ and $\alpha\|\widehat{\mathbf{w}}_j\|^2$

$$\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\mathsf{T}\overline{\mathbf{w}}_j) - \ell_t(\phi_t^\mathsf{T}\widehat{\mathbf{w}}_j) = \left(\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\mathsf{T}\overline{\mathbf{w}}_j)\right) - \left(\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\mathsf{T}\widehat{\mathbf{w}}_j)\right)$$
$$= \alpha\|\widehat{\mathbf{w}}_j\|^2 - \alpha\|\overline{\mathbf{w}}_j\|^2 + \left(\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\mathsf{T}\overline{\mathbf{w}}_j) + \alpha\|\overline{\mathbf{w}}_j\|^2\right) - \left(\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\mathsf{T}\widehat{\mathbf{w}}_j) + \alpha\|\widehat{\mathbf{w}}_j\|^2\right)$$

We will now apply the following result from Xu et al., (2017).

Proposition 5.10 — (Xu et al., 2017, Lemma 2). Suppose the loss functions ℓ_t are L-Lipschitz continuous, and $\overline{\mathbf{w}}_j = \mathbf{\Pi}_{\mathcal{I}_j}\overline{\mathbf{w}}_j = \Phi_{\mathcal{I}_j}\mathbf{U}_j\mathbf{\Sigma}_j^{-1/2}\overline{\boldsymbol{\omega}}_j$. We have

$$\frac{1}{h_j}\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\mathsf{T}\overline{\mathbf{w}}_j) + \frac{\alpha}{2}\|\overline{\mathbf{w}}_j\|^2 \leq \frac{1}{h_j}\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\mathsf{T}\widehat{\mathbf{w}}_j) + \frac{\alpha}{2}\|\widehat{\mathbf{w}}_j\|^2 + \frac{L^2}{2\alpha h_j}\|\Phi_j - \mathbf{\Pi}_{\mathcal{I}_j}\Phi_j\|_2^2.$$

First, it is important to quantify the last term in Proposition 5.10,

$$\begin{aligned}\|\Phi_j - \mathbf{\Pi}_{\mathcal{I}_j}\Phi_j\|_2^2 &= \lambda_{\max}\left(\left(\Phi_j - \mathbf{\Pi}_{\mathcal{I}_j}\Phi_j\right)^\mathsf{T}\left(\Phi_j - \mathbf{\Pi}_{\mathcal{I}_j}\Phi_j\right)\right)\\ &= \lambda_{\max}\left(\Phi_j^\mathsf{T}\Phi_j - 2\Phi_j^\mathsf{T}\mathbf{\Pi}_{\mathcal{I}_j}\Phi_j + \Phi_j^\mathsf{T}\mathbf{\Pi}_{\mathcal{I}_j}\mathbf{\Pi}_{\mathcal{I}_j}\Phi_j\right)\\ &= \lambda_{\max}\left(\Phi_j^\mathsf{T}\Phi_j - \Phi_j^\mathsf{T}\mathbf{\Pi}_{\mathcal{I}_j}\Phi_j\right) = \lambda_{\max}\left(\mathbf{K}_j - \widetilde{\mathbf{K}}_j\right) \leq \frac{\gamma\varepsilon}{1-\varepsilon},\end{aligned}$$

when in the last step we applied Lemma 4.3 that bounds the quality of the approximation. In order to apply Proposition 5.10 we also need to rescale B_j,

$$\begin{aligned}\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\mathsf{T}\overline{\mathbf{w}}_j) + \alpha\|\overline{\mathbf{w}}_j\|^2 &= h_j\left(\frac{1}{h_j}\sum_{t=t_j}^{t_{j+1}-1} \ell_t\left(\phi_t^\mathsf{T}\overline{\mathbf{w}}_j\right) + \frac{\alpha}{2}\frac{2}{h_j}\|\overline{\mathbf{w}}_j\|^2\right)\\ &\leq h_j\left(\frac{1}{h_j}\sum_{t=t_j}^{t_{j+1}-1} \ell_t\left(\phi_t^\mathsf{T}\widehat{\mathbf{w}}_j\right) + \frac{\alpha}{2}\frac{2}{h_j}\|\widehat{\mathbf{w}}_j\|^2 + \frac{L^2 h_j}{4\alpha h_j}\|\Phi_j - \mathbf{\Pi}_{\mathcal{I}_j}\Phi_j\|_2^2\right)\\ &= \sum_{t=t_j}^{t_{j+1}-1} \ell_t\left(\phi_t^\mathsf{T}\widehat{\mathbf{w}}_j\right) + \alpha\|\widehat{\mathbf{w}}_j\|^2 + \frac{L^2 h_j}{4\alpha}\|\mathbf{K}_j - \widetilde{\mathbf{K}}_j\|_2^2\\ &\leq \sum_{t=t_j}^{t_{j+1}-1} \ell_t\left(\phi_t^\mathsf{T}\widehat{\mathbf{w}}_j\right) + \alpha\|\widehat{\mathbf{w}}_j\|^2 + \frac{L^2\varepsilon}{4(1-\varepsilon)}\frac{h_j\gamma}{\alpha}.\end{aligned}$$

Therefore, the difference of the regularized losses for the best solution within epoch j when considering the whole space $\mathcal{H}$ versus subspace $\mathcal{H}_j$ is bounded as

$$\left(\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\mathsf{T}\overline{\mathbf{w}}_j) + \alpha\|\overline{\mathbf{w}}_j\|\right) - \left(\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\mathsf{T}\widehat{\mathbf{w}}_j) + \alpha\|\widehat{\mathbf{w}}_j\|\right) \leq \frac{L^2\varepsilon}{4(1-\varepsilon)}\frac{h_j\gamma}{\alpha}$$

and therefore their unregularized counterparts are bounded as

$$B_j \leq \alpha\|\widehat{\mathbf{w}}_j\|^2 - \alpha\|\overline{\mathbf{w}}_j\|^2 + \frac{L^2\varepsilon}{4(1-\varepsilon)}\frac{h_j\gamma}{\alpha}.$$

Bounding C_j

Similarly as for B_j, we add and subtract the regularizers

$$\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\intercal \widehat{\mathbf{w}}_j) - \ell_t(\phi_t^\intercal \mathbf{w}) = \left(\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\intercal \widehat{\mathbf{w}}_j)\right) - \left(\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\intercal \mathbf{w})\right)$$
$$= \alpha\|\mathbf{w}\|^2 - \alpha\|\widehat{\mathbf{w}}_j\|^2 + \left(\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\intercal \widehat{\mathbf{w}}_j) + \alpha\|\widehat{\mathbf{w}}_j\|^2\right) - \left(\sum_{t=t_j}^{t_{j+1}-1} \ell_t(\phi_t^\intercal \mathbf{w}) + \alpha\|\mathbf{w}\|^2\right)$$

By the definition of $\widehat{\mathbf{w}}_j$ as a minimizer, we have that the difference between the summations is negative or zero. Therefore, term C_j is trivially bounded as

$$C_j \leq \alpha\|\mathbf{w}\|^2 - \alpha\|\widehat{\mathbf{w}}_j\|^2.$$

Bounding the regret: We put all the bounds on decomposed regret together:

$$\sum_{t=1}^{T} \ell_t(\phi_t^\intercal \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t^\intercal \mathbf{w}) = \sum_{j=1}^{J} A_j + B_j + C_j$$
$$\leq \sum_{j=1}^{J} \frac{4}{\sigma_{\min}} d_{\text{eff}}^T\left(\frac{\alpha}{\sigma_{\min}L^2}\right) \log\left(2\sigma_{\min}L^2 \operatorname{Tr}(\mathbf{K}_j)/\alpha\right) + \alpha\|\overline{\mathbf{w}}_j\|_2^2 + R_j + \frac{L^2}{\alpha} - \|\widetilde{\boldsymbol{\omega}}_{t_{j+1}-1} - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_{t_{j+1}-1}}$$
$$+ \alpha\|\widehat{\mathbf{w}}_j\|^2 - \alpha\|\overline{\mathbf{w}}_j\|^2 + \frac{L^2\varepsilon}{4(1-\varepsilon)}\frac{h_j\gamma}{\alpha} + \alpha\|\mathbf{w}\|^2 - \alpha\|\widehat{\mathbf{w}}_j\|^2$$
$$= \left(\sum_{j=1}^{J} \frac{4}{\sigma_{\min}} d_{\text{eff}}^T\left(\frac{\alpha}{\sigma_{\min}L^2}\right) \log\left(2\sigma_{\min}L^2 \operatorname{Tr}\left(\mathbf{K}_j\right)/\alpha\right)\right) + \left(\sum_{j=1}^{J} \frac{L^2\varepsilon}{4(1-\varepsilon)}\frac{h_j\gamma}{\alpha}\right) + \frac{JL^2}{\alpha} + J\alpha\|\mathbf{w}\|$$
$$+ \sum_{j=1}^{J} R_j - \|\widetilde{\boldsymbol{\omega}}_{t_{j+1}-1} - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_{t_{j+1}-1}}$$
$$\leq \left(\sum_{j=1}^{J} \frac{4}{\sigma_{\min}} d_{\text{eff}}^T\left(\frac{\alpha}{\sigma_{\min}L^2}\right) \log\left(2\sigma_{\min}L^2 T/\alpha\right)\right) + \frac{L^2}{\alpha}\left(\frac{T\gamma\varepsilon}{4(1-\varepsilon)} + 1\right) + J\alpha\|\mathbf{w}\|$$
$$+ \sum_{j=1}^{J} R_j - \|\widetilde{\boldsymbol{\omega}}_{t_{j+1}-1} - \overline{\boldsymbol{\omega}}_j\|^2_{\widetilde{\mathbf{A}}_{t_{j+1}-1}}$$
$$\leq J\alpha\|\mathbf{w}\| + \frac{4J}{\sigma_{\min}} d_{\text{eff}}^T\left(\frac{\alpha}{\sigma_{\min}L^2}\right) \log\left(2\sigma_{\min}L^2 T/\alpha\right) + \frac{L^2}{\alpha}\left(\frac{T\gamma\varepsilon}{4(1-\varepsilon)} + 1\right) + \sum_{j=1}^{J} R_j$$
$$= 3\beta d_{\text{eff}}^T(\gamma)\log(2T)\left(\frac{4}{\sigma_{\min}} d_{\text{eff}}^T\left(\frac{\alpha}{\sigma_{\min}L^2}\right) \log\left(2\sigma_{\min}L^2 T/\alpha\right) + \alpha\|\mathbf{w}\|\right) + \frac{L^2}{\alpha}\left(\frac{T\gamma\varepsilon}{4(1-\varepsilon)} + 1\right) + \sum_{j=1}^{J} R_j$$

■

Proof of Theorem 5.9. In the special case of *squared loss*, we can obtain a different kind of guarantee. We proceed in a similar way as in the proof of Theorem 5.8 and highlight the differences. Starting from the $A_j + B_j + C_j$ decomposition, we will bound B_j differently using the following result.

Proposition 5.11 — (Zhdanov and Kalnishkan, 2010, Theorem 1). Take a kernel $\mathcal{K}$ on a domain $\mathcal{X}$ and a parameter $\alpha > 0$. Let $\mathcal{H}$ be the RKHS for the kernel $\mathcal{K}$. For any sequence $\{(\mathbf{x}_t, y_t)\}_{t=1}^T$ let $\mathbf{y}_T \in \mathbb{R}^T$ be the concatenation of the y_t target variables. Then

$$\mathcal{L}_T^*(\mathcal{H}) = \min_{f\in\mathcal{H}} \left(\sum_{t=1}^T \big(f(\mathbf{x}_t) - y_t\big)^2 + \alpha\|f\|_{\mathcal{H}}^2 \right) = \alpha \mathbf{y}_T^\intercal (\mathbf{K}_T + \alpha\mathbf{I})^{-1}\mathbf{y}_T.$$

Bounding B_j

In our particular case, we apply Proposition 5.11 to the whole space $\mathcal{H}$ and all subspaces $\mathcal{H}_j$, one for each epoch j.

$$\begin{aligned}
\mathcal{L}_j^*(\mathcal{H}) &= \sum_{t=t_j}^{t_{j+1}-1} \big(\phi_t^\intercal \widehat{\mathbf{w}}_j - y_t\big)^2 + \alpha\|\widehat{\mathbf{w}}_j\|_2^2 \\
&= \min_{\mathbf{w}\in\mathcal{H}} \left(\sum_{t=t_j}^{t_{j+1}-1} \big(\phi_t^\intercal \mathbf{w} - y_t\big)^2 + \alpha\|\mathbf{w}\|_2^2 \right) = \alpha \mathbf{y}_j^\intercal (\Phi_j^\intercal \Phi_j + \alpha\mathbf{I})^{-1}\mathbf{y}_j, \\
\mathcal{L}_j^*(\mathcal{H}_j) &= \sum_{t=t_j}^{t_{j+1}-1} \big(\phi_t^\intercal \overline{\mathbf{w}}_j - y_t\big)^2 + \alpha\|\overline{\mathbf{w}}_j\|_2^2 \\
&= \min_{\mathbf{w}\in\mathcal{H}_j} \left(\sum_{t=t_j}^{t_{j+1}-1} \big(\phi_t^\intercal \mathbf{w} - y_t\big)^2 + \alpha\|\mathbf{w}\|_2^2 \right) = \alpha \mathbf{y}_j^\intercal (\Phi_j^\intercal \mathbf{\Pi}_{\mathcal{I}_j} \Phi_j + \alpha\mathbf{I})^{-1}\mathbf{y}_j.
\end{aligned}$$

Therefore, taking account for the regularization in Proposition 5.11, for any epoch j,

$$\begin{aligned}
\sum_{t=t_j}^{t_{j+1}-1} \big(\phi_t^\intercal \overline{\mathbf{w}}_j - y_t\big)^2 &= -\alpha\|\overline{\mathbf{w}}_j\|_2^2 + \sum_{t=t_j}^{t_{j+1}-1} \big(\phi_t^\intercal \overline{\mathbf{w}}_j - y_t\big)^2 + \alpha\|\overline{\mathbf{w}}_j\|_2^2 \\
&= -\alpha\|\overline{\mathbf{w}}_j\|_2^2 + \alpha \mathbf{y}_j^\intercal (\Phi_j^\intercal \mathbf{\Pi}_{\mathcal{I}_j} \Phi_j + \alpha\mathbf{I})^{-1}\mathbf{y}_j.
\end{aligned}$$

Now using the kernel approximation guarantees of Lemma 4.3 we have

$$\begin{aligned}
\alpha \mathbf{y}_j^\intercal (\Phi_j^\intercal \mathbf{\Pi}_{\mathcal{I}_j} \Phi_j + \alpha\mathbf{I})^{-1}\mathbf{y}_j &\leq \alpha \mathbf{y}_j^\intercal \left(\Phi_j^\intercal \Phi_j - \frac{\gamma\varepsilon}{1-\varepsilon}\mathbf{I} + \alpha\mathbf{I} \right)^{-1} \mathbf{y}_j \\
&= \left(\alpha - \frac{\varepsilon}{1-\varepsilon}\gamma\right)^{-1} \alpha \left(\alpha - \frac{\varepsilon}{1-\varepsilon}\gamma\right) \mathbf{y}_j^\intercal \left(\Phi_j^\intercal \Phi_j + \left(\alpha - \frac{\varepsilon}{1-\varepsilon}\gamma\right)\mathbf{I} \right)^{-1} \mathbf{y}_j \\
&= \left(\left(\alpha - \frac{\varepsilon}{1-\varepsilon}\gamma\right)^{-1} \alpha \right) \alpha' \mathbf{y}_j^\intercal \left(\Phi_j^\intercal \Phi_j + \alpha'\mathbf{I} \right)^{-1} \mathbf{y}_j \\
&= (1+\varepsilon')\alpha' \mathbf{y}_j^\intercal \left(\Phi_j^\intercal \Phi_j + \alpha'\mathbf{I} \right)^{-1} \mathbf{y}_j.
\end{aligned}$$

where we denoted $\varepsilon' = \left(\left(\alpha - \frac{\gamma\varepsilon}{1-\varepsilon}\right)^{-1}\alpha\right) - 1$ and $\alpha' = \left(\alpha - \frac{\gamma\varepsilon}{1-\varepsilon}\right)$. Putting it together

$$\begin{aligned}
\sum_{t=t_j}^{t_{j+1}-1} \left(\phi_t^\intercal \overline{\mathbf{w}}_j - y_t\right)^2 &\leq -\alpha\|\overline{\mathbf{w}}_j\|_2^2 + (1+\varepsilon')\alpha'\mathbf{y}_j^\intercal \left(\Phi_j^\intercal\Phi_j + \alpha'\mathbf{I}\right)^{-1}\mathbf{y}_j \\
&= -\alpha\|\overline{\mathbf{w}}_j\|_2^2 + (1+\varepsilon')\left(\sum_{t=t_j}^{t_{j+1}-1} \min_{\mathbf{w}\in\mathcal{H}} \left(\phi_t^\intercal \mathbf{w}_j - y_t\right)^2 + \alpha'\|\mathbf{w}_j\|_2^2\right) \\
&\leq -\alpha\|\overline{\mathbf{w}}_j\|_2^2 + (1+\varepsilon')\left(\sum_{t=t_j}^{t_{j+1}-1} \min_{\mathbf{w}\in\mathcal{H}} \left(\phi_t^\intercal \mathbf{w}_j - y_t\right)^2 + \alpha\|\mathbf{w}_j\|_2^2\right) \\
&= -\alpha\|\overline{\mathbf{w}}_j\|_2^2 + (1+\varepsilon')\left(\sum_{t=t_j}^{t_{j+1}-1} \left(\phi_t^\intercal \widehat{\mathbf{w}}_j - y_t\right)^2 + \alpha\|\widehat{\mathbf{w}}_j\|_2^2\right) \\
&= -\alpha\|\overline{\mathbf{w}}_j\|_2^2 + \sum_{t=t_j}^{t_{j+1}-1} \left(\phi_t^\intercal \widehat{\mathbf{w}}_j - y_t\right)^2 + \varepsilon'\alpha\|\widehat{\mathbf{w}}_j\|_2^2 + \varepsilon'\left(\sum_{t=t_j}^{t_{j+1}-1} \left(\phi_t^\intercal \widehat{\mathbf{w}}_j - y_t\right)^2 + \alpha\|\widehat{\mathbf{w}}_j\|_2^2\right).
\end{aligned}$$

Therefore, extracting the B_j part of the regret we get

$$B_j \leq -\alpha\|\overline{\mathbf{w}}_j\|_2^2 + \varepsilon'\alpha\|\widehat{\mathbf{w}}_j\|_2^2 + \varepsilon'\left(\sum_{t=t_j}^{t_{j+1}-1} \left(\phi_t^\intercal \widehat{\mathbf{w}}_j - y_t\right)^2 + \alpha\|\widehat{\mathbf{w}}_j\|_2^2\right).$$

Bounding C_j

Changing slightly the regularizers that we add and subtract in the bound on C_j we obtain

$$\sum_{j=1}^{J} B_j + C_j = \sum_{j=1}^{J} -\alpha\|\overline{\mathbf{w}}_j\|_2^2 + \varepsilon'\alpha\|\mathbf{w}\|_2^2 + \varepsilon'\mathcal{L}_j^*.$$

Integrating this with the bound for A_j obtained in the proof of Theorem 5.8 we get

$$\begin{aligned}
\sum_{t=1}^{T} \ell_t\left(\widetilde{\phi}_t^\intercal \widetilde{\boldsymbol{\omega}}_t\right) - \ell_t\left(\phi_t^\intercal \mathbf{w}\right) \leq &\left(\sum_{j=1}^{J} \frac{4}{\sigma_{\min}} d_{\text{eff}}^j\left(\frac{\alpha}{\sigma_{\min}L^2}\right)\log\left(2\sigma_{\min}L^2\operatorname{Tr}(\mathbf{K}_j)/\alpha\right) + \varepsilon'\mathcal{L}_j^*\right) \\
&+ JLC + \frac{JL^2}{\alpha} + J\varepsilon'\alpha\|\mathbf{w}\|_2^2.
\end{aligned}$$

■

References

Aizerman, M. A., E. A. Braverman, and L. Rozonoer (1964). "Theoretical foundations of the potential function method in pattern recognition learning." In: *Automation and Remote Control,* Automation and Remote Control, 25, pages 821–837 (cited on page 92).

Alaoui, Ahmed El and Michael W. Mahoney (2015). "Fast randomized kernel methods with statistical guarantees". In: *Neural Information Processing Systems* (cited on pages 19, 24, 27, 38–42, 48, 61, 68, 86, 91–93, 95, 125).

Bach, Francis (2013). "Sharp analysis of low-rank kernel matrix approximations". In: *Conference on Learning Theory* (cited on pages 19, 21, 61, 62, 68, 96).

Batson, Joshua, Daniel A Spielman, and Nikhil Srivastava (2012). "Twice-ramanujan sparsifiers". In: *SIAM Journal on Computing* 41.6, pages 1704–1721 (cited on page 19).

Belkin, Mikhail, Irina Matveeva, and Partha Niyogi (2004). "Regularization and Semi-Supervised Learning on Large Graphs". In: *Proceedings of COLT* (cited on pages 103, 105, 107).

Belkin, Mikhail and Partha Niyogi (2001). "Laplacian eigenmaps and spectral techniques for embedding and clustering". In: *NIPS*. Volume 14. 14, pages 585–591 (cited on page 98).

Belkin, Mikhail, Partha Niyogi, and Vikas Sindhwani (2006). "Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples". In: *Journal of Machine Learning Research* 7, pages 2399–2434 (cited on pages 98, 103).

Bousquet, Olivier and André Elisseeff (2002). "Stability and generalization". In: *The Journal of Machine Learning Research* 2, pages 499–526. URL: http://dl.acm.org/citation.cfm?id=944801 (visited on 04/28/2015) (cited on pages 104, 106).

Cai, Zhuhua, Zekai J Gao, Shangyu Luo, Luis L Perez, Zografoula Vagena, and Christopher Jermaine (2014). "A comparison of platforms for implementing and running very large scale machine learning algorithms". In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, pages 1371–1382 (cited on page 103).

Calandriello, Daniele, Alessandro Lazaric, and Michal Valko (2015). "Large-scale semi-supervised learning with online spectral graph sparsification". In: *Resource-Efficient*

Machine Learning workshop at International Conference on Machine Learning (cited on page 106).

Calandriello, Daniele, Alessandro Lazaric, and Michal Valko (2016). "Analysis of Nyström method with sequential ridge leverage score sampling". In: *Uncertainty in Artificial Intelligence* (cited on page 48).

— (2017a). "Distributed Sequential Sampling for Kernel Matrix Approximation". In: *AISTATS* (cited on pages 28, 31, 38, 41, 47, 57, 61, 72).

— (2017b). "Efficient Second-Order Online Kernel Learning with Adaptive Embedding". In: *Advances in Neural Information Processing Systems* (cited on pages 118, 129, 131).

— (2017c). "Second-Order Kernel Online Convex Optimization with Adaptive Sketching". In: *International Conference on Machine Learning* (cited on pages 38, 49, 51, 61, 62, 117, 119–121, 123, 129).

Caponnetto, Andrea and Ernesto De Vito (2005). "Fast Rates for Regularized Least-squares Algorithm". In: *Foundations of Computational Mathematics* (cited on pages 29, 96).

Cavallanti, Giovanni, Nicolo Cesa-Bianchi, and Claudio Gentile (2007). "Tracking the best hyperplane with a simple budget perceptron". In: *Machine Learning* 69.2-3, pages 143–167 (cited on pages 117, 129, 132).

Cesa-Bianchi, Nicolo, Alex Conconi, and Claudio Gentile (2005). "A second-order perceptron algorithm". In: *SIAM Journal on Computing* 34.3, pages 640–668 (cited on pages 116, 121).

Cesa-Bianchi, Nicolo and Gábor Lugosi (2006). *Prediction, learning, and games*. Cambridge university press (cited on page 134).

Chapelle, Olivier, Bernhard Schlkopf, and Alexander Zien (2010). *Semi-Supervised Learning*. 1st. The MIT Press. ISBN: 0262514125, 9780262514125 (cited on pages 98, 102).

Ching, Avery, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan (2015). "One trillion edges: graph processing at Facebook-scale". In: *Proceedings of the VLDB Endowment* 8.12, pages 1804–1815 (cited on page 103).

Cohen, Michael B., Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford (2015a). "Uniform sampling for matrix approximation". In: *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*. ACM, pages 181–190 (cited on pages 20, 23, 46).

Cohen, Michael B., Cameron Musco, and Christopher Musco (2015b). "Ridge Leverage Scores for Low-Rank Approximation". In: *arXiv:1511.07263 [cs]*. arXiv: 1511.07263. URL: http://arxiv.org/abs/1511.07263 (visited on 12/08/2015) (cited on pages 20, 28, 45).

Cohen, Michael B, Cameron Musco, and Jakub Pachocki (2016). "Online Row Sampling". In: *International Workshop on Approximation, Randomization, and Combinatorial Optimization APPROX* (cited on pages 20, 28, 41, 45, 51, 92, 98, 100, 101).

Cortes, Corinna, Mehryar Mohri, Dmitry Pechyony, and Ashish Rastogi (2008). "Stability of transductive regression algorithms". In: *Proceedings of the 25th international conference on Machine learning*. ACM, pages 176–183 (cited on pages 103, 104, 106–108).

Cover, Thomas M and Joy A Thomas (2012). *Elements of information theory*. John Wiley & Sons (cited on page 28).

Dekel, Ofer, Shai Shalev-Shwartz, and Yoram Singer (2008). "The forgetron: A kernel-based perceptron on a budget". In: *SIAM Journal on Computing* 37.5, pages 1342–1372 (cited on pages 117, 129, 132).

Drineas, Petros, Malik Magdon-Ismail, Michael W Mahoney, and David P. Woodruff (2012). "Fast approximation of matrix coherence and statistical leverage". In: *International Conference on Machine Learning* (cited on pages 21–23).

Drineas, Petros, Michael W Mahoney, and S Muthukrishnan (2008). "Relative-error CUR matrix decompositions". In: *SIAM Journal on Matrix Analysis and Applications* 30.2, pages 844–881 (cited on pages 25, 99).

Duchi, John, Elad Hazan, and Yoram Singer (2011). "Adaptive subgradient methods for online learning and stochastic optimization". In: *Journal of Machine Learning Research* 12.Jul, pages 2121–2159. URL: http://www.jmlr.org/papers/v12/duchi11a.html (visited on 02/24/2017) (cited on page 116).

Fergus, Rob, Yair Weiss, and Antonio Torralba (2009). "Semi-Supervised Learning in Gigantic Image Collections". In: *Proceedings of NIPS*, pages 522–530 (cited on pages 102–104, 111).

Gammerman, Alex, Yuri Kalnishkan, and Vladimir Vovk (2004). "On-line prediction with kernels and the complexity approximation principle". In: *Uncertainty in artificial intelligence*. AUAI Press, pages 170–176 (cited on pages 116, 118).

Garcke, Jochen and Michael Griebel (2005). "Semi-supervised learning with sparse grids". In: *Proc. of the 22nd ICML Workshop on Learning with Partially Classified Training Data* (cited on page 104).

Ghashami, Mina, Edo Liberty, Jeff M. Phillips, and David P. Woodruff (2015). "Frequent Directions : Simple and Deterministic Matrix Sketching". In: *arXiv:1501.01711 [cs]*. arXiv: 1501.01711. URL: http://arxiv.org/abs/1501.01711 (visited on 10/23/2015) (cited on page 131).

Ghashami, Mina, Edo Liberty, Jeff M Phillips, and David P Woodruff (2016a). "Frequent directions: Simple and deterministic matrix sketching". In: *SIAM Journal on Computing* 45.5, pages 1762–1792 (cited on page 126).

Ghashami, Mina, Daniel J Perry, and Jeff Phillips (2016b). "Streaming kernel principal component analysis". In: *AISTATS* (cited on pages 97, 127).

Gleich, David F. and Michael W. Mahoney (2015). "Using Local Spectral Methods to Robustify Graph-Based Learning Algorithms". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '15. Sydney, NSW, Australia: ACM, pages 359–368. ISBN: 978-1-4503-3664-2. DOI: 10.1145/2783258.2783376 (cited on page 103).

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The elements of statistical learning*. Springer (cited on page 15).

Hazan, Elad, Adam Kalai, Satyen Kale, and Amit Agarwal (2006). "Logarithmic regret algorithms for online convex optimization". In: *Conference on Learning Theory*. Springer, pages 499–513 (cited on pages 29, 49, 88, 116–120, 139, 141).

He, Wenwu and James T. Kwok (2014). "Simple randomized algorithms for online learning with kernels". In: *Neural Networks* 60, pages 17–24. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.07.006. URL: http://www.sciencedirect.com/science/article/pii/S0893608014001634 (visited on 07/05/2016) (cited on page 117).

Horn, Roger A and Charles R Johnson (2012). *Matrix analysis*. Cambridge university press (cited on pages 14, 31, 35).

Jebara, Tony, Jun Wang, and Shih-Fu Chang (2009). "Graph construction and b-matching for semi-supervised learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, pages 441–448 (cited on page 104).

Ji, Ming, Tianbao Yang, Binbin Lin, Rong Jin, and Jiawei Han (2012). "A Simple Algorithm for Semi-supervised Learning with Improved Generalization Error Bound". In: *Proceedings of ICML*. arXiv: 1206.6412. URL: http://arxiv.org/abs/1206.6412 (cited on page 104).

Kapralov, Michael, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford (2014). "Single pass spectral sparsification in dynamic streams". In: *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*. IEEE, pages 561–570 (cited on page 112).

Kapralov, Michael and Rina Panigrahy (2012). "Spectral sparsification via random spanners". In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ACM, pages 393–398 (cited on page 100).

Kelner, Jonathan A. and Alex Levin (2013). "Spectral Sparsification in the Semi-streaming Setting". en. In: *Theory of Computing Systems* 53.2, pages 243–262. ISSN: 1432-4350, 1433-0490. DOI: 10.1007/s00224-012-9396-1. URL: http://link.springer.com/10.1007/s00224-012-9396-1 (visited on 04/22/2015) (cited on pages 12, 18, 20, 98, 100–102).

Kelner, Jonathan A, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu (2013). "A simple, combinatorial algorithm for solving SDD systems in nearly-linear time". In: *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, pages 911–920 (cited on page 100).

Kingma, Diederik and Jimmy Ba (2015). "Adam: A method for stochastic optimization". In: *ICLR* (cited on page 116).

Kivinen, J., A.J. Smola, and R.C. Williamson (2004a). "Online Learning with Kernels". en. In: *IEEE Transactions on Signal Processing* 52.8. (Visited on 02/18/2017) (cited on pages 115, 116, 122).

Kivinen, Jyrki, Alexander Smola, and Robert Williamson (2004b). "Online Learning with Kernels". In: *IEEE Transactions on Signal Processing* 52.8, pages 2165–2176 (cited on page 115).

Koutis, Ioannis, Alex Levin, and Richard Peng (2012). "Improved spectral sparsification and numerical algorithms for SDD matrices". In: *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*. Volume 14. LIPIcs, pages 266–277 (cited on page 100).

Koutis, Ioannis, Gary L. Miller, and Richard Peng (2011). "A Nearly-m log n Time Solver for SDD Linear Systems". In: *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS*, pages 590–598 (cited on pages 104, 106).

Koutis, Ioannis and Shen Chen Xu (2016). "Simple parallel and distributed algorithms for spectral graph sparsification". In: *ACM Transactions on Parallel Computing (TOPC)* 3.2, page 14 (cited on page 100).

Kulesza, Alex, Ben Taskar, et al. (2012). "Determinantal point processes for machine learning". In: *Foundations and Trends® in Machine Learning* 5.2–3, pages 123–286 (cited on page 29).

Kumar, Sanjiv, Mehryar Mohri, and Ameet Talwalkar (2012). "Sampling Methods for the Nyström Method". In: *J. Mach. Learn. Res.* 13.1, pages 981–1006. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2503308.2343678 (cited on page 103).

Kyng, Rasmus, Jakub Pachocki, Richard Peng, and Sushant Sachdeva (2017). "A framework for analyzing resparsification algorithms". In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, pages 2032–2043 (cited on pages 60, 98, 102).

Le, Quoc, Tamás Sarlós, and Alex J Smola (2013). "Fastfood - Approximating kernel expansions in loglinear time". In: *International Conference on Machine Learning* (cited on pages 117, 118).

Le, Trung, Tu Nguyen, Vu Nguyen, and Dinh Phung (2016). "Dual Space Gradient Descent for Online Learning". In: *Advances In Neural Information Processing Systems*, pages 4583–4591 (cited on page 132).

Lee, Yin Tat and He Sun (2015). "Constructing Linear-Sized Spectral Sparsification in Almost-Linear Time". In: *arXiv preprint arXiv:1508.03261* (cited on page 19).

Leskovec, Jure, Jon Kleinberg, and Christos Faloutsos (2005). "Graphs over time: densification laws, shrinking diameters and possible explanations". In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, pages 177–187 (cited on page 112).

Levy, Haim (2015). *Stochastic dominance: Investment decision making under uncertainty*. Springer (cited on page 82).

Liu, Wei, Junfeng He, and Shih-Fu Chang (2010). "Large graph construction for scalable semi-supervised learning". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 679–686 (cited on page 104).

Lu, Jing, Steven C.H. Hoi, Jialei Wang, Peilin Zhao, and Zhi-Yong Liu (2016). "Large Scale Online Kernel Learning". In: *Journal of Machine Learning Research* 17.47, pages 1–43 (cited on pages 117, 118, 129, 132).

Luo, Haipeng, Alekh Agarwal, Nicolo Cesa-Bianchi, and John Langford (2016). "Efficient second-order online learning via sketching". In: *Neural Information Processing Systems* (cited on pages 116, 118–122, 124–126, 128, 131).

Minsker, Stanislav (2011). "On some extensions of Bernstein's inequality for self-adjoint operators". In: *arXiv preprint arXiv:1112.5448* (cited on page 64).

Musco, Cameron and Christopher Musco (2016). *Provably useful kernel matrix approximation in linear time*. Technical report. arXiv: 1605.07583. URL: http://arxiv.org/abs/1605.07583 (cited on page 38).

— (2017). "Recursive Sampling for the Nyström Method". In: *NIPS* (cited on pages 12, 20, 38, 44–46, 61, 62, 68, 92–94, 96, 97).

Nesterov, Yurii and Arkadii Nemirovskii (1994). *Interior-point polynomial algorithms in convex programming*. SIAM (cited on page 124).

Orabona, Francesco and Koby Crammer (2010). "New adaptive algorithms for online classification". In: *Neural Information Processing Systems*. Curran Associates Inc., pages 1840–1848 (cited on page 116).

Orabona, Francesco, Joseph Keshet, and Barbara Caputo (2008). "The projectron: a bounded kernel-based perceptron". In: *Proceedings of the 25th international conference on Machine learning*. ACM, pages 720–727 (cited on pages 117, 129, 132).

Pachocki, Jakub (2016). "Analysis of Resparsification". In: *arXiv preprint arXiv:1605.08194* (cited on pages 60, 102).

Papailiopoulos, Dimitris, Anastasios Kyrillidis, and Christos Boutsidis (2014). "Provable deterministic leverage score sampling". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 997–1006 (cited on page 65).

Patel, Raajen, Tom Goldstein, Eva Dyer, Azalia Mirhoseini, and Richard Baraniuk (2016). "Deterministic Column Sampling for Low-Rank Matrix Approximation: Nyström vs. Incomplete Cholesky Decomposition". In: *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, pages 594–602 (cited on page 65).

Rasmussen, Carl Edward and Christopher K. I. Williams (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press. ISBN: 026218253X (cited on page 28).

Rudi, Alessandro, Raffaello Camoriano, and Lorenzo Rosasco (2015). "Less is more: Nystrom computational regularization". In: *Neural Information Processing Systems* (cited on pages 62, 91, 95, 96).

Saluja, Avneesh, Hany Hassan, Kristina Toutanova, and Chris Quirk (2014). "Graph-based semi-supervised learning of translation models from monolingual data". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL), Baltimore, Maryland, June* (cited on page 103).

Schölkopf, Bernhard, Alexander J. Smola, and Klaus-Robert Müller (1999). "Kernel principal component analysis". In: *Advances in kernel methods*. MIT Press Cambridge, MA, USA, pages 327–352 (cited on pages 91, 97).

Silva, Marcel K, Nicholas JA Harvey, and Cristiane M Sato (2016). "Sparse sums of positive semidefinite matrices". In: *ACM Transactions on Algorithms (TALG)* 12.1, page 9 (cited on page 19).

Spielman, Daniel A. (2017). *Laplacian linear equations, sparsification local graph clustering, low-stretch spanning trees, and so on.* URL: https://sites.google.com/a/yale.edu/laplacian/ (visited on 10/20/2017) (cited on page 100).

Spielman, Daniel A and Nikhil Srivastava (2011). "Graph sparsification by effective resistances". In: *SIAM Journal on Computing* 40.6, pages 1913–1926 (cited on pages 99, 100).

Srinivas, Niranjan, Andreas Krause, Matthias Seeger, and Sham M Kakade (2010). "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design". In: *International Conference on Machine Learning*, pages 1015–1022 (cited on pages 28, 29, 49, 121).

Subramanya, Amarnag and Partha Pratim Talukdar (2014). "Graph-Based Semi-Supervised Learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8.4, pages 1–125 (cited on page 103).

Sun, Yi, Jürgen Schmidhuber, and Faustino J. Gomez (2012). "On the size of the online kernel sparsification dictionary". In: *International Conference on Machine Learning* (cited on page 130).

Talwalkar, Ameet, Sanjiv Kumar, and Henry A. Rowley (2008). "Large-Scale Manifold Learning". In: *Computer Vision and Pattern Recognition (CVPR)*. URL: http://www.sanjivk.com/largeManifold.pdf (cited on page 103).

Thrun, Sebastian (1998). "Lifelong learning algorithms". In: *Learning to learn* 8, pages 181–209 (cited on page 65).

Tropp, Joel A (2012). "User-friendly tail bounds for sums of random matrices". In: *Foundations of computational mathematics* 12.4, pages 389–434 (cited on page 22).

Tropp, Joel Aaron (2011). "Freedman's inequality for matrix martingales". In: *Electronic Communications in Probability* 16, pages 262–270 (cited on page 73).

— (2015). "An introduction to matrix concentration inequalities". In: *Foundations and Trends in Machine Learning* 8.1-2, pages 1–230 (cited on pages 32, 82).

Tsang, Ivor W. and James T. Kwok (2006). "Large-Scale Sparsified Manifold Regularization." In: *NIPS*. Edited by Bernhard Schölkopf, John C. Platt, and Thomas Hoffman. MIT Press, pages 1401–1408. ISBN: 0-262-19568-2. URL: http://dblp.uni-trier.de/db/conf/nips/nips2006.html#TsangK06 (cited on page 104).

Von Luxburg, Ulrike (2007). "A tutorial on spectral clustering". In: *Statistics and computing* 17.4, pages 395–416 (cited on page 98).

Wang, Zhuang, Koby Crammer, and Slobodan Vucetic (2012). "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training". In: *Journal of Machine Learning Research* 13.Oct, pages 3103–3131 (cited on page 117).

Wathen, Andrew J. and Shengxin Zhu (2015). "On spectral distribution of kernel matrices related to radial basis functions". In: *Numerical Algorithms* 70.4, pages 709–726. ISSN:

1572-9265. DOI: 10.1007/s11075-015-9970-0. URL: http://dx.doi.org/10.1007/s11075-015-9970-0 (cited on page 130).

Williams, Christopher and Matthias Seeger (2001). "Using the Nystrom method to speed up kernel machines". In: *Neural Information Processing Systems* (cited on pages 19, 21, 91, 117).

Woodruff, David P. (2014). "Sketching as a tool for numerical linear algebra". In: *arXiv preprint arXiv:1411.4357*. URL: http://arxiv.org/abs/1411.4357 (visited on 04/22/2015) (cited on page 92).

Xu, Yi, Haiqin Yang, Lijun Zhang, and Tianbao Yang (2017). "Efficient Non-oblivious Randomized Reduction for Risk Minimization with Improved Excess Risk Guarantee". In: *AAAI Conference on Artificial Intelligence* (cited on page 142).

Yang, Tianbao, Yu-Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou (2012). "Nyström method vs random fourier features: A theoretical and empirical comparison". In: *Advances in neural information processing systems*, pages 476–484. (Visited on 05/10/2017) (cited on page 117).

Yang, Y., M. Pilanci, and M. J. Wainwright (2017). "Randomized Sketches for Kernels: Fast and Optimal Non-Parametric Regression". In: *Annals of Statistics, to appear* (cited on page 130).

El-Yaniv, Ran and Dmitry Pechyony (2006). "Stable transductive learning". In: *Proceedings of COLT*. Springer, pages 35–49 (cited on page 106).

Yu, Kai and Shipeng Yu (2005). "Blockwise supervised inference on large graphs". In: *Proc. of the 22nd ICML Workshop on Learning* (cited on page 104).

Zhang, Yuchen, John C. Duchi, and Martin J. Wainwright (2015). "Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates". In: *Journal Machine Learning Research* 16, pages 3299–3340 (cited on page 96).

Zhao, Peilin, Jialei Wang, Pengcheng Wu, Rong Jin, and Steven C H Hoi (2012). "Fast bounded online gradient descent algorithms for scalable kernel-based online learning". In: *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*. Volume 1, pages 169–176. ISBN: 9781450312851 (cited on pages 129, 132).

Zhdanov, Fedor and Yuri Kalnishkan (2010). "An Identity for Kernel Ridge Regression". en. In: *Algorithmic Learning Theory*. Edited by Springer. Lecture Notes in Computer Science, pages 405–419 (cited on pages 116, 119, 122, 144).

Zhu, C. and H. Xu (2015). "Online Gradient Descent in Function Space". In: *ArXiv e-prints* (cited on pages 116, 118, 122, 123, 126).

Zhu, Xiaojin (2008). *Semi-Supervised Learning Literature Survey*. Technical report 1530. U. of Wisconsin-Madison (cited on page 102).

Zhu, Xiaojin, Zoubin Ghahramani, and John Lafferty (2003). "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions". In: *Proceedings of ICML*, pages 912–919 (cited on pages 98, 103, 105).

Zhu, Xiaojin and John D. Lafferty (2005). "Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning". In: *Proceedings of ICML*, pages 1052–1059. DOI: 10.1145/1102351.1102484. URL: http://doi.acm.org/10.1145/1102351.1102484 (cited on page 104).

Zinkevich, Martin (2003). "Online Convex Programming and Generalized Infinitesimal Gradient Ascent". In: *International Conference on Machine Learning*, pages 928–936 (cited on pages 115, 120).