Distributed Adaptive Sampling for Kernel Matrix Approximation

Daniele Calandriello **Alessandro Lazaric** **Michal Valko**
SequeL team, INRIA Lille - Nord Europe

Abstract

Most kernel-based methods, such as kernel regression, kernel PCA, ICA, or k-means clustering, do not scale to large datasets, because constructing and storing the kernel matrix $\mathbf{K}_n$ requires at least $\mathcal{O}(n^2)$ time and space for n samples. Recent works [1, 9] show that sampling points with replacement according to their ridge leverage scores (RLS) generates small dictionaries of relevant points with strong spectral approximation guarantees for $\mathbf{K}_n$. The drawback of RLS-based methods is that computing exact RLS requires constructing and storing the whole kernel matrix. In this paper, we introduce SQUEAK, a new algorithm for kernel approximation based on RLS sampling that *sequentially* processes the dataset, storing a dictionary which creates accurate kernel matrix approximations with a number of points that only depends on the effective dimension $d_{\text{eff}}(\gamma)$ of the dataset. Moreover since all the RLS estimations are efficiently performed using only the small dictionary, SQUEAK never constructs the whole matrix $\mathbf{K}_n$, runs in linear time $\tilde{\mathcal{O}}(nd_{\text{eff}}(\gamma)^3)$ w.r.t. n, and requires only a single pass over the dataset. We also propose a parallel and distributed version of SQUEAK achieving similar accuracy in as little as $\tilde{\mathcal{O}}(\log(n)d_{\text{eff}}(\gamma)^3)$ time.

1 Introduction

One of the major limits of kernel ridge regression (KRR), kernel PCA [13], and other kernel methods is that for n samples storing and manipulating the kernel matrix $\mathbf{K}_n$ requires $\mathcal{O}(n^2)$ space, which becomes rapidly infeasible for even a relatively small n. For larger sizes (or streams) we cannot even afford to store or process the data on as single machine. Many solutions focus on how to scale kernel methods by reducing its space (and time) complexity without compromising the prediction accuracy. A popular approach is to construct low-rank approximations of the kernel matrix by randomly selecting a subset (dictionary) of m columns from $\mathbf{K}_n$, thus reducing the space complexity to $\mathcal{O}(nm)$. These methods, often referred to as *Nyström approximations*, mostly differ in the distribution used to sample the columns of $\mathbf{K}_n$ and the construction of low-rank approximations. Both of these choices significantly affect the accuracy of the resulting approximation [12]. Bach [2] showed that uniform sampling preserves the prediction accuracy of KRR (up to ε) only when the number of columns m is proportional to the maximum degree of freedom of the kernel matrix. This may require sampling $\mathcal{O}(n)$ columns in datasets with high coherence [6], i.e., a kernel matrix with weakly correlated columns. On the other hand, Alaoui and Mahoney [1] showed that sampling columns according to their ridge leverage scores (RLS) (i.e., a measure of the influence of a point on the regression) produces an accurate Nyström approximation with only a number of columns m proportional to the average degrees of freedom of the matrix, called *effective dimension*. Unfortunately, the complexity of computing RLS requires storing the whole kernel matrix, thus making this approach infeasible. However, Alaoui and Mahoney [1] proposed a fast method to compute a constant-factor approximation of the RLS and showed that accuracy and space complexity are close to the case of sampling with exact RLS at the cost of an extra dependency on the inverse of the minimal eigenvalue of the kernel matrix. Unfortunately, the minimal eigenvalue can be arbitrarily small in many problems. Calandriello et al. [3] addressed this issue by processing the dataset *incrementally* and updating estimates of the ridge leverage scores, effective dimension, and Nyström approximations on-the-fly. Although the space complexity of the resulting algorithm (INK-Estimate) does not depend on the minimal eigenvalue anymore, it introduces a dependency on the largest eigenvalue of $\mathbf{K}_n$, which in the worst case can be as big as n, thus losing the ad-

Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.

vantage of the method. In this paper we introduce an algorithm for SeQUEntial Approximation of Kernel matrices (SQUEAK), a new algorithm that builds on INK-Estimate, but uses *unnormalized* RLS. This improvement, together with a new analysis, opens the way to major improvements over current leverage sampling methods (see Sect. 6 for a comparison with existing methods) closely matching the dictionary size achieved by exact RLS sampling. First, unlike INK-Estimate, SQUEAK is simpler, does not need to compute an estimate of the effective dimension for normalization, and exploits a simpler, more accurate RLS estimator. This new estimator only requires access to the points stored in the dictionary. Since the size of the dictionary is much smaller than the n, SQUEAK needs to actually observe only a fraction of the kernel matrix $\mathbf{K}_n$, resulting in a runtime linear in n. Second, since our dictionary updates require only access to local data, our algorithm allows for distributed processing where machines operating on different dictionaries do not need to communicate with each other. In particular, intermediate dictionaries can be extracted in parallel from small portions of the dataset and they can be later merged in a hierarchical way. Third, the sequential nature of SQUEAK requires a more sophisticated analysis that take into consideration the complex interactions and dependencies between successive resampling steps. The analysis of SQUEAK builds on a new martingale argument that could be of independent interest for similar online resampling schemes. Moreover, our SQUEAK can naturally incorporate new data without the need of recomputing the whole resparsification from scratch and therefore it can be applied in streaming settings. We note there exist other ways to avoid the intricate dependencies with simpler analysis, for example by resampling [9], but with negative algorithmic side effects: these methods need to pass through the dataset multiple times. SQUEAK passes *through the dataset only once*[1] and is therefore the first provably accurate kernel approximation algorithm that can handle both *streaming and distributed* settings.

2 Background

In this section, we introduce the notation and basics of kernel approximation used through the paper.

Notation. We use curly capital letters $\mathcal{A}$ for collections. We use upper-case bold letters $\mathbf{A}$ for matrices and operators, lower-case bold letters $\mathbf{a}$ for vectors, and lower-case letters a for scalars, with the exception of f, g, and h which denote functions. We denote by $[\mathbf{A}]_{ij}$ and $[\mathbf{a}]_i$, the (i,j) element of a matrix and ith element of a vector respectively. We denote by $\mathbf{I}_n \in \mathbb{R}^{n\times n}$, the identity matrix of dimension n and by $\mathrm{Diag}(\mathbf{a}) \in \mathbb{R}^{n\times n}$ the diagonal matrix with the vector $\mathbf{a} \in \mathbb{R}^n$ on the diagonal. We use $\mathbf{e}_{n,i} \in \mathbb{R}^n$ to denote the indicator vector for element i of dimension n. 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 that $\mathbf{A}-\mathbf{B}$ is a Positive Semi-Definite (PSD) matrix or operator. Finally, the set of integers between 1 and n is denoted as $[n] := \{1,\dots,n\}$, and between i and j as $[i:j] := \{i,\dots,j\}$.

Kernel. We consider a positive definite kernel function $\mathcal{K} : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ and we denote with $\mathcal{H}$ its induced Reproducing Kernel Hilbert Space (RKHS), and with $\varphi : \mathcal{X} \to \mathcal{H}$ its corresponding feature map. Using φ, and without loss of generality, for the rest of the paper we will replace $\mathcal{H}$ with a high dimensional space $\mathbb{R}^D$ where D is large and potentially infinite. With this notation, the kernel evaluated between to points can be expressed as $\mathcal{K}(\mathbf{x},\mathbf{x}') = \langle \mathcal{K}(\mathbf{x},\cdot), \mathcal{K}(\mathbf{x}',\cdot)\rangle_{\mathcal{H}} = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}')\rangle_{\mathcal{H}} = \varphi(\mathbf{x})^\mathsf{T}\varphi(\mathbf{x}')$. Given a dataset of points $\mathcal{D} = \{\mathbf{x}_t\}_{t=1}^n$, 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. We also define the feature vectors $\boldsymbol{\phi}_i = \varphi(\mathbf{x}_i) \in \mathbb{R}^D$ and after introducing the matrix $\boldsymbol{\Phi}_t = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \dots, \boldsymbol{\phi}_t] \in \mathbb{R}^{D\times t}$ we can rewrite the kernel matrix as $\mathbf{K}_t = \boldsymbol{\Phi}_t^\mathsf{T}\boldsymbol{\Phi}_t$.

Kernel approximation by column sampling. One of the most popular strategies to have low space complexity approximations of the kernel $\mathbf{K}_t$ is to randomly select a subset of its columns (possibly reweighted) and use them to perform the specific kernel task at hand (e.g., kernel regression). More precisely, we define a column dictionary as a collection $\mathcal{I}_t = \{(i, w_i)\}_{i=1}^t$, where the first term denotes the index of the column and w_i its weight, which is set to zero for all columns that are not retained. For the theoretical analysis, we conveniently keep the dimension of any dictionary $\mathcal{I}_t$ to t, while in practice, we only store the non-zero elements. In particular, we denote by $|\mathcal{I}_t|$ be the size of the dictionary corresponding to the elements with non-zero weights w_i. Associated with a column dictionary, there is a selection matrix $\mathbf{S}_t = \mathrm{Diag}(\sqrt{w_1}\dots\sqrt{w_t}) \in \mathbb{R}^{t\times t}$ such that for any matrix $\mathbf{A}_t \in \mathbb{R}^{t\times t}$, $\mathbf{A}_t\mathbf{S}_t$ returns a $t\times t$ matrix where the columns selected by $\mathcal{I}_t$ are properly reweighted and all other columns are set to 0. Despite the wide range of kernel applications, it is possible to show that in most of them, the quality of a dictionary can be mea-

[1] Note that there is an important difference in whether the method passes through *kernel matrix* only once or through the *dataset* only once, in the former, the algorithm may still need access one data point up to n times, thus making it unsuitable for the streaming setting and less practical for distributed computation.

sured in terms of how well it approximates the projection associated to the kernel. In kernel regression, for instance, we use $\mathbf{K}_t$ to construct the projection (hat) matrix that projects the observed labels $\mathbf{y}_t$ to $\widehat{\mathbf{y}}_t$. In particular, let $\mathbf{P}_t = \mathbf{K}_t\mathbf{K}_t^+$ be the projection matrix (where $\mathbf{K}_t^+$ indicates the pseudoinverse), then $\widehat{\mathbf{y}}_t = \mathbf{P}_t\mathbf{y}_t$. If $\mathbf{K}_t$ is full-rank, then $\mathbf{P}_t = \mathbf{I}_t$ is the identity matrix and, we can reconstruct any target vector $\mathbf{y}_t$ exactly. On the other hand, the only sampling scheme which guarantees to properly approximate a full rank $\mathbf{P}_t$ requires all columns to be represented in $\mathcal{I}_t$. In fact, all columns have the same "importance" and no low-space approximation is possible. Nonetheless, kernel matrices are often either rank deficient or have extremely small eigenvalues (exponentially decaying spectrum), as a direct (and desired) consequence of embedding low dimensional points $\mathbf{x}_i$ into a high dimensional RKHS. In this case, after soft thresholding the smaller eigenvalues to a given value γ, $\mathbf{K}_t$ can be effectively approximated using a small subset of columns. This is equivalent to approximating the γ-ridge projection matrix

$$\mathbf{P}_t \overset{\text{def}}{=} (\mathbf{K}_t + \gamma\mathbf{I})^{-1/2}\mathbf{K}_t(\mathbf{K}_t + \gamma\mathbf{I})^{-1/2}.$$

We say that a column dictionary is accurate if the following condition is satisfied.

Definition 1. *A dictionary $\mathcal{I}_t = \{(i, w_i)\}_{i=1}^t$ and its associated selection matrix $\mathbf{S}_t \in \mathbb{R}^{t\times t}$ are ε-accurate w.r.t. a kernel matrix $\mathbf{K}_t = \mathbf{K}_t^{1/2}\mathbf{K}_t^{1/2}$ if* [2]

$$\|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\| \leq \varepsilon, \tag{1}$$

where for a given $\gamma > 0$, the approximated projection matrix is defined as

$$\widetilde{\mathbf{P}}_t \overset{\text{def}}{=} (\mathbf{K}_t + \gamma\mathbf{I}_t)^{-\frac{1}{2}}\mathbf{K}_t^{1/2}\mathbf{S}_t\mathbf{S}_t^\mathsf{T}\mathbf{K}_t^{1/2}(\mathbf{K}_t + \gamma\mathbf{I}_t)^{-\frac{1}{2}}.$$

Notice that this definition of accuracy is purely theoretical, since $\widetilde{\mathbf{P}}_t$ is never computed. Nonetheless, as illustrated in Sect. 5, ε-accurate dictionaries can be used to construct suitable kernel approximation in a wide range of problems.

Ridge leverage scores sampling. Alaoui and Mahoney [1] showed that an ε-accurate dictionary can be obtained by sampling columns proportionally to their γ-ridge leverage scores (RLS) defined as follows.

Definition 2. *Given a kernel matrix $\mathbf{K}_t \in \mathbb{R}^{t\times t}$, the γ-ridge leverage score (RLS) of column $i \in [t]$ is*

$$\tau_{t,i} = \mathbf{e}_{t,i}^\mathsf{T}\mathbf{K}_t(\mathbf{K}_t + \gamma\mathbf{I}_t)^{-1}\mathbf{e}_{t,i}, \tag{2}$$

Furthermore, the effective dimension $d_{\mathit{eff}}(\gamma)_t$ of the kernel matrix $\mathbf{K}_t$ is defined as

$$d_{\mathit{eff}}(\gamma)_t = \sum_{i=1}^t \tau_{t,i}(\gamma) = \operatorname{Tr}\left(\mathbf{K}_t(\mathbf{K}_t + \gamma\mathbf{I}_t)^{-1}\right). \tag{3}$$

[2] the matrix norm we use is the operator (induced) norm

Algorithm 1 The SQUEAK algorithm

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

Initialize $\mathcal{I}_0$ as empty, $\overline{q}$ (see Thm. 1)
for $t = 1, \dots, n$ **do**
  Read point $\mathbf{x}_t$ from $\mathcal{D}$
  $\overline{\mathcal{I}} = \mathcal{I}_{t-1} \cup \{(t, \widetilde{p}_{t-1,t}=1, q_{t-1,t}=\overline{q})\}$ ▷EXPAND
  $\mathcal{I}_t =$ DICT-UPDATE$(\overline{\mathcal{I}})$ using Eq. 4
end for

Subroutine 1 The DICT-UPDATE algorithm

Input: $\overline{\mathcal{I}}$
Output: $\mathcal{I}_t$

Initialize $\mathcal{I}_t = \emptyset$
for all $i \in \{1, \dots, t\}$ **do** ▷SHRINK
  if $q_{t-1,i} \neq 0$ **then**
    Compute $\widetilde{\tau}_{t,i}$ using $\overline{\mathcal{I}}$
    Set $\widetilde{p}_{t,i} = \min\{\widetilde{\tau}_{t,i}, \widetilde{p}_{t-1,i}\}$
    Set $q_{t,i} \sim \mathcal{B}(\widetilde{p}_{t,i}/\widetilde{p}_{t-1,i}, q_{t-1,i})$
  else
    $\widetilde{p}_{t,i} = \widetilde{p}_{t-1,i}$ and $q_{t,i} = q_{t-1,i}$
  end if
end for

The RLS can be interpreted and derived in many ways, and they are well studied [5, 4, 16] in the linear setting (e.g. $\boldsymbol{\phi}_t = \mathbf{x}_t$). Patel et al. [11] used them as a measure of incoherence to select important points, but their deterministic algorithm provides guarantees only when $\mathbf{K}_t$ is exactly low-rank. Here we notice that

$$\tau_{t,i} = \mathbf{e}_{t,i}^\mathsf{T}\mathbf{K}_t(\mathbf{K}_t + \gamma\mathbf{I}_t)^{-1}\mathbf{e}_{t,i} = \mathbf{e}_{t,i}^\mathsf{T}\mathbf{P}_t\mathbf{e}_{t,i},$$

which means that they correspond to the diagonal elements of the $\mathbf{P}_t$ itself. Intuitively, this correspond to selecting each column i with probability $p_{t,i} = \tau_{t,i}$ will capture the most important columns to define $\mathbf{P}_t$, thus minimizing the approximation error $\|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\|$. More formally, Alaoui and Mahoney [1] state the following.

Proposition 1. *Let $\varepsilon \in [0, 1]$ and $\mathcal{I}_n$ be the dictionary built with m columns randomly selected proportionally to RLSs $\{\tau_{n,i}\}$ with weight $w_i = 1/(m\tau_{n,i})$. If $m = \mathcal{O}(\frac{1}{\varepsilon^2}d_{\mathit{eff}}(\gamma)_n \log(\frac{n}{\delta}))$, then w.p. at least $1-\delta$, the corresponding dictionary is ε-accurate.*

Unfortunately, computing exact RLS requires storing $\mathbf{K}_n$ and this is seldom possible in practice. In the next section, we introduce SQUEAK, an RLS-based incremental algorithm able to preserve the same accuracy of Prop. 1 *without* requiring to know the RLS in advance. We prove that it generates a dictionary only a constant factor larger than exact RLS sampling.

3 Sequential RLS Sampling

In the previous section, we showed that sampling proportionally to the RLS $\{\tau_{t,i}\}$ leads to a dictionary such that $\|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\| \leq \varepsilon$. Furthermore, since the RLS correspond to the diagonal entries of $\mathbf{P}_t$, an accurate approximation $\widetilde{\mathbf{P}}_t$ may be used in turn to compute accurate estimates of $\tau_{t,i}$. The SQUEAK algorithm (Alg. 1) builds on this intuition to sequentially process the kernel matrix $\mathbf{K}_n$ so that exact RLS computed on a small matrix ($\mathbf{K}_t$ with $t \ll n$) are used to create an ε-accurate dictionary, which is then used to estimate the RLS for bigger kernels, which are in turn used to update the dictionary and so on. While SQUEAK shares a similar structure with INK-ESTIMATE [3], the sampling probabilities are computed from different estimates of the RLS $\tau_{t,i}$ and no renormalization by an estimate of $d_{\text{eff}}(\gamma)_t$ is needed. Before giving the details of the algorithm, we redefine a dictionary as a collection $\mathcal{I} = \{(i, \widetilde{p}_i, q_i)\}_i$, where i is the index of the point $\mathbf{x}_i$ stored in the dictionary, $\widetilde{p}_i$ tracks the probability used to sample it, and q_i is the number of copies (multiplicity) of i. The weights are then computed as $w_i = q_i/(\overline{q}\widetilde{p}_i)$, where $\overline{q}$ is an algorithmic parameter discussed later. We use $\widetilde{p}_i$ to stress the fact that these probabilities will be computed as approximations of the actual probabilities that should be used to sample each point, i.e., their RLS τ_i.

SQUEAK receives as input a dataset $\mathcal{D} = \{\mathbf{x}_t\}_{t=1}^n$ and processes it *sequentially*. Starting with an empty dictionary $\mathcal{I}_0$, at each time step t, SQUEAK receives a new point $\mathbf{x}_t$. Adding a new point $\mathbf{x}_t$ to the kernel matrix can either decrease the importance of points observed before (i.e., if they are correlated with the new point) or leave it unchanged (i.e., if their corresponding kernel columns are orthogonal) and thus for any $i \leq t$, the RLS evolves as follows.

Lemma 1. *For any kernel matrix $\mathbf{K}_{t-1}$ at time $t-1$ and its extension $\mathbf{K}_t$ at time t, we have that the RLS are monotonically decreasing and the effective dimension is monotonically increasing,*

$$\frac{1}{\tau_{t-1,i}+1}\tau_{t-1,i} \leq \tau_{t,i} \leq \tau_{t-1,i}, \quad d_{\mathit{eff}}(\gamma)_t \geq d_{\mathit{eff}}(\gamma)_{t-1}.$$

The previous lemma also shows that the RLS cannot decrease too quickly and since $\tau_{t-1,i} \leq 1$, they can at most halve when $\tau_{t-1,i} = 1$. After receiving the new point $\mathbf{x}_t$, we need to update our dictionary $\mathcal{I}_{t-1}$ to reflect the changes of the $\tau_{t,i}$. We proceed in two phases. During the EXPAND phase, we directly add the new element $\mathbf{x}_t$ to $\mathcal{I}_{t-1}$ and obtain a temporary dictionary $\overline{\mathcal{I}}$, where the new element t is added with a sampling probability $\widetilde{p}_{t-1,t} = 1$ and a number of copies $q_{t-1,t} = \overline{q}$, i.e., $\overline{\mathcal{I}} = \mathcal{I}_{t-1} \cup \{(t, \widetilde{p}_{t-1,t} = 1, q_{t-1,t} = \overline{q})\}$. This increases our memory usage, forcing us to update the dictionary using DICT-UPDATE, in order to decrease its size. Given as input $\overline{\mathcal{I}}$, we use the following estimator to compute the approximate RLS $\widetilde{\tau}_{t,i}$,

$$\begin{aligned}\widetilde{\tau}_{t,i} &= (1-\varepsilon)\boldsymbol{\phi}_i^\mathsf{T}(\boldsymbol{\Phi}_t\overline{\mathbf{S}}\overline{\mathbf{S}}^\mathsf{T}\boldsymbol{\Phi}_t^\mathsf{T} + \gamma\mathbf{I})^{-1}\boldsymbol{\phi}_i \\ &= \tfrac{1-\varepsilon}{\gamma}(k_{i,i} - \mathbf{k}_{t,i}^\mathsf{T}\overline{\mathbf{S}}(\overline{\mathbf{S}}^\mathsf{T}\mathbf{K}_t\overline{\mathbf{S}} + \gamma\mathbf{I}_t)^{-1}\overline{\mathbf{S}}^\mathsf{T}\mathbf{k}_{t,i}), \quad (4)\end{aligned}$$

where ε is the accuracy parameter, γ is the regularization and $\overline{\mathbf{S}}$ is the selection matrix associated to $\overline{\mathcal{I}}$. This estimator follows naturally from a reformulation of the RLS. In particular, if we consider $\boldsymbol{\phi}_i$, the RKHS representation of $\mathbf{x}_i$, the RLS $\tau_{t,i}$ can be formulated as $\tau_{t,i} = \boldsymbol{\phi}_i^\mathsf{T}(\boldsymbol{\Phi}_t\mathbf{I}_t\boldsymbol{\Phi}_t^\mathsf{T} + \gamma\mathbf{I})^{-1}\boldsymbol{\phi}_i$, where we see that the importance of point $\mathbf{x}_i$ is quantified by how orthogonal (in the RKHS) it is w.r.t. the other points. Because we do not have access to all the columns ($\overline{\mathbf{S}}\overline{\mathbf{S}}^\mathsf{T} \neq \mathbf{I}_t$), similarly to what [4] did for the special case $\boldsymbol{\phi}_i = \mathbf{x}_i$, we choose to use $\widetilde{\tau}_{t,i} \approx \boldsymbol{\phi}^\mathsf{T}(\boldsymbol{\Phi}_t\overline{\mathbf{S}}\overline{\mathbf{S}}^\mathsf{T}\boldsymbol{\Phi}_t^\mathsf{T} + \gamma\mathbf{I})^{-1}\boldsymbol{\phi}_i$, and then we use the kernel trick to derive a form that we can actually compute, resulting in Eq. 4. The approximate RLSs are then used to define the new sampling probabilities as $\widetilde{p}_{t,i} = \min\{\widetilde{\tau}_{t,i}, \widetilde{p}_{t-1,i}\}$. For each element in $\overline{\mathcal{I}}$, the SHRINK step draws a sample from the binomial $\mathcal{B}(\widetilde{p}_{t,i}/\widetilde{p}_{t-1,i}, q_{t-1,i})$, where the minimum taken in the definition of $\widetilde{p}_{t,i}$ ensures that the binomial probability is well defined (i.e., $\widetilde{p}_{t,i} \leq \widetilde{p}_{t-1,i}$). This resampling step basically *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 RLS up to time t (with high probability). We see that the new element $\mathbf{x}_t$ is only added to the dictionary with a large number of copies (from 0 to $\overline{q}$) if its estimated relevance $\widetilde{p}_{t,t}$ is high, and that over time elements originally in $\mathcal{I}_{t-1}$ are stochastically reduced to reflect the reductions of the RLSs. The lower $\widetilde{p}_{t,i}$ w.r.t. $\widetilde{p}_{t-1,i}$, the lower the number of copies $q_{t,i}$ w.r.t. $q_{t-1,i}$. If the probability $\widetilde{p}_{t,i}$ continues to decrease over time, then $q_{t,i}$ may become zero, and the column i is completely dropped from the dictionary (by setting its weight to zero). The approximate RLSs enjoy the following guarantees.

Lemma 2. *Given an ε-approximate dictionary $\mathcal{I}_{t-1}$ of matrix $\mathbf{K}_{t-1}$, construct $\overline{\mathcal{I}}$ by adding element $(t, 1, \overline{q})$ to it, and compute the selection matrix $\overline{\mathbf{S}}$. Then for all i in $\overline{\mathcal{I}}$ such that $q_{t-1,i} \neq 0$, the estimator in Eq. 4 is α-accurate, i.e., it satisfies $\tau_{t,i}/\alpha \leq \widetilde{\tau}_{t,i} \leq \tau_{t,i}$, with $\alpha = (1+\varepsilon)/(1-\varepsilon)$. Moreover, given RLS $\tau_{t-1,i}$ and $\tau_{t,i}$, and two α-accurate RLSs, $\widetilde{\tau}_{t-1,i}$ and $\widetilde{\tau}_{t,i}$, the quantity $\min\{\widetilde{\tau}_{t,i}, \widetilde{\tau}_{t-1,i}\}$ is also an α-accurate RLS.*

This result is based on the property that whenever $\mathcal{I}_{t-1}$ is ε-accurate for $\mathbf{K}_{t-1}$, the projection matrix $\mathbf{P}_t$ can be approximated by $\overline{\mathbf{P}}_t$ constructed using the temporary dictionary $\overline{\mathcal{I}}$ and thus, the RLSs can be accurately estimated and used to update $\mathcal{I}_{t-1}$ and obtain a new ε-accurate dictionary for $\mathbf{K}_t$. Since $\widetilde{\tau}_{t,i}$ is used

to sample the new dictionary $\mathcal{I}_t$, we need each point to be sampled *almost* as frequently as with the true RLS $\tau_{t,i}$, which is guaranteed by the lower bound of Lem. 2. Since RLSs are always smaller or equal than 1, this could be trivially achieved by setting $\widetilde{\tau}_{t,i}$ to 1. Nonetheless, this would keep all columns in the dictionary. Consequently, we need to force the RLS estimate to decrease as much as possible, so that low probabilities allow reducing the space as much as possible. This is obtained by the upper bound in Lem. 2, which guarantees that the estimated RLS are always smaller than the exact RLS. As a result, SHRINK sequentially preserves the overall accuracy of the dictionary and *at the same time* keeps its size as small as possible, as shown in the following theorem.

Theorem 1. *Let $\varepsilon > 0$ be the accuracy parameter, $\gamma > 1$ the regularization, and $0 < \delta < 1$ the probability of failure. Given an arbitrary dataset $\mathcal{D}$ in input together with parameters ε, γ, and δ, we run* SQUEAK *with*

$$\overline{q} = \frac{39\alpha \log(2n/\delta)}{\varepsilon^2},$$

where $\alpha = (1+\varepsilon)/(1-\varepsilon)$. Then, w.p. at least $1-\delta$, SQUEAK *generates a sequence of random dictionaries $\{\mathcal{I}_t\}_{t=1}^n$ that are ε-accurate (Eq. 1) w.r.t. any of the intermediate kernels $\mathbf{K}_t$, and the size of the dictionaries is bounded as $\max_{t=1,\dots,n} |\mathcal{I}_t| \leq 3\overline{q} d_{\mathit{eff}}(\gamma)_n$.*

As a consequence, on a successful run the overall complexity of SQUEAK *is bounded as*

$$\textit{space complexity} = \left(\max_{t=1,\dots,n} |\mathcal{I}_t|\right)^2 \leq \left(3\overline{q} d_{\mathit{eff}}(\gamma)_n\right)^2,$$
$$\textit{time complexity} = \mathcal{O}\left(n d_{\mathit{eff}}(\gamma)_n^3 \overline{q}^3\right).$$

We show later that Thm. 1 is special case of Thm. 2 and give a sketch of the proof with the statement of Thm. 2. We postpone the discussion about this result and the comparison with previous results to Sect. 6 and focus now on the space and time complexity. Note that while the dictionaries $\mathcal{I}_t$ always contain t elements for notational convenience, SHRINK actually *never* updates the probabilities of the elements with $q_{t-1,i} = 0$. This feature is particularly important, since at any step t, it only requires to compute approximate RLSs for the elements which are actually included in $\mathcal{I}_{t-1}$ and the new point $\mathbf{x}_t$ (i.e., the elements in $\overline{\mathcal{I}}$) and thus it does not require recomputing the RLSs of points $\mathbf{x}_s$ ($s < t$) that have been dropped before! This is why SQUEAK computes an ε-accurate dictionary with a *single pass over the dataset*. Furthermore, the estimator in Eq. 4 does not require computing the whole kernel column $\mathbf{k}_{t,i}$ of dimension t. In fact, the components of $\mathbf{k}_{t,i}$, corresponding to points which are no longer in $\overline{\mathcal{I}}$, are directly set to zero when computing $\mathbf{k}_{t,i}^\mathsf{T}\overline{\mathbf{S}}$. As a result, for any new point $\mathbf{x}_t$ we need to evaluate $\mathcal{K}(\mathbf{x}_s, \mathbf{x}_t)$ only for the indices s in $\overline{\mathcal{I}}$. Therefore, SQUEAK never performs more than $n(3\overline{q} d_{\mathrm{eff}}(\gamma)_n)^2$ kernel evaluations, which means that it does not even need to observe large portions of the kernel matrix. Finally, the runtime is dominated by the n matrix inversions used in Eq. 4. Therefore, the total runtime is $\mathcal{O}(n(\max_{t=1,\dots,n} |\mathcal{I}_t|)^3) = \mathcal{O}(n d_{\mathrm{eff}}(\gamma)_n^3 \overline{q}^3)$. In the next section, we introduce DISQUEAK, which improves the runtime by independently constructing separate dictionaries in parallel and then merging them recursively to construct a final ε-accurate dictionary.

Algorithm 2 The distributed SQUEAK algorithm

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

```
Partition D into disjoint sub-datasets D_i
Initialize I_{D_i} = {(j, p̃_{0,i} = 1, q_{0,i} = q̄) : j ∈ D_i}
Build set S_1 = {I_{D_i}}_{i=1}^k
for h = 1, ..., k − 1 do
  if |S_h| > 1 then                                  ▷DICT-MERGE
    Pick two dictionaries I_D, I_D' from S_h
    Ī = I_D ∪ I_D'
    I_{D,D'} = DICT-UPDATE(Ī) using Eq. 5
    Place I_{D,D'} back into S_{h+1}
  else
    S_{h+1} = S_h
  end if
end for
Return I_D, the last dictionary in S_k
```

4 Distributed RLS Sampling

In this section, we show that a minor change in the structure of SQUEAK 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. DIstributed-SQUEAK (DISQUEAK, Alg. 2) partitions $\mathcal{D}$ over multiple machines and the (small) dictionaries that are generated from different portions of the dataset are integrated in a hierarchical way. The initial dataset is partitioned over k disjoint sub-datasets $\mathcal{D}_i$ with $i = 1, \dots, k$ and k dictionaries $\mathcal{I}_{\mathcal{D}_i} = \{(j, \widetilde{p}_{0,i} = 1, q_{0,i} = \overline{q}) : j \in \mathcal{D}_i\}$ 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 SQUEAK to generate the initial dictionaries. The dictionaries $\mathcal{I}_{\mathcal{D}_i}$ are added to

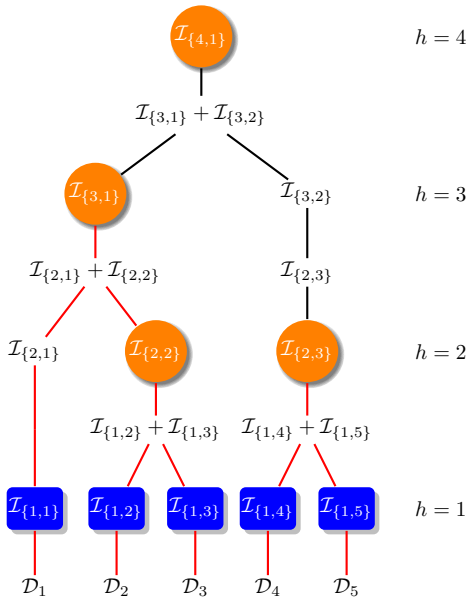

Figure 1: Merge tree for Alg. 2 with an arbitrary partitioning and merging scheme.

a dictionary collection $\mathcal{S}$, and following any predefined merge tree as in Fig. 1, these dictionaries are progressively merged together with DICT-MERGE. Given two input dictionaries $\mathcal{I}_{\mathcal{D}}$ and $\mathcal{I}_{\mathcal{D}'}$, we combine them into a single dictionary $\overline{\mathcal{I}}$ (the equivalent of the EXPAND phase in SQUEAK) and then DICT-UPDATE is run on the merged dictionaries to create an updated dictionary $\mathcal{I}_{\mathcal{D}\cup\mathcal{D}'}$, which is placed back in the dictionary collection $\mathcal{S}$. Since DICT-MERGE only takes the two dictionaries as input and does not require any information on the dictionaries in the rest of the tree, 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. Unlike in SQUEAK, DICT-UPDATE is run on the union of two distinct dictionaries rather than one dictionary and a new single point. As a result, we need to derive the "distributed" counterparts of Lemmas 1 and 2 to analyze the behavior of the RLSs and the quality of the estimator.

Lemma 3. *Given two disjoint datasets $\mathcal{D}, \mathcal{D}'$, for every $i \in \mathcal{D}\cup\mathcal{D}'$, $\tau_{i,\mathcal{D}} \geq \tau_{i,\mathcal{D}\cup\mathcal{D}'}$ and*

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

While in SQUEAK we were merging an ε-accurate dictionary $\mathcal{I}_t$ and a new point, which is equivalent to a perfect, 0-accurate dictionary, in DISQUEAK both dictionaries used in a merge are only ε-accurate. To compensate for this loss in accuracy, we introduce a new estimator,

$$\widetilde{\tau}_{\mathcal{D}\cup\mathcal{D}',i} = \tfrac{1-\varepsilon}{\gamma}(k_{i,i} - \mathbf{k}_i^{\mathsf{T}}\overline{\mathbf{S}}(\overline{\mathbf{S}}^{\mathsf{T}}\mathbf{K}\overline{\mathbf{S}} + (1+\varepsilon)\gamma\mathbf{I})^{-1}\overline{\mathbf{S}}^{\mathsf{T}}\mathbf{k}_i), \tag{5}$$

where $\overline{\mathbf{S}}$ is the selection matrix associated with the temporary dictionary $\overline{\mathcal{I}} = \mathcal{I}_{\mathcal{D}} \cup \mathcal{I}_{\mathcal{D}'}$. Eq. 5 has similar guarantees as Lem. 2, with only a slightly larger α.

Lemma 4. *Given two disjoint datasets $\mathcal{D}, \mathcal{D}'$, and two ε-approximate dictionaries $\mathcal{I}_{\mathcal{D}}$, $\mathcal{I}_{\mathcal{D}'}$, let $\overline{\mathcal{I}} = \mathcal{I}_{\mathcal{D}} \cup \mathcal{I}_{\mathcal{D}'}$ and $\overline{\mathbf{S}}$ be the associated selection matrix. Let $\mathbf{K}$ be the kernel matrix computed on $\mathcal{D}\cup\mathcal{D}'$, $\mathbf{k}_i$ its i-th column, and $\tau_{\mathcal{D}\cup\mathcal{D}',i}$ the RLS of $\mathbf{k}_i$. Then for all i in $\overline{\mathcal{I}}$ such that $q_i \neq 0$, the estimator in Eq. 5 is α-accurate, i.e. it satisfies $\tau_{\mathcal{D}\cup\mathcal{D}',i}/\alpha \leq \widetilde{\tau}_{\mathcal{D}\cup\mathcal{D}',i} \leq \tau_{\mathcal{D}\cup\mathcal{D}',i}$, with $\alpha = (1-\varepsilon)/(1+3\varepsilon)$.*

Given these guarantees, the analysis of DISQUEAK follows similar steps as SQUEAK. Given ε-accurate dictionaries, we obtain α-accurate RLS estimates $\widetilde{\tau}_{\mathcal{D}\cup\mathcal{D}',i}$ that can be used to resample all points in $\overline{\mathcal{I}}$ and generate a new dictionary $\mathcal{I}_{\mathcal{D},\mathcal{D}'}$ that is ε-accurate. To formalize a result equivalent to Thm. 1, we introduce additional notation: We index each node in the merge tree by its height h and position l. 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 by $\mathcal{S}_h = \{\mathcal{I}_{\{h,l\}}\}$. We also use $\mathbf{K}_{\{h,l\}}$ to refer to the kernel matrix 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. 1, node $\{3,1\}$ is associated with $\mathcal{I}_{\{3,1\}}$, which is an ε-approximate dictionary of the kernel matrix $\mathbf{K}_{\{3,1\}}$ 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 $(|\mathcal{D}_1|+|\mathcal{D}_2|+|\mathcal{D}_3|)$. Theorem 2 summarizes the guarantees for DISQUEAK.

Theorem 2. *Let $\varepsilon > 0$ be the accuracy parameter, $\gamma > 1$ the regularization factor, and $0 < \delta < 1$ the prob. of failure. Given an arbitrary dataset $\mathcal{D}$ and a merge tree structure of height k as input together with parameters ε, γ, and δ, we run* DISQUEAK *with*

$$\overline{q} = \frac{39\alpha\log(2n/\delta)}{\varepsilon^2}.$$

where $\alpha = (1+3\varepsilon)/(1-\varepsilon)$. Then, w.p. at least $1-\delta$, DISQUEAK *generates a sequence of collections of dictionaries $\{\mathcal{S}_h\}_{h=1}^{k}$ such that each dictionary $\mathcal{I}_{\{h,l\}}$ in $\mathcal{S}_h$ is ε-accurate (Eq. 1) w.r.t. to $\mathbf{K}_{\{h,l\}}$, and that at any node l of height h the size of the dictionary is bounded as $|\mathcal{I}_{\{h,l\}}| \leq 3\overline{q}d_{\mathit{eff}}(\gamma)_{\{h,l\}}$. The cumulative (across nodes) space and time requirementsof the algorithm depend on the exact shape of the merge tree.*

Theorem 2 gives approximation and space guarantees for *every* node of the tree. In other words, it guarantees that each intermediate dictionary processed by DISQUEAK is both an ε-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 perspective, DISQUEAK provides exactly the same guarantees of SQUEAK. Analysing the complexity of DISQUEAK is however more complex, since the order and arguments of the DICT-UPDATE operations is determined by the merge tree. We distinguish between the time and work complexity of a tree by defining 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, 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 $\mathcal{D}_i = \{\mathbf{x}_i\}$. 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 SQUEAK. Computing a solution in the fully unbalanced tree takes $\mathcal{O}(nd_{\text{eff}}(\gamma)^3_{\mathcal{D}}\bar{q}^3)$ time with a total work that is also $\mathcal{O}(nd_{\text{eff}}(\gamma)^3_{\mathcal{D}}\bar{q}^3)$, as reported in Thm. 1. On the opposite end, the fully balanced tree needs to invert a $d_{\text{eff}}(\gamma)_{\{h,l\}}$ dimensional matrix at each layer of the tree for a total of $\log(n)$ layers. Bounding all $d_{\text{eff}}(\gamma)_{\{h,l\}}$ with $d_{\text{eff}}(\gamma)_{\mathcal{D}}$, gives a complexity for computing the final solution of $\mathcal{O}(\log(n)\bar{q}^3 d_{\text{eff}}(\gamma)^3_{\mathcal{D}})$ time, with a huge improvement on SQUEAK. Surprisingly, the total work is only twice $\mathcal{O}(n\bar{q}^3 d_{\text{eff}}(\gamma)^3_{\mathcal{D}})$, 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 SQUEAK, with a comparable amount of work, but at the expense of requiring much more memory across multiple machines, since at layer h, the sum $\sum_{l=1}^{|\mathcal{S}_h|} d_{\text{eff}}(\gamma)_{\{h,l\}}$ can be much larger than $d_{\text{eff}}(\gamma)_{\mathcal{D}}$. Nonetheless, this is partly alleviated by the fact that each node $\{h,l\}$ locally requires only $d_{\text{eff}}(\gamma)^2_{\{h,l\}} \leq d_{\text{eff}}(\gamma)^2_{\mathcal{D}}$ memory.

Proof sketch: Although DISQUEAK is conceptually simple, providing guarantees on its space/time complexity and accuracy is far from trivial. The first step in the proof is 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 Alg. 2 generates the dictionary $\mathcal{I}_{\{h,l\}}$. Notice that these processes are sequential in nature and the various steps (layers in the tree) are not i.i.d. Furthermore, the variance of $\mathbf{Y}$ is potentially large, and cannot be bounded uniformly. Instead, we take a more refined approach, inspired by Pachocki [10], 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. and only then we can 3) apply i.i.d. concentrations to obtain the desired result.

5 Applications

In this section, we show how our approximation guarantees translate into guarantees for typical kernel methods. As an example, we use kernel ridge regression. We begin by showing how to get an accurate approximation $\widetilde{\mathbf{K}}_n$ from an ε-accurate dictionary.

Lemma 5. *Given an ε-accurate dictionary $\mathcal{I}_t$ of matrix $\mathbf{K}_t$, and the selection matrix $\mathbf{S}_t$, the* regularized *Nyström approximation of $\mathbf{K}_t$ is defined as*

$$\widetilde{\mathbf{K}}_n = \mathbf{K}_n\mathbf{S}_n(\mathbf{S}_n^\mathsf{T}\mathbf{K}_n\mathbf{S}_n + \gamma\mathbf{I}_m)^{-1}\mathbf{S}_n^\mathsf{T}\mathbf{K}_n, \qquad (6)$$

and satisfies

$$\mathbf{0} \preceq \mathbf{K}_t - \widetilde{\mathbf{K}}_t \preceq \frac{\gamma}{1-\varepsilon}\mathbf{K}_t(\mathbf{K}_t + \gamma\mathbf{I})^{-1} \preceq \frac{\gamma}{1-\varepsilon}\mathbf{I}. \qquad (7)$$

This is not the only choice of an approximation from dictionary $\mathcal{I}_n$. For instance, Musco and Musco [9] show similar result for an unregularized Nyström approximation and Rudi et al. [12] for a smaller $\mathbf{K}_n$, construct the estimator only for the points in $\mathcal{I}_n$. Let $m = |\mathcal{I}_n|$, $\mathbf{W} = (\mathbf{S}_n^\mathsf{T}\mathbf{K}_n\mathbf{S}_n + \gamma\mathbf{I}_m) \in \mathbb{R}^{m\times m}$ is nonsingular and $\mathbf{C} = \mathbf{K}_n\mathbf{S}_n \in \mathbb{R}^{n\times m}$, applying the Woodbury formula we compute the regression weights as

$$\begin{aligned}\widetilde{\mathbf{w}}_n =& (\widetilde{\mathbf{K}}_n + \mu\mathbf{I}_n)^{-1}\mathbf{y}_n = (\mathbf{C}\mathbf{W}^{-1}\mathbf{C}^\mathsf{T} + \mu\mathbf{I}_n)^{-1}\mathbf{y}_n \\ =& \frac{1}{\mu}\left(\mathbf{y}_n - \mathbf{C}\left(\mathbf{C}^\mathsf{T}\mathbf{C} + \mu\mathbf{W}\right)^{-1}\mathbf{C}^\mathsf{T}\mathbf{y}_n\right). \qquad (8)\end{aligned}$$

Computing $(\mathbf{C}^\mathsf{T}\mathbf{C} + \mu\mathbf{W})^{-1}$ takes $\mathcal{O}(nm^2)$ time to construct the matrix and $\mathcal{O}(m^3)$ to invert it, while the other matrix-matrix multiplication take at most $\mathcal{O}(nm^2)$ time. Overall, these operations require to store at most an $n \times m$ matrix. Therefore the final complexity of computing a KRR using the dictionary is reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(nm^2 + m^3)$ time, and from $\mathcal{O}(n^2)$ to $\mathcal{O}(nm)$ space. The following corollary provides guarantees for the empirical risk of the solution $\widetilde{\mathbf{w}}_n$ in a fixed design setting.

Corollary 1 ([1, Thm. 3])**.** *For an arbitrary dataset $\mathcal{D}$, let $\mathbf{K}$ be the kernel matrix constructed on $\mathcal{D}$. Run* SQUEAK *or* DISQUEAK *with regularization parameter γ. Then, the solution $\widetilde{\mathbf{w}}$ computed using the regularized Nyström approximation $\widetilde{\mathbf{K}}$ satisfies*

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

where μ is the regularization of kernel ridge regression and $\mathcal{R}_{\mathcal{D}}(\widetilde{\mathbf{w}})$ is the empirical risk on $\mathcal{D}$.

	Time	$\lvert\mathcal{I}_n\rvert$ (Total space $= \mathcal{O}(n\lvert\mathcal{I}_n\rvert)$)	Increm.
EXACT	n^3	n	-
Bach [2]	$\frac{nd_{\max_n}^2}{\varepsilon} + \frac{d_{\max_n}^3}{\varepsilon}$	$\frac{d_{\max,n}}{\varepsilon}$	No
Alaoui and Mahoney [1]	$n(\lvert\mathcal{I}_n\rvert)^2$	$\left(\frac{\lambda_{\min}+n\mu\varepsilon}{\lambda_{\min}-n\mu\varepsilon}\right) d_{\text{eff}}(\gamma)_n + \frac{\text{Tr}(\mathbf{K}_n)}{\mu\varepsilon}$	No
Calandriello et al. [3]	$\frac{\lambda_{\max}^2}{\gamma^2}\frac{n^2 d_{\text{eff}}(\gamma)_n^2}{\varepsilon^2}$	$\frac{\lambda_{\max}}{\gamma}\frac{d_{\text{eff}}(\gamma)_n}{\varepsilon^2}$	Yes
SQUEAK	$\frac{n^2 d_{\text{eff}}(\gamma)_n^2}{\varepsilon^2}$	$\frac{d_{\text{eff}}(\gamma)_n}{\varepsilon^2}$	Yes
RLS-SAMPLING	$\frac{nd_{\text{eff}}(\gamma)_n^2}{\varepsilon^2}$	$\frac{d_{\text{eff}}(\gamma)_n}{\varepsilon^2}$	-

Table 1: Comparison of Nyström methods. $\lambda_{\max}$ and $\lambda_{\min}$ refer to largest and smallest eigenvalues of $\mathbf{K}_n$.

For the random design setting, [12] provides a similar bound for batch RLS sampling, under some mild assumption on the kernel function $\mathcal{K}(\cdot,\cdot)$ and the dataset $\mathcal{D}$. Deriving identical guarantees for SQUEAK and DISQUEAK is straightforward.

Other applications. The projection matrix $\mathbf{P}_n$ naturally appears in some form in nearly all kernel-based methods. Therefore, in addition to KRR in the fixed design [1] and in the random design [12], any kernel matrix approximation that provides ε-accuracy guarantees on $\mathbf{P}_n$ can be used to provide guarantees for a variety of kernel-based methods. As an example, Musco and Musco [9] show this is the case for kernel PCA [13], kernel CCA with regularization, and kernel K-means clustering.

6 Discussion

Tab. 1 compares several kernel approximation methods w.r.t. their space and time complexity. For all methods, we omit $\mathcal{O}(\log(n))$ factors. We first report RLS-SAMPLING, a fictitious algorithm that receives the exact RLSs as input, as an ideal baseline for all RLS sampling algorithms. The space complexity of uniform sampling [2] scales with the maximal degree of freedom $d_{\max}$. Since $d_{\max} = n \max_i \tau_{n,i} \geq \sum_i \tau_{n,i} = d_{\text{eff}}(\gamma)_n$, uniform sampling is often outperformed by RLS sampling. While Alaoui and Mahoney [1] also sample according to RLS, their two-pass estimator does not preserve the same level of accuracy. In particular, the first pass requires to sample $\mathcal{O}\left(n\mu\varepsilon/(\lambda_{\min}-n\mu\varepsilon)\right)$ columns, which quickly grows above n^2 when $\lambda_{\min}$ becomes small. Finally, Calandriello et al. [3] require that the maximum dictionary size is fixed in advance, which implies some information about the effective dimensions $d_{\text{eff}}(\gamma)_n$, and requires estimating both $\widetilde{\tau}_{t,i}$ and $\widetilde{d}_{\text{eff}}(\gamma)_t$. This extra estimation effort causes an additional $\lambda_{\max}/\gamma$ factor to appear in the space complexity. This factor cannot be easily estimated, and it leads to a space complexity of n^3 in the worst case. Therefore, we can see from the table that SQUEAK achieves the same space complexity (up to constant factors) as knowing the RLS in advance and hence outperforms previous methods.

A recent method by Musco and Musco [9] achieves comparable space and time guarantees as SQUEAK.[3] While they rely on a similar estimator, the two approaches are very different. Their method is batch in nature, as it processes the whole dataset at the same time and it *requires multiple passes* on the data for the estimation of the leverage scores and the sampling. On the other hand, SQUEAK is intrinsically sequential and it *only requires one single pass* on $\mathcal{D}$, as points are "forgotten" once they are dropped from the dictionary. Furthermore, the different structure requires remarkably different tools for the analysis. While the method of Musco and Musco [9] can directly use i.i.d. concentration inequalities (for the price of needing several passes), we need to rely on more sophisticated martingale arguments to consider the sequential stochastic process of SQUEAK. Furthermore, DISQUEAK smoothly extends the ideas of SQUEAK over distributed architectures, while it is not clear whether [9] could be parallelized or distributed.

Future developments Both SQUEAK and DISQUEAK need to know in advance the size of the dataset n to tune $\overline{q}$. An interesting question is whether it is possible to adaptively adjust the $\overline{q}$ parameter at runtime. This would allow us to continue updating $\mathcal{I}_n$ and indefinitely process new data beyond the initial dataset $\mathcal{D}$. It is also interesting to see whether SQUEAK could be used in conjuntion with existing meta-algorithms (e.g., [7] with model averaging) for kernel matrix approximation that can leverage an accurate sampling scheme as a black-box, and what kind of improvements we could obtain.

Acknowledgements The research presented was supported by French Ministry of Higher Education and Research, Nord-Pas-de-Calais Regional Council and French National Research Agency projects ExTra-Learn (n.ANR-14-CE24-0010-01) and BoB (n.ANR-16-CE23-0003)

[3] The technical report of Musco and Musco [9] was developed independently from our work.

References

[1] Ahmed El Alaoui and Michael W. Mahoney. Fast randomized kernel methods with statistical guarantees. In *Neural Information Processing Systems*, 2015.

[2] Francis Bach. Sharp analysis of low-rank kernel matrix approximations. In *Conference on Learning Theory*, 2013.

[3] Daniele Calandriello, Alessandro Lazaric, and Michal Valko. Analysis of Nyström method with sequential ridge leverage scores. In *Uncertainty in Artificial Intelligence*, 2016.

[4] Michael B. Cohen, Cameron Musco, and Jakub Pachocki. Online row sampling. *International Workshop on Approximation, Randomization, and Combinatorial Optimization*, 2016.

[5] Michael B. Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Symposium on Discrete Algorithms*, 2017.

[6] Alex Gittens and Michael W Mahoney. Revisiting the Nyström method for improved large-scale machine learning. In *International Conference on Machine Learning*, 2013.

[7] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the Nyström method. *Journal of Machine Learning Research*, 13(Apr):981–1006, 2012.

[8] Haim Levy. *Stochastic dominance: Investment decision making under uncertainty*. Springer, 2015.

[9] Cameron Musco and Christopher Musco. Provably useful kernel matrix approximation in linear time. Technical report, 2016. URL `http://arxiv.org/abs/1605.07583`.

[10] Jakub Pachocki. Analysis of resparsification. Technical report, 2016. URL `http://arxiv.org/abs/1605.08194`.

[11] Raajen Patel, Thomas A. Goldstein, Eva L. Dyer, Azalia Mirhoseini, and Richard G. Baraniuk. oASIS: Adaptive column sampling for kernel matrix approximation. Technical report, 2015. URL `http://arxiv.org/abs/1505.05208`.

[12] Alessandro Rudi, Raffaello Camoriano, and Lorenzo Rosasco. Less is more: Nyström computational regularization. In *Neural Information Processing Systems*, 2015.

[13] Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Advances in kernel methods*, pages 327–352. MIT Press Cambridge, MA, USA, 1999.

[14] Joel Aaron Tropp. Freedman's inequality for matrix martingales. *Electronic Communications in Probability*, 16:262–270, 2011.

[15] Joel Aaron Tropp. An introduction to matrix concentration inequalities. *Foundations and Trends in Machine Learning*, 8(1-2):1–230, 2015.

[16] David P Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1–2):1–157, 2014.