# Analysis of Nyström Method with Sequential Ridge Leverage Score Sampling

**Daniele Calandriello**
SequeL team
INRIA Lille - Nord Europe

**Alessandro Lazaric**
SequeL team
INRIA Lille - Nord Europe

**Michal Valko**
SequeL team
INRIA Lille - Nord Europe

## Abstract

Large-scale kernel ridge regression (KRR) is limited by the need to store a large kernel matrix $\mathbf{K}_t$. To avoid storing the entire matrix $\mathbf{K}_t$, Nyström methods subsample a subset of columns of the kernel matrix, and efficiently find an approximate KRR solution on the reconstructed $\widetilde{\mathbf{K}}_t$. The chosen subsampling distribution in turn affects the statistical and computational tradeoffs. For KRR problems, [16, 1] show that a sampling distribution proportional to the *ridge leverage scores* (RLSs) provides strong reconstruction guarantees for $\widetilde{\mathbf{K}}_t$. While exact RLSs are as difficult to compute as a KRR solution, we may be able to approximate them well enough. In this paper, we study KRR problems in a sequential setting and introduce the INK-ESTIMATE algorithm, that *incrementally* computes the RLSs estimates. INK-ESTIMATE maintains a small *sketch* of $\mathbf{K}_t$, that at each step is used to compute an intermediate estimate of the RLSs. First, our sketch update does not require access to previously seen columns, and therefore a *single pass* over the kernel matrix is sufficient. Second, the algorithm requires a fixed, small space budget to run dependent only on the *effective dimension* of the kernel matrix. Finally, our sketch provides strong approximation guarantees on the distance $\|\mathbf{K}_t - \widetilde{\mathbf{K}}_t\|_2$, and on the statistical risk of the approximate KRR solution at *any time*, because all our guarantees hold at any intermediate step.

## 1 INTRODUCTION

Kernel ridge regression [17, 18] (KRR) is a common non-parametric regression method with well studied theoretical advantages. Its main drawback is that, for $n$ samples, storing and manipulating the kernel regression matrix $\mathbf{K}_n$ requires $\mathcal{O}(n^2)$ space, and can become quickly intractable when $n$ grows. This includes batch large scale KRR, and online KRR, where the size of the dataset $t$ grows over time as new samples are added to the problem. For this purpose, many different methods [23, 4, 10, 14, 11, 24] attempt to reduce the memory required to store the kernel matrix, while still producing an accurate solution.

For the batch case, the Nyström family of algorithms randomly selects a subset of $m$ columns from the kernel matrix $\mathbf{K}_n$ that are used to construct a low rank approximation $\widetilde{\mathbf{K}}_t$ that requires only $\mathcal{O}(nm)$ space to store. The low-rank matrix is then used to find an approximate solution to the KRR problem. The quality of the approximate solution is strongly affected by the sampling distribution and the number of columns selected [16]. For example, uniform sampling is an approach with little computational overhead, but does not work well for datasets with high coherence [7], where the columns are weakly correlated. In particular, Bach [2] shows that the number of columns $m$ necessary for a good approximation when sampling uniformly scales linearly with the maximum degree of freedoms of the kernel matrix. In linear regression, the notion of coherence is strongly related to the definition of leverage points or *leverage scores* of the dataset [6], where points with high (statistical) leverage score are more influential in the regression problem. For KRR, Alaoui and Mahoney [1] introduce a similar concept of *ridge* leverage scores (RLSs) of a square matrix, and shows that Nyström approximations sampled according to RLS have strong reconstruction guarantees of the form $\|\mathbf{K}_n - \widetilde{\mathbf{K}}_n\|_2$, that translate into good guarantees for the approximate KRR solution [1, 16]. Compared to the uniform distribution, a distribution based on RLSs better captures non-uniformities in the data, and can achieve good approximations using only a number of columns $m$, proportional to the average degrees of freedom of the matrix, called the *effective dimension* of the problem. The disadvantage of RLSs compared to uniform sampling is the high computational cost of exact RLSs, which is comparable to solving KRR itself. Alaoui and Mahoney [1] reduces this problem by showing that a distribution based on approximate RLSs can also provide the same strong guarantees, if the RLSs are approximated up to a constant er-

ror factor. They provide a fast method to compute these RLSs, but, unlike our approach, requires multiple passes over data. Another disadvantage of their approach, that we address, is the *inverse dependence on the minimal eigenvalue* of the kernel matrix in the error bound of Alaoui and Mahoney [1], which can be significant.

While Nyström methods are a typical choice in a batch setting, *online kernel sparsification* (OKS) [4, 5] examines each sample in the dataset sequentially. OKS maintains a small *dictionary* of relevant samples. Whenever a new sample arrive, if the dictionary is not able to accurately represent the new sample as a combination of the samples already stored, the dictionary is updated. This dictionary can be used to approximate KRR incrementally. OKS decides whether to include a sample using the correlation between samples in the dictionary and the new sample. This can measured using approximate linear dependency (ALD) [5], coherence [15], or the surprise criterion [12].

Generalization properties of online kernel sparsification were studied by Engel et al. [5], but depend on the empirical error and are not compared with an exact KRR solution on the whole dataset. Online kernel regression with the ALD rule was analyzed by Sun et al. [19], under the assumption that, asymptotically in $n$, the eigenvalues of the kernel matrix decay exponentially fast. Sun et al. [19] show that in this case the size of the dictionary grows sublinearly in $t$, or in other words that, asymptotically in $n$, the dictionary size converge to a fraction of $n$ that will be small whenever the eigenvalues decay fast enough. This space guarantee is weaker than the fixed space requirements of Nyström methods, one of the reasons is that these methods (unlike ours) cannot remove a sample from the dictionary after inclusion. Furthermore, Van Vaerenbergh et al. [22] studies variants of online kernel regression with a forgetting factor for time-varying series, but these methods are not well studied in the normal KRR setting. Unlike in the batch setting, in the sequential setting we often require the guarantees not only at the end but also *in the intermediate steps* and this is our objective. Inspired by the advances in the analyses of the Nyström methods, in this paper, we focus on finding a space efficient algorithm capable of solving KRR problems in the sequential setting but that would be also equipped with generalization guarantees.

**Main contributions** We propose the INK-ESTIMATE algorithm that processes a dataset $\mathcal{D}$ of size $n$ in *a single pass*. It requires only a small, fixed space budget, $\overline{q}$ proportional to the effective dimension of the problem and on the accuracy required. The algorithm maintains a Nyström approximation $\widetilde{\mathbf{K}}_t$, of the kernel matrix at time $t$, $\mathbf{K}_t$, based on RLSs estimates. At each step, it uses only the approximation and the newly received sample to incrementally update the RLSs estimate, and to compute $\widetilde{\mathbf{K}}_{t+1}$. Unlike in the batch Nyström setting, our challenge is to track RLSs and an effective dimension that *changes over time*. Sampling distributions based on RLSs can become obsolete and biased, but we show how to update them over time *without necessity of accessing previously seen samples* outside of the ones contained in $\mathbf{K}_t$. Our space budget $\overline{q}$ scales with the average degree of freedom of the matrix, and not the larger maximum degree of freedom (as by Bach [2]), and does not imposes assumptions on the ridge regularization parameter, or on the smallest eigenvalue of the problem as the result of Alaoui and Mahoney [1]. However, we provide the same strong guarantees as batch RLSs based Nyström methods on $\|\mathbf{K}_n - \widetilde{\mathbf{K}}_n\|_2$ and on the risk of the approximate KRR solution. In addition to batch Nyström methods, all of these guarantees hold at any intermediate step $t$, and therefore the algorithm can output *accurate intermediate solutions*, or it can be interrupted at *any time* and return a solution with guarantees. Finally, it operates in a *sequential* setting, requiring only a single pass over the data.

If we compare INK-ESTIMATE to other online kernel regression methods (such as OKS), our algorithm provides generalization guarantees with respect to the exact KRR solution. Furthermore, it provides a new criteria for inclusion of a sample in the dictionary, in particular the ridge leverage scores. This criterion gives us a procedure that not only randomly includes samples in the dictionary, but that also randomly discards them to satisfy space constraints not only asymptotically, but at every step.

## 2 BACKGROUND

In this section we introduce the notation used through the paper and we introduce the kernel ridge regression problem [17] and Nyström approximation of the kernel matrix with ridge leverage scores.

**Notation.** We use curly capital letters $\mathcal{A}$ for collections. We use upper-case bold letters $\mathbf{A}$ for matrices, lower-case bold letters $\mathbf{a}$ for vectors, and lower-case letters $a$ for scalars. We denote by $[\mathbf{A}]_{ij}$ and $[\mathbf{a}]_i$ the $(i,j)$ element of a matrix and $i$th 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 $\mathbf{e}_{i,n} \in \mathbb{R}^n$ to denote the indicator vector for element $i$ of dimension $n$. When the dimensionality 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 PSD matrix. Finally, the set of integers between 1 and $n$ is denoted by $[n] := \{1, \ldots, n\}$.

### 2.1 Exact Kernel Ridge Regression

**Kernel regression.** We consider a regression dataset $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$, with input $\mathbf{x}_t \in \mathcal{X} \subseteq \mathbb{R}^d$ and output $y_t \in \mathbb{R}$. We denote by $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a positive definite kernel function and by $\varphi : \mathcal{X} \to \mathbb{R}^D$ the corresponding

feature map,[1] so that the kernel is obtained as $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \varphi(\mathbf{x})^\mathsf{T}\varphi(\mathbf{x}')$. Given the dataset $\mathcal{D}$, we define the kernel matrix $\mathbf{K}_t \in \mathbb{R}^{t\times t}$ constructed on the first $t$ samples as the application of the kernel function on all pairs of input values, i.e., $[\mathbf{K}_t]_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$ for any $i, j \in [t]$ and we denote by $\mathbf{y}_t \in \mathbb{R}^t$ the vector with components $y_i$, $i \in [t]$. We also define the feature vectors $\boldsymbol{\phi}_t = \varphi(\mathbf{x}_t) \in \mathbb{R}^D$ and after introducing the feature matrix

$$\boldsymbol{\Phi}_t = \left[\ \boldsymbol{\phi}_1 \mid \boldsymbol{\phi}_2 \mid \dots \mid \boldsymbol{\phi}_t\ \right] \in \mathbb{R}^{D\times t},$$

we can rewrite the kernel matrix as $\mathbf{K}_t = \boldsymbol{\Phi}_t^\mathsf{T}\boldsymbol{\Phi}_t$. Whenever a new point $\mathbf{x}_{t+1}$ arrives, the kernel matrix $\mathbf{K}_{t+1} \in \mathbb{R}^{t+1\times t+1}$ is obtained by bordering $\mathbf{K}_t$ as

$$\mathbf{K}_{t+1} = \left[\begin{array}{c|c} \mathbf{K}_t & \overline{\mathbf{k}}_{t+1} \\ \hline \overline{\mathbf{k}}_{t+1}^\mathsf{T} & k_{t+1} \end{array}\right] \tag{1}$$

where $\overline{\mathbf{k}}_{t+1} \in \mathbb{R}^t$ is such that $[\overline{\mathbf{k}}_{t+1}]_i = \mathcal{K}(\mathbf{x}_{t+1}, \mathbf{x}_i)$ for any $i \in [t]$ and $k_{t+1} = \mathcal{K}(\mathbf{x}_{t+1}, \mathbf{x}_{t+1})$. According to the definition of the feature matrix $\boldsymbol{\Phi}_t$, we also have $\mathbf{k}_{t+1} = \boldsymbol{\Phi}_t^\mathsf{T}\boldsymbol{\phi}_{t+1}$.

At any time $t$, the objective of *sequential* kernel regression is to find the vector $\widehat{\mathbf{w}}_t \in \mathbb{R}^t$ that minimizes the regularized quadratic loss

$$\widehat{\mathbf{w}}_t = \arg\min_{\mathbf{w}} \|\mathbf{y}_t - \mathbf{K}_t\mathbf{w}\|^2 + \mu\|\mathbf{w}\|^2, \tag{2}$$

where $\mu \in \mathbb{R}$ is a regularization parameter. This objective admits the closed form solution

$$\widehat{\mathbf{w}}_t = (\mathbf{K}_t + \mu\mathbf{I})^{-1}\mathbf{y}_t. \tag{3}$$

In the following, we use $\mathbf{K}_t^\mu$ as a short-hand for $(\mathbf{K}_t + \mu\mathbf{I})$. In batch regression, $\widehat{\mathbf{w}}_n$ is computed only once when all the samples of $\mathcal{D}$ are available, solving the linear system in Eq. 3 with $\mathbf{K}_n$. In the *fixed-design* kernel regression, the accuracy of resulting solution $\widehat{\mathbf{w}}_n$ is measured by the prediction error on the input set from $\mathcal{D}$. More precisely, the prediction of the estimator $\widehat{\mathbf{w}}_n$ in each point is obtained as $[\mathbf{K}_n\widehat{\mathbf{w}}_n]_i$, 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 $x_i$ i.e., for any $i \in [n]$,

$$y_i = f^*(x_i) + \eta_i,$$

where $\eta_i$ is a zero-mean i.i.d. noise with bounded variance $\sigma^2$. Let $\mathbf{f}^* \in \mathbb{R}^n$ be the vector with components $f^*(x_i)$, then the risk of $\widehat{\mathbf{w}}_n$ is measured as

$$\mathcal{R}(\widehat{\mathbf{w}}_n) = \mathbb{E}_\eta\big[||\mathbf{f}^* - \mathbf{K}_n\widehat{\mathbf{w}}_n||_2^2\big]. \tag{4}$$

If the regularization parameter $\mu$ is properly tuned, it is possible to show that $\widehat{\mathbf{w}}_n$ has near-optimal risk guarantees (in a minmax sense). Nonetheless, the computation of $\widehat{\mathbf{w}}_n$ requires $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space, which becomes rapidly unfeasible for large datasets.

[1] where $D$ can be very large or infinite (e.g. gaussian kernel)

### 2.2 Nyström Approximation with Ridge Leverage Scores

A common approach to reduce the complexity of kernel regression is to (randomly) select a subset of $m$ samples out of $\mathcal{D}$, and compute the kernel between two points only when one of them is in the selected subset. This is equivalent to selecting a subset of columns of the $\mathbf{K}_n$ matrix. More formally, given the $n$ samples in $\mathcal{D}$, a probability distribution $\mathbf{p}_n = [p_{1,n}, \dots, p_{n,n}]$ is defined over all columns of $\mathbf{K}_n$ and $m \le n$ columns are randomly sampled with replacement according to $\mathbf{p}_n$. We define by $\mathcal{I}_n$ the sequence of $m$ indices $i \in [n]$ selected by the sampling procedure. From $\mathcal{I}_n$, we construct the corresponding selection matrix $\mathbf{S}_n \in \mathbb{R}^{n\times m}$, where each column $[\mathbf{S}_n]_{:,t} \in \mathbb{R}^n$ is all-zero except from the entry corresponding to the $t$-th element in $\mathcal{I}_n$ (i.e., $[\mathbf{S}]_{ij}$ is non-zero if at trial $j$ the element $i$ is selected). Whenever the non-zero entries of $\mathbf{S}_n$ are set to 1, sampling $m$ columns from matrix $\mathbf{K}_n$ is equivalent to computing $\mathbf{K}_n\mathbf{S}_n \in \mathbb{R}^{n\times m}$. More generally, the non-zero entries of $\mathbf{S}_n$ could be set to some arbitrary weight $[\mathbf{S}]_{ij} = b_{ij}$. The resulting regularized Nyström approximation of the original kernel $\mathbf{K}_n$ is defined as

$$\widetilde{\mathbf{K}}_n = \mathbf{K}_n\mathbf{S}_n(\mathbf{S}_n^\mathsf{T}\mathbf{K}_n\mathbf{S}_n + \gamma\mathbf{I}_m)^{-1}\mathbf{S}_n^\mathsf{T}\mathbf{K}_n, \tag{5}$$

where $\gamma$ is a regularization term (possibly different from $\mu$). At this point, $\widetilde{\mathbf{K}}_n$ can be used to solve Eq. 3. Let $\mathbf{W} = (\mathbf{S}_n^\mathsf{T}\mathbf{K}_n\mathbf{S}_n + \gamma\mathbf{I}_m)^{-1} \in \mathbb{R}^{m\times m}$ and $\mathbf{C} = \mathbf{K}_n\mathbf{S}_n\mathbf{W}^{1/2} \in \mathbb{R}^{n\times m}$, applying the Woodbury inversion formula [8] we have

$$\begin{aligned}\widetilde{\mathbf{w}}_n =& (\widetilde{\mathbf{K}}_n + \mu\mathbf{I}_n)^{-1}\mathbf{y}_n = (\mathbf{C}\mathbf{I}_m\mathbf{C}^\mathsf{T} + \mu\mathbf{I}_n)^{-1}\mathbf{y}_n \\ =& \left(\frac{1}{\mu}\mathbf{I}_n - \frac{1}{\mu^2}\mathbf{I}_n\mathbf{C}\left(\mathbf{I}_m + \frac{1}{\mu}\mathbf{C}^\mathsf{T}\mathbf{C}\right)^{-1}\mathbf{C}^\mathsf{T}\mathbf{I}_n\right)\mathbf{y}_n \\ =& \frac{1}{\mu}\left(\mathbf{y}_n - \mathbf{C}\left(\mathbf{C}^\mathsf{T}\mathbf{C} + \mu\mathbf{I}_m\right)^{-1}\mathbf{C}^\mathsf{T}\mathbf{y}_n\right).\end{aligned} \tag{6}$$

Computing $\mathbf{W}^{1/2}$ and $\mathbf{C}$ takes $\mathcal{O}(m^3)$ and $\mathcal{O}(nm^2)$ time using a singular value decomposition, and so does solving the linear system. All the operations require to store at most an $n \times m$ matrix. Therefore the final complexity 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. Rudi et al. [16] recently showed that in random design, the risk of the resulting solution $\widetilde{\mathbf{w}}_n$ strongly depends on the choice of $m$ and the column sampling distribution $\mathbf{p}_n$. Early methods sampled columns uniformly, and Bach [2] shows that the using this distribution can provide a good approximation when the maximum diagonal entry of $\mathbf{K}_n(\mathbf{K}_n + \mu\mathbf{I})^{-1}$ is small. Following on this approach, Alaoui and Mahoney [1] propose a distribution proportional to these diagonal entries and calls them $\gamma$-Ridge Leverage Scores. We now restate their definition of RLS, corresponding sampling distribution, and the effective dimension.

**Definition 1.** *Given a kernel matrix $\mathbf{K}_n \in \mathbb{R}^{n\times n}$, the $\gamma$-ridge leverage score (RLS) of column $i \in [n]$ is*

$$\tau_{i,n}(\gamma) = \mathbf{k}_{i,n}^{\mathsf{T}}(\mathbf{K}_n + \gamma\mathbf{I}_m)^{-1}\mathbf{e}_{i,n}, \tag{7}$$

*where $\mathbf{k}_{i,n} = \mathbf{K}_n\mathbf{e}_{i,n}$. Furthermore, the effective dimension $d_{\mathit{eff}}(\gamma)_n$ of the kernel is defined as*

$$d_{\mathit{eff}}(\gamma)_n = \sum_{i=1}^{n}\tau_{i,n}(\gamma) = \mathrm{Tr}\left(\mathbf{K}_n(\mathbf{K}_n + \gamma\mathbf{I}_n)^{-1}\right). \tag{8}$$

*The corresponding sampling distribution $\mathbf{p}_n$ is defined as*

$$[\mathbf{p}_n]_i = p_{i,n} = \frac{\tau_{i,n}(\gamma)}{\sum_{j=1}^{n}\tau_{i,n}(\gamma)} = \frac{\tau_{i,n}}{d_{\mathit{eff}}(\gamma)_n}. \tag{9}$$

The RLSs are directly related to the structure of the kernel matrix and the regularized regression. If we perform an eigendecomposition of the kernel matrix as $\mathbf{K}_n = \mathbf{U}_n\mathbf{\Lambda}_n\mathbf{U}_n^{\mathsf{T}}$, then the RLS of a column $i \in [n]$ is

$$\tau_{i,n}(\gamma) = \sum_{j=1}^{n}\frac{\lambda_j}{\lambda_j+\gamma}[\mathbf{U}]_{i,j}^2, \tag{10}$$

which shows how the RLS is a weighted version of the standard leverage scores (i.e., $\sum_j [U]_{i,j}^2$), where the weights depend on both the spectrum of $\mathbf{K}_n$ and the regularization $\gamma$, which plays the role of a soft threshold on the rank of $\mathbf{K}_n$. Similar to the standard leverage scores [3], the RLSs measure the relevance of each point $\mathbf{x}_i$ for the overall kernel regression problem. Another interesting property of the RLSs is that their sum is the effective dimension $d_{\text{eff}}(\gamma)_n$, which measures the intrinsic capacity of the kernel $\mathbf{K}_n$ when its spectrum is soft-thresholded by a regularization $\gamma$.[2] We refer to the overall Nyström method using RLS and sampling according to to $\mathbf{p}_n$ in Eq. 9 as BATCH-EXACT, which is illustrated in Alg. 1. We single out the DIRECT-SAMPLE subroutine (which simply draws $m$ independent samples from the multinomial distribution $\mathbf{p}_n$) to ease the introduction of our incremental algorithm in the next section.

With the following claim, Alaoui and Mahoney [1] prove that the regularized Nyström approximation $\widetilde{\mathbf{K}}_n$ obtained from Eq. 5 guarantees an accurate reconstruction of the original kernel matrix $\mathbf{K}_n$, and the risk of the associated solution $\widetilde{\mathbf{w}}_n$ is close to the risk of the exact solution $\widehat{\mathbf{w}}_n$.

**Proposition 1** (Alaoui and Mahoney [1], App. A, Lem. 1)**.** *Let $\gamma \geq 1$, let $\mathbf{K}_n$ be the full kernel matrix ($t = n$), and let $\tau_{i,n}$, $d_{\mathit{eff}}(\gamma)_n$, $p_{i,n}$ be defined according to Definition 1. For any $0 \leq \varepsilon \leq 1$, and $0 \leq \delta \leq 1$, if we run Alg. 1 using* DIRECT-SAMPLE *(Subroutine 1) with sampling budget $m$,*

$$m \geq \left(\frac{2d_{\mathit{eff}}(\gamma)}{\varepsilon^2}\right)\log\left(\frac{n}{\delta}\right),$$

[2]Notice that indeed we have $d_{\text{eff}}(\gamma)_n \leq \mathrm{Rank}(\mathbf{K}_n)$.

**Algorithm 1** BATCH-EXACT algorithm

**Input:** $\mathcal{D}$, regularization parameter $\gamma$, sampling budget $m$ and probabilities $\mathbf{p}_n$ (Eq. 9)
**Output:** Nyström approximation $\widetilde{\mathbf{K}}_n$, matrix $\mathbf{S}_n$

1: Compute $\mathcal{I}_n$ using DIRECT-SAMPLE$(\mathbf{p}_n, m)$
2: Compute $\mathbf{S}_n$ using $\mathcal{I}_n$ and weights $1/\sqrt{mp_{i,n}}$
3: Compute $\widetilde{\mathbf{K}}_n$ using $\mathbf{S}_n$ and Equation 5

**Subroutine 1** DIRECT-SAMPLE$(\mathbf{p}_n, m) \to \mathcal{I}_n$

**Input:** probabilities $\mathbf{p}_n$, sampling budget $m$
**Output:** subsampled column indices $\mathcal{I}_n$

1: **for** $j = \{1, \ldots, m\}$ **do**
2: Sample $i \sim \mathcal{M}(p_{1,n}, \ldots, p_{n,n})$
3: Add $i$ to $\mathcal{I}_n$
4: **end for**

*to compute matrix $\mathbf{S}_n$, then with probability $1-\delta$ the corresponding Nyström approximation $\widetilde{\mathbf{K}}_n$ in Eq. 5 satisfies the condition*

$$\mathbf{0} \preceq \mathbf{K}_n - \widetilde{\mathbf{K}}_n \preceq \frac{\gamma}{1-\varepsilon}\mathbf{K}_n(\mathbf{K}_n+\gamma\mathbf{I}_n)^{-1} \preceq \frac{\gamma}{1-\varepsilon}\mathbf{I}_n. \tag{11}$$

*Furthermore, replacing $\mathbf{K}_n$ by $\widetilde{\mathbf{K}}_n$ in Eq. 3 gives an approximation solution $\widetilde{\mathbf{w}}_n$ such that*

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

**Discussion** This result directly relates the number of columns selected $m$ with the accuracy of the approximation of the kernel matrix. In particular, the inequalities in Eq. 11 show that the distance $\|\mathbf{K}_n - \widetilde{\mathbf{K}}_n\|_2$ is smaller than $\gamma/(1-\varepsilon)$. This level of accuracy is then sufficient to guarantee that, when $\gamma$ is properly tuned, the prediction error of $\widetilde{\mathbf{w}}_n$ is only a factor $(1+2\varepsilon)^2$ away from the error of the exact solution $\widehat{\mathbf{w}}$. As it was shown in [1], using $\widetilde{\mathbf{K}}_n$ in place of $\mathbf{K}_n$ introduces a bias in the solution $\widetilde{\mathbf{w}}_n$ of order $\gamma$. For appropriate choices of $\gamma$ this bias is dominated by the ridge regularization bias controlled by $\mu$. As a result, $\widetilde{\mathbf{w}}_n$ can indeed achieve almost the same risk as $\widehat{\mathbf{w}}_n$ and, at the same time, ignore all directions that are whitened by the regularization and only approximate those that are more relevant for ridge regression, thus reducing both time and space complexity. The RLSs quantify how important each column is to approximate these relevant directions but computing exact RLSs $\tau_{i,n}(\gamma)$ using Eq. 7 is as hard as solving the regression problem itself. Fortunately, in many cases it is computationally feasible to find an approximation of the RLSs. Alaoui and Mahoney [1] explore this possibility, showing that the accuracy and space guarantees are robust to perturbations in the distribution $\mathbf{p}_n$, and provide a two-pass method to compute such approximations. Unfortunately, the accuracy of their RLSs approximation is proportional to the smallest eigenvalue $\lambda_{\min}(\mathbf{K}_n)$, which in

**Algorithm 2** The INK-ORACLE algorithm

**Input:** Dataset $\mathcal{D}$, regularization $\gamma$, sampling budget $\overline{q}$ and $(\alpha, \beta)$-ORACLE

**Output:** $\widetilde{\mathbf{K}}_n$, $\mathbf{S}_n$

Initialize $\mathcal{I}_0$ as empty, $\widetilde{p}_{1,0} = 1, b_{1,0} = 1$, budget $\overline{q}$
**for** $t = 0, \ldots, n-1$ **do**
  Receive new column $\overline{\mathbf{k}}_{t+1}$ and scalar $k_{t+1}$
  Receive $\alpha$-leverage scores $\widetilde{\tau}_{i,t+1}$ for any $i \in \mathcal{I}_t \cup \{t+1\}$ from $(\alpha, \beta)$-ORACLE
  Receive $\beta$-approximate $\widetilde{d}_{\text{eff}}(\gamma)_{t+1}$ from $(\alpha, \beta)$-ORACLE
  Set $\widetilde{p}_{i,t+1} = \min\{\widetilde{\tau}_{i,t+1}/\widetilde{d}_{\text{eff}}(\gamma)_{t+1}, \widetilde{p}_{i,t}\}$
  $\mathcal{I}_{t+1}, \mathbf{b}_{t+1}$ = SHRINK-EXPAND$(\mathcal{I}_t, \widetilde{\mathbf{p}}_{t+1}, \mathbf{b}_t, \overline{q})$
  Compute $\mathbf{S}_{t+1}$ using $\mathcal{I}_{t+1}$ and weights $\sqrt{b_{i,t+1}}$
  Compute $\widetilde{\mathbf{K}}_{t+1}$ using $\mathbf{S}_{t+1}$ and Equation 5
**end for**
Return $\widetilde{\mathbf{K}}_n$ and $\mathbf{S}_n$

**Subroutine 2** SHRINK-EXPAND$(\mathcal{I}_t, \widetilde{\mathbf{p}}_{t+1}, \mathbf{b}_t, \overline{q})$

**Input:** $\mathcal{I}_t$, app. pr. $\{(\widetilde{p}_{i,t+1}, b_{i,t}) : i \in \mathcal{I}_t\}$, $\widetilde{p}_{t+1,t+1}$, $\overline{q}$

**Output:** $\mathcal{I}_{t+1}$

**for all** $j \in \{1, \ldots, t\}$ **do** ▷SHRINK
  $b_{i,t+1} = b_{i,t}$
  **while** $b_{i,t+1}\widetilde{p}_{i,t+1} \leq 1/\overline{q}$ and $b_{i,t} \neq 0$ **do**
    Sample a random Bernoulli $\mathcal{B}\left(\frac{b_{i,t+1}}{b_{i,t+1}+1}\right)$
    On success set $b_{i,t+1} = b_{i,t+1} + 1$
    On failure set $b_{i,t+1} = 0$
  **end while**
**end for**
$b_{t+1,t+1} = 1$ ▷EXPAND
**while** $b_{t+1,t+1}\widetilde{p}_{t+1,t+1} \leq 1/\overline{q}$ and $b_{t+1,t+1} \neq 0$ **do**
  Sample a random Bernoulli $\mathcal{B}\left(\frac{b_{t+1,t+1}}{b_{t+1,t+1}+1}\right)$
  On success set $b_{t+1,t+1} = b_{t+1,t+1} + 1$
  On failure set $b_{t+1,t+1} = 0$
**end while**
Add to $\mathcal{I}_{t+1}$ all columns with $b_{i,t+1} \neq 0$

some cases can be very small. In the rest of the paper, we propose an *incremental* approach that requires only a *single pass* over the data and, at the same time, does not depend on $\lambda_{\min}(\mathbf{K}_n)$ to be large as in [1], or on $\max_i \tau_{i,n}$ to be small as in [2].

# 3 INCREMENTAL ORACLE KERNEL APPROXIMATION WITH SEQUENTIAL SAMPLING

Our main goal is to extend the known ridge leverage score sampling to the *sequential setting*. This comes with several challenges that needs to be addressed *simultaneously:*

1. The RLSs change when a new sample arrives. We not only need to estimate them, but to *update* this estimate over iterations.

2. The effective dimension $\widetilde{d}_{\text{eff}}(\gamma)_t$, necessary to normalize the leverage scores for the sampling distribution $\mathbf{p}_n$, depends on the interactions of all columns, including the ones that we decided *not* to keep.

3. Due to changes in RLSs, our sampling distribution $\widetilde{\mathbf{p}}_t$ *changes over time*. We need to update to dictionary to reflect these changes, or it will quickly become *biased*, but once we completely drop a column, we cannot sample it again.

In this section, we address the third challenge of incremental updates of the columns with an algorithm for the approximation of the kernel matrix $\mathbf{K}_n$, assuming that the first and second issue are addressed by an *oracle* giving *both* good approximations of leverage scores and the effective dimension.

**Definition 2.** *At any step $t$, an $(\alpha, \beta)$-oracle returns an $\alpha$-approximate ridge leverage scores $\widetilde{\tau}_{i,t}$ which satisfy*

$$\frac{1}{\alpha}\tau_{i,t}(\gamma) \leq \widetilde{\tau}_{i,t} \leq \tau_{i,t}(\gamma),$$

*for any $i \in [t]$ and and a $\beta$-approximate effective dimension $\widetilde{d}_{\text{eff}}(\gamma)_t$ which satisfy*

$$d_{\text{eff}}(\gamma)_t \leq \widetilde{d}_{\text{eff}}(\gamma)_t \leq \beta d_{\text{eff}}(\gamma)_t.$$

We address the first and second challenge in Sect. 4 with an efficient implementation and $(\alpha, \beta)$-oracle. In the following we give the *incremental* INK-ORACLE algorithm equipped with an $(\alpha, \beta)$-oracle that after $n$ steps it returns a kernel approximation with the same properties as if an $(\alpha, \beta)$-oracle was used directly at time $n$.

### 3.1 The INK-ORACLE Algorithm

Apart from an $(\alpha, \beta)$-ORACLE and the dataset $\mathcal{D}$, INK-ORACLE (Alg. 2) receives as input the regularization parameter $\gamma$ used in constructing the final Nyström approximation and a sampling budget $\overline{q}$. It initializes the index dictionary $\mathcal{I}_0$ of stored columns as empty, and the estimated probabilities as $\widetilde{p}_{i,0} = 1$. Finally it initializes a set of integer weights $b_{i,0} = 1$. These weights will represent a discretized approximation of $1/\widetilde{p}_{i,t}$ (the inverse of the probabilities). At each time step $t$, it receives a new column $\overline{\mathbf{k}}_{t+1}$ and $k_{t+1}$. This can be implemented either by hav-

ing a separate algorithm, constructing each column sequentially and stream it to INK-ORACLE, or by having INK-ORACLE store just the samples (for an additional $\mathcal{O}(td)$ space complexity) and independently compute the column once. The algorithm invokes the $(\alpha, \beta)$-oracle to compute approximate probabilities $\widetilde{p}_{i,t+1} = \widetilde{\tau}_{i,t+1}/\widetilde{d}_{\text{eff}}(\gamma)_{t+1}$, and then takes the minimum $\min\{\widetilde{p}_{i,t+1}, \widetilde{p}_{i,t}\}$ for the sampling probability. As our analysis will reveal, this step is necessary to ensure that the SHRINK-EXPAND operation remains well defined, since the true probabilities $p_{i,t}$ decrease over time. It is important to notice that differently from the batch sampling setting, the approximate probabilities do not necessarily sum to one, but it is guaranteed that $\sum_{i=1}^{t} \widetilde{p}_{i,t} \leq 1$. The SHRINK-EXPAND procedure is composed of two steps. In the SHRINK step, we update the weights of the columns already in our dictionary. To decide whether a weight should be increased or not, the product of the weight at the preceding step $b_{i,t-1}$ and the new estimate $\widetilde{p}_{i,t}$ is compared to a threshold. If the product is above the threshold, it means the probability did not change much, and no action is necessary. If the product falls below the threshold, it means the decrease of $\widetilde{p}_{i,t}$ is significant, and the old weight is not representative anymore and should be increased. To increase the weight (e.g. from $k$ to $k+1$), we draw a Bernoulli random variable $\mathcal{B}(\frac{k}{k+1})$, and if it succeeds we increase the weight to $k+1$, while if it fails we set the weight to 0. The more $\widetilde{p}_{i,t}$ decrease over time, the higher the chanches that $b_{i,t+1}$ is set to zero, and the index $i$ (and the associated column $\mathbf{k}_{i,t+1}$) is completely dropped from the dictionary. Therefore, the SHRINK step randomly reduces the size of the dictionary to reflect the evolution of the probabilities. Conversely, the EXPAND step introduces the new column in the dictionary, and quickly updates its weight $b_{t,t}$ to reflect $\widetilde{p}_{t,t}$. Depending on the relevance (encoded by the RLS) of the new column, this means that it is possible that the new column is discarded at the same iteration as it is introduced. For a whole pass over the dataset, INK-ORACLE queries the oracle for each RLS at least once, but it *never* asks again for the RLS of a columns dropped from $\mathcal{I}_t$. As we will see in the next section, this greatly simplifies the construction of the oracle. Finally, after updating the dictionary, we use the updated weights $\sqrt{b_{i,t}}$ to update the approximation $\widetilde{\mathbf{K}}_t$, that can be used at any time and not only in the end.

### 3.2 Analysis of INK-ORACLE

The main result of this section is the lower bound on the number of columns required to be kept in order to guarantee a $\gamma/(1-\varepsilon)$ approximation of $\mathbf{K}_t$.

**Theorem 1.** *Let $\gamma > 1$. Given access to an $(\alpha, \beta)$-oracle, for $0 \leq \varepsilon \leq 1$ and $0 \leq \delta \leq 1$, if we run Alg. 2 with parameter $\overline{q}$*

$$\overline{q} \geq \left(\frac{28\alpha\beta d_{\mathit{eff}}(\gamma)_t}{\varepsilon^2}\right)\log\left(\frac{4t}{\delta}\right),$$

*to compute a sequence of random matrices $\mathbf{S}_t$ with a random number of columns $Q_t$, then with probability $1-\delta$, for all $t$ the corresponding Nyström approximation $\widetilde{\mathbf{K}}_t$ (Eq. 5) satisfies condition in Eq. 11,*

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

*and the number of columns selected $Q_t$ is such that*

$$Q_t \leq 8\overline{q}.$$

**Discussion** Unlike in the batch setting, where the sampling procedure always returned $m$ samples, the number of columns $Q_t$ selected by INK-ORACLE is a random variable, but with high probability it will be not much larger than $\overline{q}$. Comparing INK-ORACLE to online kernel sparsification methods [19], we see that the number of columns, and therefore the space requirement, is guaranteed to be small not only asymptotically but at each step, and that no assumption on the spectrum of the matrix is required. Instead, the space complexity naturally scales with the effective dimension of the problem, and old samples that become superfluous are automatically discarded. Comparing Thm. 1 to Prop.1, INK-ORACLE achieves the same performance as its batch counterpart, as long as the space budget $\overline{q}$ is large enough. This budget depends on several quantities that are difficult to estimate, such as the effective dimension of the full kernel matrix. In practice, this quantity can be interpreted as the maximum amount of space that the user can afford for the algorithm to run. If the actual complexity of the problem exceeds this budget, the user can choose to run it again with another parameter $\gamma$ or a worse accuracy $\varepsilon$. It is important to notice that, as we show in the proof, the distribution induced by the sampling procedure of INK-ORACLE is not the same as the distribution obtained by the multinomial sampling of BATCH-EXACT. Nonetheless, in our analysis we show that the bias introduced by the different distribution is small, and this allows INK-ORACLE to match the approximation guarantees given by Alaoui and Mahoney [1].

We give a detailed proof of Thm. 1 in App. B. In the rest of this section we sketch the proof and give the intuition for the most relevant parts.

The SHRINK step uses the thresholding condition to guarantee that the weight $b_{i,t}$ are good approximations of the $\widetilde{p}_{i,t}$. To make the condition effective, we require that the approximate probabilities $\widetilde{p}_{j,t}$ are decreasing. Because the approximate probabilities follow the true probabilities $p_{i,t}$, we first show that this decrease happens for the exact case.

**Lemma 1.** *For any kernel matrix $\mathbf{K}_t$ at time $t$, and its bordering $\mathbf{K}_{t+1}$ at time $t+1$ we have that the probabilities $p_{i,t}$ are monotonically decreasing over time $t$,*

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

Since ridge leverage scores represent the importance of a column, when a new column arrives, there are two cases that can happen. If the column is orthogonal to the existing matrix, none of the previous leverage scores changes. If the new column can explain part of the previous columns, the previous columns should be picked less often, and we expect $\tau_{i,t}$ to decrease. Contrary to RLS, the effective dimension increases when the new sample is orthogonal to the existing matrix, while it stays the same when the new sample is a linear combination of the existing ones. In addition, the presence of $\gamma$ regularizes both cases. When the vector is nearly orthogonal, the presence of $\gamma\mathbf{I}$ in the inverse will still penalize it, while the $\gamma$ term at the denominator of $\Delta$ will reduce the influence of linearly correlated samples. Because $\tau_{i,t}$ decreases over time and $d_{\text{eff}}(\gamma)_{i,t}$ increases, the probabilities $p_{i,t}$ will overall decrease over time. This result itself is not sufficient to guarantee a well defined SHRINK step. Due to the $(\alpha, \beta)$-approximation, it is possible that $p_{i,t+1} \leq p_{i,t}$ but $\widetilde{p}_{i,t+1} \not\leq \widetilde{p}_{i,t}$. To exclude this possibility, we adapt the following idea from Kelner and Levin [9].

**Proposition 2** (Kelner and Levin [9])**.** *Given the approximate probabilities $\widetilde{\mathbf{p}}_t$ returned by an $(\alpha, \beta)$-oracle at time $t$, and the approximate probabilities $\widetilde{\mathbf{p}}_{t+1}$ returned by an $(\alpha, \beta)$-oracle at time $\{t+1\}$, then the approximate probabilities $\min^3\{\widetilde{\mathbf{p}}_t, \widetilde{\mathbf{p}}_{t+1}\}$ are also $(\alpha, \beta)$-approximate for $\{t+1\}$. Therefore, without loss of generality, we can assume that $\widetilde{p}_{i,t+1} \leq \widetilde{p}_{i,t}$.*

Combining Lemma 1 and Proposition 2, we can guarantee that at each step the $\widetilde{p}_{i,t}$-s decrease. Unlike in the batch setting [1], we have to take additional care to consider correlations between iterations, the fact that the inclusion probabilities of Algorithm 2 are different from the multinomial ones of DIRECT-SAMPLE, and that the number of columns kept at each iteration is a *random* quantity $Q_t$. We adapt the approach of Pachocki [13] to the KRR setting to analyse this process. The key aspect is that the reweighting and rejection rule on line 3 of Algorithm 2 will only happen when the probabilities are truly changing. Finally, using a concentration inequality, we show that the number $Q_t$ of columns selected is with high probability only a constant factor away from the budget $\overline{q}$ given to the algorithm.

# 4 LEVERAGE SCORES AND EFFECTIVE DIMENSION ESTIMATION

In the previous section we showed that our incremental sampling strategy based on (estimated) RLSs has strong space and approximation guarantees for $\widetilde{\mathbf{K}}_n$. While the analysis reported in the previous section relied on the existence of an $(\alpha, \beta)$-oracle returning accurate leverage scores and effective dimension estimates, in this section we show that such an oracle *exists and can be implemented efficiently.* This is obtained by *two separate estimators* for the RLSs and effective dimension that are updated incrementally and combined together to determine the sampling probabilities.

[3] element-wise mininum

## 4.1 Leverage Scores

We start by constructing an estimator that at each time $t$, takes as input an approximate kernel matrix $\widetilde{\mathbf{K}}_t$, and returns $\alpha$-approximate RLS $\widetilde{\tau}_{i,t+1}$. The incremental nature of the estimator lies in the fact that it exploits access to the columns already in $\mathbf{S}_t$ and the new (exact) column $\overline{\mathbf{k}}_{t+1}$. We give the following approximation guarantees.

**Lemma 2.** *We assume that $\widetilde{\mathbf{K}}_t$ satisfies Eq. (11), and define $\overline{\mathbf{K}}_{t+1}$ as the matrix bordered with the new row and column*

$$\overline{\mathbf{K}}_{t+1} = \left[\begin{array}{c|c} \widetilde{\mathbf{K}}_t & \overline{\mathbf{k}}_{t+1} \\ \hline \overline{\mathbf{k}}_{t+1}^\intercal & k_{t+1} \end{array}\right].$$

*Then*

$$\mathbf{0} \preceq \mathbf{K}_{t+1} - \overline{\mathbf{K}}_{t+1} \preceq \frac{\gamma}{1-\varepsilon}\mathbf{I}.$$

*Moreover let $\alpha = \frac{2-\varepsilon}{1-\varepsilon}$ and*

$$\widetilde{\tau}_{i,t+1} = \frac{1}{\alpha\gamma}\left(k_{i,i} - \mathbf{k}_{i,t+1}\left(\overline{\mathbf{K}}_{t+1} + \alpha\gamma\mathbf{I}\right)^{-1}\mathbf{k}_{i,t+1}\right). \tag{12}$$

*Then, for all $i$ such that $\mathbf{k}_{i,t+1} \in \mathcal{I}_t \cup \{t+1\}$,*

$$\frac{1}{\alpha}\tau_{i,t+1}(\gamma) \leq \widetilde{\tau}_{i,t+1} \leq \tau_{i,t+1}(\gamma).$$

**Remark** There are two important details that are used in proof of Lem. 2 (App. C). First, notice that using $\widetilde{\mathbf{K}}_t$ to approximate RLSs directly, would not be accurate enough. RLSs are defined as $\tau_{i,t}(\gamma) = \mathbf{e}_i^\intercal \mathbf{K}_t(\mathbf{K}_t + \gamma\mathbf{I})^{-1}\mathbf{e}_i$ and while the product $(\mathbf{K}_t + \gamma\mathbf{I})^{-1}\mathbf{e}_i$ can be accurately reconstructed using $(\widetilde{\mathbf{K}}_t + \gamma\mathbf{I})^{-1}\mathbf{e}_i$, the multiplication $\mathbf{K}_t\mathbf{e}_i$ cannot be approximated well using $\widetilde{\mathbf{K}}_t$. Since the nullspace of $\widetilde{\mathbf{K}}_t$ can be larger than the one of $\mathbf{K}_t$, it is possible that $\mathbf{e}_i$ partially falls into it, thus compromising the accuracy of the approximation of the RLS. In our approach, we deal with this problem by using the *actual columns* $\mathbf{k}_{i,t}$ of $\mathbf{K}_t$ to compute the RLS. This way, we preserve as much as exact information of the matrix as possible, while the expensive inversion operation is performed on the smaller approximation $\widetilde{\mathbf{K}}_t$. Since we require access to the stored columns $\mathbf{k}_{i,t}$, our approach can approximate the RLSs only for columns present in the dictionary but this is enough, since we are only interested in accurate probabilities for columns in the dictionary and for the new column $\overline{\mathbf{k}}_{t+1}$ (which is available at time $t+1$). As a comparison, the

two-pass approach of Alaoui and Mahoney [1] uses the first pass just to compute an approximation $\widetilde{\mathbf{K}}_n$, and then approximates all leverage scores with $\widetilde{\mathbf{K}}_n(\widetilde{\mathbf{K}}_n + \gamma\mathbf{I})^{-1}$. This has an impact on their approximation factor $\alpha$, that is proportional to $(\lambda_{\min}(\mathbf{K}_n) - \gamma\varepsilon)$. Therefore to have $\alpha \approx (\lambda_{\min}(\mathbf{K}_n) - \gamma\varepsilon) > 0$, it is necessary that $\gamma\varepsilon$ is of the order of $\lambda_{\min}(\mathbf{K}_n)$, which in some cases can be very small, and strongly increase the space requirements of the algorithm. Using the actual columns of the matrix in Eq. 12 allows us to compute an $\alpha$-approximation independent of the smallest eigenvalue.

### 4.2 Effective Dimension

Using Eq. 12, we can estimate all the RLSs that we need to update $\mathbf{S}_t$. Nonetheless, to prove that the number of columns selected is not too large, the proof of Thm. 1 in the appendix requires that the sum of the probabilities $\widetilde{p}_{i,t}$ is smaller than 1. Therefore we not only need to compute the RLSs, but also a normalization constant. Indeed, a naïve definition of the probability $\widetilde{p}_{i,t}$ could be $p_{i,t} = \frac{\widetilde{\tau}_{i,t}}{\sum_{j=1}^{t}\widetilde{\tau}_{i,t}}$. A major challenge in our setting is that we cannot compute the sum of the approximate RLSs, because we do not have access to all the columns. Fortunately, we know that $\sum_{j=1}^{t}\widetilde{\tau}_{i,t} \leq \sum_{j=1}^{t}\tau_{i,t}(\gamma) = d_{\text{eff}}(\gamma)_t$. Therefore, one of our technical contribution is an estimator $\widetilde{d}_{\text{eff}}(\gamma)_t$ that does not use the approximate RLSs for the the columns that we no longer have. We now define this estimator and state its approximation accuracy.

**Lemma 3.** *Assume $\widetilde{\mathbf{K}}_t$ satisfies Eq. 11. Let $\alpha = \left(\frac{2-\varepsilon}{1-\varepsilon}\right)$ and $\beta = \left(\frac{2-\varepsilon}{1-\varepsilon}\right)^2(1+\rho)$ with $\rho = \frac{\lambda_{\max}(\mathbf{K}_n)}{\gamma}$. Define*

$$\widetilde{d}_{\textit{eff}}(\gamma)_{t+1} = \widetilde{d}_{\textit{eff}}(\gamma)_t + \alpha\widetilde{\Delta}_t \tag{13}$$

*with*

$$\begin{aligned}\widetilde{\Delta}_t =& \frac{1}{k_{t+1} + \gamma - \overline{\mathbf{k}}_{t+1}^{\mathsf{T}}\left(\widetilde{\mathbf{K}}_t + \alpha\gamma\mathbf{I}\right)^{-1}\overline{\mathbf{k}}_{t+1}} \\ &\times \left(k_{t+1} - \overline{\mathbf{k}}_{t+1}^{\mathsf{T}}\left(\widetilde{\mathbf{K}}_t + \alpha\gamma\mathbf{I}\right)^{-1}\overline{\mathbf{k}}_{t+1}\right. \\ &\left. - \frac{(1-\varepsilon)^2}{4}\gamma\overline{\mathbf{k}}_{t+1}^{\mathsf{T}}(\widetilde{\mathbf{K}}_t + \gamma\mathbf{I})^{-2}\overline{\mathbf{k}}_{t+1}\right). \end{aligned} \tag{14}$$

*Then*

$$d_{\textit{eff}}(\gamma)_{t+1} \leq \widetilde{d}_{\textit{eff}}(\gamma)_{t+1} \leq \beta d_{\textit{eff}}(\gamma)_{t+1}.$$

**Discussion** Since we cannot compute accurate RLSs for columns that are not present in the dictionary, we prefer to not estimate how each RLSs changes over time, but instead we directly estimate the increment of their sum. We do it by updating our estimate $\widetilde{d}_{\text{eff}}(\gamma)_{t+1}$ using our previous estimate $\widetilde{d}_{\text{eff}}(\gamma)_t$, and $\widetilde{\Delta}_t$. $\widetilde{\Delta}_t$ captures directly the interaction of the new sample with the *aggregate* of the previous samples, and allows us to estimate the increase in effective dimension using only the current matrix approximation $\widetilde{\mathbf{K}}_t$, the new column $\overline{\mathbf{k}}_{t+1}$ and the scalar $k_{t+1}$. Differently from the other terms we studied, the numerator of $\widetilde{\Delta}_t$ contains an additional $\gamma\overline{\mathbf{k}}_{t+1}^{\mathsf{T}}(\widetilde{\mathbf{K}}_t + \gamma\mathbf{I})^{-2}\overline{\mathbf{k}}_{t+1}$ second order term. The guarantees provided by Eq. 11 are not straightforward to extend because in general if $(\mathbf{K}_t+\gamma\mathbf{I})^{-1} \succeq (\widetilde{\mathbf{K}}_t+\alpha\gamma\mathbf{I})^{-1}$, it is not guaranteed that $(\mathbf{K}_t + \gamma\mathbf{I})^{-2} \succeq (\widetilde{\mathbf{K}}_t + \alpha\gamma\mathbf{I})^{-2}$. Nonetheless, we show that $\widetilde{\mathbf{K}}_t$ is still sufficient to estimate $\widetilde{\Delta}_t$, but, unlike $\alpha$, the approximation error $\beta$ is now dependent on the spectrum.

**Algorithm 3** The INK-ESTIMATE algorithm

**Input:** Dataset $\mathcal{D}$, regularization $\gamma$, sampling budget $\overline{q}$

**Output:** $\widetilde{\mathbf{K}}_n$, $\mathbf{S}_n$

Initialize $\mathcal{I}_0$ as empty, $\widetilde{p}_{1,0} = 1, b_{1,0} = 1$, budget $\overline{q}$
**for** $t = 0, \dots, n-1$ **do**
  Receive new column $\overline{\mathbf{k}}_{t+1}$ and scalar $k_{t+1}$
  Compute $\alpha$-leverage scores $\{\widetilde{\tau}_{i,t+1} : i \in \mathcal{I}_t \cup \{t+1\}\}$, using $\overline{\mathbf{K}}_{t+1}$, $\mathbf{k}_i$, $k_{i,i}$, and Eq. (12)
  Compute $\beta$-approximate $\widetilde{d}_{\text{eff}}(\gamma)_{t+1}$ using $\widetilde{\mathbf{K}}_t$, $\overline{\mathbf{k}}_{t+1}$, $k_{t+1}$, and Eq. (13)
  Set $\widetilde{p}_{i,t+1} = \min\{\widetilde{\tau}_{i,t+1}/\widetilde{d}_{\text{eff}}(\gamma)_{t+1}, \widetilde{p}_{i,t}\}$
  $\mathcal{I}_{t+1}, \mathbf{b}_{t+1}$ = SHRINK-EXPAND$(\mathcal{I}_t, \widetilde{\mathbf{p}}_{t+1}, \mathbf{b}_t, \overline{q})$
  Compute $\mathbf{S}_{t+1}$ using $\mathcal{I}_{t+1}$ and weights $\sqrt{b_{i,t+1}}$
  Compute $\widetilde{\mathbf{K}}_{t+1}$ using $\mathbf{S}_{t+1}$ and Equation 5
**end for**
Return $\widetilde{\mathbf{K}}_n$ and $\mathbf{S}_n$

### 4.3 Analysis of INK-ESTIMATE

With the separate estimates for leverage scores (Sect. 4.1) and effective dimension (Sect. 4.2), we have the necessary ingredients for the $(\alpha, \beta)$-oracle and we are ready to present the final algorithm INK-ESTIMATE (Alg. 3).

Using the approximation guarantees of Lem. 2 and Lem. 3, we are ready to state the final result, instantiating the generic $\alpha$ and $\beta$ terms of Thm. 2 with the values obtained in this section.

**Theorem 2.** *Let $\rho = \lambda_{\max}(\mathbf{K}_t)/\gamma$, $\alpha = \left(\frac{2-\varepsilon}{1-\varepsilon}\right)$, $\beta = \left(\frac{2-\varepsilon}{1-\varepsilon}\right)^2(1+\rho)$, and $\gamma > 1$. For any $0 \leq \varepsilon \leq 1$, and $0 \leq \delta \leq 1$, if we run Alg. 3 with parameter $\overline{q}$, where*

$$\overline{q} \geq \left(\frac{28\alpha\beta d_{\textit{eff}}(\gamma)_t}{\varepsilon^2}\right)\log\left(\frac{4t}{\delta}\right),$$

*to compute a sequence of random matrices $\mathbf{S}_t$ with a random number of columns $Q_t$, then with probability $1-\delta$, for all $t$ the corresponding Nyström approximation $\widetilde{\mathbf{K}}_t$ (Eq. 5)*

*satisfies condition 11*

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

*With the same prob.,* INK-ESTIMATE *requires at most*

$$\begin{aligned}\mathcal{O}(n^2\overline{q}^2 + n\overline{q}^3) &\\ &\leq \mathcal{O}\left(\alpha^2\beta^2 n^2 d_{\mathit{eff}}(\gamma)_n^2 + \alpha^3\beta^3 n d_{\mathit{eff}}(\gamma)_n^3\right)\\ &= \mathcal{O}\left(\alpha^4(1+\rho)^2 n^2 d_{\mathit{eff}}(\gamma)_n^2 + \alpha^6(1+\rho)^3 n d_{\mathit{eff}}(\gamma)_n^3\right)\end{aligned}$$

*time and the space is bounded as*

$$\mathcal{O}(n\overline{q}) \leq \mathcal{O}\left(\alpha\beta n d_{\mathit{eff}}(\gamma)_n\right) = \mathcal{O}\left(\alpha^2(1+\rho) n d_{\mathit{eff}}(\gamma)_n\right).$$

For the space complexity, from Theorem 1 we know we will not select more than $\mathcal{O}(\overline{q})$ columns in high probability. For the time complexity, at each iteration we need to solve linear systems involving $(\overline{\mathbf{K}}_{t+1} + \alpha\gamma\mathbf{I})^{-1}$ and $(\widetilde{\mathbf{K}}_t + \alpha\gamma\mathbf{I})^{-1}$. Approximating the inverse using transformations similar to Eq. (6) takes $\mathcal{O}(t\overline{q}^2 + \overline{q}^3)$ time, again using a singular value decomposition approach. To compute all leverage scores, we need to first compute an approximate inverse in $\mathcal{O}(t\overline{q}^2 + \overline{q}^3)$ time, and then solve $Q_t$ systems, each using a multiplication costing $\mathcal{O}(tQ_t)$. With high probability, $Q_t \leq 8\overline{q}$, therefore computing all leverage scores costs $\mathcal{O}(t\overline{q}^2 + \overline{q}^3)$ for the first singular value decomposition, and $\mathcal{O}(t\overline{q})$ for each of the $\mathcal{O}(\overline{q})$ applications. To update the effective dimension estimate, we only have to compute another approximate inverse, and that costs $\mathcal{O}(t\overline{q}^2 + \overline{q}^3)$ as well. Finally, we have to sum the costs over $n$ steps, and from $\sum_{t=1}^{n} t\overline{q}^2 \leq \overline{q}^2 n^2$, we obtain the final complexity. Even with a significantly different approach, INK-ESTIMATE achieves the same approximation guarantees as BATCH-EXACT. Consequently, it provides the same risk guarantees as the known batch version [1], stated in the following corollary.

**Corollary 1.** *For every $t \in \{1, \ldots, n\}$, let $\mathbf{K}_t$ be the kernel matrix at time $t$. Run Algorithm 3 with regularization parameter $\gamma$ and space budget $\overline{q}$. Then, at any time $t$, the solution $\widetilde{\mathbf{w}}_t$ computed using the regularized Nyström approximation $\widetilde{\mathbf{K}}_t$ satisfies*

$$\begin{aligned}\mathcal{R}(\widetilde{\mathbf{w}}_t) &\leq \left(1 + \frac{\gamma}{\mu}\frac{1}{1-\varepsilon}\right)^2 \mathcal{R}(\widehat{\mathbf{w}}_t)\\ &= \left(1 + \frac{\lambda_{\max}(\mathbf{K}_t)}{\rho\mu}\frac{1}{1-\varepsilon}\right)^2 \mathcal{R}(\widehat{\mathbf{w}}_t).\end{aligned}$$

**Discussion** Thm. 2 combines the generic result of Thm. 1 with the actual implementation of an oracle that we developed in this section. All the guarantees that hold for INK-ORACLE are inherited by INK-ESTIMATE, but now we can quantify the impact of the errors $\alpha$ and $\beta$ on the algorithm. As we saw, the $\alpha$ error does not depends on the time, the spectrum of the kernel matrix or other quantities that increase over time. On the other hand, estimating the effective dimension without having access to all the leverage scores is a much harder task, and the $\beta$ factor depends on the spectrum through the $\rho$ coefficient. The influence that this coefficient exerts on the space and time complexity can vary significantly as the relative magnitude of $\lambda_{\max}(\mathbf{K}_n)$, $\gamma$ and $\mu$ changes. If the largest eigenvalue grows too large without a corresponding increase in $\gamma$, the space and time requirements of INK-ESTIMATE can grow, but the risk bound, depending on $\gamma/\mu$ remains small. On the other hand, increasing $\gamma$ without increasing $\mu$ reduces the computational complexity, but makes the guarantees on the risk of the solution $\widetilde{\mathbf{w}}_t$ much weaker. As an example, [1, Thm. 3] chooses, $\mu \geq \lambda_{\max}(\mathbf{K}_n)$ and $\gamma \approxeq \mu$. If we do the same, we recover their bound.

# 5 CONCLUSION

We presented a space-efficient algorithm for sequential Nyström approximation that requires only a single pass over the dataset to construct a low-rank matrix $\widetilde{\mathbf{K}}_n$ that accurately approximates the kernel matrix $\mathbf{K}_n$, and compute an approximate KRR solution $\widetilde{\mathbf{w}}_n$ whose risk is close to the exact solution $\widehat{\mathbf{w}}_n$. All of these guarantees do not hold only for the final matrix, but are valid for all intermediate matrices $\widetilde{\mathbf{K}}_t$ constructed by the sequential algorithm.

To address the challenges coming from the sequential setup, we introduced two separate estimators for RLSs and effective dimension that provide multiplicative error approximations of these two quantities across iterations. While the approximation of the RLSs is only a constant factor away from the exact RLSs, the error of the approximate effective dimension scales with the spectrum of the matrix through the coefficient $\rho$. A more careful analysis, or a different estimator might improve this dependence, and they can be easily plugged to the general analysis.

Our generalization results apply to the fixed design setting. An important extension of our work would be to consider a random design, such as in the work of Rudi et al. [16]. This extension would need even more careful tuning of the regularization parameter $\gamma$, needing to satisfy requirements of both generalization and the approximation of the $(\alpha, \beta)$-oracle. Finally, the runtime analysis of the algorithm does not fully exploit the sequential nature of the updates. An implementation based on decompositions more amenable to updates (e.g., Cholesky decomposition), or on low-rank solvers that can exploit hot-start might further improve the time complexity.

**Acknowledgements** The research presented in this paper was supported by CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020, French Ministry of Higher Education and Research, Nord-Pas-de-Calais Regional Council, French National Research Agency project ExTra-Learn (n.ANR-14-CE24-0010-01).

## 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 *International Conference on Learning Theory*, 2013.

[3] Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, and David P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *International Conference on Machine Learning*, 2012.

[4] Yaakov Engel, Shie Mannor, and Ron Meir. Sparse online greedy support vector regression. In *European Conference on Machine Learning*, 2002.

[5] Yaakov Engel, Shie Mannor, and Ron Meir. The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing*, 52(8):2275–2285, 2004.

[6] B. S. Everitt. *The Cambridge dictionary of statistics*. Cambridge University Press, Cambridge, 2002.

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

[8] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. Society for Industrial and Applied Mathematics, 2002.

[9] Jonathan A. Kelner and Alex Levin. Spectral sparsification in the semi-streaming setting. *Theory of Computing Systems*, 53(2):243–262, 2012.

[10] Jyrki Kivinen, Alexander J. Smola, and Robert C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2004.

[11] Quoc Le, Tamás Sarlós, and Alex J Smola. Fastfood — Approximating kernel expansions in loglinear time. In *International Conference on Machine Learning*, 2013.

[12] Weifeng Liu, Il Park, and Jose C. Principe. An information theoretic approach of designing sparse kernel adaptive filters. *IEEE Transactions on Neural Networks*, 20(12):1950–1961.

[13] Jakub Pachocki. Analysis of resparsification. *arXiv preprint arXiv:1605.08194*, 2016.

[14] Ali Rahimi and Ben Recht. Random features for large-scale kernel machines. In *Neural Information Processing Systems*, 2007.

[15] Cédric Richard, José Carlos M. Bermudez, and Paul Honeine. Online prediction of time series data with kernels. *IEEE Transactions on Signal Processing*, 57 (3):1058–1067.

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

[17] Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press, 2001.

[18] John Shawe-Taylor and Nelo Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

[19] Yi Sun, Jürgen Schmidhuber, and Faustino J. Gomez. On the size of the online kernel sparsification dictionary. In *International Conference on Machine Learning*, 2012.

[20] Joel A Tropp. Freedman's inequality for matrix martingales. *Electron. Commun. Probab*, 16:262–270, 2011.

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

[22] Steven Van Vaerenbergh, Miguel Lázaro-Gredilla, and Ignacio Santamaría. Kernel recursive least-squares tracker for time-varying regression. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(8):1313–1326, 2012.

[23] Christopher Williams and Matthias Seeger. Using the Nystrom method to speed up kernel machines. In *Neural Information Processing Systems*, 2001.

[24] Yuchen Zhang, John C. Duchi, and Martin J. Wainwright. Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates. *Journal Machine Learning Research*, 16:3299–3340, 2015.