Incremental Randomized Sketching for Online Kernel Learning

Xiao Zhang [1] **Shizhong Liao** [1]

Abstract

Randomized sketching has been used in offline kernel learning, but it cannot be applied directly to online kernel learning due to the lack of incremental maintenances for randomized sketches with regret guarantees. To address these issues, we propose a novel incremental randomized sketching approach for online kernel learning, which has efficient incremental maintenances with theoretical guarantees. We construct two incremental randomized sketches using the sparse transform matrix and the sampling matrix for kernel matrix approximation, update the incremental randomized sketches using rank-1 modifications, and construct a time-varying explicit feature mapping for online kernel learning. We prove that the proposed incremental randomized sketching is statistically unbiased for the matrix product approximation, obtains a $1+\epsilon$ relative-error bound for the kernel matrix approximation, enjoys a sublinear regret bound for online kernel learning, and has constant time and space complexities at each round for incremental maintenances. Experimental results demonstrate that the incremental randomized sketching achieves a better learning performance in terms of accuracy and efficiency even in adversarial environments.

1. Introduction

Online kernel learning must rise to the challenge of incremental maintenance in a low computational complexity with accuracy and convergence guarantees at each round. Online gradient descent is a judicious choice for online kernel learning due to its efficiency and effectiveness (Bottou, 2010; Shalev-Shwartz et al., 2011). However, the kernelization of the online gradient descent may cause a linear growth with respect to the model size and increase the hypothesis updating time significantly (Wang et al., 2012). To solve the problem, budgeted online kernel learning has been proposed to maintain a buffer of support vectors (SVs) with a fixed budget, which limits the model size to reduce the computational complexities (Crammer et al., 2003). Several budget maintenance strategies were introduced to bound the buffer size of SVs. Dekel et al. (2005) maintained the budget by discarding the oldest support vector. Orabona et al. (2008) projected the new example onto the linear span of SVs in the feature space to reduce the buffer size. But the buffer of SVs cannot be used for kernel matrix approximation directly. Besides, it is difficult to provide a lower bound of the buffer size that is key to the theoretical guarantees for budgeted online kernel learning.

Unlike the budget maintenance strategies of SVs, sketches of the kernel matrix have been introduced into online kernel learning to approximate the kernel incrementally. Engel et al. (2004) used the approximate linear dependency (ALD) rule to construct the sketches for kernel approximation in kernel regression. But the kernel matrix approximation error is not compared with the best rank-k approximation. Sun et al. (2012) proved that the size of sketches constructed by ALD rule grows sublinearly with respect to the data size when the eigenvalues decay fast enough. Recently, randomized sketches of kernel matrix were used as surrogates to perform the offline kernel approximation with finite sample bounds (Sarlós, 2006; Woodruff, 2014; Raskutti & Mahoney, 2015), but few efforts have been made to formulate suitable randomized sketches for online kernel learning. Existing online kernel learning approaches using randomized sketches usually adopted a random sampling strategy. Lu et al. (2016) applied the Nyström approach for kernel function approximation and constructed the explicit feature mapping using the singular value decomposition, but it fixed the randomized sketches after initialization, which is equivalent to the offline Nyström approach using uniform sampling that may suffer from a low approximation accuracy in adversarial environments due to the non-uniformity structure of the data (Kumar et al., 2009). Calandriello et al. (2017b) assumed that the kernel function can be expressed as an inner product in the explicit feature space, and proposed a second order online kernel learning approach using online sampling strategy, but the construction of the explicit feature mapping needs to be further explored. Calandriello et al. (2017a)

[1]College of Intelligence and Computing, Tianjin University, Tianjin 300350, China. Correspondence to: Shizhong Liao <szliao@tju.edu.cn>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

formulated an explicit feature mapping using a Nyström approximation, and performed effective second-order updates for hypothesis updating, which enjoys a logarithmic regret and has a per-step cost that grows with the effective dimension.

In this paper, we propose an incremental randomized sketching approach for online kernel learning. The proposed incremental randomized sketching inherits the efficient incremental maintenance for matrix approximations and explicit feature mappings, has a constant time complexity at each round, achieves a $1+\epsilon$ relative-error for kernel matrix approximation, and enjoys a sublinear regret for online kernel learning, therefore presenting an effective and efficient sketching approach to online kernel learning.

2. Notations and Preliminaries

Let $[\boldsymbol{A}]_{*i}$ and $[\boldsymbol{A}]_{i*}$ denote the i-th column of the matrix $\boldsymbol{A}$ and the column vector consisting of the entries from the i-th row of $\boldsymbol{A}$ respectively, $\|\boldsymbol{A}\|_{\mathrm{F}}$ and $\|\boldsymbol{A}\|_2$ the Frobenius and spectral norm of $\boldsymbol{A}$ respectively, and $\boldsymbol{A}^\dagger$ the Moore-Penrose pseudoinverse of $\boldsymbol{A}$. Let $[M]=\{1,2,\ldots,M\}$, $\mathcal{S}=\{(\boldsymbol{x}_i,y_i)\}_{i=1}^T\subseteq(\mathcal{X}\times\mathcal{Y})^T$ be the sequence of T instances, where $\mathcal{X}\subseteq\mathbb{R}^D$ and $\mathcal{Y}=\{-1,1\}$. We denote the loss function by $\ell:\mathcal{Y}\times\mathcal{Y}\to\mathbb{R}^+\cup\{0\}$, the kernel function by $\kappa:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ and its corresponding kernel matrix by $\boldsymbol{K}=(\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j))\in\mathbb{R}^{T\times T}$, the i-th eigenvalue of $\boldsymbol{K}$ by $\lambda_i(\boldsymbol{K})$, $\lambda_1(\boldsymbol{K})\geq\lambda_2(\boldsymbol{K})\geq\cdots\geq\lambda_T(\boldsymbol{K})$, and the reproducing kernel Hilbert space (RKHS) associated with κ by $\mathcal{H}_\kappa=\overline{\mathrm{span}\{\kappa(\cdot,\boldsymbol{x}):\boldsymbol{x}\in\mathcal{X}\}}$.

2.1. Kernel Matrix Approximation

Given a kernel matrix $\boldsymbol{K}\in\mathbb{R}^{T\times T}$, the sketched kernel matrix approximation problem in the offline setting is described as follows (Wang et al., 2016):

$$\boldsymbol{F}_{\mathrm{fast}}=\operatorname*{arg\,min}_{\boldsymbol{F}\in\mathbb{R}^{B\times B}}\|\boldsymbol{S}^\intercal(\boldsymbol{C}\boldsymbol{F}\boldsymbol{C}^\intercal-\boldsymbol{K})\boldsymbol{S}\|_{\mathrm{F}}^2,$$

from which we can obtain the approximate kernel matrix $\boldsymbol{K}_{\mathrm{fast}}=\boldsymbol{C}\boldsymbol{F}_{\mathrm{fast}}\boldsymbol{C}^\intercal\approx\boldsymbol{K}$, where $\boldsymbol{S}\in\mathbb{R}^{T\times s}$ is a sketch matrix, $\boldsymbol{C}\in\mathbb{R}^{T\times B}$ and $s,B>0$ are the sketch sizes. The Nyström approach uses the column selection matrix $\boldsymbol{P}$ as the sketch matrix (Williams & Seeger, 2001), i.e., $\boldsymbol{S}=\boldsymbol{P}\in\mathbb{R}^{T\times B}$, and formulate $\boldsymbol{C}$ as $\boldsymbol{C}=\boldsymbol{K}\boldsymbol{P}$, which yields the approximate kernel matrix $\boldsymbol{K}_{\mathrm{ny}}=\boldsymbol{C}\boldsymbol{W}^\dagger\boldsymbol{C}^\intercal$, where $\boldsymbol{W}=\boldsymbol{P}^\intercal\boldsymbol{K}\boldsymbol{P}\in\mathbb{R}^{B\times B}$. While $\boldsymbol{S}=\boldsymbol{I}_T\in\mathbb{R}^{T\times T}$, the modified Nyström approach (also called the prototype model) (Wang & Zhang, 2013) is obtained as

$$\boldsymbol{K}_{\mathrm{mod}}=\boldsymbol{C}\boldsymbol{F}_{\mathrm{mod}}\boldsymbol{C}^\intercal\approx\boldsymbol{K},$$

where $\boldsymbol{F}_{\mathrm{mod}}=\boldsymbol{C}^\dagger\boldsymbol{K}\left(\boldsymbol{C}^\intercal\right)^\dagger$, which is more accurate but more computationally expensive than the standard Nyström. But in the context of online kernel learning, the size of the kernel matrix increases over time, a scenario for which the offline kernel matrix approximation approaches do not readily apply due to the lack of efficient incremental maintenances with regret guarantees.

2.2. Randomized Sketches

Randomized sketches via hash functions can be described in general using hash-based sketch matrices. Without loss of generality, we assume that the sketch size s is divisible by d. Let $h_1,\ldots,h_d$ be hash functions from $\{1,\ldots,n\}$ to $\{1,\ldots,s/d\}$, and $\sigma_1,\ldots,\sigma_d$ the other d hash functions. We denote the hash-based sketch submatrix by $\boldsymbol{S}_k\in\mathbb{R}^{n\times(s/d)}$ and the hash-based sketch matrix by $\boldsymbol{S}=[\boldsymbol{S}_1,\boldsymbol{S}_2,\ldots,\boldsymbol{S}_d]\in\mathbb{R}^{n\times s}$, where $[\boldsymbol{S}_k]_{i,j}=\sigma_k(i)$ for $j=h_k(i)$ and $[\boldsymbol{S}_k]_{i,j}=0$ for $j\neq h_k(i)$, $k=1,\ldots,d$. The randomized sketch of $\boldsymbol{A}\in\mathbb{R}^{T\times n}$ is denoted by $\boldsymbol{A}\boldsymbol{S}\in\mathbb{R}^{T\times s}$. Sketch matrices of different randomized sketches can be specified by different parameters s, d and functions $\{\sigma_k\}_{k=1}^d$ as follows.

Count Sketch Matrix (Charikar et al., 2004): $d=1$ and $\sigma_1(i)\in\{-1,+1\}$ is 2-wise independent hash function.

Sparser Johnson-Lindenstrauss Transform (SJLT) (Kane & Nelson, 2014): For the block construction of SJLT, $\sigma_k(i)\in\{-1/\sqrt{d},+1/\sqrt{d}\}$ is $O(\log T)$-wise independent hash function for $k=1,\ldots,d$, where d is the number of blocks. SJLT is a generalization of the Count Sketch matrix.

3. Novel Incremental Randomized Sketching

In this section, we propose a novel incremental randomized sketching approach. Specifically, we construct the incremental randomized sketches for the kernel matrix approximation, formulate the incremental maintenance for the incremental randomized sketches, and build a time-varying explicit feature mapping. In the online setting, at round $t+1$, a new example $\boldsymbol{x}_{t+1}$ arrives and the kernel matrix $\boldsymbol{K}^{(t+1)}$ can be represented as a bordered matrix as follows:

$$\boldsymbol{K}^{(t+1)}=\begin{bmatrix}\boldsymbol{K}^{(t)} & \boldsymbol{\psi}^{(t+1)}\\ \boldsymbol{\psi}^{(t+1)\intercal} & \xi^{(t+1)}\end{bmatrix}\in\mathbb{R}^{(t+1)\times(t+1)},$$

where $\xi^{(t+1)}=\kappa(\boldsymbol{x}_{t+1},\boldsymbol{x}_{t+1})$ and

$$\boldsymbol{\psi}^{(t+1)}=[\kappa(\boldsymbol{x}_{t+1},\boldsymbol{x}_1),\kappa(\boldsymbol{x}_{t+1},\boldsymbol{x}_2),\ldots,\kappa(\boldsymbol{x}_{t+1},\boldsymbol{x}_t)]^\intercal.$$

To approximate the kernel matrix incrementally, we introduce a sketch matrix $\boldsymbol{S}_{\mathrm{p}}^{(t+1)}$ to reduce the complexity of the problem (2) and another sketch matrix $\boldsymbol{S}_{\mathrm{m}}^{(t+1)}$ to reduce the size of the approximate kernel matrix, where $\boldsymbol{S}_{\mathrm{p}}^{(t+1)}\in\mathbb{R}^{(t+1)\times s_{\mathrm{p}}}$ is an SJLT (defined in Section 2.2) and $\boldsymbol{S}_{\mathrm{m}}^{(t+1)}\in\mathbb{R}^{(t+1)\times s_{\mathrm{m}}}$ is a sub-sampling matrix. Using the two sketch matrices, we first formulate the *incremental randomized sketches* of $\boldsymbol{K}^{(t+1)}$ as follows:

$$\boldsymbol{\Phi}_{\mathrm{pm}}^{(t+1)}=\boldsymbol{S}_{\mathrm{p}}^{(t+1)\intercal}\boldsymbol{C}_{\mathrm{m}}^{(t+1)}\ \text{ and }\ \boldsymbol{\Phi}_{\mathrm{pp}}^{(t+1)}=\boldsymbol{S}_{\mathrm{p}}^{(t+1)\intercal}\boldsymbol{C}_{\mathrm{p}}^{(t+1)},$$

where

$$\boldsymbol{C}_{\mathrm{p}}^{(t+1)} = \boldsymbol{K}^{(t+1)}\boldsymbol{S}_{\mathrm{p}}^{(t+1)} \text{ and } \boldsymbol{C}_{\mathrm{m}}^{(t+1)} = \boldsymbol{K}^{(t+1)}\boldsymbol{S}_{\mathrm{m}}^{(t+1)}.$$

Then $\boldsymbol{K}^{(t+1)}$ can be approximated by

$$\boldsymbol{K}_{\mathrm{sk}}^{(t+1)} = \boldsymbol{C}_{\mathrm{m}}^{(t+1)}\boldsymbol{F}_{\mathrm{sk}}^{(t+1)}\boldsymbol{C}_{\mathrm{m}}^{(t+1)\intercal} \approx \boldsymbol{K}^{(t+1)}, \quad (1)$$

where $\boldsymbol{F}_{\mathrm{sk}}^{(t+1)} \in \mathbb{R}^{s_{\mathrm{m}} \times s_{\mathrm{m}}}$ is obtained by solving the following sketched matrix approximation problem at round $t+1$

$$\begin{aligned}\boldsymbol{F}_{\mathrm{sk}}^{(t+1)} &= \arg\min_{\boldsymbol{F}} \left\| \boldsymbol{S}_{\mathrm{p}}^{(t+1)\intercal} \boldsymbol{E}_{\boldsymbol{F}}^{(t+1)} \boldsymbol{S}_{\mathrm{p}}^{(t+1)} \right\|_{\mathrm{F}}^2 \\ &= \left(\boldsymbol{\Phi}_{\mathrm{pm}}^{(t+1)}\right)^{\dagger} \boldsymbol{\Phi}_{\mathrm{pp}}^{(t+1)} \left(\boldsymbol{\Phi}_{\mathrm{pm}}^{(t+1)\intercal}\right)^{\dagger},\end{aligned} \quad (2)$$

where $\boldsymbol{E}_{\boldsymbol{F}}^{(t+1)} = \boldsymbol{C}_{\mathrm{m}}^{(t+1)}\boldsymbol{F}\boldsymbol{C}_{\mathrm{m}}^{(t+1)\intercal} - \boldsymbol{K}^{(t+1)}$.

Next, we design the incremental maintenance for the incremental randomized sketches. We partition the sketch matrices into block matrices as

$$\boldsymbol{S}_{\mathrm{p}}^{(t+1)} = \left[\boldsymbol{S}_{\mathrm{p}}^{(t)\intercal}, \boldsymbol{s}_{\mathrm{p}}^{(t+1)}\right]^{\intercal}, \quad \boldsymbol{S}_{\mathrm{m}}^{(t+1)} = \left[\boldsymbol{S}_{\mathrm{m}}^{(t)\intercal}, \boldsymbol{s}_{\mathrm{m}}^{(t+1)}\right]^{\intercal},$$

where $\boldsymbol{s}_{\mathrm{m}}^{(t+1)} \in \mathbb{R}^{s_{\mathrm{m}}}$ is a sampling vector, $\boldsymbol{s}_{\mathrm{p}}^{(t+1)} \in \mathbb{R}^{s_{\mathrm{p}}}$ is a vector that contains d nonzero entries determined by d different hash mappings in SJLT. Then we can update the sketch $\boldsymbol{\Phi}_{\mathrm{pm}}^{(t+1)} \in \mathbb{R}^{s_{\mathrm{p}} \times s_{\mathrm{m}}}$ incrementally by rank-1 modifications

$$\boldsymbol{\Phi}_{\mathrm{pm}}^{(t+1)} = \boldsymbol{\Phi}_{\mathrm{pm}}^{(t)} + \boldsymbol{\Delta}_{\mathrm{pm}}^{(t+1)},$$

where $\boldsymbol{\Delta}_{\mathrm{pm}}^{(t+1)} = \boldsymbol{R}_{\mathrm{pm}}^{(t+1)} + \boldsymbol{R}_{\mathrm{mp}}^{(t+1)\intercal} + \boldsymbol{T}_{\mathrm{pm}}^{(t+1)}$ consists of the following three rank-1 matrices

$$\begin{aligned}\boldsymbol{R}_{\mathrm{pm}}^{(t+1)} &= \boldsymbol{s}_{\mathrm{p}}^{(t+1)}\boldsymbol{\psi}^{(t+1)\intercal}\boldsymbol{S}_{\mathrm{m}}^{(t)}, \\ \boldsymbol{R}_{\mathrm{mp}}^{(t+1)} &= \boldsymbol{s}_{\mathrm{m}}^{(t+1)}\boldsymbol{\psi}^{(t+1)\intercal}\boldsymbol{S}_{\mathrm{p}}^{(t)}, \\ \boldsymbol{T}_{\mathrm{pm}}^{(t+1)} &= \xi^{(t+1)}\boldsymbol{s}_{\mathrm{p}}^{(t+1)}\boldsymbol{s}_{\mathrm{m}}^{(t+1)\intercal}.\end{aligned}$$

Similarly, we update the sketch $\boldsymbol{\Phi}_{\mathrm{pp}}^{(t+1)} \in \mathbb{R}^{s_{\mathrm{p}} \times s_{\mathrm{p}}}$ by rank-1 modifications as follows:

$$\boldsymbol{\Phi}_{\mathrm{pp}}^{(t+1)} = \boldsymbol{\Phi}_{\mathrm{pp}}^{(t)} + \boldsymbol{\Delta}_{\mathrm{pp}}^{(t+1)},$$

where $\boldsymbol{\Delta}_{\mathrm{pp}}^{(t+1)} = \boldsymbol{R}_{\mathrm{pp}}^{(t+1)} + \boldsymbol{R}_{\mathrm{pp}}^{(t+1)\intercal} + \boldsymbol{T}_{\mathrm{pp}}^{(t+1)}$ and the modifications are performed using two rank-1 matrices

$$\begin{aligned}\boldsymbol{R}_{\mathrm{pp}}^{(t+1)} &= \boldsymbol{s}_{\mathrm{p}}^{(t+1)}\boldsymbol{\psi}^{(t+1)\intercal}\boldsymbol{S}_{\mathrm{p}}^{(t)}, \\ \boldsymbol{T}_{\mathrm{pp}}^{(t+1)} &= \xi^{(t+1)}\boldsymbol{s}_{\mathrm{p}}^{(t+1)}\boldsymbol{s}_{\mathrm{p}}^{(t+1)\intercal}.\end{aligned}$$

Figure 1 illustrates the kernel matrix approximation using the incremental randomized sketching.

Finally, we construct the time-varying explicit feature mapping using the incremental randomized sketches. We decompose $\boldsymbol{\Phi}_{\mathrm{pp}}^{(t+1)}$ via the rank-k singular value decomposition (SVD) as follows:

$$\boldsymbol{\Phi}_{\mathrm{pp}}^{(t+1)} \approx \boldsymbol{V}^{(t+1)}\boldsymbol{\Sigma}^{(t+1)}\boldsymbol{V}^{(t+1)\intercal}, \quad (3)$$

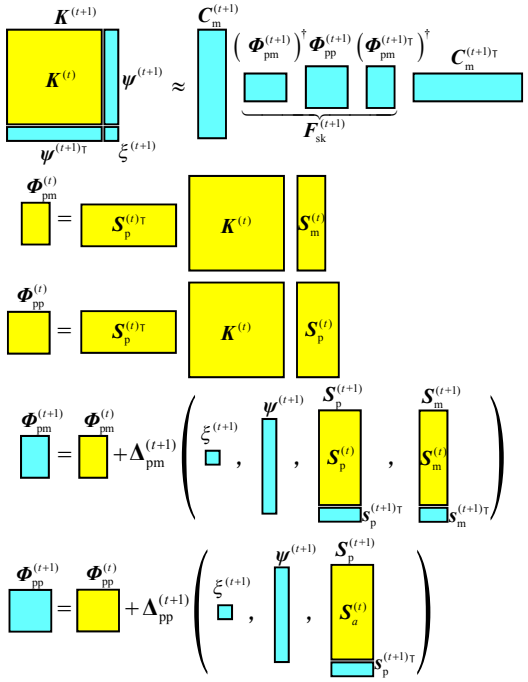


Figure 1. The proposed incremental randomized sketching for kernel matrix approximation at round $t+1$, where $\boldsymbol{K}_{\mathrm{sk}}^{(t+1)} = \boldsymbol{C}_{\mathrm{m}}^{(t+1)}\boldsymbol{F}_{\mathrm{sk}}^{(t+1)}\boldsymbol{C}_{\mathrm{m}}^{(t+1)\intercal}$ is the approximate kernel matrix, $\boldsymbol{\Phi}_{\mathrm{pm}}^{(t+1)}$ and $\boldsymbol{\Phi}_{\mathrm{pp}}^{(t+1)}$ are two incremental randomized sketches for constructing $\boldsymbol{F}_{\mathrm{sk}}^{(t+1)}$, $\boldsymbol{\Delta}_{\mathrm{pp}}^{(t+1)}$ can be seen as a modification function with input variables $\xi^{(t+1)}$, $\boldsymbol{\psi}^{(t+1)}$, $\boldsymbol{S}_{\mathrm{p}}^{(t+1)}$, and $\boldsymbol{\Delta}_{\mathrm{pm}}^{(t+1)}$ uses an additional input variable $\boldsymbol{S}_{\mathrm{m}}^{(t+1)}$. Since the incremental randomized sketches retain the key information with the efficient rank-1 modifications, our incremental randomized sketching can be computed in constant time and space complexities and enjoys a sublinear regret while applied to online kernel learning (see Section 5).

where $\boldsymbol{V}^{(t+1)} \in \mathbb{R}^{s_{\mathrm{p}} \times k}$, $\boldsymbol{\Sigma}^{(t+1)} \in \mathbb{R}^{k \times k}$ and rank $k \leq s_{\mathrm{p}}$.

Then $\boldsymbol{F}_{\mathrm{sk}}^{(t+1)}$ in (2) is approximated by

$$\boldsymbol{F}_{\mathrm{sk}}^{(t+1)} \approx \boldsymbol{Q}_{t+1}\boldsymbol{Q}_{t+1}^{\intercal},$$

where

$$\boldsymbol{Q}_{t+1} = \left(\boldsymbol{\Phi}_{\mathrm{pm}}^{(t+1)}\right)^{\dagger} \boldsymbol{V}^{(t+1)} \left(\boldsymbol{\Sigma}^{(t+1)}\right)^{\frac{1}{2}},$$

which yields the approximate kernel matrix from (1)

$$\boldsymbol{K}_{\mathrm{sk}}^{(t+1)} \approx \left(\boldsymbol{C}_{\mathrm{m}}^{(t+1)}\boldsymbol{Q}_{t+1}\right)\left(\boldsymbol{C}_{\mathrm{m}}^{(t+1)}\boldsymbol{Q}_{t+1}\right)^{\intercal}.$$

Further, the time-varying explicit feature mapping can be updated at round $t+1$ by

$$\boldsymbol{\phi}_{t+2}(\cdot) = \left(\left[\kappa(\cdot, \tilde{\boldsymbol{x}}_1), \ldots, \kappa(\cdot, \tilde{\boldsymbol{x}}_{s_{\mathrm{m}}})\right]\boldsymbol{Q}_{t+1}\right)^{\intercal}, \quad (4)$$

where $\{\tilde{\boldsymbol{x}}_i\}_{i=1}^{s_{\mathrm{m}}}$ are the sampled examples obtained by $\boldsymbol{S}_{\mathrm{m}}^{(t+1)}$, by which the kernel function value between the examples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ can be approximated by $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \approx \langle \boldsymbol{\phi}_{t+2}(\boldsymbol{x}_i), \boldsymbol{\phi}_{t+2}(\boldsymbol{x}_j)\rangle$.

4. Application to Online Kernel Learning

Using the time-varying explicit feature mapping $\phi_t(\cdot)$, we formulate the hypothesis $f_t(\boldsymbol{x}_t)$ for prediction at round t, which is an approximation of the optimal hypothesis on the first $t-1$ instances[1]:

$$f_t(\boldsymbol{x}_t) = \langle \boldsymbol{w}_t, \boldsymbol{\phi}_t(\boldsymbol{x}_t) \rangle \approx \sum_{i \in [t-1]} \alpha_i^* \kappa(\boldsymbol{x}_t, \boldsymbol{x}_i),$$

where $\boldsymbol{w}_t$ is the weight vector and $\{\alpha_i^*\}_{i\in[t-1]}$ is the set of the optimal coefficients of the optimal hypothesis over the first $t-1$ instances. When applying the hypotheses $\{f_t\}_{t\in[T]}$ to online kernel learning, the learning procedure consists of the following two stages. At the first stage, we use a buffer $\mathcal{V}_t$ to store the examples with nonzero losses, restrict the size of the buffer under a fixed budget B ($s_\mathrm{p}, s_\mathrm{m} < B \ll T$), and perform the Kernelized Online Gradient Descent (KOGD) (Kivinen et al., 2001) until the size of $\mathcal{V}_t$ reaches the budget B. Assuming that T_0 rounds have passed while the budget has been reached, let $\boldsymbol{S}_\mathrm{m}^{(T_0)}$ be the uniform sampling matrix, we sample $\{\tilde{\boldsymbol{x}}_i\}_{i=1}^{s_\mathrm{m}}$ using $\boldsymbol{S}_\mathrm{m}^{(T_0)}$ and initialize the sketches $\boldsymbol{\Phi}_\mathrm{pp}^{(T_0)}$, $\boldsymbol{\Phi}_\mathrm{pm}^{(T_0)}$ and $\boldsymbol{Q}_{T_0}$. Then we obtain $\boldsymbol{\phi}_{T_0+1}(\cdot)$ and initialize the hypothesis $\boldsymbol{w}_{T_0+1}$ such that

$$\boldsymbol{w}_{T_0+1}^\intercal \boldsymbol{\phi}_{T_0+1}(\boldsymbol{x}_{T_0}) = \sum_{\boldsymbol{x}_i \in \mathcal{V}_{T_0}} \alpha_i \kappa(\boldsymbol{x}_{T_0}, \boldsymbol{x}_i),$$

which yields

$$\boldsymbol{w}_{T_0+1}^\intercal = \sum_{\boldsymbol{x}_i \in \mathcal{V}_{T_0}} \alpha_i \kappa(\boldsymbol{x}_{T_0}, \boldsymbol{x}_i) \frac{\boldsymbol{\phi}_{T_0+1}(\boldsymbol{x}_{T_0})^\intercal}{\|\boldsymbol{\phi}_{T_0+1}(\boldsymbol{x}_{T_0})\|_2^2},$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_B]^\intercal$ is obtained by KOGD. At the second stage, we update the hypothesis using Online Gradient Descent (OGD) (Zinkevich, 2003). Denote the hypothesis at round t using the updated explicit feature mapping $\boldsymbol{\phi}_{t+1}(\cdot)$ by

$$\bar{f}_t(\boldsymbol{x}_t) = \langle \bar{\boldsymbol{w}}_t, \boldsymbol{\phi}_{t+1}(\boldsymbol{x}_t) \rangle . \tag{5}$$

We can obtain $\bar{\boldsymbol{w}}_t$ by setting $\bar{f}_t(\boldsymbol{x}_t) = f_t(\boldsymbol{x}_t)$, which yields

$$\bar{\boldsymbol{w}}_t^\intercal = f_t(\boldsymbol{x}_t) \boldsymbol{\phi}_{t+1}(\boldsymbol{x}_t)^\dagger = \frac{f_t(\boldsymbol{x}_t) \cdot \boldsymbol{\phi}_{t+1}(\boldsymbol{x}_t)^\intercal}{\|\boldsymbol{\phi}_{t+1}(\boldsymbol{x}_t)\|_2^2}.$$

Then we update the hypothesis $\bar{f}_t(\cdot) = \langle \bar{\boldsymbol{w}}_t, \boldsymbol{\phi}_{t+1}(\cdot) \rangle$ in (5) as follows:

$$\boldsymbol{w}_{t+1} = \bar{\boldsymbol{w}}_t - \eta \nabla \mathcal{L}_t(\bar{\boldsymbol{w}}_t), \tag{6}$$

where $\ell_t(\bar{\boldsymbol{w}}_t) = \ell_t(\bar{f}_t) := \ell(\bar{f}_t(\boldsymbol{x}_t), y_t)$ and $\mathcal{L}_t(\bar{\boldsymbol{w}}_t) := \ell_t(\bar{\boldsymbol{w}}_t) + \frac{\lambda}{2}\|\bar{\boldsymbol{w}}_t\|_2^2$.

[1] It follows from the representer theorem that it is sufficient to approximate the optimal hypothesis of the linear combination form.

To avoid the unbounded sizes of $\boldsymbol{\psi}^{(t+1)}$, $\boldsymbol{S}_\mathrm{p}^{(t+1)}$ and $\boldsymbol{S}_\mathrm{m}^{(t+1)}$, we fix the sampled examples $\{\tilde{\boldsymbol{x}}_i\}_{i=1}^{s_\mathrm{m}}$ in $\boldsymbol{\phi}_t(\cdot)$, $t = T_0+1, \ldots, T$, and adopt a periodic updating strategy for $\boldsymbol{Q}_t$. Specifically, we update $\boldsymbol{Q}_t$ in (4) once for every ρ examples at the second stage, where $\rho \in [T-B]$ is the update cycle. To reduce the time complexity of the matrix decomposition, we adopt the randomized SVD (Halko et al., 2011) for (3). Besides, we also use the randomized SVD for computing the Moore-Penrose pseudoinverse of $\boldsymbol{\Phi}_\mathrm{pm}^{(t+1)}$. Finally, we summarize the above stages into Algorithm 1, called sketched online gradient descent (SkeGD).

Algorithm 1 SkeGD Algorithm

Require: sketch sizes s_p and s_m, stepsize η, rank k, budget B, update cycle ρ, number of blocks d, regularization parameter λ

Initialize $f_1 = 0$ and $\mathcal{V}_1 = \emptyset$
for $t = 1, \ldots, T$ **do**
  Receive new example $\boldsymbol{x}_t$
  Predict $\hat{y}_t = \mathrm{sgn}(f_t(\boldsymbol{x}_t))$
  if $|\mathcal{V}_t| < B$ **then**
    $\mathcal{V}_{t+1} = \mathcal{V}_t \cup \{\boldsymbol{x}_t\}$ whenever the loss is nonzero
    Update f_t by kernelized online gradient descent
  else
    if $|\mathcal{V}_t| = B$ **then**
      Initialize $T_0 = t$
      Initialize $\boldsymbol{\Phi}_\mathrm{pp}^{(T_0)}$, $\boldsymbol{\Phi}_\mathrm{pm}^{(T_0)}$ and $\boldsymbol{Q}_{T_0}$
      Initialize $\boldsymbol{\phi}_{T_0+1}(\cdot)$ and $\boldsymbol{w}_{T_0+1}$
    else
      if $t \mod \rho = 1$ **then**
        Update $\boldsymbol{\Phi}_\mathrm{pp}^{(t)}$ and $\boldsymbol{\Phi}_\mathrm{pm}^{(t)}$ using rank-1 modifications
        Compute $\boldsymbol{Q}_t$ using $\boldsymbol{\Phi}_\mathrm{pp}^{(t)}$ and $\boldsymbol{\Phi}_\mathrm{pm}^{(t)}$
        $\boldsymbol{\phi}_{t+1}(\boldsymbol{x}_t) = ([\kappa(\boldsymbol{x}_t, \tilde{\boldsymbol{x}}_1), \ldots, \kappa(\boldsymbol{x}_t, \tilde{\boldsymbol{x}}_{s_\mathrm{m}})]\boldsymbol{Q}_t)^\intercal$
        $\bar{\boldsymbol{w}}_t^\intercal = f_t(\boldsymbol{x}_t)\boldsymbol{\phi}_{t+1}(\boldsymbol{x}_t)^\intercal / \|\boldsymbol{\phi}_{t+1}(\boldsymbol{x}_t)\|_2^2$
      else
        $\boldsymbol{Q}_t = \boldsymbol{Q}_{t-1}$, $\boldsymbol{\Phi}_\mathrm{pp}^{(t)} = \boldsymbol{\Phi}_\mathrm{pp}^{(t-1)}$, $\boldsymbol{\Phi}_\mathrm{pm}^{(t)} = \boldsymbol{\Phi}_\mathrm{pm}^{(t-1)}$
        $\bar{\boldsymbol{w}}_t = \boldsymbol{w}_t$
      end if
      $\boldsymbol{w}_{t+1} = \bar{\boldsymbol{w}}_t - \eta \nabla \mathcal{L}_t(\bar{\boldsymbol{w}}_t)$
    end if
  end if
end for

5. Theoretical Analysis

In this section, we demonstrate that the proposed incremental randomized sketching satisfies the low-rank approximation property, analyze the regret for online kernel learning and give the complexity analysis for SkeGD. The detailed proofs can be found in the supplementary material. For convenience, we denote $\boldsymbol{S}_\mathrm{p}^{(T)}$, $\boldsymbol{S}_\mathrm{m}^{(T)}$, $\boldsymbol{C}_\mathrm{m}^{(T)}$, $\boldsymbol{F}_\mathrm{sk}^{(T)}$ and $\boldsymbol{K}^{(T)}$ by $\boldsymbol{S}_\mathrm{p}$, $\boldsymbol{S}_\mathrm{m}$, $\boldsymbol{C}_\mathrm{m}$, $\boldsymbol{F}_\mathrm{sk}$ and $\boldsymbol{K}$, respectively.

5.1. Low-Rank Approximation Property

We first present the bias and variance analysis for approximating the matrix product using $\boldsymbol{S}_\mathrm{p}$ in the proposed incremental randomized sketches, which shows that our sketched

kernel matrix approximation problem (2) is an unbiased estimate of the modified Nyström kernel matrix approximation problem.

Lemma 1 (Matrix Product Preserving Property). *Let $\boldsymbol{A} \in \mathbb{R}^{T\times m}$ and $\boldsymbol{S}_\mathrm{p} \in \mathbb{R}^{T\times s_\mathrm{p}}$. Then,*

$$\mathrm{E}\left[\|\boldsymbol{S}_\mathrm{p}^\intercal \boldsymbol{A}\|_\mathrm{F}^2\right] = \|\boldsymbol{A}\|_\mathrm{F}^2, \quad \mathrm{Var}\left[\|\boldsymbol{S}_\mathrm{p}^\intercal \boldsymbol{A}\|_\mathrm{F}^2\right] \leq \frac{2}{s_\mathrm{p}}\|\boldsymbol{A}\|_\mathrm{F}^4.$$

Proof Sketch. We first partition $\boldsymbol{S}_\mathrm{p}$ into d hash-based sketch submatrices, and demonstrate the expectation and variance while approximating the inner product by $\boldsymbol{S}_\mathrm{p}$, which yields the unbiasedness for the matrix product approximation. Then we derive the expectation of $\|\boldsymbol{S}_\mathrm{p}^\intercal \boldsymbol{A}\|_\mathrm{F}^4$ and obtain the variance of $\|\boldsymbol{S}_\mathrm{p}^\intercal \boldsymbol{A}\|_\mathrm{F}^2$. □

By Lemma 1, we can prove that the proposed incremental randomized sketching is nearly as accurate as the modified Nyström for kernel matrix approximation, which is more accurate than the standard Nyström approach.

Theorem 1 (Low-Rank Approximation Property). *Let $\boldsymbol{K} \in \mathbb{R}^{T\times T}$ be a symmetric matrix, $\epsilon_0 \in (0,1)$. $\boldsymbol{F}_\mathrm{sk} \in \mathbb{R}^{s_\mathrm{m}\times s_\mathrm{m}}$, $\boldsymbol{C}_\mathrm{m} \in \mathbb{R}^{T\times s_\mathrm{m}}$ are matrices defined in (1). Set $\tau = s_\mathrm{m}/s_\mathrm{p}$ and $d = \Theta\left(\log^3(s_\mathrm{m})\right)$ for $\boldsymbol{S}_\mathrm{p} \in \mathbb{R}^{T\times s_\mathrm{p}}$. Let $\boldsymbol{U}_\mathrm{m} \in \mathbb{R}^{T\times s_\mathrm{m}}$ be a matrix with orthonormal columns, $\boldsymbol{U}_\mathrm{m}^\perp \in \mathbb{R}^{T\times(T-s_\mathrm{m})}$ be a matrix satisfying $\boldsymbol{U}_\mathrm{m}\boldsymbol{U}_\mathrm{m}^\intercal + \boldsymbol{U}_\mathrm{m}^\perp(\boldsymbol{U}_\mathrm{m}^\perp)^\intercal = \boldsymbol{I}_T$ and $\boldsymbol{U}_\mathrm{m}^\intercal\boldsymbol{U}_\mathrm{m}^\perp = \boldsymbol{O}$, and*

$$s_\mathrm{p} = \Omega\left(s_\mathrm{m}\,\mathrm{polylog}\left(s_\mathrm{m}\delta_0^{-1}\right)/\epsilon_0^2\right),$$

then with probability at least $1-\delta_0$ all singular values of $\boldsymbol{S}_\mathrm{p}^\intercal\boldsymbol{U}_\mathrm{m}$ are $1\pm\epsilon_0$, and with probability at least $1-\delta$

$$\|\boldsymbol{C}_\mathrm{m}\boldsymbol{F}_\mathrm{sk}\boldsymbol{C}_\mathrm{m}^\intercal - \boldsymbol{K}\|_\mathrm{F}^2 \leq (1+\epsilon)\|\boldsymbol{C}_\mathrm{m}\boldsymbol{F}_\mathrm{mod}\boldsymbol{C}_\mathrm{m}^\intercal - \boldsymbol{K}\|_\mathrm{F}^2,$$

where

$$\sqrt{\epsilon} = 2\tau\sqrt{\frac{T}{\delta_1\delta_2}} + \sqrt{\frac{2\tau}{\delta_2}}\left(\epsilon_0^2 + 2\epsilon_0 + 2\right), \tag{7}$$

δ_i is the failure probability of matrix product preserving as

$$\Pr\left\{\frac{\|\boldsymbol{B}_i\boldsymbol{A}_i - \boldsymbol{B}_i\boldsymbol{S}_\mathrm{p}\boldsymbol{S}_\mathrm{p}^\intercal\boldsymbol{A}_i\|_\mathrm{F}^2}{\|\boldsymbol{B}_i\|_\mathrm{F}^2\|\boldsymbol{A}_i\|_\mathrm{F}^2} > \frac{2}{\delta_i s_\mathrm{p}}\right\} \leq \delta_i, \quad i=1,2,$$

$\boldsymbol{A}_1 = \boldsymbol{U}_\mathrm{m}$, $\boldsymbol{B}_1 = \boldsymbol{I}_T$, $\boldsymbol{A}_2 = \boldsymbol{U}_\mathrm{m}^\perp(\boldsymbol{U}_\mathrm{m}^\perp)^\intercal\boldsymbol{K}$, $\boldsymbol{B}_2 = \boldsymbol{U}_\mathrm{m}^\intercal$, and $\delta = \delta_0 + \delta_1 + \delta_2$.

Proof Sketch. Let $\boldsymbol{A} \in \mathbb{R}^{T\times m}$, $\boldsymbol{B} \in \mathbb{R}^{p\times T}$. By Lemma 1 and the Markov's inequality, we derive the error bound for approximating $\boldsymbol{BA}$ by $\boldsymbol{BS}_\mathrm{p}\boldsymbol{S}_\mathrm{p}^\intercal\boldsymbol{A}$, which yields the bound of the kernel matrix approximation error combined with the singular values of $\boldsymbol{S}_\mathrm{p}\boldsymbol{U}_\mathrm{m}$. Then we bound the singular values of $\boldsymbol{S}_\mathrm{p}\boldsymbol{U}_\mathrm{m}$ and obtain the final upper bound of the approximation error. □

Remark 1. *Theorem 1 shows that the proposed incremental randomized sketching achieves a relative-error bound for kernel matrix approximation, for the modified Nyström is a $1+\epsilon'$ relative-error approximation with respect to the best rank-k approximation (Wang & Zhang, 2013). The kernel approximation error of the modified Nyström approach is determined by s_m. Thus, larger s_m leads to a tighter kernel approximation error bound. Besides, from (7) we can obtain $\sqrt{\tau} = \Theta(T^{-\frac{1}{4}}\epsilon^{\frac{1}{4}})$. Clearly, small τ can guarantee the approximation error ϵ. Finally, the choice of s_p depends on τ and s_m, i.e., $s_\mathrm{p} = s_\mathrm{m}/\tau$.*

5.2. Regret Analysis

Let $\boldsymbol{K}_{B,\rho} \in \mathbb{R}^{(B+\lfloor(T-B)/\rho\rfloor)\times(B+\lfloor(T-B)/\rho\rfloor)}$ be the intersection matrix of $\boldsymbol{K}$ which is constructed by $B+\lfloor(T-B)/\rho\rfloor$ examples, $\mu(\boldsymbol{K}_{B,\rho})$ be the coherence of $\boldsymbol{K}_{B,\rho}$ as

$$\mu(\boldsymbol{K}_{B,\rho}) = \frac{B+\lfloor(T-B)/\rho\rfloor}{\mathrm{rank}(\boldsymbol{K}_{B,\rho})}\max_i\|(\boldsymbol{U}_{B,\rho})_{i,:}\|_2^2,$$

where $\boldsymbol{U}_{B,\rho}$ is the singular vector matrix of $\boldsymbol{K}_{B,\rho}$. Obviously, $\mu(\boldsymbol{K}_{B,\rho})$ is independent of T when $\rho = \Theta(T-B)$. We demonstrate the regret bound for online kernel learning using our incremental randomized sketching as follows[2].

Theorem 2 (Regret Bound). *Let $\boldsymbol{K} \in \mathbb{R}^{T\times T}$ be a kernel matrix with $\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j) \leq 1$, k ($k \leq s_\mathrm{p}$) be the rank in the incremental randomized sketches, $\epsilon_0 \in (0,1)$ and δ_i ($i=0,1,2$) be the failure probabilities defined in Theorem 1. Set the update cycle $\rho = \lfloor\theta(T-B)\rfloor$, $\theta \in (0,1)$, $d = \Theta\left(\log^3(s_\mathrm{m})\right)$ and*

$$s_\mathrm{p} = \Omega\left(s_\mathrm{m}\,\mathrm{polylog}(s_\mathrm{m}\delta_0^{-1})/\epsilon_0^2\right),$$
$$s_\mathrm{m} = \Omega(\mu(\boldsymbol{K}_{B,\rho})k\log k),$$

for $\boldsymbol{S}_\mathrm{p} \in \mathbb{R}^{T\times s_\mathrm{p}}$ and $\boldsymbol{S}_\mathrm{m} \in \mathbb{R}^{T\times s_\mathrm{m}}$. Assume $\ell_t(\cdot)$ is a convex loss function that is Lipschitz continuous with Lipschitz constant L, and the eigenvalues of $\boldsymbol{K}$ decay polynomially with decay rate $\beta > 1$. Let $\boldsymbol{w}_t$, $t \in [T]$ be the sequence of hypotheses generated by (6), satisfying $|f_t(\boldsymbol{x}_t)| = |\langle\boldsymbol{w}_t, \boldsymbol{\phi}_t(\boldsymbol{x}_t)\rangle| \leq C_f$, $t\in[T]$. Then, for the optimal hypothesis f^* that minimize $\frac{\lambda}{2}\|f\|_{\mathcal{H}_\kappa}^2 + \frac{1}{T}\sum_{t=1}^T\ell_t(f)$ in the original reproducing kernel Hilbert space $\mathcal{H}_\kappa$, with probability at least $1-\delta$

$$\mathcal{R}_\mathrm{reg}(T,f^*) \leq \frac{\|\boldsymbol{w}_Z^*\|_2^2}{2\theta\eta} + \frac{\eta L^2 T}{2} + \frac{\sqrt{1+\epsilon}}{\lambda}O(\sqrt{B}) + \frac{C_f^2}{2\theta\eta} + \frac{1}{\lambda(\beta-1)}\left(\frac{3}{2} - \frac{B+1/\theta}{T}\right),$$

where $\mathcal{R}_\mathrm{reg}(T,f^*) = \sum_{t=1}^T\left(\mathcal{L}_t(\boldsymbol{w}_t) - \mathcal{L}_t(f^*)\right)$, $\delta = \delta_0 + \delta_1 + \delta_2$, ϵ is defined in (7), $\boldsymbol{w}_t^*$ is the optimal hypothesis

[2] In the theoretical analysis, we assume $T_0 = B$ and omit KOGD used at the first stage that enjoys a $O(\sqrt{B})$ regret bound.

on the incremental randomized sketches over the first t instances and $Z = \arg\max_{i\in[T]} \|\boldsymbol{w}_i^*\|_2$.

Proof Sketch. We first decompose $\mathcal{L}_t(\boldsymbol{w}_t) - \mathcal{L}_t(\boldsymbol{w}^*)$ into two terms as follows:

$$\underbrace{\mathcal{L}_t(\boldsymbol{w}_t) - \mathcal{L}_t(\boldsymbol{w}_t^*)}_{\text{Optimization error}} + \underbrace{\mathcal{L}_t(\boldsymbol{w}_t^*) - \mathcal{L}_t(\boldsymbol{w}^*)}_{\text{Estimation error}},$$

where $\boldsymbol{w}^*$ is the optimal hypothesis on the incremental randomized sketches in hindsight. Then we bound the errors and complete the proof. □

Remark 2. *The assumption of polynomial decay for eigenvalues of the kernel matrix is a general assumption, met for shift invariant kernels, finite rank kernels and convolution kernels (Liu & Liao, 2015; Belkin, 2018), that is $\exists\beta > 1$ such that $\lambda_i(\boldsymbol{K}) = O(i^{-\beta})$. Since the regret bound contains the term $1/\theta$ and the term $-1/\theta$ while SkeGD requires an additional $O(\theta^{-2})$ runtime for updating (see Section 5.3), a suitable update cycle $\rho = \lfloor\theta(T-B)\rfloor$ needs to be tuned, which is analyzed in the experiments in adversarial environments. From Remark 1, smaller τ leads to a tighter kernel matrix approximation error bound, which yields a tighter regret bound. Besides, the lower bounds of the sketch size s_p and s_m are independent of the number of rounds T.*

In Theorem 2, we can obtain a $O(\sqrt{T})$ regularized regret bound by setting $\eta = 1/\sqrt{T}$, and achieve a $O(\sqrt{T})$ unregularized regret bound by setting $\eta, \lambda = 1/\sqrt{T}$. The existing regret bound of first-order online kernel learning with Nyström is $O(\sqrt{T})$ while requiring the budget $B = O(T)$ (Lu et al., 2016), but the proposed SkeGD does not need this requirement. Under the same assumption that the eigenvalues decay polynomially with decay rate $\beta > 1$, the regret bound of existing second-order online kernel learning via adaptive sketching is $O(T^{1/\beta}\log^2 T)$ (Calandriello et al., 2017a), which is tighter than our bound when $\beta \gg 2$. But this second-order approach has a $O(T^{2/\beta})$ time complexity at each round, while our SkeGD has a constant time complexity at each round with respect to T (see Section 5.3).

5.3. Complexity Analysis for SkeGD

The running time of SkeGD is mainly consumed by the matrix decomposition and multiplication while updating the incremental randomized sketches. Table 1 summarizes the computational complexities of SkeGD at each round[3]. At the first stage ($t \le T_0$), SkeGD has constant time and space complexities per round. At the second stage ($t > T_0$), the time complexity of SkeGD at each updating round ($\boldsymbol{Q}_t$ is updated) is $O\left(\nu + s_\mathrm{p}^2 + s_\mathrm{p}s_\mathrm{m}\right)$, and the space complexity of SkeGD is $O\left(s_\mathrm{p}^2 + s_\mathrm{p}s_\mathrm{m} + \nu\right)$, where $\nu = B + \lfloor(T-B)/\rho\rfloor = B + \lfloor\theta^{-1}\rfloor$ when $\rho = \lfloor\theta(T-B)\rfloor$ $(0 < \theta < 1)$. Since $\boldsymbol{Q}_t$ is updated once for every ρ examples, SkeGD has a constant space complexity, and a $O(s_\mathrm{m})$ time complexity at each round except for $\lfloor\theta^{-1}\rfloor$ updating rounds which are in $O\left(B + s_\mathrm{p}^2 + s_\mathrm{p}s_\mathrm{m}\right)$ time complexity. Thus, the overall time complexity of SkeGD at the second stage is

$$O\left(Ts_\mathrm{m} + \left\lfloor\frac{T-B}{\rho}\right\rfloor\left(\nu + s_\mathrm{p}^2 + s_\mathrm{p}s_\mathrm{m}\right)\right).$$

Table 1. The computational complexities of SkeGD at each round, where $\nu = B + \lfloor\theta^{-1}\rfloor$ when $\rho = \lfloor\theta(T-B)\rfloor$ $(0 < \theta < 1)$.

Item	Round	Operation	Time	Space
KOGD	$t < T_0$	–	$O(B)$	$O(B)$
$\boldsymbol{\Phi}_{\mathrm{pp}}^{(T_0)}$	$t = T_0$	Initialization	$O(Bs_\mathrm{p})$	$O(Bs_\mathrm{p})$
$\boldsymbol{\Phi}_{\mathrm{pm}}^{(T_0)}$	$t = T_0$	Initialization	$O(Bs_\mathrm{m})$	$O(Bs_\mathrm{m})$
$\boldsymbol{\Phi}_{\mathrm{pp}}^{(t)}$	$t > T_0$	Updating	$O(\nu)$	$O(s_\mathrm{p}^2 + \nu)$
$\boldsymbol{\Phi}_{\mathrm{pm}}^{(t)}$	$t > T_0$	Updating	$O(s_\mathrm{m})$	$O(s_\mathrm{p}s_\mathrm{m})$
$\boldsymbol{\Phi}_{\mathrm{pp}}^{(t)}$	$t > T_0$	Decomposition	$O(s_\mathrm{p}^2)$	$O(s_\mathrm{p}^2)$
$\boldsymbol{\Phi}_{\mathrm{pm}}^{(t)}$	$t > T_0$	Decomposition	$O(s_\mathrm{p}s_\mathrm{m})$	$O(s_\mathrm{p}s_\mathrm{m})$
$\boldsymbol{Q}_t$	$t > T_0$	Multiplication	$O(s_\mathrm{p}s_\mathrm{m})$	$O(s_\mathrm{p}s_\mathrm{m})$
$\boldsymbol{\phi}_{t+1}(\boldsymbol{x})$	$t > T_0$	Multiplication	$O(s_\mathrm{m})$	$O(s_\mathrm{m})$

6. Experiments

In this section, we evaluate the effectiveness and efficiency of the proposed incremental randomized sketching for kernel approximation and online kernel learning.

6.1. Experimental Setups

All experiments are performed on a machine with 4-core Intel Core i7 3.60 GHz CPU and 16GB memory. We compare our SkeGD with the state-of-the-art online kernel learning algorithms on the well-known classification benchmarks[4]. We merge the training and testing data into a single dataset for each benchmark, which covers the number of instances ranging from $1,000$ to $581,012$. All the experiments are performed over 20 different random permutations of the datasets. The stepsizes η of all the gradient descent based algorithms are tuned in $10^{[-5:+1:0]}$, and the regularization parameters λ are tuned in $10^{[-4:+1:1]}$. Besides, we use the Gaussian kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2/2\sigma^2\right)$, where the set $\sigma \in \{2^{[-5:+0.5:7]}\}$ are adopted as the candidate kernel set. Since kernel alignment (KA) is computationally efficient and independent of learning algorithms (Cortes et al., 2012; Ding et al., 2018), which is suitable for kernel selection of online kernel learning (Zhang & Liao, 2018), we select the optimal kernels using KA for each dataset.

6.2. Kernel Approximation

First, we evaluate the performance of the proposed incremental randomized sketching (IRS) in terms of kernel ap-

[3]Since we set $k, d < s_\mathrm{m} < s_\mathrm{p}$ in the experiments, we omit k and d in the complexity analysis for SkeGD.

[4]https://www.csie.ntu.edu.tw/∼cjlin/libsvm

proximation. The kernel values are approximated using the explicit feature mapping (4) in our IRS. We compare with the incremental Nyström (INys) approach (Lu et al., 2016), which constructs the fixed intersection matrix using the first B examples in the sequence of T instances. For our IRS, we set $s_{\mathrm{p}} = 3B/4$, $s_{\mathrm{m}} = B/2$, $k = 0.2B$ and $d = 4$. Table 2 shows the relative error $\|\widetilde{\boldsymbol{K}}^{(T)} - \boldsymbol{K}\|_{\mathrm{F}}^2 / \|\boldsymbol{K}\|_{\mathrm{F}}^2$ and running time on `german` and `svmguide3` with different budget B, where $\widetilde{\boldsymbol{K}}^{(T)}$ is the approximate kernel matrix after T rounds. We can observe that IRS outperforms the INys in terms of the relative approximation error on both datasets. Besides, our IRS performs a randomized SVD and only needs to sample $s_{\mathrm{m}} = B/2$ examples while updating the incremental randomized sketches, hence is more efficient.

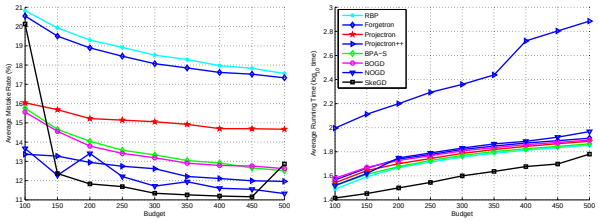
Table 2. Results of kernel approximation using IRS and INys.

Approach	`german` ($B = 100$)		`german` ($B = 200$)	
	Relative error	Time (s)	Relative error	Time (s)
INys	0.078 ± 0.012	0.054	0.047 ± 0.006	0.081
IRS	$\mathbf{0.059 \pm 0.008}$	**0.047**	$\mathbf{0.031 \pm 0.003}$	**0.066**
Approach	`svmguide3` ($B = 100$)		`svmguide3` ($B = 200$)	
	Relative error	Time (s)	Relative error	Time (s)
INys	0.037 ± 0.001	0.083	0.031 ± 0.001	0.092
IRS	$\mathbf{0.018 \pm 0.001}$	**0.065**	$\mathbf{0.012 \pm 0.002}$	**0.077**

6.3. Online Learning under a Fixed Budget

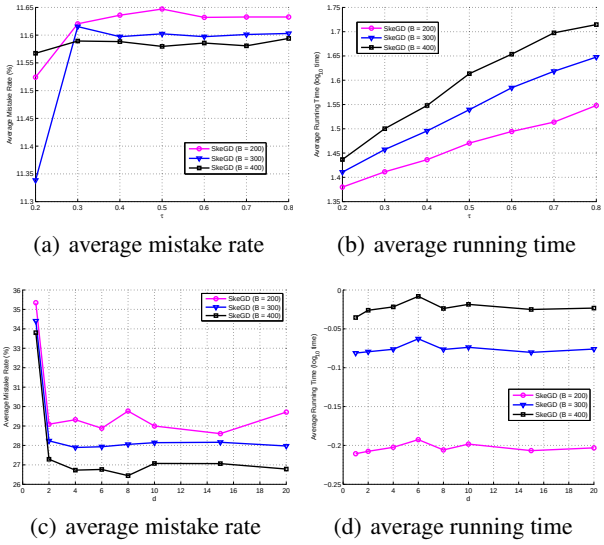
To demonstrate the performance of SkeGD for online kernel learning, we compare SkeGD with the existing sketching and budgeted online kernel learning algorithms under a fixed budget B, including RBP (Cavallanti et al., 2007), Forgetron (Dekel et al., 2005), Projectron and Projectron++ (Orabona et al., 2008), BPA-S (Wang & Vucetic, 2010), BOGD (Zhao et al., 2012), and NOGD (Lu et al., 2016). The compared algorithms are obtained from the LIBOL v0.3.0 toolbox and LSOKL toolbox[5]. For all the algorithms, we adopt the hinge loss function, set $B = 100$ for small datasets ($T \leq 10,000$) and $B = 200$ for other datasets. Besides, we set $\tau = 0.2$, $s_{\mathrm{p}} = 3B/4$, $s_{\mathrm{m}} = \tau s_{\mathrm{p}}$, $d = 4$ and $\rho = 0.3T$ in our SkeGD if not specially specified, and the rank $k = 0.1B$ for NOGD and SkeGD. The mistake rate is used to evaluate the accuracy of online kernel learning, which is computed by $\sum_{t=1}^{T} I(y_t f_t(\boldsymbol{x}_t) < 0)/T \times 100$.

Experimental results from Table 3 show that SkeGD consistently performs better than the other algorithms in terms of the mistake rate of online kernel learning. For most datasets, SkeGD is more efficient than the other algorithms under the same budget B, because the time complexity per round of SkeGD is $O(s_{\mathrm{m}})$ ($s_{\mathrm{m}} < B$) for most rounds as shown in complexity analysis, while the compared algorithms are in $\Omega(B)$ time complexity at each round. Then, we set $s_{\mathrm{p}} = 3B/4$, $s_{\mathrm{m}} = B/2$ and study the influence of the budget B. From Figure 2, we observe that the increasing budget B yields lower mistake rate but leads to higher computation time cost. This is because larger budget B means that more examples are preserved to construct the explicit feature mapping and a higher computation cost is needed to obtain a better approximation quality. For fixed budget ($B = 200, 300, 400$), we set $s_{\mathrm{p}} = 3B/4$, $s_{\mathrm{m}} = \tau s_{\mathrm{p}}$ and vary τ from 0.2 to 0.8. From Figure 3 (a), (b) we can see that SkeGD with smaller τ yields better empirical performance in terms of the average mistake rate and the average running time, which conforms to the theoretical results. For different B, we set $s_{\mathrm{p}} = 3B/4$, $s_{\mathrm{m}} = B/2$, and explore the influence of d that is the number of blocks in the sketch matrix $\boldsymbol{S}_{\mathrm{p}}$. The results in Figure 3 (c), (d) verify that large d can significantly improve the accuracy of online kernel



(a) average mistake rate (b) average running time

Figure 2. The average mistake rates and average running time w.r.t. the budget B on `cod-rna`.



(a) average mistake rate (b) average running time

(c) average mistake rate (d) average running time

Figure 3. (a), (b): average mistake rate, average running time w.r.t. $\tau = s_{\mathrm{m}}/s_{\mathrm{p}}$ on `cod-rna`; (c), (d): average mistake rate and average running time w.r.t. the number of blocks d on `spambase`.

[5] http://libol.stevenhoi.org/
http://lsokl.stevenhoi.org/

Table 3. Comparisons among RBP, Forgetron, Projectron, Projectron++, BPA-S, BOGD, NOGD and our SkeGD w.r.t. the mistake rates (%) and the running time (s).

Algorithm	german		svmguide3		spambase		a9a	
	Mistake rate	Time	Mistake rate	Time	Mistake rate	Time	Mistake rate	Time
RBP	38.924 ± 1.754	0.071	34.091 ± 1.230	**0.080**	35.533 ± 0.769	0.391	28.297 ± 0.115	15.338
Forgetron	38.387 ± 1.345	0.089	35.023 ± 1.161	0.119	36.082 ± 0.692	0.502	28.977 ± 0.380	19.503
Projectron	37.008 ± 1.232	**0.069**	34.172 ± 0.989	0.081	33.627 ± 1.531	0.381	21.376 ± 0.315	15.310
Projectron++	35.396 ± 1.262	0.162	28.902 ± 0.650	0.161	32.424 ± 0.932	0.928	18.702 ± 0.389	23.525
BPA-S	35.345 ± 1.119	0.072	28.948 ± 0.765	0.081	33.359 ± 1.351	0.392	21.840 ± 0.391	15.304
BOGD	33.985 ± 2.013	0.085	30.013 ± 1.109	0.088	33.112 ± 0.692	0.447	29.614 ± 0.244	15.569
NOGD	29.881 ± 0.510	0.075	22.891 ± 0.043	0.085	34.332 ± 1.671	0.396	18.843 ± 0.366	15.416
SkeGD	**27.932 ± 0.481**	0.076	**21.388 ± 0.122**	0.082	**31.301 ± 0.531**	**0.322**	**18.524 ± 0.283**	**11.736**
Algorithm	w7a		ijcnn1		cod-rna		covtype.binary	
	Mistake rate	Time	Mistake rate	Time	Mistake rate	Time	Mistake rate	Time
RBP	5.246 ± 0.034	17.753	16.702 ± 0.120	11.887	19.359 ± 0.114	46.182	41.321 ± 0.024	217.216
Forgetron	5.552 ± 0.045	18.579	17.092 ± 0.203	13.648	18.874 ± 0.048	54.248	41.334 ± 0.076	299.276
Projectron	4.749 ± 0.455	17.639	12.231 ± 0.020	11.682	16.138 ± 0.452	50.473	32.748 ± 0.424	226.050
Projectron++	4.236 ± 1.042	17.101	9.588 ± 0.012	17.930	12.743 ± 0.560	162.823	30.749 ± 0.311	466.637
BPA-S	3.204 ± 0.010	17.937	11.368 ± 0.035	11.610	13.967 ± 0.235	47.952	36.538 ± 0.409	215.262
BOGD	3.482 ± 0.133	17.969	12.064 ± 0.079	11.993	13.818 ± 0.049	53.349	41.858 ± 0.014	223.180
NOGD	2.903 ± 0.009	17.923	9.540 ± 0.003	11.984	12.393 ± 0.583	56.845	34.021 ± 0.617	216.327
SkeGD	**2.782 ± 0.001**	**7.452**	**9.348 ± 0.001**	**8.034**	**11.638 ± 0.050**	**25.918**	**30.493 ± 1.063**	**142.739**

learning with little sacrifice of the time efficiency, where d is equivalent to the number of nonzero entries in each row of $\boldsymbol{S}_{\mathrm{p}}$.

6.4. Online Learning in Adversarial Environments

In this experiment, we compare SkeGD with NOGD, FOGD (Lu et al., 2016) and PROS-N-KONS (Calandriello et al., 2017a) in adversarial environments. We set the budget $B = 100$ for SkeGD and NOGD, and use identical parameters to those used in Section 6.3. Besides, we use the same experimental settings for FOGD (feature dimension $= 4B$) and PROS-N-KONS as provided in the original papers. Inspired by the adversarial settings in (Calandriello et al., 2017a; Wang et al., 2018), we build adversarial datasets using the benchmark `german`. We decompose the online learning game into k_{b} blocks, where each block includes k_{r} rounds. At each round of the same block, we receive the same instance sampled from `german`. Moreover, the revealed labels are flipped at each even block by multiplying -1. Thus, the sample size of the adversarial dataset is $k_{\mathrm{b}} \times k_{\mathrm{r}}$. We set $k_{\mathrm{b}} = 500$, $k_{\mathrm{r}} = 10$ in `german-1`, and $k_{\mathrm{b}} = 500$, $k_{\mathrm{r}} = 20$ in `german-2`.

From Table 4, we can observe that SkeGD is significantly more accurate than other algorithms in adversarial environments, which demonstrates the effectiveness of our incremental randomized sketches. We also notice that SkeGD is much more efficient than the second-order algorithm PROS-N-KONS, and has the time costs comparable to the other first-order algorithms. Additionally, too large or too small values of the update cycle may lead to an increased loss that conforms to Remark 2, and $\rho \approx 0.005T$ performs well in terms of accuracy while having a comparable efficiency.

Table 4. Comparison of online kernel learning algorithms in adversarial environments w.r.t. the mistake rates (%) and the running time (s), where $\rho = \lfloor \theta(T - B) \rfloor$ is the update cycle of SkeGD.

Algorithm	german-1		german-2	
	Mistake rate	Time	Mistake rate	Time
FOGD	37.493 ± 0.724	0.140	32.433 ± 0.196	0.265
NOGD	30.918 ± 0.003	0.405	26.737 ± 0.002	0.778
PROS-N-KONS	27.633 ± 0.416	33.984	17.737 ± 0.900	98.873
SkeGD ($\theta = 0.1$)	17.320 ± 0.136	0.329	7.865 ± 0.059	0.597
SkeGD ($\theta = 0.01$)	17.272 ± 0.112	0.402	7.407 ± 0.086	0.633
SkeGD ($\theta = 0.005$)	**16.578 ± 0.360**	0.484	7.266 ± 0.065	0.672
SkeGD ($\theta = 0.001$)	16.687 ± 0.155	1.183	**6.835 ± 0.136**	1.856

7. Conclusion

In this paper, we have proposed the novel incremental randomized sketching for online kernel learning, which maintains the incremental randomized sketches using efficient rank-1 modifications and constructs a time-varying explicit feature mapping that is suitable for adversarial environments. The proposed sketching approach preserves the kernel matrix approximation accuracy with a $1 + \epsilon$ relative-error, has the lower bounds of the sketch sizes independently of the number of rounds, and enjoys a sublinear regret bound for online kernel learning. Under the same assumptions about kernel eigenvalue decay, in contrast to the second-order online kernel learning via sketching that has per-round computational complexities depending on the number of rounds, our sketching has constant computational complexities at each round, independently of the number of rounds. The theoretical formulation and algorithmic implementation may provide a sketch scheme for both online and offline large-scale kernel learning.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (No. 61673293).

References

Belkin, M. Approximation beats concentration? An approximation view on inference with smooth radial kernels. In *Proceedings of the 31st Conference on Learning Theory*, pp. 1348–1361, 2018.

Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the 19th International Conference on Computational Statistics*, pp. 177–186, 2010.

Calandriello, D., Lazaric, A., and Valko, M. Efficient second-order online kernel learning with adaptive embedding. In *Advances in Neural Information Processing Systems 30*, pp. 6140–6150, 2017a.

Calandriello, D., Lazaric, A., and Valko, M. Second-order kernel online convex optimization with adaptive sketching. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 645–653, 2017b.

Cavallanti, G., Cesa-Bianchi, N., and Gentile, C. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2):143–167, 2007.

Charikar, M., Chen, K., and Farach-Colton, M. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004.

Cortes, C., Mohri, M., and Rostamizadeh, A. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13:795–828, 2012.

Crammer, K., Kandola, J., and Singer, Y. Online classification on a budget. In *Advances in Neural Information Processing Systems 16*, pp. 225–232, 2003.

Dekel, O., Shalev-Shwartz, S., and Singer, Y. The forgetron: A kernel-based perceptron on a fixed budget. In *Advances in Neural Information Processing Systems 18*, pp. 259–266, 2005.

Ding, L., Liao, S., Liu, Y., Yang, P., and Gao, X. Randomized kernel selection with spectra of multilevel circulant matrices. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pp. 2910–2917, 2018.

Engel, Y., Mannor, S., and Meir, R. The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Pocessing*, 52(8):2275–2285, 2004.

Halko, N., Martinsson, P.-G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

Kane, D. M. and Nelson, J. Sparser Johnson-Lindenstrauss transforms. *Journal of the ACM*, 61(1):4:1–23, 2014.

Kivinen, J., Smola, A. J., and Williamson, R. C. Online learning with kernels. In *Advances in Neural Information Processing Systems 14*, pp. 785–792, 2001.

Kumar, S., Mohri, M., and Talwalkar, A. On sampling-based approximate spectral decomposition. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 553–560, 2009.

Liu, Y. and Liao, S. Eigenvalues ratio for kernel selection of kernel methods. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pp. 2814–2820, 2015.

Lu, J., Hoi, S. C., Wang, J., Zhao, P., and Liu, Z. Large scale online kernel learning. *Journal of Machine Learning Research*, 17:1613–1655, 2016.

Orabona, F., Keshet, J., and Caputo, B. The projectron: A bounded kernel-based perceptron. In *Proceedings of the 25th International Conference on Machine learning*, pp. 720–727, 2008.

Raskutti, G. and Mahoney, M. Statistical and algorithmic perspectives on randomized sketching for ordinary least-squares. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 617–625, 2015.

Sarlós, T. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pp. 143–152, 2006.

Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. Pegasos: Primal estimated sub-gradient solver for SVM. *Mathematical Programming*, 127(1):3–30, 2011.

Sun, Y., Schmidhuber, J., and Gomez, F. J. On the size of the online kernel sparsification dictionary. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 329–336, 2012.

Wang, G., Zhao, D., and Zhang, L. Minimizing adaptive regret with one gradient per iteration. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 2762–2768, 2018.

Wang, S. and Zhang, Z. Improving CUR matrix decomposition and the Nyström approximation via adaptive sampling. *Journal of Machine Learning Research*, 14: 2729–2769, 2013.

Wang, S., Zhang, Z., and Zhang, T. Towards more efficient SPSD matrix approximation and CUR matrix decomposition. *Journal of Machine Learning Research*, 17:1–49, 2016.

Wang, Z. and Vucetic, S. Online passive-aggressive algorithms on a budget. In *Proceedings of the 13rd International Conference on Artificial Intelligence and Statistics*, pp. 908–915, 2010.

Wang, Z., Crammer, K., and Vucetic, S. Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training. *Journal of Machine Learning Research*, 13:3103–3131, 2012.

Williams, C. and Seeger, M. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 14*, pp. 682–688, 2001.

Woodruff, D. P. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

Zhang, X. and Liao, S. Online kernel selection via incremental sketched kernel alignment. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 3118–3124, 2018.

Zhao, P., Wang, J., Wu, P., Jin, R., and Hoi, S. C. Fast bounded online gradient descent algorithms for scalable kernel-based online learning. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 169–176, 2012.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pp. 928–936, 2003.