Streaming Kernel Principal Component Analysis

Mina Ghashami
School of Computing
University of Utah
Salt Lake City, UT 84112
ghashami@cs.utah.edu

Daniel J. Perry
SCI Institute
University of Utah
Salt Lake City, UT 84112
dperry@cs.utah.edu

Jeff M. Phillips
School of Computing
University of Utah
Salt Lake City, UT 84112
jeffp@cs.utah.edu

Abstract

Kernel principal component analysis (KPCA) provides a concise set of basis vectors which capture nonlinear structures within large data sets, and is a central tool in data analysis and learning. To allow for nonlinear relations, typically a full $n \times n$ kernel matrix is constructed over n data points, but this requires too much space and time for large values of n. Techniques such as the Nyström method and random feature maps can help towards this goal, but they do not explicitly maintain the basis vectors in a stream and take more space than desired.

We propose a new approach for streaming KPCA which maintains a small set of basis elements in a stream, requiring space only logarithmic in n, and also improves the dependence on the error parameter. Our technique combines together random feature maps with recent advances in matrix sketching, it has guaranteed spectral norm error bounds with respect to the original kernel matrix, and it compares favorably in practice to state-of-the-art approaches.

1 Introduction

Principal component analysis (PCA) is a well-known technique for dimensionality reduction, and has many applications including visualization, pattern recognition, and data compression [11]. Given a set of centered d-dimensional (training) data points $A = [a_1; \ldots; a_n] \in \mathbb{R}^{n \times d}$, PCA diagonalizes the covariance matrix $C = \frac{1}{n} A^T A$ by solving the eigenvalue equation $Cv = \lambda v$. However, when the data points lie on a highly nonlinear space, PCA fails to concisely capture the structure of data. To overcome this, several nonlinear extension of PCA have been proposed, in particular Kernel Principal Component Analysis (KPCA) [24]. The basic idea of KPCA is to implicitly map the data into a nonlinear feature space of high (or often infinite) dimension and perform PCA in that space [24]. The nonlinear map is often denoted as $\phi : \mathbb{R}^d \to \mathcal{H}$ where $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS). While direct computation of PCA in RKHS is infeasible, we can invoke the so called *kernel trick* which exploits the fact that PCA interacts with data through only pair-wise inner products. That is $\langle \phi(x), \phi(y) \rangle_{\mathcal{H}} = K(x, y)$, for all $x, y \in \mathbb{R}^d$ for a kernel function K; we represent this as the $n \times n$ gram matrix G. However, KPCA suffers from high space and computational complexity in storing the entire kernel (gram) matrix $G \in \mathbb{R}^{n \times n}$ and in computing the decomposition of it in the training phase. Then in the testing phase it spends $O(nd)$ time to evaluate the kernel function for any arbitrary test vector with respect to all training examples. Although one can use low rank decomposition approaches [4, 23, 18, 7] to reduce the computational cost to some extent, KPCA still needs to compute and store the kernel matrix.

There have been two main approaches towards resolving this space issue. First approach is the Nyström [26] which uses a sample of the data points to construct a much smaller gram matrix. Second approach is using feature maps [22] which provide an approximate but explicit embedding of the RKHS into Euclidean space. As we describe later, both approaches can be made to operate in a stream, approximating the KPCA result in less than $O(n^2)$ time and space.

Once these approximations are formed, they reveal a $D \ll n$ dimensional space, and typically a k-dimensional subspace found through linear PCA in $\mathbb{R}^D$, which captures most of the data (e.g., a low rank-k approximation). There are two main purposes of these

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 41.

D- and k-dimensional subspaces; they start with mapping a data point $x \in \mathbb{R}^d$ into the D-dimensional space, and then often onto the k-dimensional subspace. If x is one of the training data points, then the k-dimensional representation can be used as a concise "loadings" vector. It can be used in various down-stream training and learning tasks wherein this k-dimensional space, can assume linear relations (e.g., linear separators, clustering under Euclidean distance) since the non-linearity will have already been represented through the mapping to this space. If x is not in the training set, and the training set represents some underlying distribution, then we can assess the "fit" of x to this distribution by considering the residual of its representation in the D-dimensional space when projected to the k-dimensional space.

We let TEST TIME refer to this time for mapping a single point x to the D-dimensional and k-dimensional spaces. The value of k needed to get a good fit depends on the choice of kernel and its fit to the data; but D depends on the technique. For instance in (regular) KPCA $D = n$, in Nyström $D = O(1/\varepsilon^2)$, when using random feature maps with [22] $D = O((1/\varepsilon^2)\log n)$, where $\varepsilon \in (0, 1)$ is the error parameter. We propose a new streaming approach, named as SKPCA, that will only require $D = O(1/\varepsilon)$.

We prove bounds and show empirically that SKPCA greatly outperforms existing techniques in TEST TIME, and is also comparable or better in other measures of SPACE (the cost of storing this map, and space needed to construct it), and TRAIN TIME (the time needed to construct the map to the D-dimensional and k-dimensional spaces).

Background and Notation. We indicate matrix A is $n \times d$ dimensional as $A \in \mathbb{R}^{n\times d}$. Matrices A and Z will be indexed by their row vectors $A = [a_1; a_2; \ldots; a_n]$ while other matrices $V, U, W, \ldots$ will be indexed by column vectors $V = [v_1, v_2, \ldots, v_d]$. We use I_n for the n-dimensional identity matrix and $0^{n\times d}$ as the full zero matrix of dimension $n \times d$. The Frobenius norm of a matrix A is $\|A\|_F = \sqrt{\sum_{i=1} \|a_i\|^2}$ and the spectral norm is $\|A\|_2 = \sup_{x\in\mathbb{R}^d} \frac{\|Ax\|}{\|x\|}$. We denote transpose of a matrix as A^T. The singular value decomposition of matrix $A \in \mathbb{R}^{n\times d}$ is denoted by $[U, S, V] = \mathsf{svd}(A)$. If $n \geq d$ it guarantees that $A = USV^T$, $U^TU = I_n$, $V^TV = I_d$, $U \in \mathbb{R}^{n\times n}$, $V \in \mathbb{R}^{d\times d}$, and $S = \mathsf{diag}(s_1, s_2, \ldots, s_d) \in \mathbb{R}^{n\times d}$ is a diagonal matrix with $s_1 \geq s_2 \geq \ldots \geq s_d \geq 0$. Let U_k and V_k be matrices containing the first k columns of U and V, respectively, and $S_k = \mathsf{diag}(s_1, s_2, \ldots, s_k) \in \mathbb{R}^{k\times k}$. The matrix $A_k = U_kS_kV_k^T$ is the best rank k approximation of A in the sense that $A_k = \arg\min_{C:\mathsf{rank}(C)\leq k} \|A - C\|_{2,F}$. We denote by $\pi_B(A)$ the projection of rows of A on the span of the rows of B. In other words, $\pi_B(A) = AB^\dagger B$ where $(\cdot)^\dagger$ indicates taking the Moore-Penrose psuedoinverse. Finally, expected value of a matrix is defined as the matrix of expected values.

1.1 Related Work

Matrix Sketching. Among many recent advancements in matrix sketching [27, 20], we focus on those that compress a $n \times d$ matrix A into an $\ell \times d$ matrix B. There are several classes of algorithms based on row/column sampling [4, 2] (very related to Nyström approaches [5]), random projection [23] or hashing [3] which require $\ell \approx \mathsf{c}/\varepsilon^2$ to achieve ε error. The constant c depends on the algorithm, specific type of approximation, and whether it is a "for each" or "for all" approximation. A recent and different approach, Frequent Directions (FD) [18], uses only $\ell = 2/\varepsilon$ to achieve the error bound $\|A^TA - B^TB\|_2 \leq \varepsilon\|A\|_F^2$, and runs in time $O(nd/\varepsilon)$. We use a modified version of this algorithm in our proposed approach.

Incremental Kernel PCA. Techniques of this group update/augment the eigenspace of kernel PCA without storing all training data. [13] adapted incremental PCA [9] to maintain a set of linearly independent training data points and compute top d eigenvectors such that they preserve a θ-fraction (for a threshold $\theta \in (0, 1)$) of the total energy of the eigenspace. However this method suffers from two major drawbacks. First, the set of linearly independent data points can grow large and unpredictably, perhaps exceeding the capacity of the memory. Second, under adversarial (or structured sparse) data, intermediate approximations of the eigenspace can compound in error, giving bad performance [6]. Some of these issues can be addressed using online regret analysis assuming incoming data is drawn iid (e.g., [15]). However, in the adversarial settings we consider, FD [18] can be seen as the right way to formally address these issues.

Nyström-Based Methods for Kernel PCA. Another group of methods [26, 5, 8, 14, 25], known as NYSTRÖM, approximate the kernel (gram) matrix G with a low-rank matrix $\hat{G}$, by sampling columns of G. The original version [26] samples c columns with replacement as C and estimates $\hat{G} = CW^{-1}C^T$, where W is the intersection of the sampled columns and rows; this method takes $O(nc^2)$ time and is not streaming. Later [5] used sampling with replacement and approximated G as $\hat{G} = CW_k^\dagger C^T$. They proved if sampling probabilities are of form $p_i = G_{ii}^2/\sum_{i=1}^n G_{ii}^2$, then for $\varepsilon \in (0, 1)$ and $\delta \in (0, 1)$, a Frobenius error bound $\|G - \bar{G}_k\|_F \leq \|G - G_k\|_F + \varepsilon\sum_{i=1}^n G_{ii}^2$ holds with probability $1 - \delta$ for $c = O((k/\varepsilon^4)\log(1/\delta))$, and a spectral error bound $\|G - \bar{G}_k\|_2 \leq \|G - G_k\|_2 + \varepsilon\sum_{i=1}^n G_{ii}^2$ holds with probability $1 - \delta$ for $c = O((1/\varepsilon^2)\log(1/\delta))$

samples. There exist conditional improvements, e.g., [8] shows with $c = O(\mu \frac{k \ln(k/\delta)}{\varepsilon^2})$ where μ denotes the coherence of the top k-dimensional eigenspace of G, that $\|G - \bar{G}_k\|_2 \leq (1 + \frac{n}{(1-\varepsilon)c})\|G - G_k\|_2$.

Random Fourier Features for Kernel PCA. In this line of work, the kernel matrix is approximated via randomized feature maps. The seminal work of [22] showed one can construct randomized feature maps $Z : \mathbb{R}^d \to \mathbb{R}^m$ such that for any shift-invariant kernel $K(x, y) = K(x - y)$ and all $x, y \in \mathbb{R}^d$, $\mathbf{E}[\langle Z(x), Z(y)\rangle] = K(x, y)$ and if $m = O((d/\varepsilon^2)\log(n/\delta))$, then with probability at least $1-\delta$, $|\langle Z(x), Z(y)\rangle - K(x, y)| \leq \varepsilon$. Using this mapping, instead of *implicitly* lifting data points to $\mathcal{H}$ by the kernel trick, they *explicitly* embed the data to a low-dimensional Euclidean inner product space. Subsequent works generalized to other kernel functions such as group invariant kernels [17], min/intersection kernels [21], dot-product kernels [12], and polynomial kernels [10, 1]. This essentially converts kernel PCA to linear PCA. In particular, Lopez *et al.* [19] proposed RANDOMIZED NONLINEAR PCA (RNCA), which is an exact linear PCA on the approximate data feature maps matrix $Z \in \mathbb{R}^{n\times m}$. They showed the approximation error is bounded as $\mathbf{E}[\|\hat{G}-G\|_2] \leq \Theta((n \log n)/m)$, where $\hat{G} = ZZ^T$ is not actually constructed.

1.2 Our Result vs. Previous Streaming

In this paper, we present a streaming algorithm for computing kernel PCA where the kernel is any shift-invariant function $K(x, y) = K(x - y)$. We refer to our algorithm as SKPCA (Streaming Kernel PCA) throughout the paper. Transforming the data to a m-dimensional random Fourier feature space $Z \in \mathbb{R}^{n\times m}$ (for $m \ll n$) and maintaining an approximate ℓ dimensional subspace $W \in \mathbb{R}^{m\times\ell}$ ($\ell \ll m$), we are able to show that for $\tilde{G} = ZWW^TZ^T$, the bound $\|G-\tilde{G}\|_2 \leq \varepsilon n$ holds with high probability. Our algorithm requires $O(dm+m\ell)$ space for storing m feature functions and the ℓ-dimensional eigenspace W. Moreover SKPCA needs $O(dm + ndm + nm\ell) = O(nm(d + \ell))$ time to compute W, and permits $O(dm+m\ell)$ test time to first transfer the data point to m-dimensional feature space and then update the eigenspace W.

We compare with two streaming algorithms.

RNCA[19]: It achieves $\mathbf{E}[\|\hat{G}-G\|_2] \leq \Theta((n \log n)/m)$, for $\hat{G} = ZZ^T$ and $Z \in \mathbb{R}^{n\times m}$ being the data feature map matrix. Using Markov's inequality it is easy to show that with any constant probability $\|\hat{G} - G\|_2 \leq \varepsilon n$ if $m = O((\log n)/\varepsilon)$. We extend this (in Appendix A.1) to show with $m = O((\log n)/\varepsilon^2)$ then $\|\hat{G} - G\|_2 \leq \varepsilon n$ with high probability $1 - 1/n$. This algorithm takes $O(dm+nmd+nm^2) = O(nmd+nm^2)$ time to construct m feature functions, apply them to n data points and compute the $m \times m$ covariance matrix (adding n $m \times m$ outer products). Moreover, it takes $O(dm + m^2)$ space to store the feature functions and covariance matrix. Testing on a new data point $x \in \mathbb{R}^d$ is done by applying the m feature functions on x, and projecting to the rank-k eigenspace in $O(dm + mk)$

NYSTRÖM [5]: It approximates the original Gram matrix with $\bar{G} = CW_k^{\dagger}C^T$. For shift-invariant kernels, the sampling probabilities are $p_i = G_{ii}^2/\sum_{i=1}^n G_{ii}^2 = 1/n$, hence one can construct $W \in \mathbb{R}^{c\times c}$ in a stream using c independent reservoir samplers. Note setting $k = n$ (hence $G_k = G$), their spectral error bound translates to $\|G-\bar{G}\|_2 \leq \varepsilon n$ for $c = O((1/\varepsilon^2)\log(1/\delta))$. Their algorithm requires $O(nc+dc^2 \log n)$ time to do the sampling and construct W. It also needs $O(cd+c^2)$ space for storing the samples and W. The test time step on a point $x \in \mathbb{R}^d$ evaluates $K(x, y)$ on each data point y sampled, taking $O(cd)$ time, and projects onto the c-dimensional and k-dimensional basis in $O(c^2 + ck)$ time; requiring $O(cd + c^2)$ time.

For both RNCA and NYSTRÖM we calculate the eigen-decomposition once at cost $O(m^3)$ or $O(c^3)$, respectively, at TRAIN TIME. Since SKPCA *maintains this decomposition at all steps*, and TESTing may occur at any step, this favors RNCA and NYSTRÖM.

Table 1 summarizes train/test time and space usage of above mentioned algorithms. KPCA is included in the table as a benchmark. All bounds are mentioned for high probability $\delta = 1/n$ guarantee. As a result $c = O((\log n)/\varepsilon^2)$ for NYSTRÖM and $m = O((\log n)/\varepsilon^2)$ for RNCA and SKPCA. One can use Hadamard fast Fourier transforms (Fastfood)[16] in place of Gaussian matrices to gain improvement on train/test time and space usage of SKPCA and RNCA. These matrices allow us to compute random feature maps in time $O(m \log d)$ instead of $O(md)$, and to store feature functions in space $O(m)$ instead of $O(md)$. Since d was relatively small in our examples, we did not observe much empirical benefit of this approach, and we omit it from our further discussions.

We see SKPCA wins on SPACE and TRAIN TIME by factor $(\log n)/\varepsilon$ over RNCA, and similarly on SPACE and TEST TIME by factors $(\log n)/\varepsilon$ and $(\log n)/(\varepsilon^2 k)$ over NYSTRÖM. When d is constant and $\varepsilon < ((\log^2 n)/n)^{1/3}$ it improves TRAIN TIME over NYSTRÖM. It is the first method to use SPACE sublinear (logarithmic) in n and sub-quartic in $1/\varepsilon$, and have TRAIN TIME sub-quartic in $1/\varepsilon$, even without counting the eigen-decomposition cost.

2 Algorithm and Analysis

In this section, we describe our algorithm STREAMING KERNEL PRINCIPAL COMPONENT ANALYSIS (SKPCA) for approximating the eigenspace of a data

Table 1: Asymptotic TRAIN TIME, TEST TIME and SPACE for SKPCA, KPCA [24] , RNCA [19], and NYSTRÖM [5] to achieve $\|G' - G\|_2 \leq \varepsilon n$ with high probability (KCPA is exact) and with Gaussian kernels.

	TRAIN TIME	TEST TIME	SPACE
KPCA	$O(n^2 d + n^3)$	$O(n^2 + nd)$	$O(n^2 + nd)$
NYSTRÖM	$O((n \log n)/\varepsilon^2 + (d \log^3 n)/\varepsilon^4 + (\log^3 n)/\varepsilon^6)$	$O((d \log n)/\varepsilon^2 + (\log^2 n)/\varepsilon^4)$	$O((d \log n)/\varepsilon^2 + (\log^2 n)/\varepsilon^4)$
RNCA	$O((nd \log n)/\varepsilon^2 + (n \log^2 n)/\varepsilon^4 + (\log^3 n)/\varepsilon^6)$	$O((d \log n)/\varepsilon^2 + (k \log n)/\varepsilon^2)$	$O((d \log n)/\varepsilon^2 + (\log^2 n)/\varepsilon^4)$
SKPCA	$O((nd \log n)/\varepsilon^2 + (n \log n)/\varepsilon^3)$	$O((d \log n)/\varepsilon^2 + (k \log n)/\varepsilon^2)$	$O((d \log n)/\varepsilon^2 + (\log n)/\varepsilon^3)$

Algorithm 1 SKPCA

Input: $A \in \mathbb{R}^{n \times d}$ as data points, a shift-invariant kernel function K, and $\ell, m \in \mathbb{Z}^+$
Output: Feature maps $[f_1, \cdots, f_m]$ and their approximate best ℓ-dim subspace W
$[f_1, \cdots, f_m] = \text{FEATUREMAPS}(K, m)$
$B \leftarrow 0^{\ell \times m}$
for $i \in [n]$ **do**
  $z_i = \sqrt{\frac{2}{m}}[f_1(a_i), \cdots, f_m(a_i)]$
  $B \leftarrow z_i$ #insert z_i as a row to B
  if B has no zero valued rows **then**
    $[Y, \Sigma, W] \leftarrow \mathsf{svd}(B)$
    $B \leftarrow \sqrt{\max\{0, \Sigma^2 - \Sigma^2_{\ell/2, \ell/2} I_\ell\}} \cdot W^T$
Return $[f_1, \cdots, f_m]$ and W $\quad \#W \in \mathbb{R}^{m \times \ell}$

set which exists on a nonlinear manifold and is received in streaming fashion one data point at a time. SKPCA consists of two implicit phases. In the first phase, a set of m data oblivious random feature functions $(f_1, \cdots, f_m)$ are computed to map data points to a low dimensional Euclidean inner product space. These feature functions are used to map each data point $a_i \in \mathbb{R}^d$ to $z_i \in \mathbb{R}^m$. In the second phase, each approximate feature vector z_i is fed into a modified FREQUENTDIRECTIONS [18] which is a small space streaming algorithm for computing an approximate set of singular vectors; the matrix $W \in \mathbb{R}^{m \times \ell}$.

However, in the actual algorithm these phases are not separated. The feature mapping functions are precomputed (oblivious to the data), so the approximate feature vectors are immediately fed into the matrix sketching algorithm, so we never need to fully materialize and store the full $n \times m$ matrix Z. Also, perhaps unintuitively, we do not sketch the m-dimensional column-space of Z, rather its n-dimensional row-space. Yet, since the resulting ℓ-dimensional row-space of W (with $\ell \ll m$) encodes a lower dimensional subspace within $\mathbb{R}^m$, it serves to represent our kernel principal components. Pseudocode is provided in Algorithm 1.

Approximate feature maps. To make the algorithm concrete, we consider the approximate feature maps described in the general framework of Rahimi and Recht [22]; label this instantiation of the FEATUREMAPS function as RANDOM-FOURIER FEATUREMAPS (or RFFMAPS). This works for positive definite shift-invariant kernels $K(x, y) = K(x - y)$ (e.g. Gaussian kernel $K(x, y) = (1/2\pi)^{d/2} \exp(-\|x - y\|^2/2)$). It computes a randomized feature map $z : \mathbb{R}^d \to \mathbb{R}^m$ so that $\mathbf{E}[z(x)^T z(y)] = K(x, y)$ for any $x, y \in \mathbb{R}^d$. To construct the mapping z, they define m functions of the form $f_i(x) = \cos(r_i^T x + \gamma_i)$, where $r_i \in \mathbb{R}^d$ is a sample drawn uniformly at random from the Fourier transform of the kernel function, and $\gamma_i \sim \mathsf{Unif}(0, 2\pi]$, uniformly at random from the interval $(0, 2\pi]$. Applying each f_i on a datapoint x, gives the ith coordinate of $z(x)$ in $\mathbb{R}^m$ as $z(x)_i = \sqrt{2/m} f_i(x)$. This implies each coordinate has squared value of $(z(x)_i)^2 \leq 2/m$.

We consider $m = O((1/\varepsilon^2) \log n)$ and $\ell = O(1/\varepsilon)$.

SPACE: We store the m functions f_i, for $i = 1, \ldots, m$; since for each function uses a d-dimensional vector r_i, it takes $O(dm)$ space in total. We compute feature map $z(x)$ and get a m-dimensional row vector $z(a_i)$ for each data point $a_i \in A$, which then is used to update the sketch $B \in \mathbb{R}^{\ell \times m}$ in FREQUENTDIRECTIONS. Since we need an additional $O(\ell m)$ for storing B and W, the total space usage of Algorithm 1 is $O(dm + \ell m) = O((d \log n)/\varepsilon^2 + (\log n)/\varepsilon^3)$.

TRAIN TIME: Applying the feature map takes $O(n \cdot dm)$ time and computing the modified FREQUENTDIRECTIONS sketch takes $O(n\ell m)$ time, so the training time is $O(ndm + n\ell m) = O(n \log n(d/\varepsilon^2 + 1/\varepsilon^3))$.

TEST TIME: For a test point x_{test}, we can lift it to $\mathbb{R}^m$ in $O(dm)$ time using $\{f_1, \ldots, f_m\}$, and then use W to project it to $\mathbb{R}^\ell$ in $O(\ell m)$ time. In total it takes $O(dm + \ell m) = O((d + 1/\varepsilon)/\varepsilon^2 \cdot \log n)$ time.

Although W approximates the eigenspace of A, it will be useful to analyze the error by also considering all of the data points lifted to $\mathbb{R}^m$ as the $n \times m$ matrix Z; and then its projection to $\mathbb{R}^\ell$ as $\tilde{Z} = ZW \in \mathbb{R}^{n \times \ell}$. Note $\tilde{Z}$ would need an additional $O(n\ell)$ to store, and another pass over A (similar for NYSTRÖM and RFF); we *do not* compute and store $\tilde{Z}$, only analyze it.

2.1 Spectral Error Analysis

Let $G = \Phi\Phi^T$ be the exact kernel matrix in RKHS. Let $\hat{G} = ZZ^T$ be an approximate kernel matrix using $Z \in \mathbb{R}^{n \times m}$; it consists of mapping the n points to $\mathbb{R}^m$ using m RFFMAPS. Then we consider $\tilde{G} = ZWW^TZ^T$,

as the kernel matrix which could be constructed from output W of Algorithm 1 using RFFMaps. We ultimately will show that $\|G-\tilde{G}\|_2 \leq \varepsilon n$. The main technical challenge is that FD bounds are typically for the covariance matrix W^TW not the gram matrix WW^T, and thus new ideas are required.

We show in Lemma A.1 in Appendix A.1 that with $m = O((1/\varepsilon^2)\log(n/\delta))$ then $\|G - \hat{G}\|_2 \leq \varepsilon n$ with probability at least $1-\delta$. Note this is a "for all" result which (see Lemma 2.3) is equivalent to, for *all* unit vectors x that $|\|\Phi^T x\|^2 - \|Z^T x\|^2| \leq \varepsilon n$. If we loosen this to a "for each" result, where the above inequality holds for any one unit vector x (say we only want to test with a single vector x_{test}), then we show in Appendix A.2 that this holds with $m = O((1/\varepsilon^2)\log(1/\delta))$. This makes partial progress towards an open question [1] (can m be independent of n?) about creating oblivious subspace embeddings for Gaussian kernel features.

Next, we show that applying the modified Frequent Directions step to Z does not asymptotically increase the error. To do so, we first show that spectrum of Z along directions that FrequentDirections fails to capture is small. We prove this for any $n \times m$ matrix A that is approximated as $B \in \mathbb{R}^{\ell \times m}$ by FrequentDirections.

Lemma 2.1. *Consider an $A \in \mathbb{R}^{n\times m}$ matrix with $m \leq n$, and let B be an $\ell \times m$ matrix resulting from running Frequent Directions on A with ℓ rows. For any unit vector $y \in \mathbb{R}^n$ with $\|y^TAB^\dagger B\| = 0$, it holds that $\|y^TA\|^2 \leq \|A - A_k\|_F^2/(\ell - k)$, for all $k \leq \ell$, including $k = 0$ where $A - A_k = A$.*

Proof. Let $[U, S, V] = \mathsf{svd}(A)$ be the svd of A. Consider any unit vector $y \in \mathbb{R}^n$ that lies in the column space of A and the null space of B, that is $\|y^TAB^\dagger B\| = 0$ and $\|y^TAA^\dagger\| = \|y\| = 1$. Since $U = [u_1, u_2, \ldots, u_n]$ provides an orthonormal basis for $\mathbb{R}^n$, we can write

$$y = \sum_{i=1}^{n} \alpha_i u_i \quad \text{such that} \quad \alpha_i = \langle y, u_i\rangle, \quad \sum_{i=1}^{n} \alpha_i^2 = 1$$

Since $1 = \sum_{i=1}^n \alpha_i^2 = \|y\| = \|y^TAA^\dagger\| = \|y^TU_mU_m^T\| = \sum_{i=1}^m \alpha_i^2$, therefore $\alpha_i = 0$ for $i > m$. Moreover $\|y^TA\|^2 = \sum_{i=1}^m s_i^2\langle y, u_i\rangle^2 = \sum_{i=1}^m s_i^2\alpha_i^2$. This implies there exists a *unit* vector $x = \sum_{i=1}^m \alpha_i v_i \in \mathbb{R}^m$ with $\alpha_i = \langle x, v_i\rangle = \langle y, u_i\rangle$ for $i = 1, \cdots, m$ such that $\|y^TA\| = \|Ax\|$ and importantly $\|Bx\| = 0$, which we will prove shortly.

Then, due to the Frequent Directions bound [7], for any unit vector $\bar{x} \in \mathbb{R}^m$, $\|A\bar{x}\|^2 - \|B\bar{x}\|^2 \leq \|A - A_k\|_F^2/(\ell - k)$, and for our particular choice of x with $\|Bx\| = 0$, we obtain $\|y^TA\| = \|Ax\|^2 \leq \|A - A_k\|_F^2/(\ell - k)$, as desired.

Now to see that $\|Bx\| = 0$, we will assume that $\|Bx\| > 0$ and prove a contradiction. Since $\|Bx\| > 0$, then x is not in the null space of B, and $\|\pi_B(x)\| > 0$ for any unit vector x. Let $\Sigma = \mathsf{diag}(\sigma_1, \ldots, \sigma_\ell)$, assuming $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_\ell > 0$, are the singular values of B, and $W = [w_1, \ldots, w_\ell] \in \mathbb{R}^{mx\ell}$ are its right singular vectors. Then $\|Bx\| = \|\Sigma Wx\|$ and if $\|\pi_B(x)\| > 0$, then setting $\bar{\Sigma} = \mathsf{diag}(1, 1, \ldots, 1)$ and $\bar{B} = \bar{\Sigma}W = W_\ell$ to remove the scaling from B, we have $\|\pi_{\bar{B}}(x)\| > 0$. Similarly, if $\|y^TUSV^TB^\dagger B\| = \|y^T\pi_B(A)\| = 0$, then setting $\bar{S} = \mathsf{diag}(1, \ldots, 1)$ and $\bar{A} = U\bar{S}V^T$ to remove scale from A, we have $\|y^T\pi_B(\bar{A})\| = 0$. Hence

$$\begin{aligned} 0 < \|\pi_{\bar{B}}(x)\| &= \|xB^\dagger B\| = \|xW_\ell W_\ell^T\| \\ &= \|\sum_{j=1}^{\ell}\langle x, w_j\rangle\| = \|\sum_{j=1}^{\ell}\sum_{i=1}^{m}\alpha_i\langle v_i, w_j\rangle\| \end{aligned}$$

and

$$\begin{aligned} 0 = \|y^T\pi_B(\bar{A})\| &= \|\sum_{i=1}^{m}\langle y, u_i\rangle v_i^TW_\ell W_\ell^T\| \\ &= \|\sum_{i=1}^{m}\alpha_i v_i^TW_\ell\| = \|\sum_{j=1}^{\ell}\sum_{i=1}^{m}\alpha_i\langle v_i, w_j\rangle\|. \end{aligned}$$

Since last terms of each line match, we have a contradiction, and hence $\|Bx\| = 0$. □

Lemma 2.2. *Let $\tilde{Z} = ZW$, and $\tilde{G} = \tilde{Z}\tilde{Z}^T = ZWW^TZ^T$ be the corresponding gram matrix from $Z \in \mathbb{R}^{n\times m}$ and $W \in \mathbb{R}^{m\times\ell}$ constructed via Algorithm 1 with $\ell = 2/\varepsilon$. Comparing to $\hat{G} = ZZ^T$, then $\|\tilde{G} - \hat{G}\|_2 \leq \varepsilon n$.*

Proof. Consider any unit vector $y \in \mathbb{R}^n$, and note that $y^TZ = [y^TZ]_W + [y^TZ]_{\perp W}$ where $[y^TZ]_W = y^TZWW^T$ lies on the column space spanned by W, and $[y^TZ]_{\perp W} = y^TZ(I - WW^T)$ is in the null space of W. Then first off $\|y^TZ\|^2 = \|[y^TZ]_W\|^2 + \|[y^TZ]_{\perp W}\|^2$ since two components are perpendicular to each other. Second $[y^TZ]_WW = y^TZWW^TW = y^TZW$ and $[y^TZ]_{\perp W}W = y^TZ(I - WW^T)W = y^TZ(W - W) = 0$. Knowing these two we can say

$$\begin{aligned} &y^T(ZZ^T - \tilde{Z}\tilde{Z}^T)y \\ &= (y^TZ)(y^TZ)^T - (y^TZ)WW^T(y^TZ)^T \\ &= \|y^TZ\|^2 - \left([y^TZ]_W + [y^TZ]_{\perp W}\right)WW^T([y^TZ]_W \\ &\quad + [y^TZ]_{\perp W})^T \\ &= \|y^TZ\|^2 - (y^TZW)(y^TZW)^T \\ &= \left(\|[y^TZ]_W\|^2 + \|[y^TZ]_{\perp W}\|^2\right) - \|y^TZW\|^2 \\ &= \|[y^TZ]_{\perp W}\|^2. \end{aligned}$$

The last inequality holds because $\|y^TZW\| = \|y^TZWW^T\| = \|[y^TZ]_W\|$ as W is an orthonormal matrix.

To show $\|[y^T Z]_{\perp W}\| \leq \varepsilon \|Z\|_F^2$, consider vector $v = y^T Z(I - WW^T)Z^\dagger$ and let $y^* = v/\|v\|$. Clearly y^* satisfies requirement of Lemma 2.1 as it is a unit vector in $\mathbb{R}^n$ and $\|y^* ZWW^T\| = 0$ as

$$\begin{aligned}\|y^* ZWW^T\| &= \|y^T Z(I - WW^T)Z^\dagger ZWW^T\|/\|v\| \\ &= \|y^T Z(I - WW^T)WW^T\|/\|v\| = 0.\end{aligned}$$

Therefore it satisfies $\|y^* Z\| \leq \|Z - Z_k\|_F^2/(\ell - k)$. Since $\|Z\|_F^2 \leq 2n$ for $k = 0$ and $\ell = 2/\varepsilon$, we obtain

$$\begin{aligned}\|[y^T Z]_{\perp W}\|^2 &= \|y^T Z(I - WW^T)\| \\ &= \|y^T Z(I - WW^T)Z^\dagger Z\|^2 = \|y^* Z\|^2 \|v\|^2 \\ &\leq \|Z\|_F^2 \|v\|^2/\ell \leq \varepsilon n \|v\|^2.\end{aligned}$$

It is left to show that $\|v\| \leq 1$. For that, note $\pi_{ZW}\left(\pi_Z(y^T)\right) = \pi_{ZW}(y^T ZZ^\dagger) = y^T ZZ^\dagger (ZW)(ZW)^\dagger = y^T ZW(ZW)^\dagger = \pi_{ZW}(y^T)$. The finally we obtain

$$\begin{aligned}\|v\|^2 &= \|y^T Z(I - WW^T)Z^\dagger\|^2 \\ &= \|y^T ZZ^\dagger - y^T ZWW^T Z^\dagger\|^2 \\ &= \|\pi_Z(y^T) - \pi_{ZW}(y^T)\|^2 \\ &= \|\pi_Z(y^T) - \pi_{ZW}(\pi_Z(y^T))\|^2 \\ &\leq \|\pi_Z(y^T)\|^2 \leq \|y\|^2 = 1. \qquad \square\end{aligned}$$

Combining Lemmas A.1 and 2.2 and using triangle inequality, we get our main result.

Theorem 2.1. *Let $G = \Phi\Phi^T$ be the exact kernel matrix over n points. Let $\tilde{G} = ZW^T WZ^T$ be the result of Z from $m = O((1/\varepsilon^2)\log(n/\delta))$ RFFMaps and W from running Algorithm 1 with $\ell = 4/\varepsilon$. Then with probability at least $1 - \delta$, we have $\|G - \tilde{G}\|_2 \leq \varepsilon n$.*

2.2 Frobenius Error Analysis

Let the true gram matrix be $G = \Phi\Phi^T$, and consider $G' = YY^T$, for any Y including when $Y = ZW$. First we write the bound in terms of Φ and Y.

Lemma 2.3. $\|G - G'\|_2 = \max_{\|x\|=1} |\|\Phi^T x\|^2 - \|Y^T x\|^2|$.

Proof. Recall we can rewrite spectral norm as $\|G - G'\|_2 = \max_{\|x\|=1} |x^T Gx - x^T G'x| = \max_{\|x\|=1} |x^T \Phi\Phi^T x - x^T YY^T x| = \max_{\|x\|=1} |\|\Phi^T x\|^2 - \|Y^T x\|^2|$. First line follows by definition of top eigenvalue of a symmetric matrix, and last line is true because $\|y\|^2 = y^T y$ for any vector y. $\square$

Thus if $\|G - G'\|_2 \leq \varepsilon n$ where $G' = YY^T$ could be reconstructed by any of the algorithms we consider, then it implies $\max_{\|x\|=1} |\|\Phi^T x\|^2 - \|Y^T x\|^2| \leq \varepsilon n$. We can now generalize the spectral norm bound to Frobenius norm. Let $G - G' = U\Lambda U^T$ be the eigen decomposition of $G - G'$. Recall that one can write each eigenvalue as $\Lambda_{i,i} = u_i^T (G - G')u_i$, and the definition of the Frobenius norm implies $\|G - G'\|_F^2 = \sum_{i=1}^n \Lambda_{i,i}^2$ Hence

$$\begin{aligned}\|G - G'\|_F^2 &= \sum_{i=1}^n (u_i^T (G - G')u_i)^2 \\ &= \sum_{i=1}^n (\|\Phi^T u_i\|^2 - \|Y^T u_i\|^2)^2 \leq \sum_{i=1}^n (\varepsilon n)^2 \leq \varepsilon^2 n^3.\end{aligned}$$

Therefore $\|G - G'\|_F \leq \varepsilon n^{1.5}$. We can also show a more interesting bound by considering G_k and G'_k, the best rank k approximations of G and G' respectively.

Lemma 2.4. *Given that $\|G - G'\|_2 \leq \varepsilon n$ we can bound $\|G - G'_k\|_F \leq \|G - G_k\|_F + \varepsilon\sqrt{k}n$.*

Proof. Let $[u_1, \ldots, u_n]$ and $[v_1, \ldots, v_n]$ be eigenvectors of G and $G - G'$, respectively. Then

$$\begin{aligned}\|G - G'_k\|_F^2 &= \sum_{i=1}^k (v_i^T (G - G'_k)v_i)^2 \\ &+ \sum_{i=k+1}^n (v_i^T (G - G'_k)v_i)^2 \\ &\leq \sum_{i=1}^k (v_i^T (G - G'_k)v_i)^2 + \sum_{i=k+1}^n (v_i^T Gv_i)^2 \\ &\leq \sum_{i=1}^k (v_i^T (\Phi\Phi^T - YY^T)v_i)^2 + \sum_{i=k+1}^n (u_i^T Gu_i)^2 \\ &= \sum_{i=1}^k (\|\Phi^T v_i\|^2 - \|Y^T v_i\|^2)^2 + \|G - G_k\|_F^2 \\ &\leq k(\varepsilon n)^2 + \|G - G_k\|_F^2.\end{aligned}$$

The second transition is true because G' is positive semidefinite, therefore $v_i^T (G - G'_k)v_i \leq v_i^T Gv_i$, and third transition holds because if u_i is ith eigenvector of G then, $u_i^T Gu_i \geq v_i^T Gv_i$ where v_i is ith eigenvector of $G - G'$. Taking square root yields

$$\|G - G'_k\|_F^2 \leq \|G - G'_k\|_F + \varepsilon n\sqrt{k}. \qquad \square$$

Thus we can get error bounds for the best rank-k approximation of the data in RKHS that depends on "tail" $\|G - G_k\|_F$ which is typically small. We can also make the second term $\varepsilon n\sqrt{k}$ equal to $\varepsilon' n$ by using a value of $\varepsilon = \varepsilon'/\sqrt{k}$ in the previously described algorithms.

3 Experiments

We measure the Space, Train time, and Test time of our SKPCA algorithms with ℓ taking values $\{2, 5, 10, 20, 30, 50\}$. We use spectral and Forbenious

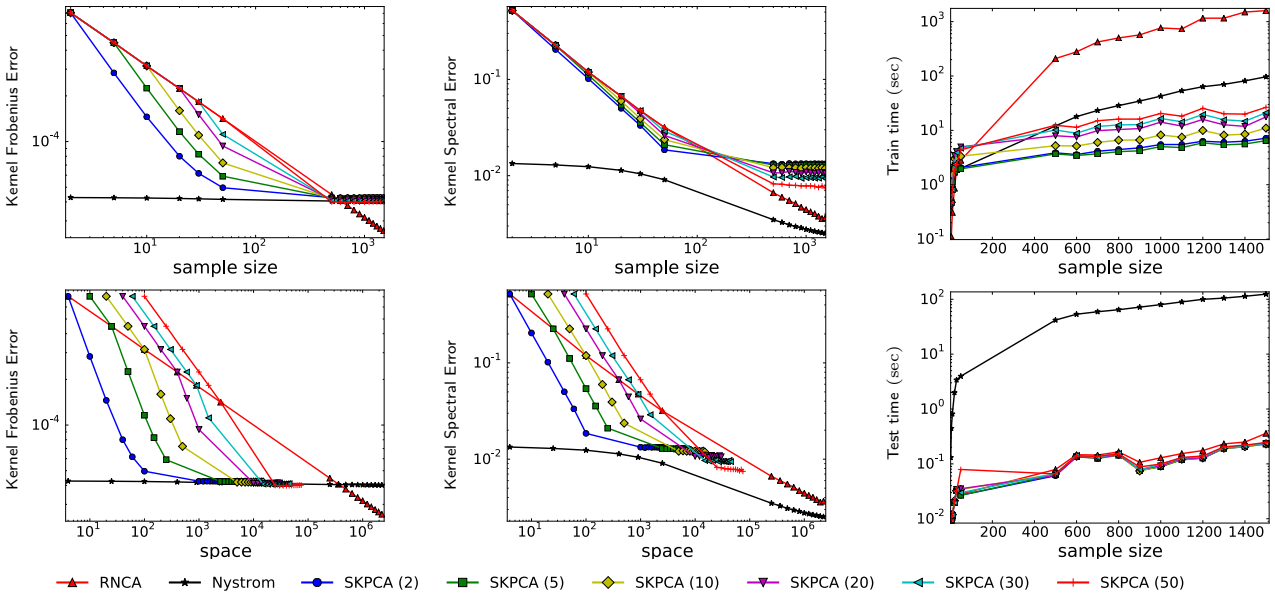

Figure 1: RANDOMNOISY dataset showing Error and Test and Train time versus Sample Size (m or c) and Space.

error measures and compare against the NYSTRÖM and the RNCA approach using RFFMAPS. All methods are implemented in Julia, run on an OpenSUSE 13.1 machine with 80 Intel(R) Xeon(R) 2.40GHz CPU and 750 GB of RAM.

Data sets. We run experiments on several real (CPU, ADULT, FOREST) and synthetic (RANDOMNOISY) data sets. Each data set is an $n \times d$ matrix A (CPU is 7373×21, FOREST is 523910×54, ADULT is 33561×123, and RANDOMNOISY is 20000×1000) with n data points and d attributes. For each a random subset is removed as the test set of size 1000, except CPU where the test set size is 800. We generate the RANDOMNOISY synthetic dataset using the approach by [18]. We create $A = SDU + F/\zeta$, where SDU is an s-dimensional signal matrix (for $s < d$) and F/ζ is (full) d-dimensional noise with ζ controlling the signal to noise ratio. Each entry $F_{i,j}$ of F is generated i.i.d. from a normal distribution $N(0,1)$, and we use $\zeta = 10$. For the signal matrix, $S \in \mathbb{R}^{n \times s}$ again we generate each $S_{i,j} \sim N(0,1)$ i.i.d; D is diagonal with entries $D_{i,i} = 1 - (i-1)/d$ linearly decreasing; and $U \in \mathbb{R}^{s \times d}$ is just a random rotation. We set $s = 50$.

Error measures. We consider two error measures comparing the true gram matrix G and an approximated gram matrix (constructed in various ways). Kernel Spectral Error $= \|G - G'\|_2/n$ represents the worst case error. Kernel Frobenius Error $= \|G - G'\|_F/n^2$ represents the global error. We normalized the error measures by $1/n$ and $1/n^2$, respectively, so they are comparable across data sets. These measures require another pass on the data to compute, but give a more holistic view of how accurate our approaches are.

We measure the SPACE requirements of each algorithm as follows. SKPCA sketch has space $md + m\ell$, NYSTRÖM is $c^2 + cd$, and RNCA is $m^2 + md$, where m is the number of RFFMAPS, and c is the number of samples in NYSTRÖM. In our experiments, we set m and c similarly, calling these parameters SAMPLE SIZE. Note that SAMPLE SIZE and SPACE usage are different: both RNCA and Nyström have SPACE quadratic in SAMPLE SIZE, while for SKPCA it is linear.

Results. Figures 1, 2, and 3 show log-log plots of results for RANDOM NOISY, CPU, and ADULT datasets. See also Figure 4 for FOREST in the Appendix.

For small SAMPLE SIZE we observe that NYSTRÖM performs quite well under all error measures, corroborating results reported by Lopez *et al.* [19]. However, all methods have a *very small error range*, typically less than 0.01. For Kernel Frobenius Error we typically observe a cross-over point where RNCA and often most versions of SKPCA have better error for that size. Under Kernel Spectral Error we often see a cross-over point for SKPCA, but not for RNCA. We suspect that this is related to how FD only maintains the most dominate directions while ignoring other (potentially spurious) directions introduced by the RFFMAPS.

In general, SKPCA has as good or better error than RNCA for the same size, with smaller size being required with smaller ℓ values. This difference is more pronounced in SPACE than SAMPLE SIZE, where our theoretical results expect a polynomial advantage.

In timing experiments, especially TRAIN TIME we see SKPCA has a very large advantage. As a function of SAMPLE SIZE RNCA is slowest for the TRAIN TIME, and NYSTRÖM is slowest for TEST TIME by several

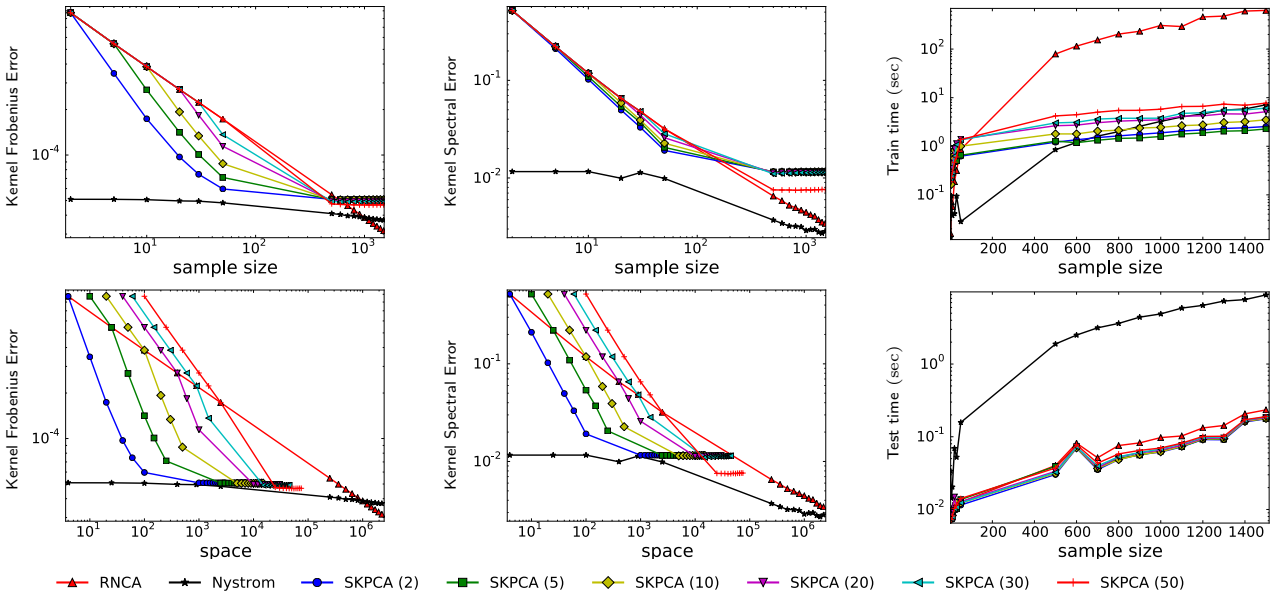

Figure 2: CPU dataset showing Error and Test and Train time versus Sample Size (m or c) and Space.

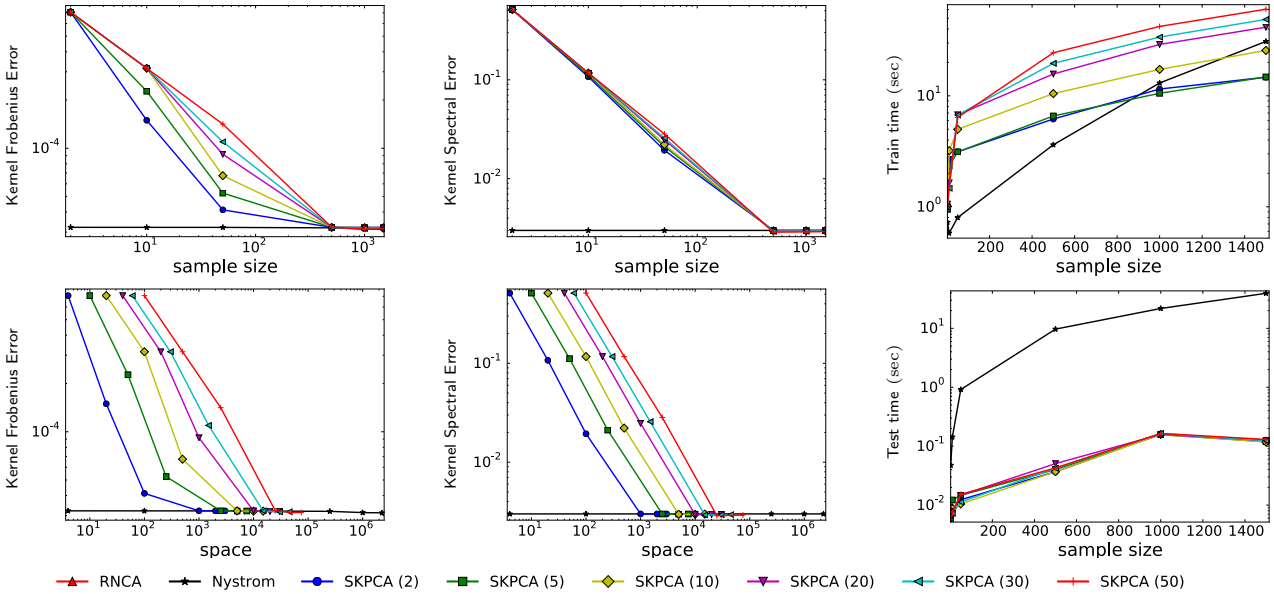

Figure 3: ADULT dataset showing Error and Test and Train time versus Sample Size (m or c) and Space.

orders of magnitude. In both cases all versions of SKPCA are among the fastest algorithms. For the TRAIN TIME results, RNCA's slow time is dominated by summing n outer products, of dimensions $m \times m$. This is avoided in SKPCA by only keeping the top ℓ dimensions, and only requiring similar computation on the order of $\ell \times m$, where typically $\ell \ll m$. NYSTRÖM only updates the $c \times c$ gram matrix when a new point replaces an old one, expected $c \log n$ times.

NYSTRÖM is comparatively very slow in TEST TIME. It computes a new row and column of the gram matrix, and projects onto this space, taking $O(cd + c^2)$ time. Both RNCA and SKPCA avoid this by directly computing an m dimensional representation of a test data point in $O(dm)$ time. Recall we precompute the eigenstructure for RNCA and NYSTRÖM, whereas SKPCA maintains it at all times, so if this step were counted, SKPCA's advantage here would be even larger.

Summary. Our proposed method SKPCA has superior timing and error results to RNCA, by sketching in the kernel feature space. Its error is typically a bit worse than a NYSTRÖM approach, but the difference is quite small, and SKPCA is far superior to NYSTRÖM in TEST TIME, needed for any data analysis.

Acknowledgements. Thanks to support from NSF CCF-1350888, IIS-1251019, ACI-1443046, and CNS-1514520.

References

[1] Haim Avron, Huy L. Nguyen, and David P. Woodruff. Subspace embeddings for the polynomial kernel. In *NIPS*, 2014.

[2] Christos Boutsidis, Michael W. Mahoney, and Petros Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of 20th ACM-SIAM Symposium on Discrete Algorithms*, 2009.

[3] Kenneth L Clarkson and David P Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th Annual ACM symposium on Theory of computing*, 2013.

[4] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36(1):158–183, 2006.

[5] Petros Drineas and Michael W Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research*, 6:2153–2175, 2005.

[6] Mina Ghashami, Amey Desai, and Jeff M. Phillips. Improved practical matrix sketching with guarantees. In *Proceedings 22nd Annual European Symposium on Algorithms*, 2014.

[7] Mina Ghashami and Jeff M. Phillips. Relative errors for deterministic low-rank matrix approximations. In *SODA*, pages 707–717, 2014.

[8] Alex Gittens and Michael W Mahoney. Revisiting the nystrom method for improved large-scale machine learning. *arXiv preprint arXiv:1303.1849*, 2013.

[9] Peter M Hall, A David Marshall, and Ralph R Martin. Incremental eigenanalysis for classification. In *BMVC*, volume 98, pages 286–295, 1998.

[10] Raffay Hamid, Ying Xiao, Alex Gittens, and Dennis DeCoste. Compact random feature maps. *arXiv preprint arXiv:1312.4626*, 2013.

[11] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

[12] Purushottam Kar and Harish Karnick. Random feature maps for dot product kernels. *arXiv preprint arXiv:1201.6530*, 2012.

[13] Shosuke Kimura, Seiichi Ozawa, and Shigeo Abe. Incremental kernel pca for online learning of feature space. In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, volume 1, pages 595–600. IEEE, 2005.

[14] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nyström method. *The Journal of Machine Learning Research*, 13(1):981–1006, 2012.

[15] Dima Kuzmin and Manfred K Warmuth. Online kernel pca with entropic matrix updates. In *Proceedings of the 24th international conference on Machine learning*, pages 465–472. ACM, 2007.

[16] Quoc Le, Tamás Sarlós, and Alex Smola. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, 2013.

[17] Fuxin Li, Catalin Ionescu, and Cristian Sminchisescu. Random fourier approximations for skewed multiplicative histogram kernels. In *Pattern Recognition*, pages 262–271. Springer, 2010.

[18] Edo Liberty. Simple and deterministic matrix sketching. In *KDD*, pages 581–588, 2013.

[19] David Lopez-Paz, Suvrit Sra, Alex Smola, Zoubin Ghahramani, and Bernhard Schölkopf. Randomized nonlinear component analysis. *arXiv preprint arXiv:1402.0119*, 2014.

[20] Michael W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3, 2011.

[21] Subhransu Maji and Alexander C Berg. Max-margin additive classifiers for detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 40–47. IEEE, 2009.

[22] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2007.

[23] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *FOCS*, pages 143–152, 2006.

[24] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Artificial Neural Networks—ICANN'97*, pages 583–588. Springer, 1997.

[25] Ameet Talwalkar and Afshin Rostamizadeh. Matrix coherence and the nystrom method. *arXiv preprint arXiv:1004.2008*, 2010.

[26] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Proceedings of the 14th Annual Conference on Neural Information Processing Systems*, number EPFL-CONF-161322, pages 682–688, 2001.

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

.