Nyström sketches

Daniel J. Perry (✉), Braxton Osting, and Ross T. Whitaker

University of Utah, USA,
{dperry@cs,osting@math,whitaker@cs}.utah.edu

Abstract. Despite prolific success, kernel methods become difficult to use in many large scale unsupervised problems because of the evaluation and storage of the full Gram matrix. Here we overcome this difficulty by proposing a novel approach: compute the optimal small, out-of-sample Nyström sketch which allows for fast approximation of the Gram matrix via the Nyström method. We demonstrate and compare several methods for computing the optimal Nyström sketch and show how this approach outperforms previous state-of-the-art Nyström coreset methods of similar size. We further demonstrate how this method can be used in an online setting and explore a simple extension to make the method robust to outliers in the training data.

Keywords: Nyström approximation, kernel PCA, kernel preimage

1 Introduction

Kernel-based learning methods have gained popularity in the last few decades due to their simplicity and powerful capability. Specifically, for unsupervised problems kernel PCA [18], spectral clustering [12], and many other methods have been used successfully on a variety of problems, see, *e.g.*, [25, 22, 19]. However, like other non-parametric learning models, their application to large data sets can be limited by the need to retain all the training data.

One of the most successful approximation techniques, generally referred to as the Nyström approximation, was introduced in [26] and further analyzed in [5] where a small subset of the data, which we will call a Nyström coreset, is used to approximate the full Gram matrix. The primary difficulty with Nyström approximation in an unsupervised setting is to select the right coreset to populate the dictionary—if the coreset is too large unneeded computational cost is incurred and if the coreset doesn't contain a representative set then too much error is incurred. While there has been significant work in the area of coreset selection (see, *e.g.*, [3, 1, 2, 13, 20, 24, 21]), these methods all select a subset of the training data for use in the Nyström approximation. This is counterintuitive because it is well known [14, 7] that the optimal basis on which to project data in an unsupervised setting are the PCA basis vectors, which for clarity we will refer to as a sketch instead of a coreset because the summary basis are a derivation (via SVD) rather than a subset fo the input data. We will show that, for very small element sizes, a Nyström sketch obtained by solving a particular optimization problem (similar to the kPCA optimization problem) are better able to describe the data (have smaller projection error) than a Nyström coreset of similar size. In applications, one frequently needs to project a dataset onto the coreset, which is a computationally intensive task, and a smaller coreset/sketch (with similar projection error) reduces these computational costs.

More recently, there has been significant effort to develop techniques that make use of random projections to estimate the lifting function directly [15, 16, 9, 11]. These methods can be thought of as a type of random sketch for kernel approximation, where instead of storing a coreset of the data from the input set, a set of random directions are retained for computing the nonlinear lifting. While theoretically interesting and demonstrably useful, there has also been work exhibiting significant advantages of the Nyström coreset approach [29], leading to the conclusion that, with regards to projection error, sampling from the dataset itself will always yield specific advantages to random projections. Here we demonstrate that with the same size dictionary, the proposed Nyström sketch obtains superior results than a random projection based kernel approximation.

In this paper, we propose a novel approach to kernel approximation that uses a Nyström sketch, similar to a PCA basis, instead of a Nyström coreset [5] or a random projection approximation [15, 9]. By formulating this Nyström sketch as an optimization problem, we also incorporate well known optimality ideas from the immensely successful PCA basis selection.

2 Background

While Nyström approximation of the Gram matrix is generally useful in both supervised and unsupervised settings, in order to focus the discussion we will primarily be investigating how different methods compare in estimating the kernel PCA (kPCA), and our primary measure of accuracy will be projection error resulting from the different approximations to kPCA.

2.1 Nyström coreset and sketch

Because there is some ambiguity around the terms matrix coreset and matrix sketch and to clarify discussion within this paper, we now introduce a more strict definition on *coreset* and *sketch* with respect to a given data matrix $X \in \mathbb{R}^{d \times n}, X = [x_1, \dots, x_n]$ for the purposes of Nyström approximation.

Definition 1. *A Nyström* coreset *is a subset matrix* $R \in \mathbb{R}^{d \times m}, R = [r_1, \dots, r_m]$ *where each of the columns* r_i *has a corresponding column in the input dataset,* $r_i = x_j$*, so that* $R = XQ$*, where* $Q \in \mathbb{R}^{n \times m}$ *is a column sampling matrix chosen for use in a Nyström approximation.*

Definition 2. *A Nyström* sketch *is a matrix* $S \in \mathbb{R}^{d \times c}, S = [s_1, \dots, s_c]$ *where each of the columns is derived from the input dataset for purposes of Nyström approximation, but the columns do not necessarily need to come from* X*–in this sense the columns are "out of sample".*

A Nyström sketch is a more general matrix, and so while a sketch could be a coreset (if the derived columns happened to correspond to columns in the dataset), a coreset is *not* necessarily a sketch.

2.2 PCA and kPCA

Linear PCA can be computed via a truncated SVD of $X = U\Sigma V^T$ at cost $\mathcal{O}(n^3)$, or the top p components can be computed using the Lanczos method (similar to power iteration) at cost $\mathcal{O}(npz)$ where z is the number of matrix multiplies required to compute one eigenvector, and depends on the singular value gap. For computational efficiency the eigendecomposition is typically performed on the outer product or covariance matrix, $XX^TU = U\Sigma^2$ when $d \ll n$, or on the inner product or Gram matrix, $X^TXV = V\Sigma^2$ situations where $d \gg n$. In either case, a subspace of dimension p is selected based on the eigenvalues, by selecting the subspace spanned by the p eigenvectors corresponding to the largest p eigenvalues. KPCA is a non-linear extension of PCA, where the data elements are lifted in a higher dimensional feature space $\mathbb{H}$ using a non-linear lifting function $\phi\colon \mathbb{R}^d \to \mathbb{H}$ prior to PCA, resulting in a data matrix in $\mathbb{H}$, $\Phi := \phi(X) = [\phi(x_1), \ldots, \phi(x_n)]$, and corresponding decomposition $\Phi = U\Sigma V^T$. The idea is that PCA in $\mathbb{H}$ will reveal nonlinear relationships in the lower dimensional input space $\mathbb{R}^d$. Because the dimension of $\mathbb{H}$ is potentially infinite, we compute the right singular vectors spanning $\mathrm{Im}(\Phi^T)$ using the Gram matrix eigendecomposition, $\Phi^T\Phi V = V\Sigma^2$. We recover the left singular vectors of Φ spanning $\mathrm{Im}(\Phi)$: $U = \Phi V\Sigma^{-1}$. A projection onto the kPCA subspace can be computed using the reproducing kernel, $y_i = \phi(x_i)^T U = \phi(x_i)^T \Phi V \Sigma^{-1} = k(x_i, X)V\Sigma^{-1}$, where the kernel function $k(a,b) = \phi(a)^T\phi(b)$ corresponds to an inner product in feature space. The lifting function $\phi(\cdot)$ and corresponding reproducing kernel $k(\cdot,\cdot)$ are typically chosen so that computing $k(a,b)$ is more efficient than the explicit expansion and evaluation of the inner product in $\mathbb{H}$.

2.3 kPCA projection error and Nyström approximation

For a collection of data points $\{x_i\}_{i=1}^n$, the mean projection error, E^p_{kpca}, for projection onto a p-dimensional PCA subspace is the mean of the lengths of the orthogonal projections,

$$E^p_{\mathrm{kpca}} = \frac{1}{n}\sum_{i=1}^{n} \|(I - V_pV_p^T)\phi(x_i)\|^2 = \frac{1}{n}\|(I - V_pV_p^T)\Phi\|_F^2. \tag{1}$$

Here $\|\cdot\|_F$ denotes the Frobenius norm. The error can be computed using a kernel trick,

$$\begin{aligned} E^p_{\mathrm{kpca}} &= \frac{1}{n}\sum_{i=1}^{n} \|(I - V_pV_p^T)\phi(x_i)\|^2 \qquad (2) \\ &= \frac{1}{n}\sum_{i=1}^{n} k(x_i, x_i) - k(X, x_i)^T k(X, X)_p^{-1} k(X, x_i), \end{aligned}$$

where $k(X,X)_p$ is the best rank-p approximation of the matrix $k(X,X)$.

Consider a Nyström coreset specified by the matrix $R \in R^{d\times m}$. Let $\Phi_R = \phi(R)$ have SVD $\Phi_R^T = U_R \Sigma_R V_R^T$ and $k(R,R) = \Phi_R^T \Phi_R \in \mathbb{R}^{m\times m}$ be the Gram matrix of the dictionary elements. We define the mean projection error, $E_{\text{kpca}}(R)$, for projection onto Φ_R analogously to (1) and simplify this expression using the kernel trick from (2),

$$\begin{aligned} E_{\text{kpca}}(R) &:= \frac{1}{n}\|(I - V_R V_R^T)\Phi\|^2 \\ &= \frac{1}{n}\sum_{i=1}^{n} k(x_i,x_i) - k(R,x_i)^T k(R,R)^{-1} k(R,x_i). \end{aligned} \tag{3}$$

This approximation to $k(x_i,x_i)$ is known as the Nyström approximation, which we aim to optimize.

3 General formulation of the Nyström sketch learning problem

We now consider learning an optimal Nyström sketch with $m < n$ elements that describes the collection of n data points $\{x_i\}_{i=1}^n \subset \mathbb{R}^d$. Consider a sketch specified by the matrix $R \in R^{d\times m}$ with columns given by m sketch elements $\{r_j\}_{j=1}^m \subset \mathbb{R}^d$.

Definition 3. Nyström sketch learning problem. *Compute the Nyström sketch $R^\star$ by solving the optimization problem,*

$$\min_{R\in R^{d\times m}} \; E_{kpca}(R), \tag{4}$$

where the objective function is defined in (2).

We will show below that the solution to this Nyström sketch learning problem for a linear kernel is the PCA basis of dimension m. For any nonlinear space the Nyström sketch learning problem is essentially tracking the preimage of the kPCA basis as it is computed. We discuss later how this approach differs from precomputing the kPCA basis and then finding a preimage afterwards, as well as some empirical evidence showing that the proposed sketch learning method performs much better in practice.

3.1 Optimal Nyström sketch is the PCA basis for the trivial lifting function

As an example, we consider the Nyström sketch learning problem (4) for the trivial lifting function, $\phi(x) = x$, with corresponding reproducing kernel $k(x,y) =$

$x^T y$. In this case, the objective function for the Nyström sketch learning problem can be written

$$E_{\text{kpca}}(R) = \frac{1}{n} \sum_{i=1}^{n} x_i^T x_i - x_i^T R(R^T R)^{-1} R^T x_i. \tag{5a}$$

Assume $m < d$. We observe that minimizing E_{kpca} over all $R \in R^{d \times m}$ is equivalent to maximizing

$$\begin{aligned} \sum_{i=1}^{n} x_i^T R(R^T R)^{-1} R^T x_i &= \operatorname{tr}\left[X^T R(R^T R)^{-1} R^T X\right] \\ &= \langle XX^T, R(R^T R)^{-1} R^T \rangle_F. \end{aligned}$$

where $\langle \cdot, \cdot, \rangle_F$ denote the Frobenius inner product. The quantity $R(R^T R)^{-1} R^T$ is simply the projection onto the image of R. Since the optimal Nyström sketch clearly has rank m, we can let $\tilde{U} \in \mathbb{R}^{d \times m}$ have columns which are an orthonormal basis for the image of R. Thus, $R(R^T R)^{-1} R^T = \tilde{U}\tilde{U}^T$. It follows from Von Neumann's trace inequality that $\langle XX^T, \tilde{U}\tilde{U}^T \rangle_F \leq \sum_{i=1}^{m} \lambda_i(XX^T)$ with equality only when $\operatorname{span}(\tilde{U})$ is the span of the first m left singular vectors of X. Thus the optimal Nyström sketch is any rank m matrix, R, with image equal to the span of the first m left singular vectors of X, or the rank-m PCA basis.

4 Optimal Nyström sketch using non-linear least-squares methods

In this section we show how the Nyström sketch learning problem (4) can be formulated as a nonlinear least squares problem, for which a variety of optimization algorithms, such as Gauss-Newton or Levenberg-Marquardt, can be applied to find the optimal Nyström sketch, R [27]. To apply such methods, we will require the gradient and Hessian of $E_{\text{kpca}}(R)$ with respect to the sketch, R.

We assume that the sketch R and kernel k have been chosen such that $k(R, R)$ is an invertible matrix. We first compute

$$\begin{aligned} &\frac{\partial}{\partial R_{jk}} k(R,x)^T k(R,R)^{-1} k(R,x) = \\ &\qquad 2k(R,x)^T k(R,R)^{-1} \left[\frac{\partial}{\partial R_{jk}} k(R,x)\right] \\ &\qquad + k(R,x)^T \left[\frac{\partial}{\partial R_{jk}} k(R,R)^{-1}\right] k(R,x) \end{aligned}$$

To compute the second term, we use the identity

$$\frac{\partial}{\partial R_{jk}} k(R,R)^{-1} = -k(R,R)^{-1} \left[\frac{\partial}{\partial R_{jk}} k(R,R)\right] k(R,R)^{-1}.$$

Thus we have

$$\frac{\partial}{\partial R_{jk}} k(R,x)^T k(R,R)^{-1} k(R,x) = \tag{6}$$
$$2k(R,x)^T k(R,R)^{-1} \left[\frac{\partial}{\partial R_{jk}} k(R,x)\right]$$
$$- k(R,x)^T k(R,R)^{-1} \left[\frac{\partial}{\partial R_{jk}} k(R,R)\right] k(R,R)^{-1} k(R,x).$$

Now write the objective

$$\begin{aligned} E_{\text{kpca}}(R) &= \frac{1}{n}\sum_{i=1}^{n} \|(I - V_R V_R^T)\phi(x_i)\|^2 \\ &= \frac{1}{n}\sum_{i=1}^{n} \|r_i\|^2 = \frac{1}{n}\|R\|^2, \end{aligned}$$

where the i-th residual vector and total residual vector are defined

$$r_i = \phi(x_i) - V_R V_R^T \phi(x_i) \quad \text{and} \quad R = \begin{pmatrix} r_1^T & \cdots & r_n^T \end{pmatrix}^T .$$

Using (6), the gradient of $E_{\text{kpca}}(R)$ is computed

$$\begin{aligned} \frac{\partial E_{\text{kpca}}}{\partial R_{jk}} &= \frac{1}{n}\sum_{i=1}^{n} \frac{\partial \|r_i\|^2}{\partial R_{jk}} \\ &= -\frac{1}{n}\sum_{i=1}^{n} 2k(R,x_i)^T k(R,R)^{-1} \left[\frac{\partial}{\partial R_{jk}} k(R,x_i)\right] \\ &\quad - k(R,x_i)^T k(R,R)^{-1} \left[\frac{\partial}{\partial R_{jk}} k(R,R)\right] k(R,R)^{-1} k(R,x_i). \end{aligned}$$

In particular, we find that the gradient of $E_{\text{kpca}}(R)$ can be evaluated in terms of the kernel function.

As an example, we compute the gradient for the linear and a general symmetric stationary kernels below.

4.1 Linear kernel

For the linear kernel, we have $k(x,y) = x^T y$, $k(R,x) = R^T x$, and $k(R,R) = R^T R$.

$$\frac{\partial [R^T R]_{l,m}}{\partial R_{j,k}} = R_{j,m}\delta_{k,l} + R_{j,l}\delta_{k,m} \tag{7}$$

$$\frac{\partial [R^T x]_l}{\partial R_{j,k}} = x_j \delta_{k,l} \tag{8}$$

where $\delta_{a,b} = \begin{cases} 1 & a = b \\ 0 & a \neq b \end{cases}.$

Algorithm 1 A least-squares based method for solving the Nyström sketch learning problem (4).

```
Input Mercer kernel $k(\cdot,\cdot)$, input data $X$, and sketch size $m$
Output Nyström sketch $R \in \mathbb{R}^{d\times m}$
Initialize: Let $R$ be a random subset of $X$ or use, e.g., [13]
Solve (4) using the Levenberg-Marquardt method (or alternative non-linear least squares method) to find the optimal parameter $R$.
```

4.2 Symmetric stationary kernel

For any symmetric stationary kernel of the form $k(x, y) = f(\|x - y\|^2)$.

$$\begin{aligned}\frac{\partial[k(R,R)]_{l,m}}{\partial R_{j,k}} &= f'(\|R_l - R_m\|^2)2\left[(R_{j,k} - R_{j,m})\delta_{l,k}\right.\\&\quad\left.+(R_{j,m} - R_{j,l})\delta_{m,k}\right]\\\frac{\partial[k(R,x)]_l}{\partial R_{j,k}} &= f'(\|R_l - x\|^2)2\delta_{l,k}\left[R_{j,k} - x_j\right]\end{aligned}$$

4.3 Batch algorithms

An algorithm for solving the Nyström sketch problem (4) is given in Algorithm 1. In this formulation we explicitly take advantage of the least-squares aspect of the optimal Nyström sketch problem. By using the special structure of a non-linear least-squares problem we are able to perform better than comparable gradient descent based methods. In Section 5, we compare the performance of Algorithm 1 with current state-of-the-art Nyström coreset selection and using other optimization strategies such as BFGS on both real and simulated data sets.

A second algorithm for estimating the optimal Nyström sketch is to first solve for the optimal basis in feature space using kPCA and then compute the preimage of the basis. Because there often is not a one-to-one mapping between input space and feature space, the preimage is an estimation problem, however it seems intuitive that if the optimal basis in feature space is the kPCA basis, then finding the corresponding preimage of those bases should provide a reasonable Nyström sketch.

While this approach has been used in other contexts—for example [4] use preimages of their incremental kPCA basis as a sketch—we have not seen it proposed as an explicit solution to the Nyström sketch learning problem. We consequently propose this technique as a "straw hat" approach to solving the sketch learning problem (details Algorithm 2). However, in Section 5, we exhibit several instances where a direct solution to the Nyström sketch learning problem via Algorithm 1 results in a more descriptive Nyström sketch. Furthermore the online and robust extensions to Algorithm 1 introduced in Sections 4.4 and 4.5 are not directly transferable to the preimage approach.

Algorithm 2 A preimage based solution to the Nyström sketch learning problem.

```
**Input** Mercer kernel $k(\cdot, \cdot)$, input data $X$, and sketch size $m$
**Output** Approximate optimal Nyström sketch $R \in \mathbb{R}^{d \times m}$
Initialize: Run kPCA to compute the bases $\{V_1, \ldots, V_m\}$ of the kPCA space of dimension $m$.
Apply a preimage algorithm, such as [17], to each $V_i$ independently, resulting in $m$ sketch elements, $d_i = \text{preimage}(V_i)$.
```

Algorithm 3 A stochastic gradient descent based solution to the online Nyström sketch learning problem (4).

```
**Input** Mercer kernel $k(\cdot, \cdot)$, input data $X$, sketch size $m$, and batch size $b$
**Output** Optimal Nyström sketch $R \in \mathbb{R}^{m \times d}$
Initialize stage: Let $R$ be the first $m$ elements observed of $X$.
Solve (4) using stochastic gradient descent over the next $b$ observed data elements, $\{x_i, \ldots, x_{i+b}\}$.
```

4.4 Online algorithm

In addition to the batch formulation in Algorithm 1, we also propose two explicit extensions that are useful in online and noisy contexts.

First, by design, a least squares solver will need to revisit each data point repeatedly, which becomes costly as the data set grows. While Algorithm 1 can be used for small- to medium-sized data sets, it does not scale to very large datasets. To address this problem we modify Algorithm 1 to use a stochastic gradient descent (SGD) method, resulting in Algorithm 3. We explore how this algorithm performs on large data sets (large enough that the Gram matrix can't be explicitly formed, making the approach in Algorithm 2 untenable) by comparing to current state-of-the-art kernel approximation techniques such as Random Kitchen Sinks [15, 16] and Fastfood [9]. We are able to demonstrate using the same sized dictionaries that the proposed optimal Nyström sketch obtains a smaller projection error on an unseen test set than these state-of-the-art kernel approximation techniques.

4.5 Robust online algorithm

A second extension is the use of an alternative loss to least-squares, possibly biasing the solution for certain desirable properties. Because the batch and online algorithms can use gradient descent methods instead of least squares methods, it is possible extend the underlying model to be something other than an ℓ_2-norm loss. One particular loss of interest in data mining is the least-absolute-deviations or ℓ_1-norm loss, which has some nice properties, such as being more robust to outliers than the ℓ_2-norm loss. Other work has adapted batch kPCA to use ℓ_1-norm loss, for example [28] developed an algorithm to solve for the kPCA basis using an ℓ_1-norm loss which yields a subspace basis which characterizes

Algorithm 4 A gradient descent based solution to the robust Nyström sketch learning problem (9).

```
Input Mercer kernel $k(\cdot,\cdot)$, input data $X$, sketch size $m$, batch size $b$, and regularization $\epsilon$
Output Optimal Nyström sketch $R \in \mathbb{R}^{m\times d}$
Initialize stage: Let $R$ be the first $m$ elements observed of $X$.
Apply (stochastic) gradient descent to find the optimal parameter $R$ to the least-absolute-deviations problem in (9) after observing $b$ new data elements $\{x_i, \ldots, x_{i+b}\}$.
```

the primary signal in the data more than the outliers. We propose the following modified projection error,

$$\begin{aligned} E_{\text{rkpca}}(R) &= \frac{1}{n}\sum_{i=1}^{n} \|(I - V_p V_p^T)\phi(x_i)\|_1 \\ &\approx \frac{1}{n}\sum_{i=1}^{n} \sqrt{k(x_i,x_i) - k(R,x_i)^T k(R,R)^{-1} k(R,x_i) + \epsilon} \end{aligned} \tag{9}$$

where the approximation is valid as $\epsilon \to 0$. This approximation is (i) differentiable for $\epsilon > 0$ and (ii) allows for evaluation using the kernel trick. The derivative of (9) with respect to the sketch can be calculated as in Section 4. To compute solutions of (9), we propose a gradient descent based method in the batch setting or SGD in the online setting, as detailed in Algorithm 4.

5 Experiments

We ran several experiments to explore how well the proposed Nyström sketch learning strategy works compared to current state-of-the-art methods in Nyström coreset selection.

To compare methods in batch mode, we generate a simple simulated dataset of a swirl with outliers, very similar to the swirl data set presented in [28]. This synthesized dataset facilitates exploring and visualizing the performance of the various selection techniques, including the effects of the robust loss function. For a view of real-world data, we also compared the batch algorithms on a subsample of the large forest cover data set from th UCI repository [10].

To compare the methods in an online setting, we use the forest cover, cpu, and two gas datatsets, gas-CO and gas-methane from the UCI repository [10, 6]. The datasets are large enough that it becomes difficult to compute kPCA exactly, making Algorithm 1 or Algorithm 2 unusable. Instead we compare to a random projection based kernel approximation methods. We first ran the proposed algorithm on a training subset, and compare the resulting Nyström approximation to the current state-of-the-art method for the Gaussian kernels on large data sets - random Fourier features (RFF) [15]. We note that the proposed method is learning from the training data while the RFF method is not; this is part of the point - by learning a small sketch for Nyström approximation, the proposed method is able to outperform the current state-of-the-art kernel approximation methods for similar sized sketches.

5.1 Methods

We present results using the least-squares method or *lsq* (Algorithm 1) using the *preimage* method (Algorithm 2), and the stochastic gradient descent method or *sgd* (Algorithm 3). For comparison we also include two non-stochastic gradient based variations on Algorithm 3, the first uses the BFGS method which we refer to as *bfgs*, and the second uses a typical first order gradient descent method *gd*. For details on the Levenberg-Marquadt method used in *lsq*, gradient descent, and the BFGS method we refer the reader to [27]. To compare to the current state-of-the-art kernel approximations, we use the seminal random Fourier features *rff* from [15, 16].

The *preimage* method makes use of the preimage approach in [17], which uses a fixed-point iteration solver to compute the preimage. We initially evaluated the preimage approaches in [8] which uses an interpolation of a neighborhood of points and [23] which combines both methods, but found that [17] performed best overall on the problems considered.

To compare against Nyström coreset methods we use the well studied uniform random coreset selection method *random* [5]. Although simple, this method is a common benchmark for Nyström coreset selection; we include it here for reference to other coreset methods.

5.2 Parameter selection

We specifically present examples using the Gaussian kernel, $k(x_1, x_2) = \frac{\|x_1 - x_2\|_2^2}{2\sigma^2}$, because of it's universal use and to demonstrate that even on problems where the feature space has high curvature and infinite feature size our method performs well. The parameter σ for the synthetic swirl data set was chosen small enough "spread out" the curve as a line in feature space. For the other datasets σ was selected according to the average inter-point distance of the data set, specific parameters are in Table 1.

For each dataset we estimated the number of kPCA components, p, needed to capture 90% of the kPCA spectral energy and set that as the smallest Nyström sketch. We then explore sketches of that size and larger, $m \geq p$.

5.3 Optimal Nyström sketch learning

The first experiment used a synthetic swirl dataset. This type of dataset is interesting in an unsupervised case, because it provides some view into how well the method is learning the (obvious) structure in the data. We generated a swirl with some clear outlier data in a way similar to [28], specifically we sample from the following region uniformly,

$$\{(0.07\gamma + \alpha)(\cos\gamma, \sin\gamma) \colon \alpha \in [0, 0.1], \gamma \in [0, 6\pi]\},$$

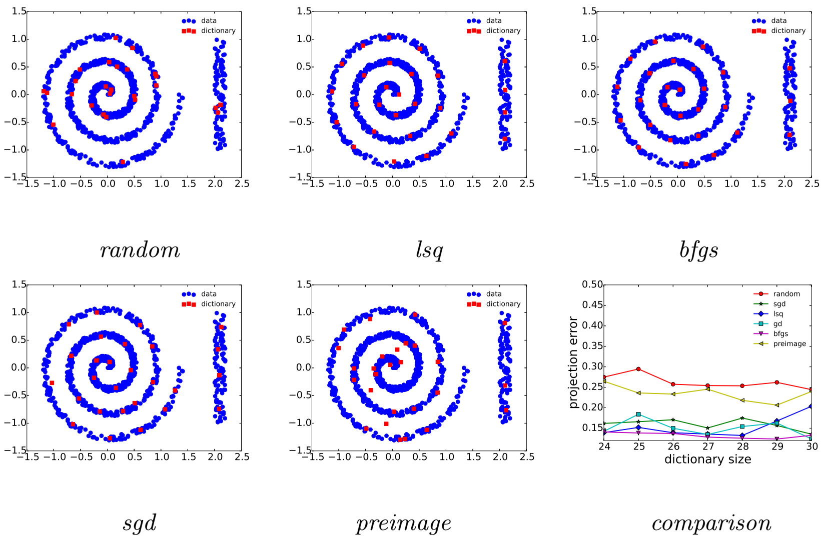

Fig. 1. Results running the various solution methods in batch mode on a synthesized swirl dataset. Top row and bottom left figures are visualizations comparing the sketch and coreset selection for each method. Bottom right figure is a plot of the projection error of each method.

as well as uniformly from a square region off to the side to provide some indication on the effect of obvious outliers on the result. We sample a total of 1000 points with 10% of those sampled from the outlier region. A summary of attributes for all data sets used is given in Table 1.

We are specifically interested in how the results change when the restriction of Nyström coreset selection to a more general Nyström sketch affects results. We have shown that the optimal Nyström sketch for a linear kernel are the first p PCA basis vectors. We hypothesize that relaxing the coreset restriction to a sketch learning problem in the non-linear kernel case also provides an advantage.

We present a visual and quantitative comparison of the Nyström coreset selection and sketch learning using the *random*, *lsq*, and *preimage* methods on the swirl data set in Figure 1. The plot of projection error shows that the proposed *lsq* approach performs the best for each size of coreset or sketch, and that both *lsq* and *preimage* sketches perform better than the random coreset of the same size. Visually we can see that each method obtains good coverage of the structure of the swirl, but *lsq* places the sketch elements at more strategic locations, for example the center of the swirl with high curvature area has more samples, while areas with less curvature have less sampling density.

Note that the *random* method maintains coreset, ie “in-sample” points, while the *preimage* and *lsq* methods can obtain sketch elements that are “out-of-sample”. We emphasize that this flexibility in sketch learning provides advantages coreset selection when the sketch/coreset is restricted in size.

data set	***d***	***n***	σ
swirl	2	1000	$\sqrt{0.10}$
forest (subsampled)	54	1000	2.59×10^3
cpu-train	21	6554	5.88×10^5
cpu-test	21	819	5.88×10^5
forest-train	54	522910	2.53×10^3
forest-test	54	58102	2.53×10^3
gas-CO-train	16	3708261	6.44×10^3
gas-CO-test	16	500000	6.44×10^3
gas-methane-train	16	3678504	4.68×10^3
gas-methane-test	16	500000	4.68×10^3

Table 1. data sets and parameters

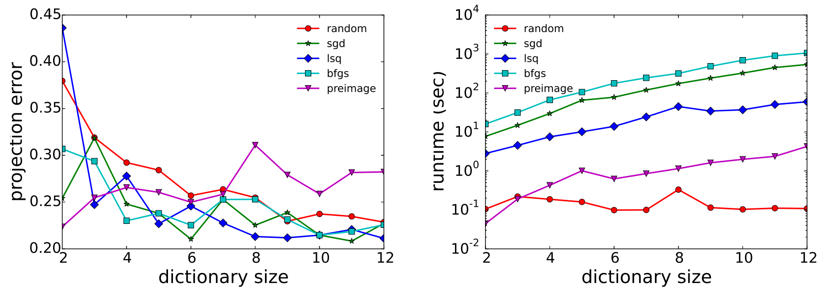

Fig. 2. Projection error (left) and runtime (right) for various solution methods in batch mode on a small subsample of the forest cover dataset.

We are also interested in understanding how various methods perform at solving the optimal Nyström sketch learning problem.

For this we ran a second experiment on a uniformly randomly subsample of the UCI forest cover data set [10]. We solved for the optimal sketch using *lsq*, *preimage*, *random*, as well as two gradient descent methods, *bfgs* and *sgd*. The results are plotted in Figure 2. Note that all three solvers for the optimal formulation, *lsq*, *bfgs*, and *sgd* perform the best for all sketch sizes. It's interesting that the *preimage* method doesn't perform as well as the sketch size increases.

We also plot the runtime of the various methods. Note that the optimal solvers also all take the most amount of time, and grow in cost as the sketch size increases. This indicates that computing an optimal sketch will primarily be worth the effort for very small sketches. We note that *lsq* runs faster than *sgd*, only because we gave a very loose convergence criteria for *lsq* (1.0×10^{-3}), while *sgd* must run through the entire data set. The *bfgs* method had the same convergence criteria but takes longer than both *lsq* and *sgd*.

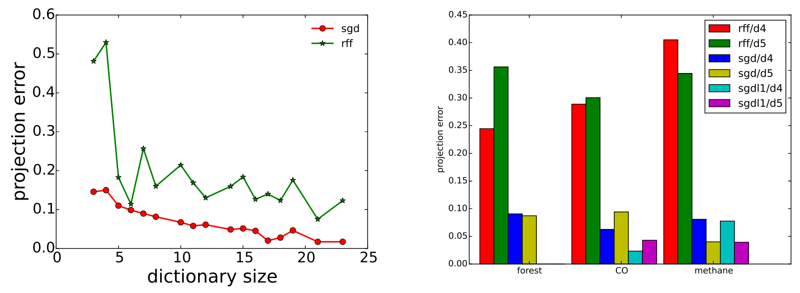

Fig. 3. Results for various solution methods in online mode on the real world datasets. (left) The projection error for the *sgd* and *rff* methods on the cpu dataset. (right) A summary of similar results for the forest, gas-CO, and gas-methane datasets, using sketch of size 4 and 5 (*d*4 and *d*5).

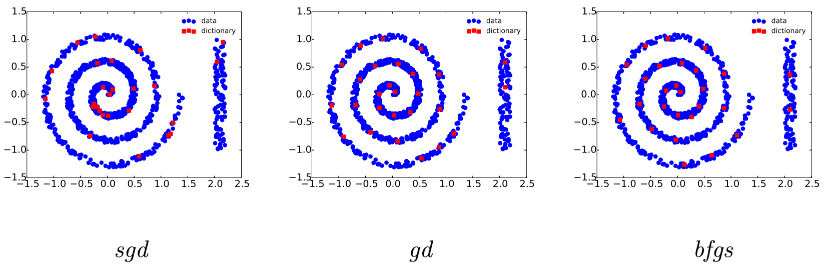

Fig. 4. Results for the gradient-descent-based methods on the robust sketch learning problem with a synthesized swirl dataset. Compare to Figure 1.

5.4 Online extension

As described previously, using a gradient descent method to solve the least-squares problem, opens the possibility of using a method like stochastic gradient descent to extend our model to be used in an online setting, as detailed in Algorithm 3.

The basic idea is that while observing new data we can continually update the sketch so that we alway have a very representative Nyström sketch.

To evaluate this idea, we simulated an online setting by splitting the UCI forest and cpu datasets into a training set and a test set each. Then we run the sketch learning algorithm on the (large) training set to learn the modified sketch as we “observe” new data. Since we are interested in using the kPCA in some kind of dimension reduction or other application, we evaluate the learned sketch on unseen data (the test set).

Because the datasets in an online setting are potentially very large, methods that need to revisit each data point repeatedly, like *lsq* and *bfgs*, are too slow.

However the *sgd* method is specifically useful in an online setting. Note that because the data set is too large for direct decomposition, the *preimage* method cannot be used on a dataset that large. Consequently we compare the sgd method with the current state-of-the-art in kernel approximation using *rff*, for the online setting.

The results of this experiment for the cpu dataset are shown in Figure 3 (left). Note that while the *sgd* was trained over a large training data set (cpu-train), the *rff* method doesn't require any training. Consequently projection error for the *rff* consists of projecting the data point onto the preselected random directions, taking the inner product, and then computing a difference with exact kernel inner product. Because the *sgd* method is trained on the same distribution of data and computes an approximate optimal sketch, it performs better for all sketch sizes evaluated.

The results on the forest, gas-CO, and gas-methane datasets are summarized in Figure 3 (right). This plot summarizes the projection error for the three datasets using the *rff*, *sgd*, and the ℓ_1 version of *sgd*. Each method was evaluated on a sketch of size 4 or 5, as indicated.

Again these experiments specifically point out the advantage of using a learned sketch and the Nyström approximation over using a random basis of the same size. While random projection methods excel when the basis size is very large, for very small sketches the Nyström approximation obtains a better projection error, as shown here.

5.5 Robust extension

We ran similar experiments using the robust extension described in Algorithm 4. The results for the swirl dataset are shown in Figure 4. Note that compared to the least-squares results in Figure 1 (especially *lsq*), the robust formulation places fewer sketch elements in the outlier section of the swirl data set. This is an important effect caused by the use of the ℓ_1 norm, reducing the effect of outliers on the sketch learning.

We also used the robust extension in an online setting on the gas-CO and gas-methane datasets, with results in Figure 3 (right). Note how in the gas-CO case the ℓ_1 result improved on the ℓ_2 result, while in the gas-methane dataset the results were similar to the ℓ_2 results. This real world example illustrates one case where a robust result improved generalization for the gas-CO dataset.

6 Conclusion

We introduced a novel out-of-sample dictionary learning method, and shown that it better describes the data than current state-of-the-art methods; in particular for the same sized-dictionaries, the proposed method yields dictionaries with smaller projection error. We proved that for the linear kernel, the proposed method gives a dictionary that is equivalent to the PCA subspace of the given size. We also demonstrated that because of the difficulty in mapping back into the input space, the proposed method for finding the optimal dictionary performs better than the pre-image of the kPCA subspace basis.

Acknowledgements

B. Osting was partially funded by NSF DMS 16-19755.

References

1. Ahmed El Alaoui and Michael W. Mahoney. Fast Randomized Kernel Methods With Statistical Guarantees. *arXiv*, pages 1–17, 2014.
2. Carlos Alzate and Johan AK Suykens. Kernel component analysis using an epsilon-insensitive robust loss function. *Neural Networks, IEEE Transactions on*, 19(9):1583–1598, 2008.
3. Daniele Calandriello, Alessandro Lazaric, and Michal Valko. Distributed Adaptive Sampling for Kernel Matrix Approximation. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 54:1421–1429, 2017.
4. Tat-Jun Chin and David Suter. Incremental kernel principal component analysis. *Image Processing, IEEE Transactions on*, 16(6):1662–1674, 2007.
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. Jordi Fonollosa, Sadique Sheik, Ramón Huerta, and Santiago Marco. Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring. *Sensors and Actuators B: Chemical*, 215:618–629, 2015.
7. Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.
8. JT-Y Kwok and Ivor W Tsang. The pre-image problem in kernel methods. *Neural Networks, IEEE Transactions on*, 15(6):1517–1525, 2004.
9. Quoc Le, Tamás Sarlós, and Alex Smola. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, 2013.
10. M. Lichman. UCI machine learning repository, 2013.
11. David Lopez-Paz, Suvrit Sra, Alex Smola, Zoubin Ghahramani, and Bernhard Schölkopf. Randomized nonlinear component analysis. *arXiv preprint arXiv:1402.0119*, 2014.
12. Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.
13. Marie Ouimet and Yoshua Bengio. Greedy spectral embedding. In *Proc. 10th Int. Workshop on Artificial Intelligence and Statistics*, pages 253–260. Citeseer, 2005.
14. Karl Pearson. Principal components analysis. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 6(2):559, 1901.
15. Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2007.
16. Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in neural information processing systems*, pages 1313–1320, 2009.

17. Bernhard Schölkopf, Sebastian Mika, Alex Smola, Gunnar Rätsch, and Klaus-Robert Müller. Kernel pca pattern reconstruction via approximate pre-images. In *ICANN 98*, pages 147–152. Springer, 1998.
18. Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.
19. Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. Generalized outlier detection with flexible kernel density estimates. In *Proceedings of the 14th SIAM International Conference on Data Mining (SDM), Philadelphia, PA*, pages 542–550, 2014.
20. Alex J Smola, Olvi L Mangasarian, and Bernhard Schölkopf. Sparse kernel feature analysis. In *Classification, Automation, and New Media*, pages 167–178. Springer, 2002.
21. Alex J Smola and Bernhard Schökopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 911–918. Morgan Kaufmann Publishers Inc., 2000.
22. Patrick Snape and Stefanos Zafeiriou. Kernel-pca analysis of surface normals for shape-from-shading. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1059–1066. IEEE, 2014.
23. Ana R Teixeira, Ana Maria Tomé, Kurt Stadlthanner, and Elmar Wolfgang Lang. Kpca denoising and the pre-image problem revisited. *Digital Signal Processing*, 18(4):568–580, 2008.
24. Michael E Tipping. Sparse kernel principal component analysis. In *Advances in Neural Information Processing Systems*, pages 633–639, 2001.
25. Jingdong Wang, Jianguo Lee, and Changshui Zhang. Kernel trick embedded gaussian mixture model. In *Algorithmic Learning Theory*, pages 159–174. Springer, 2003.
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. SJ Wright and J Nocedal. *Numerical optimization*, volume 2. Springer New York, 1999.
28. Yingchao Xiao, Huangang Wang, Wenli Xu, and Junwu Zhou. L1 norm based kpca for novelty detection. *Pattern Recognition*, 46(1):389–396, 2013.
29. Tianbao Yang, Yu-Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *Advances in neural information processing systems*, pages 476–484, 2012.