Stochastic Optimization for Kernel PCA*

Lijun Zhang[1,2] and **Tianbao Yang**[3] and **Jinfeng Yi**[4] and **Rong Jin**[5] and **Zhi-Hua Zhou**[1,2]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[2]Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China
[3]Department of Computer Science, the University of Iowa, Iowa City, USA
[4]IBM Thomas J. Watson Research Center, Yorktown Heights, USA
[5]Alibaba Group, Seattle, USA
{zhanglj, zhouzh}@lamda.nju.edu.cn, tianbao-yang@uiowa.edu, jinfengy@us.ibm.com, jinrong.jr@alibaba-inc.com

Abstract

Kernel Principal Component Analysis (PCA) is a popular extension of PCA which is able to find nonlinear patterns from data. However, the application of kernel PCA to large-scale problems remains a big challenge, due to its quadratic space complexity and cubic time complexity in the number of examples. To address this limitation, we utilize techniques from stochastic optimization to solve kernel PCA with *linear* space and time complexities per iteration. Specifically, we formulate it as a stochastic composite optimization problem, where a nuclear norm regularizer is introduced to promote low-rankness, and then develop a simple algorithm based on stochastic proximal gradient descent. During the optimization process, the proposed algorithm always maintains a low-rank factorization of iterates that can be conveniently held in memory. Compared to previous iterative approaches, a remarkable property of our algorithm is that it is equipped with an explicit rate of convergence. Theoretical analysis shows that the solution of our algorithm converges to the optimal one at an $O(1/T)$ rate, where T is the number of iterations.

Introduction

Principal Component Analysis (PCA) is a powerful dimensionality reduction method that has been widely used in various applications including data mining, information retrieval, and pattern recognition (Duda, Hart, and Stork 2000). While the classical PCA is limited to identifying linear structures, kernel PCA, a non-linear extension of PCA, has been proposed for extracting non-linear patterns from data (Schölkopf, Smola, and Müller 1998). The key idea is to map the data into a kernel-induced Hilbert space, where dot product between points can be computed efficiently through the kernel evaluation. Given a set of n training examples, kernel PCA needs to perform eigendecomposition of the $n \times n$ kernel matrix K. As it takes $O(n^2)$ space to store K and $O(n^3)$ time to eigendecompose it, kernel PCA is prohibitively expensive for big data, where n is very large.

Existing studies for reducing the computational cost of kernel PCA can be classified into two categories: approximate and iterative. Approximate approaches (Lopez-Paz et al. 2014) construct a low-rank approximator of the kernel matrix, and use its eigensystems as an alternative. Due to the low-rank structure, the approximator can be easily stored and manipulated. The major limitation of approximate approaches is that there always exists a non-vanishing gap between their solution and that found by eigendecomposing K directly. Iterative approaches (Kim, Franz, and Schölkopf 2005) use partial information of K in each round to estimate the top eigenvectors, and thus do not need to keep the entire matrix in memory. With appropriate initialization, the solution will converge to the groundtruth asymptotically. However, there is no guarantee of the convergence rate or the global convergence property for general initial conditions.

Inspired by the recent progresses in stochastic optimization (Avron et al. 2012; Rakhlin, Shamir, and Sridharan 2012), we develop a novel iterative algorithm for kernel PCA that has a solid convergence guarantee. The staring point is the following observation:

> Since only the top eigensystems of K are used in kernel PCA, it is sufficient to find a low-rank matrix $\widehat{K}$ from which we can recover the top eigensystems of K.

In this paper, we choose $\widehat{K}$ as the low-rank matrix obtained by applying Singular Value Shrinkage (SVS) operator (Cai, Candès, and Shen 2010) to K. Thus, the problem becomes how to estimate $\widehat{K}$ without constructing K explicitly. To this end, we formulate the SVS operation as a stochastic composite optimization problem, and develop an efficient algorithm based on Stochastic Proximal Gradient Descent (SPGD). The advantage of the stochastic formulation is that only a low-rank estimate of K is needed during the optimization process. Since the SVS operation is applied in each iteration, all the iterates are prone to be low-rank. Furthermore, the low-rankness of iterates in turn makes the SVS operation very efficient. As a result, in each iteration, both space and time complexities are linear in n.

By exploiting the strong convexity of the objective, we prove that the last iterate of SPGD converges to $\widehat{K}$ at an $O(1/T)$ rate, where T is the number of iterations. It implies we can simply take the last iterate as the final solution, and thus avoid the averaging operation in the traditional algorithms (Hazan and Kale 2011; Rakhlin, Shamir, and Sridharan 2012). Finally, we examine the empirical performance of the proposed algorithm on two benchmark data sets.

*This research was supported by NSFC (61333014, 61321491), NSF (IIS-1463988, IIS-1545995), and MSRA collaborative research project (FY14-RES-OPP-110).

Related Work

In this section, we briefly review the related work on kernel PCA and stochastic optimization.

Kernel PCA

The basic idea of kernel PCA is to map the input data into a Reproducing Kernel Hilbert Space (RKHS) induced by a kernel function, and perform PCA in that space (Schölkopf, Smola, and Müller 1998). Let $\mathcal{H}$ be a RKHS with a kernel function $\kappa(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y})$, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, where $\phi : \mathbb{R}^d \mapsto \mathcal{H}$ is a possibly nonlinear feature mapping. For the sake of simplicity, we assume the data are centered, i.e., $\sum_{i=1}^n \phi(\mathbf{x}_i) = 0$. The covariance matrix in $\mathcal{H}$ is given by $C = \frac{1}{n}\sum_{i=1}^n \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^\top$. We now have to find the eigenvalues $\lambda \geq 0$ and eigenvectors $\mathbf{v} \in \mathcal{H} \setminus \{0\}$ satisfying $C\mathbf{v} = \lambda\mathbf{v}$. Since all solutions $\mathbf{v}$ with $\lambda \neq 0$ lie within the span of $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)$, we can represent $\mathbf{v}$ as $\mathbf{v} = \Phi\mathbf{u}$, where $\Phi = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)]$ and $\mathbf{u} \in \mathbb{R}^n$. As a result, it is equivalent to consider the following problem

$$\frac{1}{n}\Phi\Phi^\top\Phi\mathbf{u} = \lambda\Phi\mathbf{u}. \tag{1}$$

Let $K \in \mathbb{R}^{n\times n}$ be the kernel matrix with $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ for $i, j = 1 \ldots, n$. Multiplying both sides of (1) by $\Phi^\top$, we obtain $K^2\mathbf{u} = \lambda n K\mathbf{u}$, which can be simplified to the eigenvalue problem $K\mathbf{u} = \lambda n\mathbf{u}$ (Schölkopf, Smola, and Müller 1998, Appendix A). Let $(\mathbf{u}_i, \lambda_i)$ be the i-th eigenvector and eigenvalue pair of K, with normalization $\lambda_i\|\mathbf{u}_i\|_2^2 = 1$. Then, the i-th eigenvector of C is given by $\mathbf{v}_i = \Phi\mathbf{u}_i$, which has unit length as indicated below

$$\mathbf{v}_i^\top\mathbf{v}_i = \mathbf{u}_i^\top\Phi^\top\Phi\mathbf{u}_i = \mathbf{u}_i^\top K\mathbf{u}_i = \lambda_i\|\mathbf{u}_i\|_2^2 = 1.$$

Generally speaking, it takes $O(dn^2)$ time to calculate K, $O(n^2)$ space to store, and $O(n^3)$ time to eigendecompose it. Thus, the vanilla procedure described above becomes computationally expensive when n is large. Approximate approaches for kernel PCA (Achlioptas, Mcsherry, and Schölkopf 2002; Ouimet and Bengio 2005; Zhang, Tsang, and Kwok 2008; Lopez-Paz et al. 2014) adopt matrix approximation techniques, such as the Nyström method (Williams and Seeger 2001; Drineas and Mahoney 2005), to construct a low-rank approximator of K, and then perform eigendecomposition of this low-rank matrix. For approximate approaches, there is a dilemma between the space complexity and the accuracy of their solution. The smaller the memory, the larger the approximation error, and vice visa. On the other hand, iterative approaches can find an accurate solution with a small memory, at the cost of a longer time. The most popular iterative approach for kernel PCA is the Kernel Hebbian Algorithm (KHA) (Kim, Franz, and Schölkopf 2005; Günter, Schraudolph, and Vishwanathan 2007), which is a kernelized version of the generalized Hebbian algorithm designed for linear PCA (Oja 1982; Sanger 1989). Similar to the algorithm proposed here, KHA is also a stochastic approximation algorithm. However, due to the non-convexity of its formulation, there is no global convergence guarantee for KHA.

While the work referenced above focus on reducing the cost of kernel PCA during training, there are some studies that aim to reduce its cost in testing. In particular, sparse kernel PCA (Tipping 2001) has been proposed to express each eigenvector $\mathbf{v}_i$ in terms of a small number of training examples. It was later extended to online setting (Honeine 2012), where training examples arrive sequentially.

Finally, we note that it is always possible to cast the problem of kernel PCA as a special case of linear PCA, which can be solved efficiently by online algorithms designed for linear PCA. To do this, we simply treat columns of K as feature vectors, evaluate them sequentially, and pass them to online algorithms for linear PCA. In this way, we can find the top eigensystems of K^2, from which we can derive the top eigensystems of K. However, this kind of approaches suffers from one of the following limitations.

1. Some online PCA algorithms, such as the generalized Hebbian algorithm, are only able to find top eigenvectors. But for kernel PCA, we need both top eigenvectors and eigenvalues.
2. Many online algorithms for linear PCA, such as capped MSG (Arora, Cotter, and Srebro 2013) and incremental SVD (Brand 2006), lack formal theoretical guarantees.
3. Although certain online algorithms are equipped with regret bounds (Warmuth and Kuzmin 2008), the difference between the eigenvectors returned by online algorithms and the ground-truth remains unclear.

Stochastic Optimization

Stochastic optimization refers to the setting where we can only access to the stochastic gradient of the objective function (Hazan and Kale 2011; Zhang, Mahdavi, and Jin 2013). For general Lipschitz continuous convex functions, Stochastic Gradient Descent (SGD) exhibits the unimprovable $O(1/\sqrt{T})$ rate of convergence (Nemirovski and Yudin 1983). For strongly convex functions, some variants of SGD (Hazan and Kale 2011; Rakhlin, Shamir, and Sridharan 2012; Zhang et al. 2013) achieve the optimal $O(1/T)$ rate (Agarwal et al. 2012).

Recently, a special case of stochastic optimization, namely Stochastic Composite Optimization (SCO), has received significant interest in optimization and learning communities (Ghadimi and Lan 2012; Lin, Chen, and Peña 2014; Zhang et al. 2014). In SCO, the objective function is given by the summation of non-smooth and smooth stochastic components (Lan 2012). The most popular non-smooth components are the ℓ_1-norm regularizer for vectors and the nuclear norm regularizer for matrices, which enforce sparseness and low-rankness, respectively. Although the generic algorithms designed for stochastic optimization can also be applied to SCO, by replacing gradient with subgradient, they can not utilize the structure of the objective function to generate sparse or low-rank intermediate solutions. The specialized algorithms for SCO are all built upon Stochastic Proximal Gradient Descent (SPGD), where the power of the non-smooth term is preserved (Lan 2012; Ghadimi and Lan 2012; Chen, Lin, and Peña 2012; Lin, Chen, and Peña 2014).

A major limitation of existing algorithms for SCO is that they did not treat memory as a limited resource. If we apply them to the SCO problem considered in this paper, the

memory complexity is still $O(n^2)$. We do find a heuristic algorithm (Avron et al. 2012) in the literature which combines truncated SVD with SGD to control the space complexity. But it relies on the assumption that the objective value can be evaluated easily, which unfortunately does not hold in our case. That is the reason why we choose the basic SPGD instead of more advanced methods to optimize our problem and establish a novel convergence guarantee for SPGD.

Algorithm

We first formulate kernel PCA as a SCO problem, then develop the optimization algorithm, next discuss implementation issues, and finally present the theoretical guarantee.

Reformulation of Kernel PCA

Denote the eigendecomposition of the kernel matrix K by $U\Lambda U^\top$, where $U = [\mathbf{u}_1, \ldots, \mathbf{u}_n]$, $\Lambda = \mathrm{diag}[\lambda_1, \ldots, \lambda_n]$, and $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_n$. To train kernel PCA, it is sufficient to find a low-rank matrix $\widehat{K}$ from which the top eigensystems of K can be recovered. The ideal low-rank matrix would be the truncated SVD of K, i.e., $\sum_{i=1}^k \lambda_i \mathbf{u}_i \mathbf{u}_i^\top$ for some integer $k > 0$. However, the truncated SVD operation is non-convex, making it difficult to design a principled algorithm. Instead, we consider the low-rank matrix $\widehat{K}$ obtained by applying the Singular Value Shrinkage (SVS) operator to K with threshold λ (Cai, Candès, and Shen 2010), [1] i.e.,

$$\widehat{K} = \mathcal{D}_\lambda[K] = \sum_{i:\lambda_i > \lambda} (\lambda_i - \lambda)\mathbf{u}_i\mathbf{u}_i^\top.$$

From the above expression, we observe that eigenvectors of $\widehat{K}$ with nonzero eigenvalues are the top eigenvectors of K. Furthermore, nonzero eigenvalues of $\widehat{K}$ are the top eigenvalues of K minus λ. As a result, we can recover the top eigensystems of K (with eigenvalues larger than λ) from the eigendecomposition of $\widehat{K}$.

In the following, we formulate the SVS operation as a SCO problem. First, it is well-known $\widehat{K}$ is the optimal solution to the following convex composite optimization problem

$$\min_{Z \in \mathbb{R}^{n\times n}} \frac{1}{2}\|Z - K\|_F^2 + \lambda\|Z\|_* \tag{2}$$

where $\|\cdot\|_*$ is the nuclear norm of matrices. Let ξ be a low-rank random matrix which is an unbiased estimate of K, i.e.,

$$K = \mathrm{E}[\xi].$$

We list examples of such random matrices below.

1. For general kernel matrix K, we can construct ξ by sampling its rows or columns randomly. Let $\{i_1, \ldots, i_k\}$ be a set of random indices sampled from $[n]$ uniformly, K_{*i_j} be the i_j-th column of K, $\mathbf{e}_{i_j}$ be the i_j-th canonical base. Then,
$$\xi = \frac{n}{k}\sum_{j=1}^k K_{*i_j}\mathbf{e}_{i_j}^\top$$
is an unbiased estimate of K with rank at most k. If a symmetric matrix is desired, we can set
$$\xi = \frac{n}{2k}\left(\sum_{j=1}^k K_{*i_j}\mathbf{e}_{i_j}^\top + \sum_{j=1}^k \mathbf{e}_{i_j}K_{*i_j}^\top\right)$$
which is an unbiased estimate of K with rank at most $2k$.
2. When the kernel matrix K is generated by a shift-invariant kernel, such as the Gaussian kernel and the Laplacian kernel. We can construct ξ by the random Fourier features (Rahimi and Recht 2008). Let $\kappa(\mathbf{x}, y)$ be the shift-invariant kernel with Fourier representation
$$\kappa(\mathbf{x}, \mathbf{y}) = \int p(\mathbf{w})\exp(j\mathbf{w}^\top(\mathbf{x} - \mathbf{y}))d\mathbf{w}$$
where $p(\mathbf{w})$ is a density function. Let $\mathbf{w}$ be a Fourier component randomly sampled from $p(\mathbf{w})$, and let $\mathbf{a}(\mathbf{w})$ and $\mathbf{b}(\mathbf{w})$ be the feature vectors generated by $\mathbf{w}$, i.e.,
$$\mathbf{a}(\mathbf{w}) = [\cos(\mathbf{w}^\top\mathbf{x}_1), \ldots, \cos(\mathbf{w}^\top\mathbf{x}_n)]^\top,$$
$$\mathbf{b}(\mathbf{w}) = [\sin(\mathbf{w}^\top\mathbf{x}_1), \ldots, \sin(\mathbf{w}^\top\mathbf{x}_n)]^\top.$$
By drawn k independent samples from $p(\mathbf{w})$, denoted by $\mathbf{w}_1, \ldots, \mathbf{w}_k$, we construct ξ as
$$\xi = \frac{1}{k}\sum_{i=1}^k \mathbf{a}(\mathbf{w}_i)\mathbf{a}(\mathbf{w}_i)^\top + \mathbf{b}(\mathbf{w}_i)\mathbf{b}(\mathbf{w}_i)^\top$$
which is an unbiased estimate of K with rank at most $2k$.
3. For dot product kernels such as the polynomial kernel, we can generate the random matrix ξ in a similar way (Kar and Karnick 2012).

Then, we rewrite $\|Z - K\|_F^2$ in (2) as

$$\begin{aligned}&\|Z - K\|_F^2 = \|Z\|_F^2 - 2\,\mathrm{tr}(Z^\top K) + \|K\|_F^2\\ =&\|Z\|_F^2 - 2\,\mathrm{tr}(Z^\top \mathrm{E}[\xi]) + \mathrm{E}[\|\xi\|_F^2] + \|K\|_F^2 - \mathrm{E}[\|\xi\|_F^2]\\ =&\mathrm{E}\left[\|Z - \xi\|_F^2\right] + \|K\|_F^2 - \mathrm{E}[\|\xi\|_F^2]\end{aligned}$$

Since $\|K\|_F^2 - \mathrm{E}[\|\xi\|_F^2]$ is a constant term with respect to Z, (2) is equivalent to

$$\min_{Z \in \mathbb{R}^{n\times n}} \frac{1}{2}\mathrm{E}\left[\|Z - \xi\|_F^2\right] + \lambda\|Z\|_* \tag{3}$$

a standard SCO problem with a nuclear norm regularizer.

Optimization by Stochastic Proximal Gradient Descent (SPGD)

At this point, one may consider applying existing algorithms for stochastic optimization to the problem in (3). Unfortunately, we find that all the previous algorithms can not be applied directly due to the high space complexity or unrealistic assumptions, as explained below.

1. The generic algorithms for stochastic optimization (Nemirovski et al. 2009; Hazan and Kale 2011; Rakhlin, Shamir, and Sridharan 2012; Shamir and Zhang 2012) are built up SGD, and thus cannot utilize the structure of (3) to enforce low-rankness. Furthermore, those algorithms return the average of iterates as the final solution, which could be full-rank.

[1] For a matrix $X \in \mathbb{R}^{m\times n}$ with singular value decomposition $U\Sigma V^\top$, where $\Sigma = \mathrm{diag}[\sigma_1, \ldots, \sigma_{\min(m,n)}]$, $\mathcal{D}_\lambda[X]$ is given by $\mathcal{D}_\lambda[X] = U\mathcal{D}_\lambda[\Sigma]V^\top$ and $\mathcal{D}_\lambda[\Sigma] = \mathrm{diag}\left[\max(0, \sigma_1 - \lambda), \ldots, \max(0, \sigma_{\min(m,n)} - \lambda)\right]$.

2. Although the specialized algorithms for SCO can generate low-rank iterates based on SPGD, they need to keep track of the averaged iterates as an auxiliary variable (Chen, Lin, and Peña 2012; Lin, Chen, and Peña 2014) or as the final solution (Lan 2012; Ghadimi and Lan 2012). Thus, the space complexity is still $O(n^2)$.
3. Although the heuristic algorithm in (Avron et al. 2012) is able to make the space complexity linear in n, it needs to evaluate the objective value in each iteration, which is impossible for the SCO problem in (3). Furthermore, it is designed for general SCO problems and thus cannot exploit the strong convexity of (3).

Due to the above reasons, we develop a new algorithm to optimize (3), which is purely based on SPGD and takes its last iterate as the final solution. Denote by Z_t the solution at the t-th iteration. In this iteration, we first sample a random matrix $\xi_t \in \mathbb{R}^{n\times n}$, and it is easy to verify that $Z_t - \xi_t$ is an unbiased estimate of the gradient of $\frac{1}{2}\mathrm{E}\left[\|Z-\xi\|_F^2\right]$. Then, we update the current solution by the SPGD, which is essentially a stochastic variant of composite gradient mapping (Nesterov 2013)

$$\begin{aligned}
&Z_{t+1}\\
=&\operatorname*{argmin}_{Z\in\mathbb{R}^{n\times n}} \frac{1}{2}\|Z-Z_t\|_F^2 + \eta_t\langle Z-Z_t, Z_t-\xi_t\rangle + \eta_t\lambda\|Z\|_*\\
=&\operatorname*{argmin}_{Z\in\mathbb{R}^{n\times n}} \frac{1}{2}\left\|Z-[(1-\eta_t)Z_t+\eta_t\xi_t]\right\|_F^2 + \eta_t\lambda\|Z\|_*\\
=&\mathcal{D}_{\eta_t\lambda}\left[(1-\eta_t)Z_t+\eta_t\xi_t\right]
\end{aligned}$$

where $\eta_t > 0$ is the step size. Let $Z'_{t+1} = (1-\eta_t)Z_t + \eta_t\xi_t$. The SVS operation applies a soft-thresholding rule to the singular values of Z'_{t+1}, effectively shrinking them toward zero. In particular, singular values of Z'_{t+1} that are below the threshold $\eta_t\lambda$ vanish, and thus Z_{t+1} tends to be a low-rank matrix.

Let Z_{T+1} be the final solution obtained after T iterations. If Z_{T+1} is symmetric, we will eigendecompose Z_{T+1} and obtain its eigensystems $\{(\mathbf{u}_i,\sigma_i)\}_{i=1}^k$ with nonzero eigenvalues. Otherwise, we will use the eigensystems of $(Z_{T+1}+Z_{T+1}^\top)/2$ instead of Z_{T+1}. Note that $(Z_{T+1}+Z_{T+1}^\top)/2$ is symmetric and always more close to $\widehat{K}$ than Z_{T+1}, since

$$\begin{aligned}
&\left\|\frac{1}{2}(Z_{T+1}+Z_{T+1}^\top)-\widehat{K}\right\|_F\\
\le&\frac{1}{2}\|Z_{T+1}-\widehat{K}\|_F + \frac{1}{2}\|Z_{T+1}^\top-\widehat{K}\|_F\\
=&\|Z_{T+1}-\widehat{K}\|_F.
\end{aligned}$$

Finally, we return $\{(\mathbf{u}_i,\sigma_i+\lambda)\}_{i=1}^k$ as the top eigensystems of K. The above procedure is summarized in Algorithm 1.

Although we assume that data are centered in RKHS, our algorithm can be immediately extend to the general case. If data are uncentered, kernel PCA (Schölkopf, Smola, and Müller 1998) needs the top eigensystems of $K+\Theta$, where

$$\Theta = \frac{1}{n^2}(\mathbf{1}_n^\top K\mathbf{1}_n)\mathbf{1}_n\mathbf{1}_n^\top - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top K - \frac{1}{n}K\mathbf{1}_n\mathbf{1}_n^\top$$

Algorithm 1 A Stochastic algorithm for Kernel PCA

Input: The number of trials T, and the regularization parameter λ

1: Initialize $Z_1 = 0$
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Sample a random matrix ξ_t
4: $\eta_t = 2/t$
5: $Z_{t+1} = \mathcal{D}_{\eta_t\lambda}\left[(1-\eta_t)Z_t + \eta_t\xi_t\right]$
6: **end for**
7: Calculate the nonzero eigensystems of $\frac{1}{2}(Z_{T+1} + Z_{T+1}^\top)$: $\{(\mathbf{u}_i,\sigma_i)\}_{i=1}^k$
8: **return** $\{(\mathbf{u}_i,\sigma_i+\lambda)\}_{i=1}^k$

and $\mathbf{1}_n$ is a n-dimensional vector of all ones. If ξ is an unbiased estimate of K, then it is easy to verify $\xi+\theta$ where

$$\theta = \frac{1}{n^2}(\mathbf{1}_n^\top\xi\mathbf{1}_n)\mathbf{1}_n\mathbf{1}_n^\top - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\xi - \frac{1}{n}\xi\mathbf{1}_n\mathbf{1}_n^\top$$

is an unbiased estimate of $K+\Theta$. To find the top eigensystems of $K+\Theta$, we just need to replace the random matrix ξ in our algorithm with $\xi+\theta$ and all the rest is the same.

Implementation Issues

In this section, we discuss how to ensure all the iterates are represented in low-rank factorization form and how to accelerate the SVS operation by utilizing this fact.

First, the random matrices ξ_t can always be represented by $\xi_t = \zeta_t\chi_t^\top$, where $\zeta_t, \chi_t \in \mathbb{R}^{n\times a_t}$ are two rectangular matrices with $a_t \ll n$. Now, suppose Z_t is also represented by $Z_t = U_tV_t^\top$, where $U_t, V_t \in \mathbb{R}^{n\times b_t}$ are two rectangular matrices with $b_t \ll n$.[2] Then, $Z_{t+1} = \mathcal{D}_{\eta_t\lambda}\left[(1-\eta_t)U_tV_t^\top + \eta_t\zeta_t\chi_t^\top\right]$ can be solved efficiently according to Lemma 3.4 of (Avron et al. 2012). Specifically, we introduce two matrices $X_t, Y_t \in R^{n\times(a_t+b_t)}$ such that

$$\begin{aligned}
X_t =&[\sqrt{1-\eta_t}U_t, \sqrt{\eta_t}\zeta_t],\\
Y_t =&[\sqrt{1-\eta_t}V_t, \sqrt{\eta_t}\chi_t], \text{ and } Z_{t+1} = \mathcal{D}_{\eta_t\lambda}[X_tY_t^\top].
\end{aligned}$$

Next, we perform a reduced QR decomposition (Golub and Van Loan 1996) of $X_t = Q_XR_X$ and $Y_t = Q_YR_Y$, and find the SVD of $R_XR_Y^\top = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^\top$. Define $\widehat{U}_t = Q_X\widetilde{U}$ and $\widehat{V}_t = Q_Y\widetilde{V}$. It is easy to verify that $\widehat{U}_t\widetilde{\Sigma}\widehat{V}_t^\top$ is the SVD of $X_tY_t^\top$, from which Z_{t+1} can be calculated trivially, and represented in the from of $Z_{t+1} = U_{t+1}V_{t+1}^\top$.

From the above discussion, it is clear that the space complexity is $O(n(a_t+b_t))$ in each iteration. The running time is dominated by calculating ξ_t, which takes $O(nda_t)$ time, and QR decompositions, which take $O(n(a_t+b_t)^2)$ time. In summary, the time complexity is $O(n[da_t+(a_t+b_t)^2])$. Thus, both space and time complexities are linear in n.

Theoretical Guarantee

The following theorem shows that with a high probability, Z_{T+1} converges to $\widehat{K}$, the optimal solution to (3), at an $O(1/T)$ rate.

[2] At least, we can represent Z_1 in this form since $Z_1 = 0$.

Theorem 1 *Assume the Frobenius norm of the random matrix ξ is upper bounded by some constant $C > 0$. By setting $\eta_t = 2/t$, with a probability at least $1 - \delta$, we have*

$$\begin{aligned}&\|Z_{T+1} - \widehat{K}\|_F^2 \\ \leq &\frac{8}{T}\left[C\lambda \max_{t\in[T]} \sqrt{r_t} + C^2\left(8 + 6\log\frac{\lceil 2\log_2 T\rceil}{\delta}\right)\right] \\ = &O\left(\frac{\log\log T}{T}\right)\end{aligned}$$

where r_t is the rank of Z_t.

Note that the $O(1/T)$ convergence rate matches the lower-bound of stochastic optimization of strongly convex functions (Agarwal et al. 2012). Our result differs from previous studies of SPGD (Rosasco, Villa, and Vũ 2014) in the sense that we prove a high probability bound instead of an expectation bound. Although a similar result has been proved for SGD (Rakhlin, Shamir, and Sridharan 2012), this is the first time such a guarantee is established for SPGD. The proof of this theorem relies on the recent analysis of SGD (Rakhlin, Shamir, and Sridharan 2012) and concentration inequalities (Bartlett, Bousquet, and Mendelson 2005; Cesa-Bianchi and Lugosi 2006). Due to space limitations, details are provided in the supplementary material.

Experiments

In this section, we perform several experiments to examine the performance of our method.

Experimental Setting

We compare our stochastic algorithm for kernel PCA (**SKPCA**) with the following methods.

1. **Baseline** (Schölkopf, Smola, and Müller 1998), which calculates the kernel matrix K explicitly and eigendecomposes it.
2. Approximation based on the **Nyström** method (Drineas and Mahoney 2005; Zhang, Tsang, and Kwok 2008), which uses the Nyström method to find a low-rank approximator of K, and eigendecomposes it.
3. Kernel Hebbian Algorithm (**KHA**) (Kim, Franz, and Schölkopf 2005), which is an iterative approach for kernel PCA.

In order to run SKPCA, we need to decide the value of the parameter λ in (3), which in turn determines the number of eigenvectors used in kernel PCA. To minimize the generalization error, we would like to find a λ such that eigenvalues of K that are smaller than it fall quickly (Shawe-Taylor et al. 2005). However, it is infeasible to calculate eigenvalues of K for large n, so we will use eigenvalues of a small kernel matrix $\overline{K}$ of m examples to estimate λ. Note that eigenvalues of K/n and $\overline{K}/m$ both converges to those of the integral operator (Braun 2006). Although the optimal step size of KHA in theory is $1/t$, we found it led to very slow convergence, and thus set it to be 0.05 as suggested by (Kim, Franz, and Schölkopf 2005).

We choose the Gaussian kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2))$, and set the kernel width σ to the 20-th percentile of the pairwise distances (Mallapragada et al. 2009). The random matrix in SKPCA is constructed by random Fourier features (Rahimi and Recht 2008). The experiments are done on two benchmark data sets: Mushrooms (Chang and Lin 2011) and Magic (Frank and Asuncion 2010), which contain $8,124$ and $19,020$ examples, respectively. We choose those two medium-size data sets, because they can be handled by Baseline and thus allow us to compare different methods quantitatively. For all the experiments, we repeat them 10 times and report the averaged result.

Experimental Results

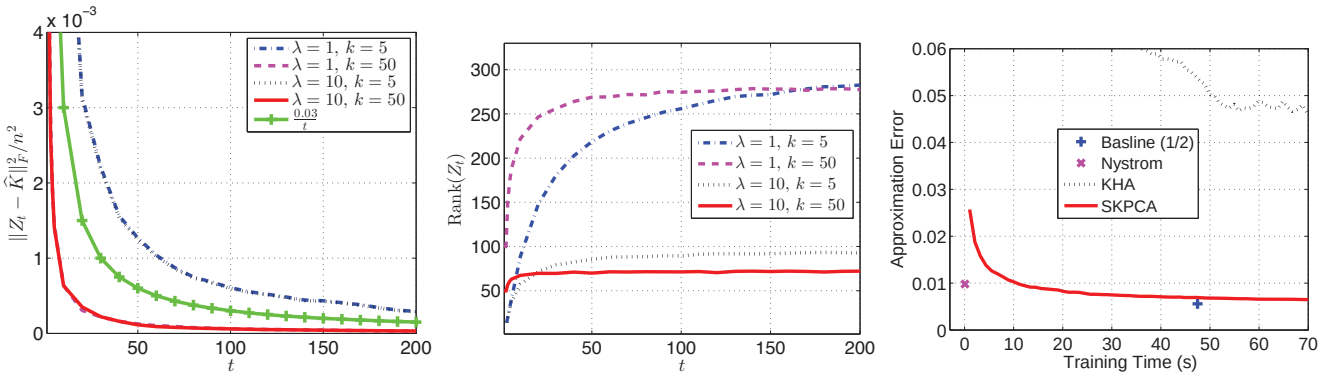
We first examine the convergence rate of SKPCA. We run SKPCA with four different combinations of the parameter λ and the number of random Fourier components k. In Fig. 1(a), we report the normalized recover error $\|Z_t - \widehat{K}\|_F^2/n^2$ with respect to the number of iterations t on the Mushrooms data set. For comparison, we also plot the curve of $0.03/t$. From the similarity among those curves, we believe the proposed algorithm achieves the $O(1/T)$ rate. As can be seen, the two curves of $k = 5$ (or $k = 50$) almost overlap with each other. That is probably because on this data set λ is not the dominating term in the upper bound given in Theorem 1. On the other hand, the convergence rate highly depends on the number of Fourier components k. The curves of $k = 50$ converge significantly faster than those of $k = 5$. The reason is that the larger k is, the closer ξ and K are, and the smaller the constant C in Theorem 1 is.

Then, we check the rank of the intermediate iterate Z_t, denoted by $\mathrm{rank}(Z_t)$, which determines the computational complexity of the t-th round. Fig. 1(b) plots $\mathrm{rank}(Z_t)$ as a function of t, which first increases and then converges to certain constant. The rank of the target matrix $\widehat{K}$ is 158 when $\lambda = 1$ and 55 when $\lambda = 10$. As can be seen, $\mathrm{rank}(Z_t)$ is just a constant factor larger than $\mathrm{rank}(\widehat{K})$. To compare different methods, we use the top 50 eigensystems returned by each algorithm to construct a rank-50 approximator of K, denoted by K^{50}, and report the approximation error $\|K^{50} - K\|_F/n$ in Fig. 1(c). In order to fit the figure, the training time of Baseline was divided by 2. The result returned by Baseline is optimal, but it takes a longer time and a much larger memory. Although Nyström is able to find a good solution, it cannot further reduce the approximation error. In comparison, SKPCA is able to refine its solution continuously and outperforms Nyström after 10 seconds. Finally, we note that SKPCA is much faster than KHA.

Experimental results on the Magic data set are provided in Fig. 2, which exhibits similar behaviors. On this data set, The rank of the $\widehat{K}$ is 89 when $\lambda = 10$ and 17 when $\lambda = 100$. The training time of Baseline was divided by 20 in Fig. 2(c).

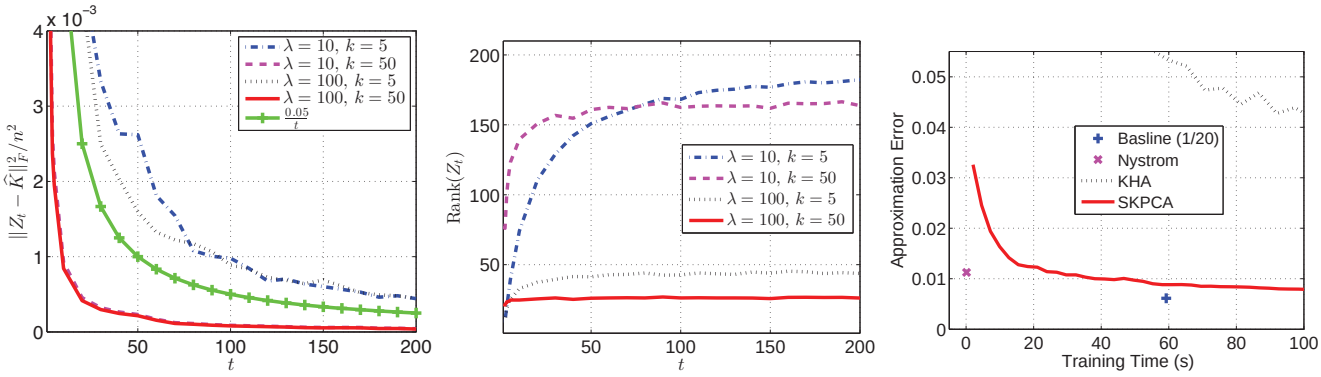
Conclusions

In this paper, we have formulated kernel PCA as a stochastic composite optimization problem with a nuclear norm regularizer, and then develop an iterative algorithm based on the stochastic proximal gradient descent algorithm. The main advantages of our method are i) both space and time complexes are linear in the number of samples; and ii) it is guaranteed to converge at an $O(1/T)$ rate, where T is the num-



(a) $\|Z_t - \widehat{K}\|_F^2/n^2$ versus t (b) $\mathrm{rank}(Z_t)$ versus t (c) Approximation error for K

Figure 1: Experimental Results on the Mushrooms data set. To fit the figure, the training time of Baseline was divided by 2 in (c).



(a) $\|Z_t - \widehat{K}\|_F^2/n^2$ versus t (b) $\mathrm{rank}(Z_t)$ versus t (c) Approximation error for K

Figure 2: Experimental Results on the Magic data set. To fit the figure, the training time of Baseline was divided by 20 in (c).

ber of iterations. Experiments on two benchmark data sets illustrate the efficiency and effectiveness of the proposed method.

References

Achlioptas, D.; Mcsherry, F.; and Schölkopf, B. 2002. Sampling techniques for kernel methods. In *Advances in Neural Information Processing Systems 14*, 335–342.

Agarwal, A.; Bartlett, P. L.; Ravikumar, P.; and Wainwright, M. J. 2012. Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization. *IEEE Transactions on Information Theory* 58(5):3235–3249.

Arora, R.; Cotter, A.; and Srebro, N. 2013. Stochastic optimization of pca with capped msg. In *Advances in Neural Information Processing Systems 26*, 1815–1823.

Avron, H.; Kale, S.; Kasiviswanathan, S.; and Sindhwani, V. 2012. Efficient and practical stochastic subgradient descent for nuclear norm regularization. In *Proceedings of the 29th International Conference on Machine Learning*, 1231–1238.

Bartlett, P. L.; Bousquet, O.; and Mendelson, S. 2005. Local rademacher complexities. *The Annals of Statistics* 33(4):1497–1537.

Brand, M. 2006. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications* 415(1):20–30.

Braun, M. L. 2006. Accurate error bounds for the eigenvalues of the kernel matrix. *Journal of Machine Learning Research* 7:2303–2328.

Cai, J.-F.; Candès, E. J.; and Shen, Z. 2010. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization* 20(4):1956–1982.

Cesa-Bianchi, N., and Lugosi, G. 2006. *Prediction, Learning, and Games*. Cambridge University Press.

Chang, C.-C., and Lin, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2(3):27:1–27:27.

Chen, X.; Lin, Q.; and Peña, J. 2012. Optimal regularized dual averaging methods for stochastic optimization. In *Advances in Neural Information Processing Systems 25*, 404–412.

Drineas, P., and Mahoney, M. W. 2005. On the nyström method for approximating a gram matrix for improved

kernel-based learning. *Journal of Machine Learning Research* 6:2153–2175.

Duda, R. O.; Hart, P. E.; and Stork, D. G. 2000. *Pattern Classification*. Wiley-Interscience Publication.

Frank, A., and Asuncion, A. 2010. UCI machine learning repository.

Ghadimi, S., and Lan, G. 2012. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization i: A generic algorithmic framework. *SIAM Journal on Optimization* 22(4):1469–1492.

Golub, G. H., and Van Loan, C. F. 1996. *Matrix computations, 3rd Edition*. Johns Hopkins University Press.

Günter, S.; Schraudolph, N. N.; and Vishwanathan, S. V. N. 2007. Fast iterative kernel principal component analysis. *Journal of Machine Learning Research* 8:1893–1918.

Hazan, E., and Kale, S. 2011. Beyond the regret minimization barrier: an optimal algorithm for stochastic strongly-convex optimization. In *Proceedings of the 24th Annual Conference on Learning Theory*, 421–436.

Honeine, P. 2012. Online kernel principal component analysis: A reduced-order model. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(9):1814–1826.

Kar, P., and Karnick, H. 2012. Random feature maps for dot product kernels. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, 583–591.

Kim, K. I.; Franz, M. O.; and Schölkopf, B. 2005. Iterative kernel principal component analysis for image modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(9):1351–1366.

Lan, G. 2012. An optimal method for stochastic composite optimization. *Mathematical Programming* 133:365–397.

Lin, Q.; Chen, X.; and Peña, J. 2014. A sparsity preserving stochastic gradient methods for sparse regression. *Computational Optimization and Applications* 58(2):455–482.

Lopez-Paz, D.; Sra, S.; Smola, A. J.; Ghahramani, Z.; and Schölkopf, B. 2014. Randomized nonlinear component analysis. In *Proceedings of the 31st International Conference on Machine Learning*.

Mallapragada, P. K.; Jin, R.; Jain, A. K.; and Liu, Y. 2009. Semiboost: Boosting for semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(11):2000–2014.

Nemirovski, A., and Yudin, D. B. 1983. *Problem complexity and method efficiency in optimization*. John Wiley & Sons Ltd.

Nemirovski, A.; Juditsky, A.; Lan, G.; and Shapiro, A. 2009. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization* 19(4):1574–1609.

Nesterov, Y. 2013. Gradient methods for minimizing composite functions. *Mathematical Programming* 140(1):125–161.

Oja, E. 1982. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology* 15(3):267–273.

Ouimet, M., and Bengio, Y. 2005. Greedy spectral embedding. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, 253–260.

Rahimi, A., and Recht, B. 2008. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, 1177–1184.

Rakhlin, A.; Shamir, O.; and Sridharan, K. 2012. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Conference on Machine Learning*, 449–456.

Rosasco, L.; Villa, S.; and Vũ, B. C. 2014. Convergence of stochastic proximal gradient algorithm. *ArXiv e-prints* arXiv:1403.5074.

Sanger, T. D. 1989. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks* 2(6):459–473.

Schölkopf, B.; Smola, A.; and Müller, K.-R. 1998. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation* 10(5):1299–1319.

Shamir, O., and Zhang, T. 2012. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. *ArXiv e-prints* arXiv:1212.1824.

Shawe-Taylor, J.; Williams, C. K. I.; Cristianini, N.; and Kandola, J. 2005. On the eigenspectrum of the gram matrix and the generalization error of kernel-pca. *IEEE Transactions on Information Theory* 51(7):2510–2522.

Tipping, M. E. 2001. Sparse kernel principal componen-t analysis. In *Advances in Neural Information Processing Systems 13*, 633–639.

Warmuth, M. K., and Kuzmin, D. 2008. Randomized online pca algorithms with regret bounds that are logarithmic in the dimension. *Journal of Machine Learning Research* 9:2287–2320.

Williams, C., and Seeger, M. 2001. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, 682–688.

Zhang, L.; Yang, T.; Jin, R.; and He, X. 2013. $O(\log T)$ projections for stochastic optimization of smooth and strongly convex functions. In *Proceedings of the 30th International Conference on Machine Learning*.

Zhang, W.; Zhang, L.; Hu, Y.; Jin, R.; Cai, D.; and He, X. 2014. Sparse learning for stochastic composite optimization. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 893–899.

Zhang, L.; Mahdavi, M.; and Jin, R. 2013. Linear convergence with condition number independent access of full gradients. In *Advance in Neural Information Processing Systems 26*, 980–988.

Zhang, K.; Tsang, I. W.; and Kwok, J. T. 2008. Improved nyström low-rank approximation and error analysis. In *Proceedings of the 25th International Conference on Machine Learning*, 1232–1239.