Projection techniques to update the truncated SVD of evolving matrices with applications

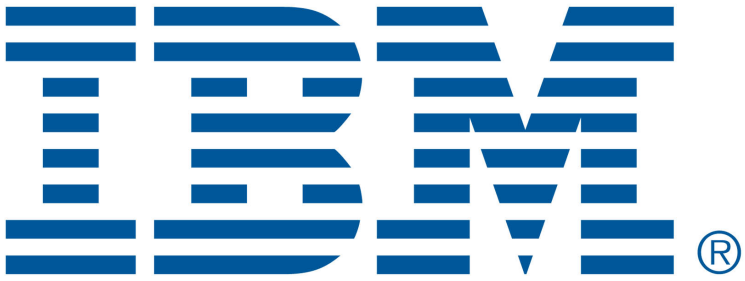
Vassilis Kalantzis, Georgios Kollias, Shashanka Ubaru,
Athanasios N. Nikolakopoulos, Lior Horesh,
and Kenneth L. Clarkson

June 2021

EPrint ID: 2021.2

IBM Research
Thomas J. Watson Research Center

Preprints available from:
https://researcher.watson.ibm.com/researcher/view.php?person=ibm-vkal



Projection techniques to update the truncated SVD of evolving matrices with applications

Vassilis Kalantzis [1] **Georgios Kollias** [1] **Shashanka Ubaru** [1]
Athanasios N. Nikolakopoulos [2] **Lior Horesh** [1] **Kenneth L. Clarkson** [3]

Abstract

Updating the rank-k truncated Singular Value Decomposition (SVD) of a matrix subject to the periodic addition of new rows (and/or columns) is a major computational kernel in important real-world applications such as latent semantic indexing and recommender systems. In this work we propose a new algorithm to update the truncated SVD of evolving matrices, i.e., matrices which are periodically augmented with a new set of rows (and/or columns). The proposed algorithm undertakes a projection viewpoint and builds a pair of subspaces which approximate the linear span of the sought singular vectors of the evolving matrix. We discuss and analyze two different choices to form the projection subspace, with the second approach being slower but leading to higher accuracy. Experiments on matrices from different applications suggest that the proposed algorithm can lead to higher qualitative accuracy than previous state-of-the-art approaches, as well as more accurate approximations of the truncated SVD. Moreover, the new algorithm is generally faster than other competitive approaches.

1. Introduction

This paper considers the problem of updating the rank-k truncated SVD of a sparse matrix subject to additions of new rows and/or columns. More specifically, let $B \in \mathbb{C}^{m\times n}$ be a matrix for which its rank-k (truncated) SVD B_k is available. Our goal is to obtain an approximate rank-k SVD A_k of matrix

$$A = \begin{pmatrix} B \\ E \end{pmatrix}, \text{ or } A = \begin{pmatrix} B & E \end{pmatrix},$$

where E denotes the matrix of newly added rows or columns. This process can be repeated several times, where at each instance matrix A becomes matrix B in the next level. A similar problem, not explored in this paper, is to approximate the rank-k SVD of B after modifying its existing entries, e.g., see (Zha & Simon, 1999).

Updating the SVD of evolving matrices is an important task in several real-world applications. One such example is Latent Semantic Indexing (LSI) in which the truncated SVD of the current term-document matrix needs to be updated as new terms/documents are added to the collection (Berry et al., 1995; Deerwester et al., 1990; Zha & Simon, 1999; Chen & Saad, 2009). Another example is the update of latent-factor models of user-item rating matrices in top-N recommendation (Cremonesi et al., 2010; Nikolakopoulos et al., 2019; Sarwar et al., 2002). Additional applications can be found in geostatistical screening (Horesh et al., 2015), and dimensionality reduction (Chen & Saad, 2008).

The standard approach to update the truncated SVD is to disregard any previously available information and apply directly to matrix A an off-the-shelf, high-performance library (Baglama & Reichel, 2005; Hernandez et al., 2005; Wu & Stathopoulos, 2015; Halko et al., 2011; Ubaru et al., 2019). This approach might be feasible when the original matrix is updated only once or twice, however becomes increasingly impractical as multiple row/column updates take place over time. Therefore, it becomes crucial to develop algorithms which return a reasonable approximation of the leading singular triplets while taking advantage of previous efforts. Such schemes have already been considered extensively for the case of full SVD (Brand, 2003; Gu et al., 1994; Moonen et al., 1992) and rank-k SVD (Berry et al., 1995; Sarwar et al., 2002; Vecharynski & Saad, 2014; Zha & Simon, 1999; Baker et al., 2012).

Contributions

1. We propose and analyze a projection scheme to update the rank-k SVD of evolving matrices. Our scheme uses a right singular projection subspace equal to $\mathbb{C}^n$, and only determines the left singular projection subspace.

[1]IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 [2]Amazon, 550 Terry Ave N, Seattle, WA 98109 (work done prior to joining Amazon) [3]IBM Research, Almaden Research Center, San Jose, CA 95120. Correspondence to: Vassilis Kalantzis <vkal@ibm.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

2. We propose and analyze two different options to set the left singular projection subspace. In both cases, the projected problem is solved in matrix-free fashion by the Lanczos algorithm instead of dense SVD solver. A complexity analysis is presented in the Supplementary Material.
3. We present experiments performed on evolving matrices originating from applications in LSI. These experiments suggest that the proposed scheme can be both faster and more accurate than state-of-the-art algorithms. Applications in face recognition and the Eigenfaces technique are also explored.

2. Background and notation

The (full) SVD of matrix B is denoted as $B = U\Sigma V^H$ where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary matrices whose jth column is equal to the left singular vector $u^{(j)}$ and right singular vector $v^{(j)}$, respectively. The matrix $\Sigma \in \mathbb{R}^{m\times n}$ has non-zero entries only along its main diagonal, and these entries are equal to the singular values $\sigma_1 \geq \cdots \geq \sigma_{\mathtt{min}(m,n)}$. Moreover, we define the matrices $U_j = \left[u^{(1)}, \ldots, u^{(j)}\right]$, $V_j = \left[v^{(1)}, \ldots, v^{(j)}\right]$, and $\Sigma_j = \mathtt{diag}\left(\sigma_1, \ldots, \sigma_j\right)$. The rank-$k$ truncated SVD of matrix B can then be written as $B_k = U_k\Sigma_k V_k^H = \sum_{j=1}^{k} \sigma_j u^{(j)} \left(v^{(j)}\right)^H$. We follow the same notation for matrix A with the exception that a circumflex is added on top of each variable, i.e., $A_k = \widehat{U}_k\widehat{\Sigma}_k\widehat{V}_k^H = \sum_{j=1}^{k} \widehat{\sigma}_j\widehat{u}^{(j)} \left(\widehat{v}^{(j)}\right)^H$, with $\widehat{U}_j = \left[\widehat{u}^{(1)}, \ldots, \widehat{u}^{(j)}\right]$, $\widehat{V}_j = \left[\widehat{v}^{(1)}, \ldots, \widehat{v}^{(j)}\right]$, and $\widehat{\Sigma}_j = \mathtt{diag}\left(\widehat{\sigma}_1, \ldots, \widehat{\sigma}_j\right)$.

The routines $\mathtt{nr}(K)$ and $\mathtt{nnz}(K)$ return the number of rows and non-zero entries of matrix K, respectively. Throughout this paper $\|\cdot\|$ will stand for the ℓ_2 norm when the input is a vector, and the spectral norm when the input is a matrix. Moreover, the term $\mathtt{range}(K)$ denotes the column space of matrix K, and $\mathtt{span}(\cdot)$ denotes the linear span of a set of vectors. The identity matrix of size n will be denoted by I_n.

2.1. Related work.

The problem of updating the SVD of an evolving matrix has been considered extensively in the context of LSI. Consider first the case $A = \begin{pmatrix} B \\ E \end{pmatrix}$, and let $(I - V_k V_k^H)E^H = QR$ such that Q is orthonormal and R is upper trapezoidal. The scheme in (Zha & Simon, 1999) writes

$$\begin{pmatrix} B \\ E \end{pmatrix} \approx \begin{pmatrix} U_k\Sigma_k V_k^H \\ E \end{pmatrix} = \begin{pmatrix} U_k & \\ & I_s \end{pmatrix} \begin{pmatrix} \Sigma_k \\ EV_k \;\; R^H \end{pmatrix} \begin{pmatrix} V_k & Q \end{pmatrix}^H = \left(\begin{pmatrix} U_k & \\ & I_s \end{pmatrix} F \right) \Theta \left(\begin{pmatrix} V_k & Q \end{pmatrix} G \right)^H$$

where the matrix product $F\Theta G^H$ denotes the compact SVD of the matrix $\begin{pmatrix} \Sigma_k \\ EV_k \;\; R^H \end{pmatrix}$.

The above idea can be also applied to $A = \begin{pmatrix} B & E \end{pmatrix}$. Indeed, if matrices Q and R are now determined as $(I - U_k U_k^H)E = QR$, we can approximate

$$\begin{aligned} \begin{pmatrix} B & E \end{pmatrix} &\approx \begin{pmatrix} U_k\Sigma_k V_k^H & E \end{pmatrix} \\ &= \begin{pmatrix} U_k & Q \end{pmatrix} \begin{pmatrix} \Sigma_k & U_k^H E \\ & R \end{pmatrix} \begin{pmatrix} V_k^H & \\ & I_s \end{pmatrix} \\ &= \left(\begin{pmatrix} U_k & Q \end{pmatrix} F \right) \Theta \left(\begin{pmatrix} V_k & \\ & I_s \end{pmatrix} G \right)^H \end{aligned}$$

where the matrix product $F\Theta G$ now denotes the compact SVD of the matrix $\begin{pmatrix} \Sigma_k & U_k^H E \\ & R \end{pmatrix}$.

When B_k coincides with the compact SVD of B, the above schemes compute the exact rank-k SVD of A, and no access to matrix B is required. Nonetheless, the application of the method in (Zha & Simon, 1999) can be challenging. For general updating problems, or problems where A does not satisfy a "low-rank plus shift" structure (Zha & Zhang, 2000), replacing B by B_k might not lead to a satisfactory approximation of A_k. Moreover, the memory/computational cost associated with the computation of the QR and SVD decompositions in each one of the above two scenarios might be prohibitive. The latter was recognized in (Vecharynski & Saad, 2014) where it was proposed to adjust the method in (Zha & Simon, 1999) by replacing matrices $(I - V_k V_k^H)E^H$ and $(I - U_k U_k^H)E$ with a low-rank approximation computed by applying the Golub-Kahan Lanczos bidiagonalization procedure (Golub & Kahan, 1965). Similar ideas have been suggested in (Yamazaki et al., 2017) and (Ubaru & Saad, 2019) where the Golub-Kahan Lanczos bidiagonalization procedure was replaced by randomized SVD (Halko et al., 2011; Ubaru et al., 2015) and graph coarsening (Ubaru & Saad, 2019), respectively.

3. The projection viewpoint

The methods discussed in the previous section can be recognized as instances of a Rayleigh-Ritz projection procedure and can be summarized as follows (Vecharynski & Saad, 2014; Yamazaki et al., 2017):

1. Compute orthonormal matrices Z and W such that $\mathtt{range}(Z)$ and $\mathtt{range}(W^H)$ approximately capture $\mathtt{range}(\widehat{U}_k)$ and $\mathtt{range}(\widehat{V}_k^H)$, respectively.
2. Compute $[\Theta_k, F_k, G_k] = \mathtt{svd}_k(Z^H AW)$ where Θ_k, F_k, and G_k denote the k leading singular values and associated left and right singular vectors of $Z^H AW$, respectively.
3. Approximate A_k by the product $(ZF_k)\Theta_k(WG_k)^H$.

Table 1. Different options to set the projection matrices Z and W for the row updating problem.

Method	Z	W
(Berry et al., 1995)		V_k
(Zha & Simon, 1999)	$Z = \begin{pmatrix} U_k & \\ & I_s \end{pmatrix}$	$[V_k, Q]$
(Vecharynski & Saad, 2014)		$[V_k, X_l]$
Alg. 1 (option I)	$Z = \begin{pmatrix} U_k & \\ & I_s \end{pmatrix}$	I_n
Alg. 1 (option II)	$Z = \begin{pmatrix} U_k, X_{\lambda,r} & \\ & I_s \end{pmatrix}$	I_n

Ideally, the matrices Z and W should satisfy

$$\texttt{span}\left(\widehat{u}^{(1)}, \ldots, \widehat{u}^{(k)}\right) \subseteq \texttt{range}(Z), \quad \text{and}$$
$$\texttt{span}\left(\widehat{v}^{(1)}, \ldots, \widehat{v}^{(k)}\right) \subseteq \texttt{range}(W).$$

Moreover, the size of matrix $Z^H AW$ should be as small as possible to avoid high computational costs during the computation of $[\Theta_k, F_k, G_k] = \texttt{svd}(Z^H AW)$.

Table 1 summarizes a few options to set matrices Z and W for the row updating problem. The method in (Vecharynski & Saad, 2014) considers the same matrix Z as in (Zha & Simon, 1999) but sets $W = [V_k, X_l]$ where X_l denotes the $l \in \mathbb{N}$ leading left singular vectors of $(I - V_k V_k^H)E^H$. The choice of matrices Z and W listed under the option "Algorithm 1" is explained in the next section. Note that the first variant of Algorithm 1 uses the same Z as in (Zha & Simon, 1999) and (Vecharynski & Saad, 2014) but different W. This choice leads to higher accuracy than the scheme in (Zha & Simon, 1999) which is also achieved asymptotically faster. A detailed comparison is deferred to the Supplementary Material. The second variant of Algorithm 1 is generally slower but more accurate.

3.1. The proposed algorithm.

Consider again the SVD update of matrix $A = \begin{pmatrix} B \\ E \end{pmatrix}$, with $E \in \mathbb{C}^{s \times n}$. The right singular vectors of A trivially satisfy $\widehat{v}^{(i)} \subseteq \texttt{range}(I_n)$, $i = 1, \ldots, n$. Therefore, we can simply set $W = I_n$ and compute the k leading singular triplets $\left(\theta_i, f^{(i)}, g^{(i)}\right)$ of the matrix $Z^H AW = Z^H A$. Indeed, this choice of W is ideal in terms of accuracy while it also removes the need to compute an approximate factorization of matrix $(I - V_k V_k^H)E^H$. On the other hand, the number of columns in matrix $Z^H AW$ is now equal to n instead of $k + s$ in (Zha & Simon, 1999) and $k + l$, $l \ll s$, in (Vecharynski & Saad, 2014; Yamazaki et al., 2017).

The approach proposed in this paper does not invoke any dense SVD solvers; rather computes the singular values of $Z^H A$ in a matrix-free fashion. Moreover, it skips the computation of the right singular vectors G_k. Indeed, the matrix G_k is only needed to approximate the k leading singular vectors $\widehat{V}_k$ of A. Assuming that an approximation $\overline{U}_k$ and $\overline{\Sigma}_k$ of the matrices $\widehat{U}_k$ and $\widehat{\Sigma}_k$ is available, $\widehat{V}_k$ can be approximated as $\overline{V}_k = A^H \overline{U}_k \overline{\Sigma}_k^{-1}$.

Algorithm 1 The proposed algorithm (row updates).

1: **Input:** B, k
2: Solve $[U_k, \Sigma_k, V_k] = \texttt{svd}_k(B)$
3: Receive matrix E
4: Build matrix Z as described in Section 4
5: Solve $[\Theta_k, F_k] = \texttt{svd}_k(Z^H A)$ where $A = \begin{pmatrix} B \\ E \end{pmatrix}$
6: Set $\overline{U}_k = ZF_k$ and $\overline{\Sigma}_k = \Theta_k$
7: Set $\overline{V}_k = A^H \overline{U}_k \overline{\Sigma}_k^{-1}$
8: If new rows need to be added, set $B \equiv A$ and repeat from Step 3, else proceed to Step 9 and exit
9: **Output:** $\overline{U}_k \approx \widehat{U}_k, \overline{\Sigma}_k \approx \widehat{\Sigma}_k, \overline{V}_k \approx \widehat{V}_k$

The proposed algorithm is sketched in Algorithm 1. In terms of computational cost, Steps 6 and 7 require approximately $2\texttt{nnz}(Z)k$ and $(2\texttt{nnz}(A) + n)k$ Floating Point Operations (FLOPS), respectively. The complexity of Step 5 will generally depend on the algorithm used to compute the matrices Θ_k and F_k. We compute these by applying the unrestarted Lanczos method to matrix $Z^H AA^H Z$ in a matrix-free fashion (Saad, 2011). The Lanczos process terminates after $\delta \in \mathbb{N}$ iterations, where δ typically is a small multiple of k. The computational cost of Step 5 is then roughly equal to $2\texttt{mv}(Z^H A)\delta + 2\texttt{nr}(Z^H)\delta^2$ FLOPS, where $\texttt{mv}(Z^H A)$ denotes the FLOPS required to perform a Matrix-Vector product (MV) with matrix $Z^H A$. The exact complexity of Lanczos will depend on the choice of matrix Z. A detailed asymptotic analysis of the complexity of Algorithm 1, as well as a comparison with the schemes in (Zha & Simon, 1999) and (Vecharynski & Saad, 2014; Yamazaki et al., 2017) are deferred to the Supplementary Material.

Throughout the remainder of this paper we focus on updating the rank-k SVD of row update problems $A = \begin{pmatrix} B \\ E \end{pmatrix}$. The discussion extends trivially to column update problems of the form $A = \begin{pmatrix} B & E \end{pmatrix}$ by converting the latter to a row update problem of the matrix $A^H = \begin{pmatrix} B^H \\ E^H \end{pmatrix}$. A detailed summary is provided in Algorithm 2. Note that by combining Algorithms 1 and 2 we can approximate the k leading singular triplets of evolving matrices in which both new rows and columns are added.

Algorithm 2 The proposed algorithm (column updates).

1: **Input:** $B,\ k$
2: Solve $[U_k, \Sigma_k, V_k] = \texttt{svd}_k(B)$
3: Receive matrix E
4: Build matrix Z as described in Section 4
5: Solve $[\Theta_k, F_k] = \texttt{svd}_k(Z^H A^H)$ where $A = \begin{pmatrix} B & E \end{pmatrix}$
6: Set $\overline{V}_k = ZF_k$ and $\overline{\Sigma}_k = \Theta_k$
7: Set $\overline{U}_k = A\overline{V}_k\overline{\Sigma}_k^{-1}$
8: If new columns need to be added, set $B \equiv A$ and repeat from Step 3, else proceed to Step 9 and exit
9: **Output:** $\overline{U}_k \approx \widehat{U}_k, \overline{\Sigma}_k \approx \widehat{\Sigma}_k, \overline{V}_k \approx \widehat{V}_k$

4. Building the projection matrix Z

The accuracy of the k approximate leading singular triplets returned by Algorithm 1 depends on how well $\texttt{range}(Z)$ captures the singular vectors $\widehat{u}^{(1)}, \ldots, \widehat{u}^{(k)}$ (Jia & Stewart, 2001; Nakatsukasa, 2017). Therefore, our focus now turns in constructing a matrix Z such that the distance between the subspace $\texttt{range}(Z)$ and the left singular vectors $\widehat{u}^{(1)}, \ldots, \widehat{u}^{(k)}$ is as small as possible.

4.1. Exploiting the left singular vectors of B.

The following proposition presents a closed-form expression of the ith left singular vector of matrix $A = \begin{pmatrix} B \\ E \end{pmatrix}$.

Proposition 1. *The left singular vector $\widehat{u}^{(i)}$ associated with singular value $\widehat{\sigma}_i$ is equal to*

$$\widehat{u}^{(i)} = \begin{pmatrix} -(BB^H - \widehat{\sigma}_i^2 I_m)^{-1} B E^H \widehat{y}^{(i)} \\ \widehat{y}^{(i)} \end{pmatrix},$$

where $\widehat{y}^{(i)}$ satisfies the equation

$$\left[E \left(\sum_{j=1}^{n} v^{(j)} \left(v^{(j)}\right)^H \frac{\widehat{\sigma}_i^2}{\widehat{\sigma}_i^2 - \sigma_j^2} \right) E^H - \widehat{\sigma}_i^2 I_s \right] \widehat{y}^{(i)} = 0,$$

and $\sigma_j = 0$ for any $j = m+1, \ldots, n$ (when $n > m$).

Proof. Deferred to the Supplementary Material. □

The above representation of $\widehat{u}^{(i)}$ requires the solution of a nonlinear eigenvalue problem to compute $\widehat{y}^{(i)}$. Alternatively, we can express $\widehat{u}^{(i)}$ as follows.

Proposition 2. *The left singular vector $\widehat{u}^{(i)}$ associated with singular value $\widehat{\sigma}_i$ is equal to*

$$\widehat{u}^{(i)} = \begin{pmatrix} u^{(1)}, \ldots, u^{(\texttt{min}(m,n))} & \\ & I_s \end{pmatrix} \begin{pmatrix} \chi_{1,i} \\ \vdots \\ \chi_{\texttt{min}(m,n),i} \\ \widehat{y}^{(i)} \end{pmatrix},$$

where the scalars $\chi_{j,i}$ are equal to

$$\chi_{j,i} = -\left(Ev^{(j)}\right)^H \widehat{y}^{(i)} \frac{\sigma_j}{\sigma_j^2 - \widehat{\sigma}_i^2}.$$

Proof. Deferred to the Supplementary Material. □

Proposition 2 suggests that setting $Z = \begin{pmatrix} u^{(1)}, \ldots, u^{(\texttt{min}(m,n))} & \\ & I_s \end{pmatrix}$ should lead to an exact (in the absence of round-off errors) computation of $\widehat{u}^{(i)}$. In practice, we only have access to the k leading left singular vectors of B, $u^{(1)}, \ldots, u^{(k)}$. The following proposition suggests that the distance between $\widehat{u}^{(i)}$ and the range space of $Z = \begin{pmatrix} u^{(1)}, \ldots, u^{(k)} & \\ & I_s \end{pmatrix}$ is at worst proportional to the ratio $\frac{\sigma_{k+1}}{\sigma_{k+1}^2 - \widehat{\sigma}_i^2}$.

Proposition 3. *Let matrix Z in Algorithm 1 be defined as*

$$Z = \begin{pmatrix} u^{(1)}, \ldots, u^{(k)} & \\ & I_s \end{pmatrix},$$

and set $\gamma = \left\| E^H \widehat{y}^{(i)} \right\|$.

Then, for any $i = 1, \ldots, k$:

$$\min_{z \in \texttt{range}(Z)} \|\widehat{u}^{(i)} - z\| \leq \left| \frac{\gamma \sigma_{k+1}}{\sigma_{k+1}^2 - \widehat{\sigma}_i^2} \right|.$$

Proof. Deferred to the Supplementary Material. □

Proposition 3 implies that left singular vectors associated with larger singular values of A are likely to be approximated more accurately.

4.1.1. The structure of matrix $Z^H A$.

Setting the projection matrix Z as in Proposition 3 gives

$$Z^H A = \left(V_k \Sigma_k \ E^H\right)^H.$$

Therefore, each MV product with matrix $Z^H A A^H Z$ requires two MV products with matrices Σ_k, V_k and E, for a total cost of about $4(nk + \texttt{nnz}(E))$ FLOPS. Moreover, we have $\texttt{nr}(Z^H) = s + k$, and thus a rough estimate of the cost of Step 5 in Algorithm 1 is $4(nk + \texttt{nnz}(E))\delta + 2(s+k)\delta^2$ FLOPS.

4.2. Exploiting resolvent expansions.

The choice of Z presented in Section 4.1 will compute the exact k leading singular triplets of A provided that the rank of B is exactly k. Nonetheless, when the rank of B is larger than k and the singular values $\sigma_{k+1}, \ldots, \sigma_{\texttt{min}(m,n)}$ are not small, the accuracy of the approximation returned

by Algorithm 1 might not be high enough. This section presents an approach to enhance the projection matrix Z.

Recall that the top part of $\widehat{u}^{(i)}$ is equal to $\widehat{f}^{(i)} = -(BB^H - \widehat{\sigma}_i^2 I_m)^{-1} BE^H \widehat{y}^{(i)}$. In practice, even if we knew the unknown quantities $\widehat{\sigma}_i^2$ and $\widehat{y}^{(i)}$, the application of matrix $(BB^H - \widehat{\sigma}_i^2 I_m)^{-1}$ for each $i = 1, \ldots, k$, is too costly. The idea presented in this section considers the approximation of $(BB^H - \widehat{\sigma}_i^2 I_m)^{-1}$ by $(BB^H - \lambda I_m)^{-1}$ for some fixed scalar $\lambda \in \mathbb{R}$.

Lemma 1. *Let*

$$B(\lambda) = (I_m - U_k U_k^H)(BB^H - \lambda I_m)^{-1}$$

such that $\lambda > \widehat{\sigma}_k^2$. Then, we have that for any $i = 1, \ldots, k$:

$$B(\widehat{\sigma}_i^2) = B(\lambda) \sum_{\rho=0}^{\infty} \left[(\widehat{\sigma}_i^2 - \lambda) B(\lambda) \right]^{\rho}.$$

Proof. Deferred to the Supplementary Material. □

Clearly, the closer λ is to $\widehat{\sigma}_i^2$, the more accurate the approximation in Lemma 1 should be. We can now provide an expression for $\widehat{u}^{(i)}$ similar to that in Proposition 2.

Proposition 4. *The left singular vector $\widehat{u}^{(i)}$ associated with singular value $\widehat{\sigma}_i$ is equal to*

$$\widehat{u}^{(i)} = \begin{pmatrix} u^{(1)}, \ldots, u^{(k)} & \\ & I_s \end{pmatrix} \begin{pmatrix} \chi_{1,i} \\ \vdots \\ \chi_{k,i} \\ \widehat{y}^{(i)} \end{pmatrix} - \begin{pmatrix} B(\lambda) \sum_{\rho=0}^{\infty} \left[(\widehat{\sigma}_i^2 - \lambda) B(\lambda) \right]^{\rho} BE^H \widehat{y}^{(i)} \\ \\ \end{pmatrix}.$$

Proof. Deferred to the Supplementary Material. □

Proposition 4 suggests a way to enhance the projection matrix Z shown in Proposition 3. For example, we can approximate $B(\lambda) \sum_{\rho=0}^{\infty} \left[(\widehat{\sigma}_i^2 - \lambda) B(\lambda) \right]^{\rho}$ by $B(\lambda)$, which gives the following bound for the distance of $\widehat{u}^{(i)}$ from $\texttt{range}(Z)$.

Proposition 5. *Let matrix Z in Algorithm 1 be defined as*

$$Z = \begin{pmatrix} u^{(1)}, \ldots, u^{(k)} & -B(\lambda) BE^H \\ & I_s \end{pmatrix}$$

and set $\gamma = \left\| E^H \widehat{y}^{(i)} \right\|$.

Then, for any $\lambda \geq \widehat{\sigma}_1^2$ and $i = 1, \ldots, k$:

$$\min_{z \in \texttt{range}(Z)} \|\widehat{u}^{(i)} - z\| \leq \left| \frac{\gamma \sigma_{k+1} (\widehat{\sigma}_i^2 - \lambda)}{(\sigma_{k+1}^2 - \widehat{\sigma}_i^2)(\sigma_{k+1}^2 - \lambda)} \right|.$$

Proof. Deferred to the Supplementary Material. □

Compared to the bound shown in Proposition 3, the bound in Proposition 5 is multiplied by $\dfrac{\widehat{\sigma}_i^2 - \lambda}{\sigma_{k+1}^2 - \lambda}$. In practice, due to cost considerations, we choose a single value of λ that is more likely to satisfy the above consideration, e.g., $\lambda \geq \widehat{\sigma}_1$.

4.2.1. Computing the matrix $B(\lambda)BE^H$.

The construction of matrix Z shown in Lemma 5 requires the computation of the matrix $-B(\lambda)BE^H$. The latter is equal to the matrix X that satisfies the equation

$$-(BB^H - \lambda I_m) X = (I_m - U_k U_k^H) BE^H. \quad (1)$$

The eigenvalues of the matrix $-(BB^H - \lambda I_m)$ are equal to $\{\lambda - \sigma_i^2\}_{i=1,\ldots,m}$, and for any $\lambda > \sigma_1^2$, the matrix $-(BB^H - \lambda I_m)$ is positive definite. It is thus possible to compute X by repeated applications of the Conjugate Gradient method.[1]

Proposition 6. *Let $K = -(BB^H - \lambda I_m)$ and $\|e_j\|_K$ denote[2] the K-norm of the error after j iterations of the Conjugate Gradient method applied to the linear system $-(BB^H - \lambda I_m)x = b$, where $b \in \texttt{range}\left((I_m - U_k U_k^H) BE^H\right)$. Then,*

$$\|e_j\|_K \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^j \|e_0\|_K,$$

where $\kappa = \dfrac{\sigma_{\texttt{min}(m,n)}^2 - \lambda}{\sigma_{k+1}^2 - \lambda}$ is the effective condition number and $\lambda > \widehat{\sigma}_1^2 \geq \sigma_1^2$.

Proof. Since $b \in \texttt{range}((I_m - U_k U_k^H) BE^H)$, the vector x satisfies the equation

$$-(I_m - U_k U_k^H)(BB^H - \lambda I_m)(I_m - U_k U_k^H) x = b. \quad (2)$$

The proof can be found in (Saad et al., 2000). □

Corollary 1. *The effective condition number satisfies the inequality $\kappa \leq \dfrac{\lambda}{\lambda - \sigma_{k+1}^2}$.*

Proposition 6 applies independently to each one of the s right-hand sides in (1). Assuming that the matrix $(I_m - U_k U_k^H) BE^H$ can be formed and stored, the effective condition number can be reduced even further. For example, solving (1) by the block Conjugate Gradient method leads to

[1] Linear systems with deflated right-hand sides can be also found in sparse eigenvalue algorithms such as Jacobi-Davidson (Sleijpen & Van der Vorst, 2000).

[2] We define $\|e_j\|_K = \sqrt{e_j^T K e_j}$

an effective condition number $\kappa = \frac{\sigma^2_{\texttt{min}(m,n)} - \lambda}{\sigma^2_{k+s+1} - \lambda}$ (O'Leary, 1980). Additional techniques to solve linear systems with multiple right-hand sides can be found in (Kalantzis et al., 2013; 2018; Stathopoulos & Orginos, 2010). As the value of λ increases, the effective condition number κ decreases. Thus from a convergence viewpoint, it is better to choose $\lambda \gg \widehat{\sigma}_i^2$. On the other hand, increasing λ leads to worse bounds in Proposition 5.

4.2.2. Truncating the matrix $B(\lambda)BE^H$.

When the number of right-hand sides in (1), i.e., number of rows in matrix E, is too large, an alternative is to consider $-B(\lambda)BE^H \approx X_{\lambda,r}S_{\lambda,r}Y_{\lambda,r}^H$, where $X_{\lambda,r}S_{\lambda,r}Y_{\lambda,r}^H$ denotes the rank-r truncated SVD of matrix $-B(\lambda)BE^H$. We can then replace $-B(\lambda)BE^H$ by $X_{\lambda,r}$, since $\texttt{range}\left(X_{\lambda,r}S_{\lambda,r}Y_{\lambda,r}^H\right) \subseteq \texttt{range}\left(X_{\lambda,r}\right)$. The matrix $X_{\lambda,r}$ can be approximated by randomized SVD as described in (Halko et al., 2011; Clarkson & Woodruff, 2009). This approach replaces $X_{\lambda,r}$ by the rank-r SVD of the matrix $B(\lambda)BE^HR$, where R is a real matrix with at least r columns whose entries are i.i.d. Gaussian random variables of zero mean and unit variance.

4.2.3. The structure of matrix Z^HA.

Setting the basis matrix Z as in Proposition 5 leads to

$$Z^HA = \left(V_k\Sigma_k \; B^HX_{\lambda,r} \; E^H\right)^H.$$

Each MV product with matrix Z^HAA^HZ requires two MV products with matrices Σ_k, V_k, E and $B^HX_{\lambda,r}$, for a total cost of $4(n(k+r) + \texttt{nnz}(E))$ FLOPS. Moreover, we have $\texttt{nr}(Z^H) = k+r+s$, and thus a rough estimate of the cost of Step 5 in Algorithm 1 is $4(n(k+r)+\texttt{nnz}(E))\delta + 2(s+k+r)\delta^2$ FLOPS.

5. Experiments

In this section we benchmark the performance of Algorithm 1 on problems from the areas of LSI and face recognition. The experiments were conducted in a Matlab environment (version R2020a), using 64-bit arithmetic, on a single core of a computing system equipped with an 2.5GHz Quad-Core Intel Core i7 processor and 16 GB of system memory. Due to space limitations we present results only for the case of adding new rows. Results for problems where we add new columns were very similar and thus omitted. Open source implementation of the method is available at: https://github.com/nikolakopoulos/TruncatedSVDupdates.

Our experiments are organized as follows. Let $A \in \mathbb{C}^{m\times n}$ denote the matrix whose rank-k SVD we seek to compute. We first load the leading $\mu \in \mathbb{N}$ rows of A and set this submatrix equal to B, i.e., $B = A(1:\mu,:)$ in Matlab-like notation. Once we compute the rank-k SVD of matrix B, we update it by adding the remaining $m-\mu$ rows of A in $\phi \in \mathbb{N}$ batches, one batch at a time. The number of rows in each batch is equal to $\tau = \lfloor (m-\mu)/\phi \rfloor$ with the exception of the very last batch which is generally slightly smaller in size. In matrix notation, during the first update we approximate the k leading singular triplets of matrix $A^{(1)} = \begin{pmatrix} B \equiv A(1:\mu,:) \\ E \equiv A(\mu+1:\mu+\tau,:) \end{pmatrix}$, then of matrix $A^{(2)} = \begin{pmatrix} B \equiv A^{(1)} \\ E \equiv A(\mu+\tau+1:\mu+2\tau,:) \end{pmatrix}$, etc. The default values used throughout our experiments are $\mu = \lceil m/10 \rceil$ and $\phi = 10$, i.e., the initial matrix B is formed by the leading 10% of the rows matrix A, and the remaining 90% of rows is added in ten batches, with each batch roughly representing 9% of the total rows of matrix A.

Table 2. Text mining test matrices (obtained by http://web.eecs.utk.edu/research/lsi/ and https://github.com/ZJULearning/MatlabFunc).

Matrix	# of rows	# of columns	avg. nnz
MED	5,735	1,033	8.9
CRAN	4,563	1,398	17.8
CISI	5,544	1,460	12.2
Reuters-21578	18,933	8,293	20.6

5.1. Latent semantic indexing.

In this section we test the qualitative performance of Algorithm 1 when applied to updating problems in LSI. Unless mentioned otherwise, we will set the projection matrix Z as in Proposition 3. As a baseline, we consider the SVD updating algorithm of Zha and Simon (Zha & Simon, 1999), a well-known algorithm for updating problems in LSI, which we outline in detail in Section 2.1. We also consider the modification of the latter suggested in (Vecharynski & Saad, 2014) in which the QR factorization of $(I_s - V_kV_k^H)E^H$ is replaced by a rank-l partial SVD. Based on a back-of-the-envelope profiling and the settings suggested in (Vecharynski & Saad, 2014), we set $l = 10$.

Table 2 lists the test matrices considered throughout our experiments along with their dimensions and average number of nonzero entries per row. These test matrices represent term-document matrices and are commonly used to benchmark the performance of text mining techniques. The first three collections are equipped with a set of queries and a corresponding list of relevant documents. Ideally, the rank-k SVD of the term-document matrices should create a model which, for each query, assigns the highest similarity scores to the true relevant documents.

Table 3. Average precision ("prec"), mean-squared error ("MSE"), and corresponding wall-clock times ("time").

		MED			CRAN			CISI		
	Method	prec	time	MSE	prec	time	MSE	prec	time	MSE
$k = 25$	ZhaSimon	38%	1.11	$7.3\text{e}-3$	61%	0.70	$1.8\text{e}-2$	59%	1.06	$5.9\text{e}-3$
	Alg. 1	**59%**	**0.26**	$\mathbf{1.3\text{e}-3}$	**81%**	**0.24**	$\mathbf{2.1\text{e}-3}$	**89%**	**0.30**	$\mathbf{5.6\text{e}-4}$
$k = 50$	ZhaSimon	40%	1.15	$1.0\text{e}-2$	62%	0.76	$2.3\text{e}-2$	60%	1.11	$7.2\text{e}-3$
	Alg. 1	**66%**	**0.51**	$\mathbf{1.7\text{e}-3}$	**85%**	**0.49**	$\mathbf{2.3\text{e}-3}$	**91%**	**0.59**	$\mathbf{5.1\text{e}-4}$
$k = 100$	ZhaSimon	42%	1.41	$1.4\text{e}-2$	65%	**0.96**	$2.9\text{e}-2$	61%	**1.37**	$9.8\text{e}-3$
	Alg. 1	**75%**	**1.26**	$\mathbf{1.8\text{e}-3}$	**89%**	1.27	$\mathbf{2.8\text{e}-3}$	**94%**	1.44	$\mathbf{9.2\text{e}-4}$

Table 3 lists the average precision returned by Algorithm 1 and the algorithm described in (Zha & Simon, 1999), as $k = 25$, $k = 50$, and $k = 100$. These results were computed using the 11-point interpolated precision (Kolda & O'leary, 1998). Updating the SVD by Algorithm 1 leads to higher accuracy compared to (Zha & Simon, 1999), regardless of the value of k. Moreover, Algorithm 1 is generally faster, especially for lower values of k. The qualitative results returned by the algorithm in (Vecharynski & Saad, 2014) are omitted since they were similar with those returned by (Zha & Simon, 1999). Finally, we list the mean-squared error (MSE) between the rank-k truncated SVD $A_k = \widehat{U}_k\widehat{\Sigma}_k\widehat{V}_k^H$ and its approximation $\overline{U}_k\overline{\Sigma}_k\overline{V}_k^H$ returned by both Algorithm 1 and (Zha & Simon, 1999).

Figure 1 plots the precision/recall curves produced by Algorithm 1 and the algorithm in (Zha & Simon, 1999) for datasets "MED", "CRAN", and "CISI". In the context of information retrieval, precision is a measure of output relevancy, while recall is a measure of how many correctly relevant results are returned. Algorithm 1 leads to better qualitative results for all datasets as $k = \{25, 50\}$. The precision/recall curves produced by Algorithm 1 as $k = \{25, 50, 100\}$ are also plotted together so as to highlight the effects of the value of k (we omit dataset "CISI" for reasons of economy of space).

Figure 2 plots the wall-clock time of Algorithm 1 and the algorithms in (Zha & Simon, 1999) and (Vecharynski & Saad, 2014), when applied to dataset "Reuters". For the option of matrix Z listed in Proposition 5, we set $\lambda = 1.01\widehat{\sigma}_1^2$, and the matrix $X_{\lambda,r}$ is built by applying one iteration of block Conjugate Gradient to approximate matrix $B(\lambda)BE^HR$, where R is an i.i.d. matrix real matrix with at $2r$ columns, and $r = 10$. In the leftmost plot we vary $k = \{25, 50, 75, 100, 125\}$ and fix $\phi = 10$, while in the middle plot we vary $\phi = \{2, 4, 6, 8, 10\}$ and fix $k = 50$. Consider first the case where we increase k. Algorithm 1 where Z is built as in Proposition 3 starts as the fastest algorithm but loses ground to the algorithm in (Vecharynski & Saad, 2014). Nonetheless, the latter can be slower than Algorithm 1 when we increase the number 'l' of computed singular triplets of matrix $(I_n - VV^H)E^H$. In addition, the accuracy in the approximation of the leading singular triplets of A returned by the algorithms in (Zha & Simon, 1999) and (Vecharynski & Saad, 2014) is typically inferior to that returned by Algorithm 1; we will return to this shortly. Moving on to the middle plot, increasing ϕ has a positive effect on methods such as those proposed in (Zha & Simon, 1999) and (Vecharynski & Saad, 2014); especially the former, since its complexity depends cubically on the number of rows of matrix E. Algorithm 1 becomes relatively slower, since each pass requires a Matrix-MultiVector product with matrix A. On the other hand, smaller values of ϕ, i.e., "thicker" updates, heavily favor Algorithm 1. Finally, the rightmost plot shows the residual norm in the approximation of the $k = 50$ leading singular triplets. In this particular example, Algorithm 1 achieves the same level of accuracy regardless of the choice of Z. On the other hand, the accuracy achieved by the algorithms in (Zha & Simon, 1999) and (Vecharynski & Saad, 2014) is rather low, indicating that they might not be appropriate for general-purpose updating problems.

Similarly, Figures 3 and 4 plots the relative singular value error and residual norm (scaled by the corresponding singular value approximation) returned by applying Algorithm 1 to datasets "MED" and "CRAN" for the case $k = 50$. To monitor the deterioration of accuracy as more updates are added into the system, we plot the relative errors and residual norms after the first, fifth, and tenth update. The accuracy marked by the tenth update is the accuracy for the singular triplets of the entire term-document matrix. Setting Z as in Proposition 5, we see that accuracy deteriorates at a slower. This is a behavior we verified for all datasets.

Table 4 lists the relative error and residual norm associated with the approximation of the singular triplet $(\hat{\sigma}_{50}, \hat{u}^{(50)}, \hat{v}^{(50)})$ as r varies from ten to fifty in increments of ten. As a reference, we list the same quantity for the case $Z = \begin{pmatrix} U_k & \\ & I_s \end{pmatrix}$. As expected, enhancing the projection matrix Z by $X_{\lambda,r}$ leads to higher accuracy, especially for higher values of r.

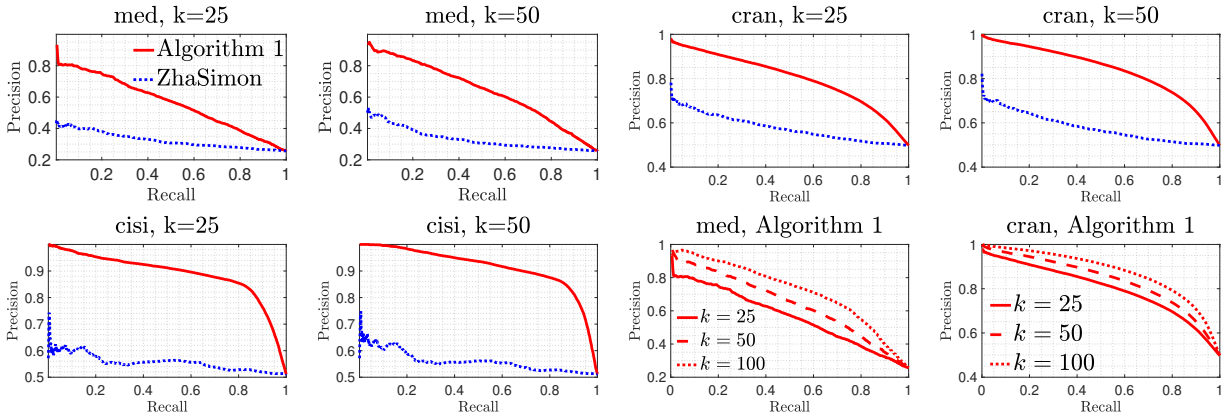


Figure 1. Precision/recall curves produced by Algorithm 1 and the algorithm described in (Zha & Simon, 1999) as $k = 25$ and $k = 50$. We also plot the precision/recall curves produced by Algorithm 1 as k varies.

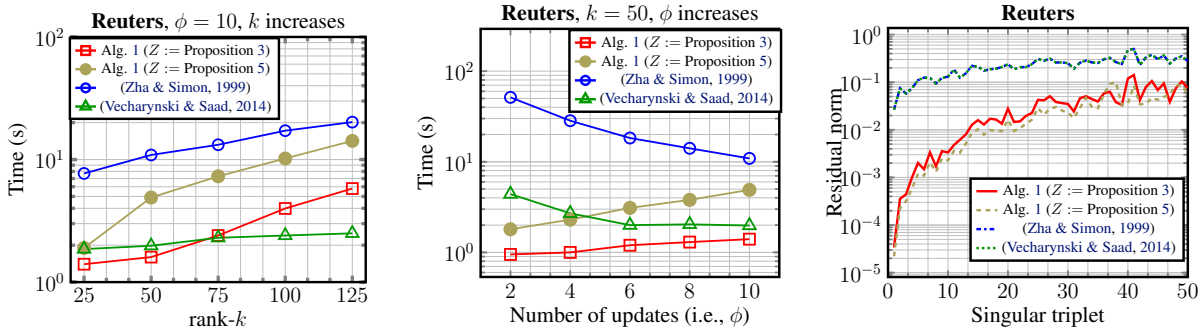


Figure 2. Left plot: wall-clock times as $k = \{25, 50, 75, 100, 125\}$. Middle plot: wall-clock times as the bottom $m - \mu$ rows of "Reuters" dataset are added sequentially in $\phi = \{2, 4, 6, 8, 10\}$ batches. Right plot: residual norm of the approximation of the $k = 50$ leading singular triplets of "Reuters" ($\phi = 10$).

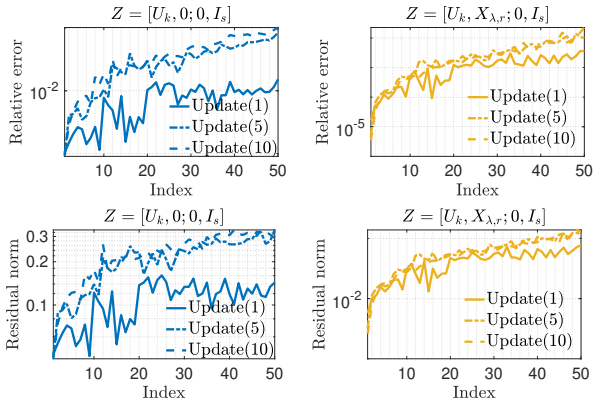


Figure 3. Approximation of the leading $k = 50$ singular triplets by Algorithm 1 (dataset "MED") for both options of matrix Z.

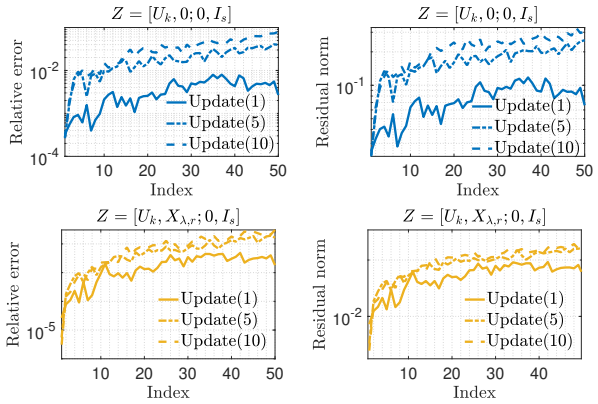


Figure 4. Approximation of the leading $k = 50$ singular triplets by Algorithm 1 (dataset "CRAN") for both options of matrix Z.

5.2. Eigenfaces

In this section we consider SVD updating problems in the context of face recognition. In particular, we consider the *Eigenfaces* technique (Turk & Pentland, 1991; Sirovich & Kirby, 1987), and replace the standard approach to compute the principal components (i.e., by computing eigenvectors of the covariance matrix) by our SVD updating scheme. Updating algorithms might be useful when the cost of standard PCA is too high, or the images dataset is so large that

Table 4. Relative error and residual norm associated with the approximation of the singular triplet $(\widehat{\sigma}_{50}, \widehat{u}^{(50)}, \widehat{v}^{(50)})$.

		MED		CRAN		CISI	
	r	err.	res.	err.	res.	err.	res.
$Z = \begin{pmatrix} U_k & X_{\lambda,r} \\ & I_s \end{pmatrix}$	$r = 10$	0.036	0.234	0.026	0.176	0.025	0.214
	$r = 20$	0.031	0.184	0.021	0.155	0.023	0.189
	$r = 30$	0.021	0.114	0.017	0.134	0.017	0.161
	$r = 40$	0.009	0.091	0.013	0.111	0.012	0.134
	$r = 50$	0.004	0.053	0.007	0.098	0.007	0.081
$Z = \begin{pmatrix} U_k & \\ & I_s \end{pmatrix}$	N/A	0.045	0.269	0.045	0.199	0.287	0.250

Table 5. Classification mean error-rate and standard deviation.

	Yale		AT&T	
Method	$k = 25$	$k = 50$	$k = 25$	$k = 50$
ZhaSimon	38.49 ± 6.89	36.89 ± 7.08	37.78 ± 6.01	35.20 ± 5.40
Alg. 1	29.60 ± 5.99	27.56 ± 5.86	6.17 ± 2.45	5.90 ± 2.89
Standard PCA	29.91 ± 5.68	27.78 ± 5.90	6.30 ± 2.40	5.60 ± 2.47

only a small portion of it can fit in system memory. Our idea is to avoid forming the covariance matrix, and instead apply Algorithm 1 to the training data after subtracting their mean. The mean of each individual pixel across all images in the training dataset is computed by performing a single pass over the training dataset. Once the set of eigenfaces is determined by Algorithm 1, test images are classified by: (i) subtracting the mean and projecting them onto the k computed eigenfaces, and (ii) classifying the test image according to the class of those images in the training dataset which are nearest neighbors of the test image in the projection subspace. For this task we use ρ-nearest neighbors with $\rho = 5$ (Fix, 1985). The number of computed eigenfaces is set as $k = 25$ and $k = 50$. A detailed description of the *Eigenfaces* technique is outlined in the Supplementary Material.

Our experiments are performed on the Yale and AT&T face databases.[3] The Yale face database consists of 165 gray scale images of 15 individuals, each individual has 11 images, obtained under different lighting conditions and facial expression. Similarly, the AT&T face database consists of a total of 400 face images of 40 different people, captured at different expressions and facial appearances. In both datasets, the size of each image is 64×64 pixels.

Table 5 lists the classification mean error-rate averaged over fifty splits, where each split contains eight images per individual in the training set. The mean error-rate achieved by Algorithm 1 is almost identical to that obtained by standard PCA, which is ideal since Algorithm 1 operates on the dataset in pieces and thus is more flexible. In contrast, results obtained using the algorithm in (Zha & Simon, 1999) are less accurate.

[3]These files were obtained in a Matlab-ready format from `http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html`; see (Cai et al., 2007a;b)

6. Conclusion

This paper presented an algorithm to update the rank-k truncated SVD of matrices subject to a periodic addition of rows/columns. The proposed algorithm builds on a projection viewpoint and aims on building a pair of subspaces which approximately include the linear span of the k left/right leading singular vectors of the updated matrix. We considered two different options to set these subspaces, where the first approach aims on reducing the wall-clock time of current state-of-the-art, while the second one aims on increasing the accuracy of the returned approximate truncated SVD. Experiments performed on matrices stemming from applications in LSI and face recognition verified the effectiveness of the proposed scheme in terms of speed as well as numerical and qualitative accuracy.

Acknowledgments

We are grateful to the reviewers for their efforts and comments. We thank Jie Chen for sharing with us the Matlab implementation of the algorithm described in (Chen & Saad, 2009). Part of this implementation was used to compute the precision and recall quantities listed in our paper.

References

Baglama, J. and Reichel, L. Augmented implicitly restarted Lanczos bidiagonalization methods. *SIAM Journal on Scientific Computing*, 27(1):19–42, 2005.

Baker, C. G., Gallivan, K. A., and Van Dooren, P. Low-rank incremental methods for computing dominant singular subspaces. *Linear Algebra and its Applications*, 436(8): 2866–2888, 2012.

Berry, M. W., Dumais, S. T., and O'Brien, G. W. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

Brand, M. Fast online SVD revisions for lightweight recommender systems. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pp. 37–46. SIAM, 2003.

Cai, D., He, X., and Han, J. Spectral regression for efficient regularized subspace learning. In *Proc. Int. Conf. Computer Vision (ICCV'07)*, 2007a.

Cai, D., He, X., Hu, Y., Han, J., and Huang, T. Learning a spatially smooth subspace for face recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition Machine Learning (CVPR'07)*, 2007b.

Chen, J. and Saad, Y. Lanczos vectors versus singular vectors for effective dimension reduction. *IEEE Transactions on Knowledge and Data Engineering*, 21(8):1091–1103, 2008.

Chen, J. and Saad, Y. Divide and conquer strategies for effective information retrieval. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pp. 449–460. SIAM, 2009.

Clarkson, K. L. and Woodruff, D. P. Numerical linear algebra in the streaming model. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 205–214, 2009.

Cremonesi, P., Koren, Y., and Turrin, R. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pp. 39–46, 2010.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

Fix, E. *Discriminatory analysis: nonparametric discrimination, consistency properties*, volume 1. USAF school of Aviation Medicine, 1985.

Golub, G. and Kahan, W. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224, 1965.

Gu, M., Stanley, and Eisenstat, S. C. A stable and fast algorithm for updating the singular value decomposition. Technical report, 1994.

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

Hernandez, V., Roman, J. E., and Vidal, V. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):351–362, 2005.

Horesh, L., Conn, A. R., Jimenez, E. A., and van Essen, G. M. Reduced space dynamics-based geo-statistical prior sampling for uncertainty quantification of end goal decisions. In *Numerical Analysis and Optimization*, pp. 191–221. Springer, 2015.

Jia, Z. and Stewart, G. An analysis of the Rayleigh-Ritz method for approximating eigenspaces. *Mathematics of computation*, 70(234):637–647, 2001.

Kalantzis, V., Bekas, C., Curioni, A., and Gallopoulos, E. Accelerating data uncertainty quantification by solving linear systems with multiple right-hand sides. *Numerical Algorithms*, 62(4):637–653, 2013.

Kalantzis, V., Malossi, A. C. I., Bekas, C., Curioni, A., Gallopoulos, E., and Saad, Y. A scalable iterative dense linear system solver for multiple right-hand sides in data analytics. *Parallel Computing*, 74:136–153, 2018.

Kolda, T. G. and O'leary, D. P. A semidiscrete matrix decomposition for latent semantic indexing information retrieval. *ACM Transactions on Information Systems (TOIS)*, 16(4):322–346, 1998.

Moonen, M., Van Dooren, P., and Vandewalle, J. A singular value decomposition updating algorithm for subspace tracking. *SIAM Journal on Matrix Analysis and Applications*, 13(4):1015–1038, 1992.

Nakatsukasa, Y. Accuracy of singular vectors obtained by projection-based SVD methods. *BIT Numerical Mathematics*, 57(4):1137–1152, 2017.

Nikolakopoulos, A. N., Kalantzis, V., Gallopoulos, E., and Garofalakis, J. D. Eigenrec: generalizing PureSVD for effective and efficient top-N recommendations. *Knowledge and Information Systems*, 58(1):59–81, 2019.

O'Leary, D. P. The block Conjugate Gradient algorithm and related methods. *Linear Algebra and its Applications*, 29: 293–322, 1980.

Saad, Y. *Numerical methods for large eigenvalue problems: revised edition*. SIAM, 2011.

Saad, Y., Yeung, M., Erhel, J., and Guyomarc'h, F. A deflated version of the Conjugate Gradient algorithm. *SIAM Journal on Scientific Computing*, 21(5):1909–1926, 2000.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth international conference on computer and information science*, volume 1, pp. 27–8. Citeseer, 2002.

Sirovich, L. and Kirby, M. Low-dimensional procedure for the characterization of human faces. *Josa a*, 4(3): 519–524, 1987.

Sleijpen, G. L. and Van der Vorst, H. A. A jacobi–davidson iteration method for linear eigenvalue problems. *SIAM review*, 42(2):267–293, 2000.

Stathopoulos, A. and Orginos, K. Computing and deflating eigenvalues while solving multiple right-hand side linear systems with an application to quantum chromodynamics. *SIAM Journal on Scientific Computing*, 32(1):439–462, 2010.

Turk, M. A. and Pentland, A. P. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*, pp. 586–587. IEEE Computer Society, 1991.

Ubaru, S. and Saad, Y. Sampling and multilevel coarsening algorithms for fast matrix approximations. *Numerical Linear Algebra with Applications*, 26(3):e2234, 2019.

Ubaru, S., Mazumdar, A., and Saad, Y. Low rank approximation using error correcting coding matrices. In *International Conference on Machine Learning*, pp. 702–710, 2015.

Ubaru, S., Seghouane, A.-K., and Saad, Y. Find the dimension that counts: Fast dimension estimation and krylov pca. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pp. 720–728. SIAM, 2019.

Vecharynski, E. and Saad, Y. Fast updating algorithms for latent semantic indexing. *SIAM Journal on Matrix Analysis and Applications*, 35(3):1105–1131, 2014.

Wu, L. and Stathopoulos, A. A preconditioned hybrid SVD method for accurately computing singular triplets of large matrices. *SIAM Journal on Scientific Computing*, 37(5): S365–S388, 2015.

Yamazaki, I., Tomov, S., and Dongarra, J. Sampling algorithms to update truncated SVD. In *2017 IEEE International Conference on Big Data (Big Data)*, pp. 817–826. IEEE, 2017.

Zha, H. and Simon, H. D. On updating problems in latent semantic indexing. *SIAM Journal on Scientific Computing*, 21(2):782–791, 1999.

Zha, H. and Zhang, Z. Matrices with low-rank-plus-shift structure: partial SVD and latent semantic indexing. *SIAM Journal on Matrix Analysis and Applications*, 21 (2):522–536, 2000.