A Scalable CUR Matrix Decomposition Algorithm: Lower Time Complexity and Tighter Bound

Authored by:

Zhihua Zhang
Shusen Wang

Abstract

The CUR matrix decomposition is an important extension of Nystr?m approximation to a general matrix. It approximates any data matrix in terms of a small number of its columns and rows. In this paper we propose a novel randomized CUR algorithm with an expected relative-error bound. The proposed algorithm has the advantages over the existing relative-error CUR algorithms that it possesses tighter theoretical bound and lower time complexity, and that it can avoid maintaining the whole data matrix in main memory. Finally, experiments on several real-world datasets demonstrate significant improvement over the existing relative-error algorithms.

1 Paper Body

Large-scale matrices emerging from stocks, genomes, web documents, web images and videos everyday bring new challenges in modern data analysis. Most efforts have been focused on manipulating, understanding and interpreting large-scale data matrices. In many cases, matrix factorization methods are employed to construct compressed and informative representations to facilitate computation and interpretation. A principled approach is the truncated singular value decomposition (SVD) which finds the best low-rank approximation of a data matrix. Applications of SVD such as eigenface [20, 21] and latent semantic analysis [4] have been illustrated to be very successful. However, the basis vectors resulting from SVD have little concrete meaning, which makes it very difficult for us to understand and interpret the data in question. An example in [10, 19] has well ? shown this viewpoint; that is, the vector [(1/2)age ? (1/ 2)height + (1/2)income], the sum of the significant uncorrelated features from a dataset of people?s features, is not particularly informative. The authors of [17] have also claimed: ?it would be interesting to try to find basis vectors for all experiment vectors, using actual experiment vectors and not artificial bases that offer little insight.? Therefore, it is of great interest to represent a data matrix in terms of a small number of actual columns and/or actual rows of the matrix. The CUR matrix decomposition provides such techniques, and it has been shown to be very useful in high dimensional data analysis [19]. Given a matrix A, the CUR technique selects a subset of columns of A to construct a matrix C and a subset of rows of A to construct a matrix R, and ? = CUR best approximates A. The typical CUR algorithms [7, computes a matrix U such that A 8, 10] work in a two-stage manner. Stage 1 is a standard column selection procedure, and Stage 2 does row selection from A and C simultaneously. Thus Stage 2 is more complicated than Stage 1. The CUR matrix decomposition problem is widely studied in the literature [7, 8, 9, 10, 12, 13, 16, 18, 19, 22]. Perhaps the most widely known work on the CUR problem is [10], in which the authors devised a randomized CUR algorithm called the subspace sampling algorithm. Particularly, the algorithm has (1 + ?) relative-error ratio with high probability (w.h.p.). 1

Unfortunately, all the existing CUR algorithms require a large number of columns and rows to be chosen. For example, for an m ? n matrix A and a target rank k ? min{m, n}, the state-ofthe-art CUR algorithm ? the subspace sampling algorithm in [10] ? requires exactly O(k 4 ??6) rows or O(k??4 log2 k) rows in expectation to achieve (1 + ?) relative-error ratio w.h.p. Moreover, the computational cost of this algorithm is at least the cost of the truncated SVD of A, that is, O(min{mn2 , nm2 }).1 The algorithms are therefore impractical for large-scale matrices. In this paper we develop a CUR algorithm which beats the state-of-the-art algorithm in both theory and experiments. In particular, we show in Theorem 5 a novel randomized CUR algorithm with lower time complexity and tighter theoretical bound in comparison with the state-of-the-art CUR algorithm in [10]. The rest of this paper is organized as follows. Section 3 introduces several existing column selection algorithms and the state-of-the-art CUR algorithm. Section 4 describes and analyzes our novel CUR algorithm. Section 5 empirically compares our proposed algorithm with the state-of-the-art algorithm.

2

Notations

For a matrix A = [aij] ? Rm?n , let a(i) be its i-th row and aj be its j-th column. Let ?A?1 = ? ? 2 1/2 be the Frobenius norm, and ?A?2 be the spectral i,j —aij — be the ?1 -norm, ?A?F = (i,j aij) norm. Moreover, let Im denote an m ? m identity matrix, and 0mn denotes an m ? n zero matrix. ?? T T T T = UA,k ?A,k VA,k + UA,k? ?A,k? VA,k? be the Let A = UA ?A VA = i=0 ?A,i uA,i vA,i SVD of A, where ? = rank(A), and UA,k , ?A,k , and VA,k correspond to the top k singular values. T T We denote Ak = UA,k ?A,k VA,k . Furthermore, let A? = UA,? ??1 A,? VA,? be the Moore-Penrose inverse of A [1].

3 Related Work Section 3.1 introduces several relative-error column selection algorithms related to this work. Section 3.2 describes the state-of-the-art CUR algorithm in [10]. Section 3.3 discusses the connection between the column selection problem and the CUR problem. 3.1

Relative-Error Column Selection Algorithms

Given a matrix A ? Rm?n , column selection is a problem of selecting c columns of A to construct C ? Rm?c to minimize ?A ? CC? A?F . Since there are (nc) possible choices of constructing C, so selecting the best subset is a hard problem. In recent years, many polynomial-time approximate algorithms have been proposed, among which we are particularly interested in the algorithms with relative-error bounds; that is, with c ? k columns selected from A, there is a constant ? such that ?A ? CC? A?F ? ??A ? Ak ?F . We call ? the relative-error ratio. We now present some recent results related to this work. We first introduce a recently developed deterministic algorithm called the dual set sparsification proposed in [2, 3]. We show their results in Lemma 1. Furthermore, this algorithm is a building block of some more powerful algorithms (e.g., Lemma 2), and our novel CUR algorithm also relies on this algorithm. We attach the algorithm in Appendix A. Lemma 1 (Column Selection via Dual Set Sparsification Algorithm). Given a matrix A ? Rm?n of rank ? and a target rank k (¡ ?), there exists a deterministic algorithm to select c (¿ k) columns of A and form a matrix C ? Rm?c such that ?

1

? ? ? Ak .

A ? CC A ? 1 +

A F F (1 ? k/c)2 1 Although some partial SVD algorithms, such as Krylov subspace methods, require only O(mnk) time, they are all numerical unstable. See [15] for more discussions.

2

Moreover, the matrix C can be computed in TVA,k +O(mn+nck 2), where TVA,k is the time needed to compute the top k right singular vectors of A. There are also a variety of randomized column selection algorithms achieving relative-error bounds in the literature: [3, 5, 6, 10, 14]. An randomized algorithm in [2] selects only c = 2k ? (1 + o(1)) columns to achieve the expected relative-error ratio (1 + ?). The algorithm is based on the approximate SVD via random projection [15], the dual set sparsification algorithm [2], and the adaptive sampling algorithm [6]. Here we present the main results of this algorithm in Lemma 2. Our proposed CUR algorithm is motivated by and relies on this algorithm. Lemma 2 (Near-Optimal Column Selection Algorithm). Given a matrix A ? Rm?n of rank ?, a target rank k (2 ? k ¡ ?), and 0 ¡ ? ¡ 1, there exists a randomized algorithm to select at most) 2k (1 + o(1) c= ? columns of A to form a matrix C ? Rm?c such that E2 ?A ? CC? A?F ? E?A ? CC? A?2F ? (1 + ?)?A ? Ak ?2F , where the expectations are taken w.r.t. C. Furthermore, the matrix C can be computed in O((mnk+ nk 3)??2/3). 3.2

The Subspace Sampling CUR Algorithm

Drineas et al. [10] proposed a two-stage randomized CUR algorithm which has a relative-error bound w.h.p. Given a matrix A ? Rm?n and a target rank k, in the first stage the algorithm chooses exactly c = O(k 2 ??2 log ? ?1) columns (or c = O(k??2 log k log ? ?1) in expectation) of A to construct C ? Rm?c ; in the second stage it chooses exactly r = O(c2 ??2 log ? ?1) rows (or r = O(c??2 log c log ? ?1) in expectation) of A and C simultaneously to construct

R and U. With probability at least 1 ? ?, the relative-error ratio is 1 + ?. The computational cost is dominated by the truncated SVD of A and C. Though the algorithm is ?-optimal with high probability, it requires too many rows get chosen: at least r = O(k??4 log2 k) rows in expectation. In this paper we seek to devise an algorithm with mild requirement on column and row numbers. 3.3

Connection between Column Selection and CUR Matrix Decomposition

The CUR problem has a close connection with the column selection problem. As aforementioned, the first stage of existing CUR algorithms is simply a column selection procedure. However, the second stage is more complicated. If the second stage is na??vely solved by a column selection algorithm on AT , then the error ratio will be at least (2 + ?). For a relative-error CUR algorithm, the first stage seeks to bound a construction error ratio of ?A?CC? A?F ?A?CC? AR? R?F given C. Actually, the first ?A?Ak ?F , while the section stage seeks to bound ?A?CC? A?F stage is a special case of the second stage where C = Ak . Given a matrix A, if an algorithm solv? AR? R?F ing the second stage results in a bound ?A?CC ? ?, then this algorithm also solves the ?A?CC? A?F T column selection problem for A with an ? relative-error ratio. Thus the second stage of CUR is a generalization of the column selection problem.

4

Main Results

In this section we introduce our proposed CUR algorithm. We call it the fast CUR algorithm because it has lower time complexity compared with SVD. We describe it in Algorithm 1 and give a theoretical analysis in Theorem 5. Theorem 5 relies on Lemma 2 and Theorem 4, and Theorem 4 relies on Theorem 3. Theorem 3 is a generalization of [6, Theorem 2.1], and Theorem 4 is a generalization of [2, Theorem 5]. 3

Algorithm 1 The Fast CUR Algorithm.

( ) 1: Input: a real matrix A ? Rm?n , target rank k, ? ? (0, 1], target column number c = 2k 1 + o(1) , target ? ( ) row number r = 2c 1 + o(1) ; ? 2: // Stage 1: select c columns of A to construct C ? Rm?c ? k? ? kV ? k; 3: Compute approximate truncated SVD via random projection such that Ak ? U ? k? ? kV ? k); V1 ? columns of V ? kT ; 4: Construct U1 ? columns of (A ? U 5: Compute s1 ? Dual Set Spectral-Frobenius Sparsification Algorithm (U1 , V1 , c ? 2k/?); 6: Construct C1 ? ADiag(s1), and then delete the all-zero columns; 7: Residual matrix D ? A ? C1 C?1 A; 8: Compute sampling probabilities: pi = ?di ?22 /?D?2F , i = 1, ? ? ? , n; 9: Sampling c2 = 2k/? columns from A with probability {p1 , ? ? ? , pn } to construct C2 ; 10: // Stage 2: select r rows of A to construct R ? Rr?n ? k? ? kV ? k)T ; V2 ? columns of U ? Tk ; 11: Construct U2 ? columns of (A ? U 12: Compute s2 ? Dual Set Spectral-Frobenius Sparsification Algorithm (U2 , V2 , r ? 2c/?); 13: Construct R1 ? Diag(s2)A, and then delete the all-zero rows; 14: Residual matrix B ? A ? AR?1 R1 ; Compute qj = ?b(j) ?22 /?B?2F , j = 1, ? ? ? , m; 15: Sampling r2 = 2c/? rows from A with probability {q1 , ? ? ? , qm } to construct R2 ; 16: return C = [C1 , C2], R = [RT1 , RT2]T , and U = C? AR? .

4.1

Adaptive Sampling

The relative-error adaptive sampling algorithm is established in [6, Theorem 2.1]. The algorithm is based on the following idea: after selecting a proportion of columns from A to form C1 by an arbitrary algorithm, the algorithms randomly samples additional c2 columns according to the residual A ? C1 C?1 A. Boutsidis et al. [2] used the adaptive sampling algorithm to decrease the residual of the dual set sparsification algorithm and obtained an (1 + ?) relative-error bound. Here we prove a new bound for the adaptive sampling algorithm. Interestingly, this new bound is a generalization of the original one in [6, Theorem 2.1]. In other words, Theorem 2.1 of [6] is a direct corollary of our following theorem in which C = Ak is set. Theorem 3 (The Adaptive Sampling Algorithm). Given a matrix A ? Rm?n and a matrix C ? Rm?c such that rank(C) = rank(CC? A) = ?, (? ? c ? n), we let R1 ? Rr1 ?n consist of r1 rows of A, and define the residual B = A ? AR?1 R1 . Additionally, for i = 1, ? ? ? , m, we define pi = ?b(i) ?22 /?B?2F . We further sample r2 rows i.i.d. from A, in each trial of which the i-th row is chosen with probability pi . Let R2 ? Rr2 ?n contains the r2 sampled rows and let R = [RT1 , RT2]T ? R(r1 +r2)?n . Then the following inequality holds: ? E?A ? CC? AR? R?2F ? ?A ? CC? A?2F + ?A ? AR?1 R1 ?2F , r2 where the expectation is taken w.r.t. R2 . 4.2

The Fast CUR Algorithm

Based on the dual set sparsification algorithm of of Lemma 1 and the adaptive sampling algorithm of Theorem 3, we develop a randomized algorithm to solve the second stage of CUR problem. We present the results of the algorithm in Theorem 4. Theorem 5 of [2] is a special case of the following theorem where C = Ak . Theorem 4 (The Fast Row Selection Algorithm). Given a matrix A ? Rm?n and a matrix C ? Rm?c such that rank(C) = rank(CC? A) = ?, (? ? c ? n), and a target rank k (? ?), the r?n proposed randomized algorithm selects r = 2? , such ? (1 + o(1)) rows of A to construct R ? R that E?A ? CC? AR? R?2F ? ?A ? CC? A?2F + ??A ? Ak ?2F , where the expectation is taken w.r.t. R. Furthermore, the matrix R can be computed in O((mnk + mk 3)??2/3) time. Based on Lemma 2 and Theorem 4, here we present the main theorem for the fast CUR algorithm. 4

Table 1: A summary of the datasets. Dataset Type size Source Redrocknatural image18000 ? 4000 http://www.agarwala.org/efficient gdc/ Arcene biology 10000 ? 900 http://archive.ics.uci.edu/ml/datasets/Arcene Dexter bag of words 20000 ? 2600http://archive.ics.uci.edu/ml/datasets/Dexter

Theorem 5 (The Fast CUR Algorithm). Given a matrix A ? Rm?n and a positive integer k ? min{m, n}, the fast CUR algorithm (described in Algorithm 1) randomly selects c = 2k ? (1 + o(1)) columns of A to construct C ? Rm?c with the near-optimal column selection algorithm of Lemma 2, r?n and then selects r = 2c with the fast row selection ? (1 + o(1)) rows of A to construct R ? R algorithm of Theorem 4. Then we have E?A ? CUR?F = E?A ? C(C? AR?)R?F ? (1 + ?)?A ? Ak ?F . ( ) Moreover, the algorithm runs in time O mnk??2/3 + (m + n)k 3 ??2/3 + mk 2 ??2 + nk 2 ??4 . Since k, c, r ? min{m, n} by the assumptions, so the time complexity of the fast CUR algorithm is lower than that of the SVD of A. This is the main reason why we call it the fast CUR algorithm. Another advantage of this algorithm is avoiding loading the whole m ? n data matrix A into main memory. None of three steps ? the randomized SVD, the dual set sparsification algorithm, and the adaptive sampling algorithm ? requires loading the whole of A into memory. The most memoryexpensive operation throughout the fast CUR Algorithm is computing the Moore-Penrose inverse of C and R, which requires maintaining an m ? c matrix or an r ? n matrix in memory. In comparison, the subspace sampling algorithm requires loading the whole matrix into memory to compute its truncated SVD.

5

Empirical Comparisons

In this section we provide empirical comparisons among the relative-error CUR algorithms on several datasets. We report the relative-error ratio and the running time of each algorithm on each data set. The relative-error ratio is defined by ?A ? CUR?F Relative-error ratio = , ?A ? Ak ?F where k is a specified target rank. We conduct experiments on three datasets, including natural image, biology data, and bags of words. Table 1 briefly summarizes some information of the datasets. Redrock is a large size natural image. Arcene and Dexter are both from the UCI datasets [11]. Arcene is a biology dataset with 900 instances and 10000 attributes. Dexter is a bag of words dataset with a 20000-vocabulary and 2600 documents. Each dataset is actually represented as a data matrix, upon which we apply the CUR algorithms. We implement all the algorithms in MATLAB 7.10.0. We conduct experiments on a workstation with 12 Intel Xeon 3.47GHz CPUs, 12GB memory, and Ubuntu 10.04 system. According to the analysis in [10] and this paper, k, c, and r should be integers far less than m and n. For each data set and each algorithm, we set k = 10, 20, or 50, and c = ?k, r = ?c, where ? ranges in each set of experiments. We repeat each set of experiments for 20 times and report the average and the standard deviation of the error ratios. The results are depicted in Figures 1, 2, 3. The results show that the fast CUR algorithm has much lower relative-error ratio than the subspace sampling algorithm. The experimental results well match our theoretical analyses in Section 4. As for the running time, the fast CUR algorithm is more efficient when c and r are small. When c and r become large, the fast CUR algorithm becomes less efficient. This is because the time complexity of the fast CUR algorithm is linear in ??4 and large c and r imply small ?. However, the purpose of CUR is to select a small number of columns and rows from the data matrix, that is, c ? n and r ? m. So we are not interested in the cases where c and r are large compared with n and m, say k = 20 and ? = 10. 5

Running Time

Running Time

700

700

600

600

600

500

500
Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
300
500 Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
400
Time (s)
400
Time (s)
Time (s)
Running Time 700
300
400 300
200
200
200
100
100
100
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
0 2
?
4
Construction Error (Frobenius Norm)
8
0 2
10 12 14 16 18 20 22 24
?
Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
1 0.9 0.8 0.7
1.2
1 0.9 0.8 0.7
0.6
?
?
12
14
16
18
1.1 1 0.9 0.8 0.7 0.6 0.5
0.6 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
10
Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
1.2
1.1
0.5 2
8

Construction Error (Frobenius Norm)
Relative Error Ratio
1.1
6
1.3 Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
1.3 Relative Error Ratio
1.2
4
Construction Error (Frobenius Norm) 1.4
1.3
Relative Error Ratio
6
Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
4
6
8
0.4 2
10 12 14 16 18 20 22 24
?
4
6
8
10
?
12
14
16
18

(a) k = 10, c = ?k, and r = ?c. (b) k = 20, c = ?k, and r = ?c. (c) k = 50, c = ?k, and r = ?c.

Figure 1: Empirical results on the Redrock data set.

Running Time
Running Time
14
20
10
12
18 16
6
Time (s)
10
8 Time (s)
Time (s)
Running Time 12
8 6
4 4 Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR

2 0 2
4
6
0 2
8 10 12 14 16 18 20 22 24 26 28 30
?
4
Construction Error (Frobenius Norm)
8
10
?
12
14
16
18
10 Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
6 4 2
20
Construction Error (Frobenius Norm)
4
6
8
?
10
12
14
Construction Error (Frobenius Norm) 1.3
Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
1.3 Relative Error Ratio
1.3 1.2 1.1 1 0.9 0.8
Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
1.2 1.1 1 0.9 0.8
0.7
0.7
0.6
0.6
Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
1.2 Relative Error Ratio
1.4
Relative Error Ratio
6
12
8
Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
2
14

1.1 1 0.9 0.8 0.7 0.6 0.5
2
4
6
8 10 12 14 16 18 20 22 24 26 28 30
?
2
4
6
8
10
?
12
14
16
18
20
0.4 2
4
6
8
?
10
12
14
(a) k = 10, c = ?k, and r = ?c. (b) k = 20, c = ?k, and r = ?c. (c) k = 50, c = ?k, and r = ?c.

Figure 2: Empirical results on the Arcene data set.

6
Running Time
Running Time
250
Running Time
300 Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
200
400 Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
250
350
Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
100
Time (s)
150
Time (s)
Time (s)
300 200 150 100
250 200 150 100

50
50
0 2
4
6
0 2
8 10 12 14 16 18 20 22 24 26 28 30
?
50 4
8
0 2
10 12 14 16 18 20 22 24
?
1.1 1.05 1 0.95 0.9 2
4
6
8 10 12 14 16 18 20 22 24 26 28 30
?
1.15 1.1 1.05 1 0.95
?
16
18
1
0.9 0.85 0.8
10 12 14 16 18 20 22 24
14
0.95
0.75 8
12
1.1
0.9
6
?
1.05
0.85 2
4
10
Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
1.15 Relative Error Ratio
Relative Error Ratio
1.15
8
1.2
Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
1.2
6

Construction Error (Frobenius Norm)
1.25 Subspace Sampling (Exactly) Subspace Sampling (Expected) Fast CUR
1.2
4
Construction Error (Frobenius Norm)
Construction Error (Frobenius Norm) 1.25
Relative Error Ratio
6
2
4
6
8
10
?
12
14
16
18
(a) k = 10, c = ?k, and r = ?c. (b) k = 20, c = ?k, and r = ?c. (c) k = 50, c = ?k, and r = ?c.
Figure 3: Empirical results on the Dexter data set.
6
Conclusions
In this paper we have proposed a novel randomized algorithm for the CUR matrix decomposition problem. This algorithm is faster, more scalable, and more accurate than the state-of-the-art algorithm, i.e., the subspace sampling algorithm. Our algorithm requires only c = 2k??1 (1 + o(1)) columns and r = 2c??1 (1 + o(1)) rows to achieve (1+?) relative-error ratio. To achieve the same relative-error bound, the subspace sampling algorithm requires c = O(k??2 log k) columns and r = O(c??2 log c) rows selected from the original matrix. Our algorithm also beats the subspace sampling algorithm in time-complexity. Our algorithm costs O(mnk??2/3 + (m + n)k 3 ??2/3 + mk 2 ??2 + nk 2 ??4) time, which is lower than O(min{mn2 , m2 n}) of the subspace sampling algorithm when k is small. Moreover, our algorithm enjoys another advantage of avoiding loading the whole data matrix into main memory, which also makes our algorithm more scalable. Finally, the empirical comparisons have also demonstrated the effectiveness and efficiency of our algorithm.

A The Dual Set Sparsification Algorithm For the sake of completeness, we attach the dual set sparsification algorithm here and describe some implementation details. The dual set sparsification algorithms are deterministic algorithms established in [2]. The fast CUR algorithm calls the dual set spectral-Frobenius sparsification algorithm [2, Lemma 13] in both stages. We show this algorithm in Algorithm 2 and its bounds in Lemma 6. Lemma 6 (Dual Set Spectral-Frobenius Sparsification). Let U = {x1 , ? ? ? , xn } ? Rl , (l ¡ n), l?n k contains the columns of an arbitrary matrix ?n X ? RT . Let V = {v1 , ? ? ? , vn } ? R , (k ¡ n), be a decompositions of the identity, i.e. v v = I . Given an integer r with k ¡ r ¡ n, k i=1 i i Algorithm 2 deterministically computes a set of weights si ? 0 (i = 1, ? ? ? , n) at most r of which are non-zero, such that ?) n n (? (?)) (k 2 T and tr ?k si xi xTi ? ?X?2F . si vi vi ? 1 ? r i=1 i=1 7

Algorithm 2 Deterministic Dual Set Spectral-Frobenius Sparsification Algorithm.

?n l n k T 1: Input: U = {xi }n i=1 ? R , (l ¡ n); V = {vi }i=1 ? R , with i=1 vi vi = Ik (k ¡ n); k ¡ r ¡ n; 2: Initialize: s0 = 0m?1 , A0 = 0k?k ; ?n ?x ?2 3: Compute ?xi ?22 for i = 1, ? ? ? , n, and then compute ?U = i=1? i 2 ; 1?

k/r

4: for ? = 0 to r ? 1 do 5: Compute the eigenvalue decomposition of A? ; 6: Find an index j in {1, ? ? ? , n} and compute a weight t ¿ 0 such that ( )?2 ( )?1 vjT A? ? (L? + 1)Ik vj ?1 2 ?1 ?U ?xj ?2 ? t ? vj ; ? vjT A? ? (L? + 1)Ik ?(L? + 1, A?) ? ?(L? , A?) where ?(L, A) =

k ( )?1 ? ?i (A) ? L ,

L? = ? ?

?

rk;

i=1

7: Update the j-th component of s? and A? : 8: end for ? 1? k/r 9: return s = sr . r

s? +1 [j] = s? [j] + t,

A? +1 = A? + tvj vjT ;

The weights si can be computed deterministically in O(rnk 2 + nl) time. Here we would like to mention the implementation of Algorithm 2, which is not described in detailed by [2]. In each iteration the algorithm performs once eigenvalue decomposition: A? = W?WT . (A? is guaranteed to be positive semi-definite in each iteration). Since ( )q ( ) A? ? ?Ik = WDiag (?1 ? ?)q , ? ? ? , (?k ? ?)q WT , we can efficiently compute (A? ? (L? + 1)Ik)q based on the eigenvalue decomposition of A? . With the eigenvalues at hand, ?(L, A?) can also be computed directly.

Acknowledgments This work has been supported in part by the Natural Science Foundations of China (No. 61070239), the Google visiting faculty program, and the Scholarship Award for Excellent Doctoral Student granted by Ministry of Education.

2 References

[1] Adi Ben-Israel and Thomas N.E. Greville. Generalized Inverses: Theory and Applications. Second Edition. Springer, 2003. [2] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near-optimal column-based matrix reconstruction. CoRR, abs/1103.0995, 2011. [3] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near optimal column-based matrix reconstruction. In Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS ?11, pages 305?314, 2011. [4] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and

Richard Harshman. Indexing by latent semantic analysis. Journal of The American Society for Information Science, 41(6):391?407, 1990. [5] Amit Deshpande and Luis Rademacher. Efficient volume sampling for row/column subset selection. In Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, FOCS ?10, pages 329?338, 2010. [6] Amit Deshpande, Luis Rademacher, Santosh Vempala, and Grant Wang. Matrix approximation and projective clustering via volume sampling. Theory of Computing, 2(2006):225?247, 2006. [7] Petros Drineas. Pass-efficient algorithms for approximating large matrices. In In Proceeding of the 14th Annual ACM-SIAM Symposium on Dicrete Algorithms, pages 223?232, 2003. 8

[8] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition. SIAM Journal on Computing, 36(1):184?206, 2006. [9] Petros Drineas and Michael W. Mahoney. On the Nystr?om method for approximating a gram matrix for improved kernel-based learning. Journal of Machine Learning Research, 6:2153? 2175, 2005. [10] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions. SIAM Journal on Matrix Analysis and Applications, 30(2):844?881, September 2008. [11] A. Frank and A. Asuncion. UCI machine learning repository, 2010. [12] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. A theory of pseudoskeleton approximations. Linear Algebra and Its Applications, 261:1?21, 1997. [13] S. A. Goreinov, N. L. Zamarashkin, and E. E. Tyrtyshnikov. Pseudo-skeleton approximations by matrices of maximal volume. Mathematical Notes, 62(4):619?623, 1997. [14] Venkatesan Guruswami and Ali Kemal Sinop. Optimal column-based low-rank matrix reconstruction. In Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA ?12, pages 1207?1214. SIAM, 2012. [15] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM Review, 53(2):217?288, 2011. [16] John Hopcroft and Ravi Kannan. Computer Science Theory for the Information Age. 2012. [17] Finny G. Kuruvilla, Peter J. Park, and Stuart L. Schreiber. Vector algebra in the analysis of genome-wide expression data. Genome Biology, 3:research0011?research0011.1, 2002. [18] Lester Mackey, Ameet Talwalkar, and Michael I. Jordan. Divide-and-conquer matrix factorization. In Advances in Neural Information Processing Systems 24. 2011. [19] Michael W. Mahoney and Petros Drineas. CUR matrix decompositions for improved data analysis. Proceedings of the National Academy of Sciences, 106(3):697?702, 2009. [20] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. Journal of the Optical Society of America A, 4(3):519?524, Mar 1987. [21] Matthew Turk and Alex Pentland. Eigenfaces for recognition. Journal of Cognitive Neuroscience, 3(1):71?86, 1991. [22] Eugene E. Tyrtyshnikov. Incomplete cross approximation in the mosaic-skeleton method. Computing, 64:367?380, 2000.

9