A randomized algorithm for approximating the SVD of a matrix

Joint work with Per-Gunnar Martinsson (U. of Colorado) and Vladimir Rokhlin (Yale)

Mark Tygert
Program in Applied Mathematics
Yale University

2007

Outline of the talk

- Related work
- Theoretical properties of a randomized algorithm
- Steps in the algorithm
- Empirical results of the algorithm
- Proof of accuracy bounds
- Applications, generalizations, and conclusions

Related deterministic algorithms

- T. Chan and P. C. Hansen (1992)
- M. Gu and S. C. Eisenstat (1996)
- E. Tyrtyshnikov, S. A. Goreinov, and N. L. Zamarashkin (1997a, 1997b, 1999, 2000)
- G. W. Stewart, M. W. Berry, and S. A. Pulatova (1999, 2005)

Analogous randomized algorithms

- C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala (2000)
- A. Frieze, R. Kannan, and S. Vempala (1999, 2004)
- D. Achlioptas and F. McSherry (2001)
- P. Drineas, R. Kannan, M. W. Mahoney, and S. Muthukrishnan (2006a, 2006b, 2006c, 2006d)
- S. Har-Peled (2006)
- A. Deshpande and S. Vempala (2006)
- S. Friedland, M. Kaveh, A. Niknejad, and H. Zare (2006)
- T. Sarlós (2006a, 2006b, 2006c)

SVD of a matrix, 1

Given an $m \times n$ matrix A, there exist

orthonormal $m \times 1$ vectors u^1, u^2, ..., u^{k-1}, u^k,

orthonormal $n \times 1$ vectors v^1, v^2, ..., v^{k-1}, v^k, and

nonnegative numbers $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{k-1} \geq \sigma_k$, such that

$$A = \sum_{j=1}^{k} u^j \, \sigma_j \, (v^j)^{\mathsf{T}}, \tag{1}$$

where k is the rank of A.

SVD of a matrix, 2

In matrix notation,

$$A = U \Sigma V^{\mathsf{T}}, \tag{2}$$

$$U = \left(\, u^1 \, \middle| \, u^2 \, \middle| \, \ldots \, \middle| \, u^{k-1} \, \middle| \, u^k \, \right), \tag{3}$$

$$V = \left(\, v^1 \, \middle| \, v^2 \, \middle| \, \ldots \, \middle| \, v^{k-1} \, \middle| \, v^k \, \right), \tag{4}$$

$$\Sigma_{i,j} = \begin{cases} \sigma_j, & i = j \\ 0, & i \neq j. \end{cases} \tag{5}$$

The columns of U are orthonormal;
the columns of V are orthonormal.

Low-rank matrix approximation

Pick a positive integer k less than the rank of A. Then,

$$\min_{\mathrm{rank}(B)=k} \|A - B\| = \sigma_{k+1}(A), \tag{6}$$

where $\|A - B\|$ denotes the spectral norm of $A - B$.

The minimum is attained by the following matrix:

$$B = \sum_{j=1}^{k} u^j \, \sigma_j(A) \, (v^j)^{\mathsf{T}}. \tag{7}$$

Why low rank?

- Compression of subblocks of matrices arising from the discretization of smoothing integral operators and their inverses.

- Classic statistical method (principal component analysis):

- Noise removal.

- Tractable representation of term-document matrices in latent semantic analysis/indexing (search engines).

Accuracy of a randomized algorithm

Suppose A is an $m \times n$ matrix.

Then, the algorithm computes a rank-k matrix B such that

$$\|A - B\| \leq 10 \sqrt{(k + 20)\, n}\ \ \sigma_{k+1}(A) \tag{8}$$

with probability at least $1 - 10^{-17}$.

Furthermore, if $\sigma_j(A) \leq \sigma_{k+1}(A)/\sqrt{n}$, then

$$\|A - B\| \leq 20 \sqrt{(k + 20)\, j}\ \ \sigma_{k+1}(A) \tag{9}$$

with probability at least $1 - 10^{-17}$.

Cost of the randomized algorithm

The algorithm requires applying A to $k + 20$ vectors, and A^{T} to k vectors, in addition to $\mathcal{O}(k^2\,(m+n))$ floating-point operations and words of memory.

Moreover, the constant hidden by the $\mathcal{O}$-notation is quite small — "$\mathcal{O}(k^2\,(m+n))$" is the cost of computing SVDs using the standard algorithms for an $m \times (k+20)$ matrix and a $k \times n$ matrix, as well as for multiplying an $m \times k$ matrix and a $k \times k$ matrix.

Examples of rapidly applicable matrices

- Sparse
- Discrete Fourier transform
- Toeplitz
- Fast multipole method
- Blockwise combinations and compositions of subblocks of the above

Step 1 of the algorithm

We test A on $k+20$ vectors whose entries are i.i.d. $N(0,1)$ random variables, in order to identify the range of A. That is, we multiply the $m \times n$ matrix A and an $n \times (k+20)$ matrix R whose entries are i.i.d. $N(0,1)$ random variables:

$$P_{m\times(k+20)} = A_{m\times n} \cdot R_{n\times(k+20)}. \tag{10}$$

Step 2 of the algorithm

Via an SVD, we find the left singular vectors corresponding to the k greatest singular values of the $m \times (k + 20)$ matrix P, and form an $m \times k$ matrix Q with the left singular vectors of P as its columns. We will prove that the set of columns of Q is close to an orthonormal basis for the range of A. (How close is determined both by $\sigma_{k+1}(A)$ and by the condition number of the random matrix R with which we multiplied A.)

Step 3 of the algorithm

Now that we have an orthonormal basis for the range of A, the rest is fairly straightforward. We apply A^{T} to the vectors which constitute the orthonormal basis. That is, we multiply the $n \times m$ matrix A^{T} and the $m \times k$ matrix Q:

$$S_{n\times k} = (A^{\mathsf{T}})_{n\times m} \cdot Q_{m\times k}. \tag{11}$$

Step 4 of the algorithm

We compute an SVD of the $k \times n$ matrix S^{T}:

$$(S^{\mathsf{T}})_{k \times n} = T_{k \times k} \cdot \Sigma_{k \times k} \cdot (V^{\mathsf{T}})_{k \times n}, \tag{12}$$

where the entries of the $k \times k$ matrix Σ are all nonnegative and are zero off of the diagonal, and where the columns of the $k \times k$ matrix T are orthonormal, as are the columns of the $n \times k$ matrix V.

Step 5 of the algorithm

We multiply the $m \times k$ matrix Q and the $k \times k$ matrix T:

$$U_{m \times k} = Q_{m \times k} \cdot T_{k \times k}. \tag{13}$$

The product $U \, \Sigma \, V^{\mathsf{T}}$ is the desired approximation to A.

Recap of the algorithm

1. Form $P_{m\times(k+20)} = A_{m\times n} \cdot R_{n\times(k+20)}$; R is random.

2. Form $Q_{m\times k}$, the matrix whose columns consist of the left singular vectors corresponding to the k greatest singular values of $P_{m\times(k+20)}$.

3. Form $S_{n\times k} = (A^{\mathsf{T}})_{n\times m} \cdot Q_{m\times k}$.

4. Compute the SVD $T_{k\times k} \cdot \Sigma_{k\times k} \cdot (V^{\mathsf{T}})_{k\times n}$ of $(S^{\mathsf{T}})_{k\times n}$.

5. Form $U_{m\times k} = Q_{m\times k} \cdot T_{k\times k}$.

Output of the algorithm

Combining the equations on the previous transparency yields

$$A_{m\times n} \approx Q_{m\times k} \cdot (Q^{\mathsf{T}})_{k\times m} \cdot A_{m\times n} = U_{m\times k} \cdot \Sigma_{k\times k} \cdot (V^{\mathsf{T}})_{k\times n}. \quad (14)$$

Recall that the entries of Σ are all nonnegative and are zero off of the diagonal, and the columns of U are orthonormal, as are the columns of V.

Empirical accuracy of the algorithm

As n ranges from 100 to 1,000,000, with $k = 10$

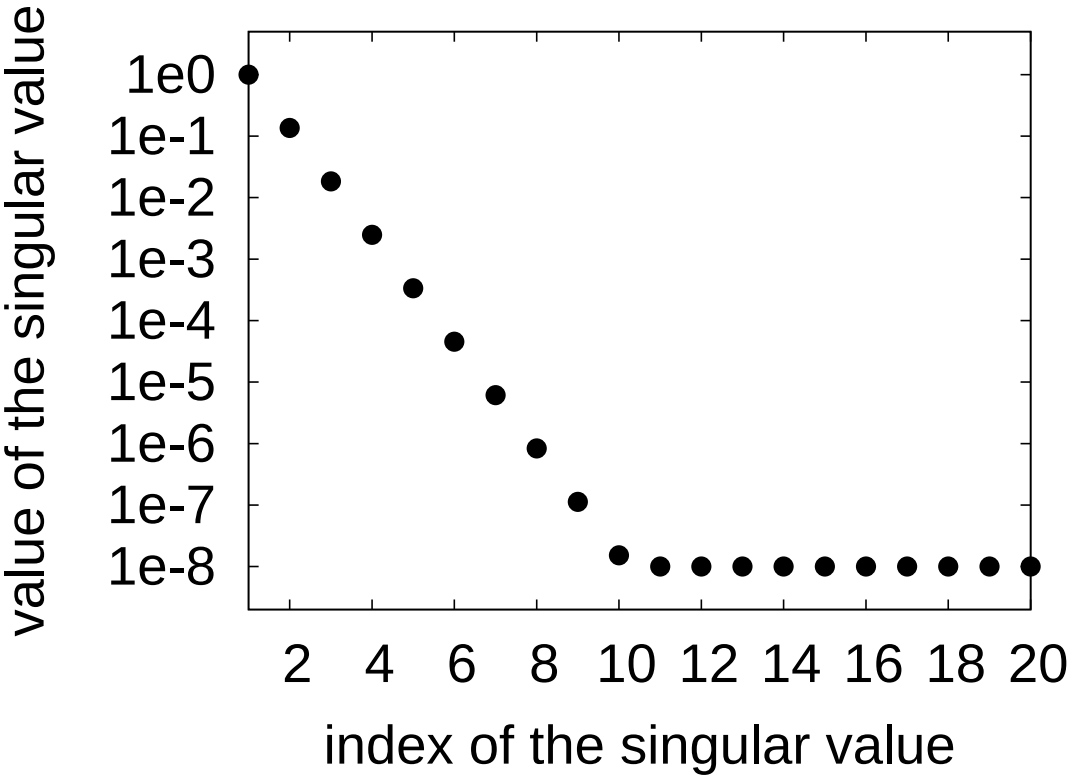
the rank-k approximation B to the $n \times n$ matrix A

is such that $\|A - B\|$ ranges from 10^{-7} to $2 \cdot 10^{-7}$,

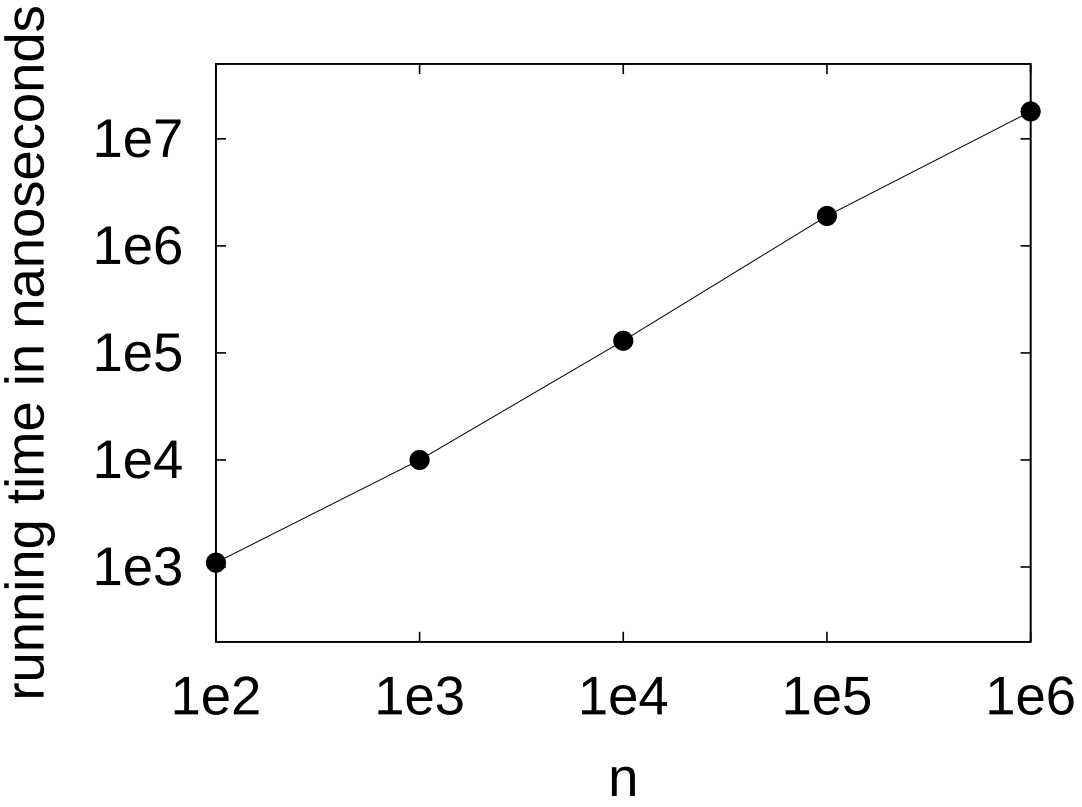
while $\sigma_{k+1}(A) = 10^{-8}$ and $\sigma_1(A) = 1$ for all n.

We applied A to k random vectors, instead of $k + 20$.

Singular values of the matrices



Costs are proportional to n with $k = 10$ for an $n \times n$ matrix of rank 20



Stratagem for proof of the accuracy bounds

1. Apply the triangle inequality multiple times, breaking the estimate into two contributions, one which involves optimization over a space of dim. k, and the other requiring the existence of a good bound in a high-dimensional space.

2. Optimize over the space of dim. k directly via brute force computation as a step in the algorithm (compute an SVD).

3. To prove the existence of a bound in the high-dimensional space, work all estimates on paper within the basis of the right singular vectors of the matrix A to be recovered. Observe that the condition number of the matrix R whose entries are i.i.d. $N(0,1)$ is not too large, with high prob.

Proof of the accuracy bounds, 1

Suppose that A is an $m \times n$ matrix, Q is an $m \times k$ matrix whose columns are orthonormal, R is an $n \times (k + 20)$ matrix, W is a $(k + 20) \times n$ matrix, and Z is a $k \times (k + 20)$ matrix.

Then, multiple applications of the triangle inequality yield that

$$\|A - Q\,Q^{\mathsf{T}}\,A\| \leq 2\,\|A - A\,R\,W\| + 2\,\|A\,R - Q\,Z\|\,\|W\|. \qquad (15)$$

By choosing W and Z appropriately, we would like to show that

$$\|A - Q\,Q^{\mathsf{T}}\,A\| \lesssim \sigma_{k+1}(A). \qquad (16)$$

Proof of the accuracy bounds, 2

The algorithm chooses the columns of Q to consist of the left singular vectors of $A\,R$ which correspond to the k greatest singular values of $A\,R$. Therefore, with an appropriate choice for Z, we obtain that

$$\|A\,R - Q\,Z\| \leq \sigma_{k+1}(A\,R) \leq \|R\|\,\sigma_{k+1}(A). \tag{17}$$

Combining formulae (15) and (17) yields that

$$\|A - Q\,Q^{\mathsf{T}}\,A\| \leq 2\,\|A - A\,R\,W\| + 2\,\|R\|\,\|W\|\,\sigma_{k+1}(A). \tag{18}$$

Proof of the accuracy bounds, 3

Again,

$$\|A - Q\,Q^{\mathsf{T}}\,A\| \leq 2\,\|A - A\,R\,W\| + 2\,\|R\|\,\|W\|\,\sigma_{k+1}(A). \qquad (19)$$

Since the norm $\|R\|$ of the random matrix R is not too large with very high probability, it suffices to show that in principle there exists a $(k+20) \times n$ matrix W such that $\|A - A\,R\,W\|$ is of the order of $\sigma_{k+1}(A)$, and $\|W\|$ is not too large, with very high probability.

Key idea in the existence proof

The entries have the same joint probability distribution with respect to any orthonormal basis for a vector whose entries in one orthonormal basis are i.i.d. $N(0,1)$ random variables.

So, within the proof that there exists a linear operator which recovers a specified matrix A from the vectors resulting from applying A to $k+20$ such random vectors, we may conduct all calculations in the basis of the right singular vectors of A.

Estimates from random matrix theory

The existence proof is almost trivial using the "key idea" from the previous transparency, since the condition number of a matrix whose entries are i.i.d. $N(0,1)$ random variables seldom gets too large. Goldstine and Von Neumann (1951) and Chen & Dongarra (2005) give particularly useful bounds on the condition numbers of such matrices. Their bounds are remarkably simple, and reasonably tight.

A peculiarity

As the condition number of R is not too large with high prob., we are able to recover A from $A\,R$ to within about $\sigma_{k+1}(A)$ by multiplying the product $A\,R$ from the right by a matrix, say W, which depends on R, but is independent of $\sigma_1(A)$, ..., $\sigma_k(A)$.

In principle, we could recover A better by taking into account $\sigma_1(A)$, ..., $\sigma_k(A)$, but doing so makes the analysis unwieldy.

It seems to be sufficient to recover A solely by knowing which subspace constitutes A's range, without knowing the structure of A within that subspace.

Recap of the proof of the accuracy bounds

1. Apply the triangle inequality multiple times, breaking the estimate into two contributions, one which involves optimization over a space of dim. k, and the other requiring the existence of a good bound in a high-dimensional space.

2. Optimize over the space of dim. k directly via brute force computation as a step in the algorithm (compute an SVD).

3. To prove the existence of a bound in the high-dimensional space, work all estimates on paper within the basis of the right singular vectors of the matrix A to be recovered. Observe that the condition number of the matrix R whose entries are i.i.d. $N(0,1)$ is not too large, with high prob.

Applications of the algorithm

- Optimal compression into standard forms (SVDs, for example) of suboptimally compressed matrices.

- Compression of subblocks of matrices arising from the discretization of smoothing integral operators and their inverses.

- Noise removal.

- Tractable representation of term-document matrices in latent semantic analysis/indexing (search engines).

What about the Lanczos method (for low-rank matrix approximation)?

- The Lanczos method with complete reorthogonalization works about as well in finite-precision arithmetic as in exact.

- Its costs are similar to the randomized algorithm's costs. However, the Lanczos method is iterative; unlike the direct randomized algorithm, it can require many iterations.

- Even in exact arithmetic, the Lanczos method has trouble when singular values are degenerate. Applications arising in engineering frequently involve symmetries, and hence often involve substantially degenerate singular values.

Interpolative decomposition algorithm

An algorithm similar to that described in the present talk can efficiently compute an approximation to the so-called interpolative decomposition of a matrix A for which both A and A^{T} can be applied efficiently to arbitrary vectors.

The similar algorithm is generally somewhat faster than the SVD-based one, and simplifies the implementation of direct/locally-adaptive solvers for integral equations.

Definition of interpolative decompositions

An interpolative decomposition of an $m \times n$ matrix A consists of an $m \times k$ matrix C whose columns consist of a subset of the columns of A, as well as a $k \times n$ matrix P, such that

1. some subset of the columns of P makes up the $k \times k$ identity,
2. every entry of P has an absolute value of at most 2, and
3. $A = C\,P$.

Conclusion

There exists a robust, efficient randomized algorithm for computing an approximation to an SVD of a matrix A for which A & A^{T} may be applied rapidly to arbitrary vectors.

Given any positive integer k, the algorithm constructs a rank-k approximation whose accuracy is of the same order as the accuracy of the best possible rank-k approximation.

The algorithm has a rather negligible probability of failure (10^{-17} is not atypical), and operates reliably independently of the structure of A (unlike the classical Lanczos method for computing an approximation to an SVD of a matrix A).

mark.tygert@yale.edu http://www.cs.yale.edu/∼tygert