SPARSE REDUCED-RANK APPROXIMATIONS TO SPARSE MATRICES

G. W. Stewart
Department of Computer Science
Institute for Advanced Computer Studies
University of Maryland
College Park
Mathematical and Computational Sciences Division
NIST

REDUCED-RANK DECOMPOSITIONS

- In many applications we are given an $m \times n$ $(m \geq n)$ matrix A and require a reduced-rank approximation:

$$A \cong \tilde{A} = XTY^{\mathrm{T}}. \qquad (*)$$

  - X and Y^{T} are of full rank k.
  - T is nonsingular of order k.

- If A is $m \times n$ then it requires
  - mn words to store A
  - mn adds and mults to form Ax

- For the form $(*)$ it requires
  - $(m+n+k)k$ words to store A
  - $(m+n+k)k$ adds and mults to form Ax

SPARSE A

$$A \cong \tilde{A} = XTY^{\mathrm{T}}. \qquad (*)$$

$$\text{———————} \diamond \text{———————}$$

- If A is sparse, then the counts become `nnz` — the number of nonzeros in A.
  - To be compared with $(m+n+k)k$ for the approximation $(*)$.
- If A has, say, ℓ elements per row, the approximation $(*)$ looses when $k > \ell$.

THE SINGULAR VALUE DECOMPOSITION

- We can write A in the form

$$A = U\Sigma V^{\mathrm{T}}.$$

  - $U = (u_1 \ \cdots \ u_n)$ is orthonormal.
  - $V = (v_1 \ \cdots \ v_n)$ is orthogonal.
  - $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$, where $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$.

- If we partition

$$A = (U_1^{(k)} \ U_2^{(k)}) \begin{pmatrix} \Sigma_1^{(k)} & 0 \\ 0 & \Sigma_2^{(k)} \end{pmatrix} \begin{pmatrix} V_1^{(k)\mathrm{T}} \\ V_2^{(k)\mathrm{T}} \end{pmatrix} = U_1^{(k)}\Sigma_1^{(k)}V_1^{(k)\mathrm{T}} + U_2^{(k)}\Sigma_2^{(k)}V_2^{(k)\mathrm{T}}.$$

We have the reduced-rank decomposition

$$A \cong \tilde{A}_k = U_1^{(k)}\Sigma_1^{(k)}V_1^{(k)\mathrm{T}}.$$

- The error is $\|\Sigma_2^{(k)}\|$.

PROS AND CONS

$$A \cong U_1^{(k)} \Sigma_1^{(k)} V_1^{(k)\mathrm{T}}. \qquad (*)$$

———— ◇ ————

- Pro: The approximation is optimal.
- Pro: The SVD reliably determines numerical rank.
- Pro: There are stable algorithms to compute $(*)$ without computing the entire SVD.
- Con: The algorithms are expensive and their behavior depends on the spectrum of the matrix.
- Con: The factors U_k and V_k are not sparse.

THE PIVOTED QR FACTORIZATION

- We can write AP in the form

$$AP = B = QR.$$

  - Q is $m \times n$ and orthonormal.
  - R is $n \times n$ and upper triangular.
  - P is a permutation matrix.

- If we partition

$$B = (B_1^{(k)} \;\; B_2^{(k)}) = (Q_1^{(k)} \;\; Q_2^{(k)}) \begin{pmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ 0 & R_{22}^{(k)} \end{pmatrix},$$

  where $R_{11}^{(k)}$ is $k \times k$, then

$$B = AP \cong Q_1^{(k)} (R_{11}^{(k)} \;\; R_{12}^{(k)}).$$

- The error is $\|R_{22}^{(k)}\|$.

PROS AND CONS

$$B = AP \cong Q_1^{(k)}(R_{11}^{(k)} \;\; R_{12}^{(k)}).$$

———— ◇ ————

- Pro: The approximations are generally good.
- Pro: The decomposition usually reveals rank.
- Pro: The algorithms are fast.
- Con: The factors Q_k and $(R_{11}^{(k)} \;\; R_{12}^{(k)})$ are not sparse.

———— ◇ ————

- In this talk we will show how to sparsify the QR approximation.

RELATION TO THE SVD

- Let $A = XY^{\mathrm{T}} + E$ be a reduced-rank approximation to A. People worry that $\mathcal{X} = \mathcal{R}(X)$ will not be a good approximation to $\mathcal{R}(U_1^{(k)})$ from the SVD.
- If $Av = \sigma u$, then

$$\sin \angle(u, \mathcal{X}) \leq \frac{\|E\|}{\sigma}. \qquad (*)$$

  - If σ is well above the error level, its singular vector is well represented in $\mathcal{X}$.
- Proof: Let $X_\perp$ be a ON basis for $\mathcal{X}_\perp$. Then $\sin \angle(u, \mathcal{X}) = \|X_\perp^{\mathrm{T}} u\|$.
  1. $X_\perp^{\mathrm{T}} A v = \sigma X_\perp^{\mathrm{T}} u$
  2. $X_\perp^{\mathrm{T}} A v = X_\perp^{\mathrm{T}}(XY^{\mathrm{T}} - E)v = X_\perp^{\mathrm{T}} XY^{\mathrm{T}} v - X_\perp^{\mathrm{T}} E v = -X_\perp^{\mathrm{T}} E v.$

  Hence $\sigma \|X_\perp^{\mathrm{T}} u\| \leq \|E\|$, which is equivalent to $(*)$

THE PIVOTED QR DECOMPOSITION: SOME IMPORTANT FACTS

$$B = AP = QR$$

- $\|r_j\| = \|b_j\|$
  - The norm of the jth columns of R and B are the same.
- $R = Q^{\mathrm{T}}B$
  - We can compute the jth row of R from the jth column of Q and B.
- $Q = BR^{-1}$
  - We can compute Q from B and R.

THE PARTITIONED DECOMPOSITION

- Partition the decomposition in the form

$$(B_1^{(k)} \;\; B_2^{(k)}) = (Q_1^{(k)} \;\; Q_2^{(k)}) \begin{pmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ 0 & R_{22}^{(k)} \end{pmatrix}.$$

  Our QR approximation is

$$B \cong Q_1^{(k)} (R_{11}^{(k)} \;\; R_{12}^{(k)}).$$

  with error

$$\|R_{22}^{(k)}\|.$$

- This error estimate — if we can compute it — can be used to determine k.

PIVOTING

- We will not compute the entire decomposition. Instead we will bring in columns of A one at a time to compute successively

$$Q_1^{(k)}(R_{11}^{(k)} \;\; R_{12}^{(k)}) \quad \text{from} \quad Q_1^{(k-1)}(R_{11}^{(k-1)} \;\; R_{12}^{(k-1)})$$

- The column to be brought in — the pivot column — corresponds the largest column of $R_{22}^{(k-1)}$.

- Again we need norm information about $R_{22}^{(k-1)}$.

COLUMN NORMS OF $R_{22}^{(k-1)}$

$$(B_1^{(k-1)} \;\; B_2^{(k-1)}) = (Q_1^{(k-1)} \;\; Q_2^{(k-1)}) \begin{pmatrix} R_{11}^{(k-1)} & R_{12}^{(k-1)} \\ 0 & R_{22}^{(k-1)} \end{pmatrix}$$

———— ◇ ————

- Let r_j and b_j be the jth columns of R and B. Then $\|r_j\| = \|b_j\|$.
- Partition

$$r_j = \begin{pmatrix} r_1^{(j)} \\ r_2^{(j)} \end{pmatrix},$$

  where $r_1^{(j)}$ is a $(k-1)$-vector. Then

$$\|b_j\|^2 = \|r_1^{(j)}\|^2 + \|r_2^{(j)}\|^2.$$

- Hence we can downdate the norms according to the formula

$$\|r_2^{(j)}\|^2 = \|b_j\|^2 - \|r_1^{(j)}\|^2 = \|b_j\|^2 - r_{1j}^2 - r_{2j}^2 - \cdots - r_{k-1,j}^2.$$

A FLY IN THE OINTMENT

- Consider the problem of computing

$$\nu^2 = \beta^2 - \rho^2$$

  in IEEE double precision with rounding unit $\epsilon_{\mathsf{M}} \cong 10^{-16}$.

- If $\nu^2/\beta^2 = 10^{-t}$, then ν^2 will have about $16 - t$ significant figures.
  - In particular, when $t = 16$, ν^2 has no significant figures.

- If $\nu/\beta \leq 10^{-8}$, our norm computation fails.

- For our applications, this is acceptable.

THE GRAM–SCHMIDT ALGORITHM

- Suppose we have a QR factorization $B = QR$ of B and wish to compute a QR factorization

$$(B \;\; a) = (Q \;\; q) \begin{pmatrix} R & r \\ 0 & \rho \end{pmatrix}$$

- We have

$$a = Qr + \rho q.$$

Hence

$$r = Q^{\mathrm{T}} a.$$

Since $\|q\| = 1$, we have

$$\rho = \|a - Qr\| \quad \text{and} \quad q = \rho^{-1}(a - Qr).$$

- The red equations are effectively an algorithm for extending our original QR factorization.

REORTHOGONALIZATION

- Cancellation in the formula $q = \rho^{-1}(a - Qr)$ can cause loss of orthogonality.

- The cure is to orthogonalize twice.

```
r = Q'*a
q = a - Q*r
s = Q'*q
r = r + s
q = q - Q*s
rho = norm(q)
q = q/rho
```

- Typically, q is orthogonal to Q to working accuracy.
  - Most people don't sweat the atypical stuff.

COMPUTATION OF THE QR APPROXIMATION

- If we apply the GS algorithm to

$$B_1^{(k-1)} = Q_1^{(k-1)} R_{11}^{(k-1)}$$

with a a column of A, we get

$$B_1^{(k)} = Q_1^{(k)} R_{11}^{(k)}$$

- From the relation $R = Q^{\mathrm{T}} B$, we can compute the kth row of $R_{12}^{(k)}$ in the form

$$R_{12}^{(k)}(k, k{+}1{:}n) = Q_1^{(k)}(:, k)^{\mathrm{T}} B^{(k)}(:, k{+}1{:}n).$$

  - This is likely to be the most expensive part of the algorithm.
  - Even if we don't want R_{12}, we must perform this computation to downdate norms.

THE QUASI-GRAM–SCHMIDT ALGORITHM I

- If m is very large, we may be unable to store Q.
- If we use the relation $Q = BR^{-1}$, we can form the product $r = Q^{\mathrm{T}}a = R^{-\mathrm{T}}B^{\mathrm{T}}a$ by the following algorithm.

```
d = B'*a
r = R'\d
```

- We can also compute $q = a - Qr = a - BR^{-1}r$ by

```
p = R\r
q = a - B*p
```

THE QUASI-GRAM–SCHMIDT ALGORITHM II

- These considerations lead to the following code for the quasi-Gram–Schmidt algorithm.

```
d = B'*a                    r = Q'*a
r = R'\d
p = R\r                     q = a - Q*r
q = a - B*p
d = B'*q                    s = Q'*q
s = R'\d
r = r + s                   r = r + s
p = R\s                     q = q - Q*s
q = q - B*s
rho = norm(q)               rho = norm(q)
q = q/rho                   q = q/rho
```

- Applying this code to computing the QR approximation gives us a Q-less approximation that we call the semi-QR approximation.

NUMERICAL PROPERTIES I

- The Gram–Schmidt algorithm with reorthogonalization produces a Q that is orthonormal to working accuracy.

- In the quasi-Gram–Schmidt the approximation BR^{-1} may have columns that are not orthonormal.
  - This would be true even if we had the correctly rounded R.

- However, the loss of orthonormality is proportional to the condition number $\|R\|\|R^{-1}\|$ of R, which is the best we can expect.

NUMERICAL PROPERTIES II

- Let $R+E$ be the correctly rounded R from the semi-QR factorization of B. Then

$$\|E\| \leq \gamma \|R\| \epsilon_{\mathsf{M}}.$$

- Set

$$\tilde{Q} = B(R+E)^{-1} \cong Q - QER^{-1}.$$

  Hence

$$\tilde{Q}^{\mathrm{T}}\tilde{Q} \cong Q^{\mathrm{T}}Q - Q^{\mathrm{T}}QER^{-1} - R^{-\mathrm{T}}EQ^{\mathrm{T}}Q.$$

- Let $W = Q^{\mathrm{T}}Q - I$ and $\omega = \|W\|$. Then

$$\tilde{\omega} \lesssim \omega + 2\|Q^{\mathrm{T}}Q\|\|R^{-1}\|\|E\| \leq \omega + 2\gamma\kappa(R)\epsilon_{\mathsf{M}},$$

  where $\kappa(R) = \|R\|\|R^{-1}\|$.

REVIEW

- Our semi-QR approximation has the form

$$B \cong (B_1^{(k)} R_{11}^{(k)-1})(R_{11}^{(k)} \;\; R_{12}^{(k)})$$

- The decomposition can be computed with pivoting with only sparse matrix-vector multiplications and low order triangular solves.

- If A has only -1, 0, or 1 for elements, then the matrix-vector multiplication can be done with additions and subtractions only.

- The columns of Q are computed, used to compute a row R, and then are discarded.

- The stopping point for the algorithm can be determined by using the value of $\|R_{22}^{(k)}\|$.

A SPARSE PIVOTED QR (SPQR) APPROXIMATION

- If n is large we may not be able to store $(R_{11}^{(k)} \; R_{12}^{(k)})$.
- Apply the quasi-GS algorithm to A to get a selection of columns of X of A and a triangular matrix R.
  - Let the error be ϵ_{col}.
- Apply the quasi-GS algorithm to A^{T} to get a selection of rows of Y^{T} of A and a triangular matrix S.
  - Let the error be ϵ_{row}.
- If

$$T = R^{-1}R^{-\text{T}}(X^{\text{T}}AY)S^{-1}S^{-\text{T}}.$$

  then

$$\|A - XTY^{\text{T}}\| \leq \sqrt{\epsilon_{\text{col}}^2 + \epsilon_{\text{row}}^2} \quad \text{is minimal.}$$

- Since the dimensions of T are small, XTY^{T} is essentially a sparse approximation to A.

SOME TIMINGS

- A is a random sparse matrix of order $10{,}000$ with singular values having equally spaced common logarithms between 0 and -6.
- For $k = 10{:}5{:}40$ we computed the SPQR approximation of rank k. This was compared with the Matlab function `svds(A,k)`. The results are summarized in the following table.

k	SPQR	SVD
10	2.5	40.2
15	3.0	40.0
20	3.2	51.1
25	3.6	55.7
30	4.1	70.2
35	4.5	92.7
40	4.8	120.0

- The SVD times are worse by factors ranging from 17 for $k = 10$ to 31 for $k = 40$.

ANOTHER EXPERIMENT

- In the previous experiment, the singular values had no gap.
- In another experiment the singular values were generated by

```
s = logspace(0, -4, n);
s(20:n) = 1e-6*s(20:n);
```

  This places a multiplicative gap of about 10^{-6} between the 19th and 20th singular values.
- The results

nc	SQR	SVD
19	1.8	4.4
20	1.8	323.6

- This illustrates the sensitivity of the SVD algorithm to the spectrum.
  - For nc $= 19$, the algorithm must find a well separated eigenspace.
  - For nc $= 20$, the eigenspace is ill-separated.
  - But we still have to compute it!

IMPLEMENTATION

- Michael Berry, Shakhina Pulatova, and I have implemented the SPQR approximation in Matlab [TOMS 31 (2005) 252–269].

- A report and the code are available at
  `ftp://thales.cs.umd.edu/pub/reports/Contents.html`