SPARSE REDUCED-RANK APPROXIMATIONS TO SPARSE MATRICES

G. W. Stewart

Department of Computer Science

University of Maryland

A PROBLEM

- We are given the following.
  - An $m \times n$ matrix A.
  - An $m \times k$ matrix X.
  - An $n \times \ell$ matrix Y.

- Find a $k \times \ell$ matrix T such that

$$\|A - XTY^*\| = \min .$$

  - Here $\| \cdot \|$ denotes the Frobenius norm.

SOLUTION

$$\|A - XTY^*\| = \min .$$

- Let P_X be the orthogonal projection onto the column space of X and $P_X^\perp = I - P_X$.

- Similarly for P_Y and $P_Y^\perp$.

- Then
$$\hat{A} = P_X A P_Y .$$

ERROR

- The error is

$$A - \hat{A} = P_X A P_Y^{\perp} + P_X^{\perp} A P_Y + P_X^{\perp} A P_Y^{\perp}.$$

- If
  - $\mathcal{R}(X)$ approximates the dominant part of the column space of A and
  - $\mathcal{R}(Y)$ approximates the dominant part of the row space of A,

then the error will be small

FORMULAS

- Let X and Y be of full rank and let

$$X = Q_X R_X \quad \text{and} \quad Y = Q_Y R_Y$$

be the QR factorizations of X and Y.

- Then

$$T = (R_X^* R_X)^{-1} X^* A Y (R_Y^* R_Y)^{-1}.$$

- Note that the formulas do not involve Q_X and Q_Y.

SPARSITY CONSIDERATIONS

$$T = (R_X^* R_X)^{-1} X^* A Y (R_X^* R_X)^{-1}.$$

- Suppose:
  - k and ℓ are small compared to m and n.
  - A, X, and Y are sparse.
- Then the formation of X^*AY can be done with sparse matrix-vector multiplications.
- The formation of T requires dense matrix operations, but the matrices are small.
- The matrix $\hat{A} = XTY^*$ can be efficiently manipulated from its factored form.

THE PIVOTED QR DECOMPOSITION

- We need to get sparse X and Y that approximate the column and row spaces of A.
  - We will focus on X.

- The matrix A has a pivoted QR decomposition of the form
$$AJ = (X \;\; X_\bullet) = (Q_X \;\; Q_\bullet) \begin{pmatrix} R_X & R_{X\bullet} \\ 0 & R_{\bullet\bullet} \end{pmatrix}.$$
  - J is a permutation matrix chosen dynamically to make the columns of X independent.

EXPANDING THE DECOMPOSITION

- Given $X = Q_X R_X$,
  - Choose a column a of A that is not in X
  - Orthogonalize a against Q_X by the Gram–Schmidt method.

1. $r = Q_X^* a$
2. $q = a - Qr$
3. $\rho = \|q\|$; $q = q/\rho$
4. $X = (X\ a)$; $Q_X = (Q_X\ q)$
5. $R_X = \begin{pmatrix} R_X & r \\ 0 & \rho \end{pmatrix}$

- In practice the orthogonalization must be repeated.

THE QUASI-GRAM–SCHMIDT ALGORITHM

- We can get rid of Q_X, which is dense, by using the relation $Q_X = XR_X^{-1}$
  1. Solve the system $R_X^* r = X^* a$
  2. Solve the system $R_X d = Xr$
  3. $q = a - d$
  4. $\rho = \|q\|;\ q = q/\rho$
  5. $X = (X\ a);\ Q_X = (Q_X\ q)$
  6. $R_X = \begin{pmatrix} R_X & r \\ 0 & \rho \end{pmatrix}$

- Reorthogonalization is required.

- The algorithm breads down at half precision.

ON $R_{\bullet\bullet}$

$$AJ = (X \;\; X_\bullet) = (Q_X \;\; Q_\bullet) \begin{pmatrix} R_X & R_{X\bullet} \\ 0 & R_{\bullet\bullet} \end{pmatrix}.$$

- $\|R_{\bullet\bullet}\| = \|P_X^\perp A\|$.
  - Thus $\|R_{\bullet\bullet}\|$ can be used to stop the expansion.
- The norms of the columns of $R_{\bullet\bullet}$ are used to choose the column to bring into the computation.
- These norms can be computed initially and downdated as the expansion proceeds.
  - This process also breaks down at half precision.
  - It is the most expensive part of the algorithm.

A THEOREM ON SINGULAR VECTORS

Let $A = XY^* + E$. Let $\sigma > 0$ be a singular value of A with normalized left and right singular vectors u and v. Then

$$\sin\angle(u, \mathcal{R}(X)),\ \sin\angle(v, \mathcal{R}(Y)) \le \frac{\|E\|_2}{\sigma}.$$

◇

- The theorem shows that if the error in the approximation is small, then its column and row spaces must contain good approximations to left and right singular vectors corresponding to the larger singular values.

COMPARISON WITH THE SVD (I)

- The A is of order $10{,}000$, with singular values given by

```
s = logspace(0, -6, n).
```

- The timings compared with Matlab's `svds` are

k	SPQR	SVD
10	2.6	42.4
15	3.0	35.7
20	3.4	52.6
25	3.7	57.3
30	4.1	70.5
35	4.4	91.4
40	4.8	120.0

COMPARISON WITH THE SVD (II)

- We now consider a problem with gaps.

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

- The timings are

k	SPQR	SVD
19	1.8	4.4
20	1.8	323.6

- When the SVD has to cross the gap, its timing jumps.
  - But it has to cross to find the gap.

SOME LITERATURE

- G. W. Stewart. Four algorithms for the the efficient computation of truncated pivoted QR approximations to a sparse matrix. *Numerische Mathematik*, 83:313–323, 1999.
- Michael W. Berry, Shakhina A. Pulatova, and G. W. Stewart. Computing Sparse Reduced-Rank Approximations to Sparse Matrices. *ACM Transactions on Mathematical Software (TOMS)*, 31:252–269, 2005.
- Z. Zhang, H. Zha, and H. Simon. Low-rank approximations with sparse factors I: Basic algorithms and error analysis. *SIAM Journal on Matrix Analysis and Applications*, 23:706–727, 2002.