University of Maryland College Park

Institute for Advanced Computer Studies TR–98–12
Department of Computer Science TR–3875

Two Algorithms for the The Efficient Computation of Truncated Pivoted QR Approximations to a Sparse Matrix*

G. W. Stewart†

ABSTRACT

In this note we propose two algorithms to compute truncated pivoted QR approximations to a sparse matrix. One is based on the Gram–Schmidt algorithm, and the other on Householder triangularization. Both algorithms leave the original matrix unchanged, and the only additional storage requirements are arrays to contain the factorization itself. Thus, the algorithms are particularly suited to determining low-rank approximations to a sparse matrix.

*This report is available by anonymous ftp from `thales.cs.umd.edu` in the directory `pub/reports` or on the web at `http://www.cs.umd.edu/ stewart/`.

†Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. This work was supported by the National Science Foundation under grant CCR-95-03126.

Two Algorithms for the The Efficient Computation of Truncated Pivoted QR Approximations to a Sparse Matrix[*]

G. W. Stewart[†]

Dedicated to Olof Widlund on his sixtieth birthday.

ABSTRACT

In this note we propose two algorithms to compute truncated pivoted QR approximations to a sparse matrix. One is based on the Gram–Schmidt algorithm, and the other on Householder triangularization. Both algorithms leave the original matrix unchanged, and the only additional storage requirements are arrays to contain the factorization itself. Thus, the algorithms are particularly suited to determining low-rank approximations to a sparse matrix.

1. Introduction

Let X be an $n \times p$ matrix with $n > p$. This paper concerns the approximation of X by a matrix $\hat{X}$ of rank $k < p$. The most elegant solution to this problem is the truncated singular value approximation which gives an $\hat{X}$ that deviates minimally from X in both the spectral and Frobenius norms.[1]

A different approximation may be had from the pivoted QR decomposition. This decomposition is a factorization of the form

$$X = Q \begin{pmatrix} R \\ 0 \end{pmatrix} \Pi^{\mathrm{T}},$$

where Q is orthogonal, R is upper triangular, and Π is a permutation matrix that corresponds to column interchanges in X (i.e., pivoting). If we partition[2]

$$X = (Q_{11}\ \ Q_{1,k+1}\ \ Q_{1,p+1}) \begin{pmatrix} R_{11} & R_{1,k+1} \\ 0 & R_{k+1,k+1} \\ 0 & 0 \end{pmatrix} \Pi^{\mathrm{T}}, \tag{1}$$

[*]This report is available by anonymous ftp from `thales.cs.umd.edu` in the directory `pub/reports` or on the web at `http://www.cs.umd.edu/ stewart/`.

[†]Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. This work was supported by the National Science Foundation under grant CCR-95-03126.

[1]Since the purpose of this note is to describe rearrangements of two standard algorithms, we assume that the reader is familiar with the basics, which are covered in an excellent book by Å. Björck [1]

[2]In the partitions to follow we use northwest indexing, in which each block has the index of the element in the northwest corner.

where R_{11} is of order k, then

$$\hat{X} = Q_{11}(R_{11} \;\; R_{1,k+1})\Pi^{\mathrm{T}} \tag{2}$$

is a rank k approximation to X with $\|\hat{X} - X\| = \|R_{k+1,k+1}\|$ in the spectral and Frobenius norms. If $R_{k+1,k+1}$ is sufficiently small, the approximation $\hat{X}$ solves our problem. We will call (2) the truncated pivoted QR approximation to X.

The singular value and pivoted QR decompositions loose some of their their appeal when X is large and sparse. The problem is that the conventional algorithms for computing these decomposition proceed by transformations that quickly destroy the sparsity of X.[3] For the singular value decomposition, recourse must be had to iterative approximations. The purpose of this note is to show that the pivoted QR decomposition can be implemented in such a way that the sparsity of X is not compromised.

We actually give two algorithms, one based on Gram–Schmidt orthogonalization and the other on Householder triangularization. The key idea of both is to build up the Q-factor column by column and the R-factor row by row, leaving the original matrix X unchanged. Only when a column of X is needed to form a column of Q is it transformed. The Gram–Schmidt variant computes the explicit factorization and is conceptually simpler. The algorithm based on Householder triangularization provides an implicit basis for the orthogonal complement of the column space of the decomposition. Both have essentially the same complexity.

2. The Gram–Schmidt variant

Let $X = (x_1 \;\cdots\; x_p)$ be partitioned by columns, and let $X_k = (x_1 \;\cdots\; x_k)$. Suppose we have computed a QR factorization.

$$X_{k-1} = Q_{k-1}R_{11},$$

where R_{11} is upper triangular of order $k-1$. Then the corresponding QR approximation is

$$\hat{X}_{k-1} = Q_{k-1}(R_{11} \;\; R_{1k}),$$

where

$$R_{1k} = Q^{\mathrm{T}}(x_k \;\cdots\; x_p).$$

Now suppose we want to update this approximation to include X_k. This can be accomplished by one step of the Gram–Schmidt algorithm:

[3] It should be stressed that the problem is not in the lack of sparsity of approximation $\hat{X}$, which in both approaches is represented economically in factored form.

$$
\begin{array}{ll}
1. & r_{1k} = Q_{k-1}^{\mathrm{T}} x_k \\
2. & q_k = x_k - Q_{k-1} r_{1k} \\
3. & r_{kk} = \|q_k\| \\
4. & q_k = q_k / r_{kk}.
\end{array}
\tag{3}
$$

Then

$$
X_k = (X_{k-1}\ \ x_k) = (Q_{k-1}\ \ q_k) \begin{pmatrix} R_{11}^{(k-1)} & r_{1k} \\ 0 & r_{kk} \end{pmatrix}
$$

is the updated QR factorization. To compute the rest of the approximation, we need only compute the kth row of R, which has the form

$$
r_{k,k+1}^{\mathrm{T}} = q_k^{\mathrm{T}} (x_{k+1}\ \cdots\ x_p). \tag{4}
$$

This updating procedure is very close to what we promised above. Each step generates a new column of Q and a new row of R. The only part of X that is modified is the column that eventually becomes the new column of Q. The only other use made of X is a vector-matrix product in (4) to fill out the kth row of R. However, there remain two problems.

The first problem is that to get good approximations we must incorporate column pivoting into our algorithm. Specifically, at the beginning of the kth stage of the algorithm, we must select the column x_j of $(x_k\ \cdots\ x_p)$ for which $\|(I - Q_{k-1}Q_{k-1}^{\mathrm{T}})x_j\|$ is maximal and interchange it with x_k. The computation of $(I - Q_{k-1}Q_{k-1}^{\mathrm{T}})x_j$ for all $j \geq k$ effectively transforms X into a dense matrix.

The answer to the this problem lies in the following well-known formula:

$$
\|(I - Q_{k-1}Q_{k-1}^{\mathrm{T}})x_j\| = \|x_j\|^2 - \sum_{i=1}^{k-1} r_{ij}^2.
$$

This formula allows us to downdate the column norms of X as we compute rows of R to give the norms of the projected columns.

The second problem is that the Gram–Schmidt algorithms as implemented above is not guaranteed to produce a matrix Q_k whose columns are orthogonal to working accuracy. The cure for this problem is a process in which q_k is reorthogonalized against the previous q_j. The details of this process, which also alters the kth columns of R have been well treated in [1], and we omit them.

Figure 1 contains an algorithm implementing this method. The algorithm continues until the projected columns norms, which correspond to the norms of the columns of $R_{k+1,k+1}$ in (1), become sufficiently small, at which point the process terminate with the rank k approximation $\hat{X} = Q[:, 1{:}k] * R[1{:}k, 1{:}p]$. For ease of exposition we have made explicit interchanges in X. The only unusual feature is the omission of the step

Given an $n \times p$ ($n \geq p$) matrix X this algorithm returns a truncated pivoted QR decomposition of X. Initially, the matrices Q and R are void.

1. $\nu_j = \|X[:,j]\|^2, \qquad j = 1, \ldots, p$
2. Determine an index p_1 such that ν_{p_1} is maximal
3. **for** $k = 1$ **to** p
4. $\quad X[:,k] \leftrightarrow X[:,p_k]$
5. $\quad R[1{:}k{-}1,k] \leftrightarrow R[1{:}k{-}1,p_k]$
6. $\quad Q[:,k] = X[:,k] - Q[:,1{:}k{-}1]{*}R[1{:}k{-}1,k]$
7. $\quad R[k,k] = \|Q[:,k]\|$
8. $\quad Q[:,k] = Q[:,k]/R[k,k]$
9. $\quad$ If necessary reorthogonalize $Q[:,k]$ and adjust $R[1{:}k{-}1,k]$
10. $\quad R[k,k{+}1{:}p] = Q[:,k]^{\mathrm{T}}{*}X[:,k{+}1{:}p]$
11. $\quad \nu_j = \nu_j - R[k,j]^2, \qquad j = k{+}1, \ldots, n$
12. $\quad$ Determine an index $p_{k+1} \geq k{+}1$ such that $\nu_{p_{k+1}}$ is maximal
13. $\quad$ **if** ($\nu_{p_{k+1}}$ is sufficiently small) **leave** k **fi**
14. $\quad$ **end for** k.

Figure 1: Truncated pivoted QR factorization: Gram–Schmidt version

1. $\qquad R[1{:}k{-}1,k] = Q[:,1{:}k{-}1]^{\mathrm{T}}{*}X[:,k]$

that corresponds to statement 1 in (3). The reason is that these numbers have already been computed as we have added rows to R.

3. Orthogonal triangularization

We now turn to using orthogonal triangularization to compute truncated pivoted QR approximations. We begin with a brief description of how the pivoted QR decomposition is computed via Householder transformations. Before the kth stage of the reduction, we have determined Householder transformations $H_1, \ldots, H_{k-1}$ and a permutation matrix Π_{k-1} such that

$$H_{k-1} \cdots H_1 X \Pi_{k-1} = \begin{pmatrix} R_{11} & \bar{R}_{1k} \\ 0 & \bar{X}_{kk} \end{pmatrix}, \tag{5}$$

where R_{11} is triangular of order $k{-}1$. One then determines the column of largest 2-norm of $\bar{X}_{kk}$ and swaps it with the first, along with the corresponding columns of R_{1k}, to get

$$H_{k-1} \cdots H_1 X \Pi_{k-1} \bar{\Pi}_k = \begin{pmatrix} R_{11} & \tilde{R}_{1k} \\ 0 & \tilde{X}_{kk} \end{pmatrix}.$$

Here $\bar{\Pi}_k$ is a permutation matrix representing the interchange of columns. Now a Householder transformation $\tilde{H}_k = I - \tilde{u}_k\tilde{u}_k^{\mathrm{T}}$ is determined so that

$$\tilde{H}_k\tilde{X}_{kk} = \begin{pmatrix} r_{kk} & \bar{r}_{k,k+1}^{\mathrm{T}} \\ 0 & \bar{X}_{k+1,k+1} \end{pmatrix}.$$

If we set $u_k^{\mathrm{T}} = (0\ \tilde{u}_k^{\mathrm{T}})$, $H_k = I - u_k u_k^{\mathrm{T}}$, and $\Pi_k = \Pi_{k-k}\bar{\Pi}_k$, then (5) holds with k advanced by one.

If after the kth stage the largest column norm of $\bar{X}_{k+1,k+1}$ is sufficiently small, we may terminate the procedure with a truncated approximation to X of rank k, in which the matrix Q^{T} is represented as the product $H_k \cdots H_1$.

In these terms, the pivoting strategy introduced above amounts to pivoting on the column of X for which the corresponding column norm of $\bar{X}_{kk}$ is maximal. As above, the square of the norm of the jth column $x_j^{(kk)}$ is (ignoring interchanges) given by

$$\|x_j^{(kk)}\|^2 = \|x_j\|^2 - \sum_{i=1}^{k-1} r_{ij}^2.$$

If we know the first $k-1$ rows of R, we can downdate the column norms of X to give the column norms of $\bar{X}_{kk}$.

Moreover to compute the Householder transformation at the kth stage we need only one column from $\bar{X}_k$ to compute u_k. This column can be computed on the spot from the corresponding column of X, used to generate u_k, and discarded. Thus there is no need to transform the entire matrix X.

These ideas are best implemented using an elegant representation for the product of Householder transformations due to Schreiber and Van Loan [2]. Specifically, if we set

$$U_k = (u_1\ \cdots\ u_k),$$

then

$$(I - u_k u_k^{\mathrm{T}})\cdots(I - u_1 u_1^{\mathrm{T}}) = U_k T_k^{\mathrm{T}} U_k^{\mathrm{T}}, \tag{6}$$

where the unit lower triangular matrix T_k can be generated by the following recurrence:

$$T_k = \begin{pmatrix} T_{k-1} & -T_{k-1}U_{k-1}^{\mathrm{T}}u_k \\ 0 & 1 \end{pmatrix}. \tag{7}$$

We will call (6) a UTU representation of the product on the left-hand side.

If we maintain the pivoted Q-factor of X in UTU form, we can compute the truncated pivoted factorization without modifying X. Specifically, suppose that just before the

Given an $n \times p$ $(n \geq p)$ matrix X this algorithm returns a truncated pivoted QR decomposition of X. The Q-factor is returned in UTU form. Initially the matrices U, T, R, and S are void.

$\nu_j = \|X[:,j]\|^2, \qquad j = 1, \ldots, p$
Determine an index p_1 such that ν_{p_1} is maximal
for $k = 1$ **to** p
  $X[:,k] \leftrightarrow X[:,p_k]$
  $R[1{:}k{-}1, k] \leftrightarrow R[1{:}k{-}1, p_k]$
  $S[1{:}k{-}1, k] \leftrightarrow S[1{:}k{-}1, p_k]$
  $\bar{x} = X[:,k] - U[:,1{:}k{-}1]{*}T^{\mathrm{T}}{*}U[:,1{:}k{-}1]^{\mathrm{T}}{*}X[:,k]$
  Determine u from $\bar{x}$
  $T = \begin{pmatrix} T & -T{*}U^{\mathrm{T}}{*}u \\ 0 & 1 \end{pmatrix}$
  $U = (U \;\; u)$
  $S = \begin{pmatrix} S \\ u^{\mathrm{T}}X \end{pmatrix}$
  $R = \begin{pmatrix} R \\ X[k,:] - U[k,:]{*}T^{\mathrm{T}}{*}S \end{pmatrix}$
  $\nu_j = \nu_j - R[k,j]^2, \qquad j = k{+}1, \ldots, n$
  Determine an index $p_{k+1} \geq k{+}1$ such that $\nu_{p_{k+1}}$ is maximal
  if ($\nu_{p_{k+1}}$ is sufficiently small) **leave** k **fi**
end for k

Figure 2: Truncated pivoted QR decomposition: UTU version

kth step, we have U_{k-1}, T_{k-1}, and an auxiliary matrix $S_{k-1} = U_{k-1}^{\mathrm{T}} X_{k-1}$, along with the current downdated column norms of X. Then the pivot column can be determined from the downdated norms. If x denotes the value of this column in the original matrix X, the transformed value can be calculated as $\bar{x} = x - U_{k-1} T_{k-1}^{\mathrm{T}} U_{k-1}^{\mathrm{T}} x$. From this the kth Householder vector u_k can be computed, and U_{k-1}, T_{k-1}, and S_{k-1} can be updated. Finally, if u^k denotes the kth row of U_k and x^k denotes the kth row of X, then the kth row of R can be computed in the form $x^k - u^k T_k S_k$. From this the new column norms can be computed and the process repeated.

The code in Figure 2 implements this sketch. As with the Gram–Schmidt version, the algorithm is quite efficient in terms of both operations and storage. Excepting interchanges, which we have incorporated explicitly to simplify the exposition, the array X is never modified. One column of it is used at each stage to compute the current Householder transformation, and the vector-matrix product $u^{\mathrm{T}}X$ must be formed to update the auxiliary matrix S. If the additional storage for S is too burdensome, the rows of R may be dropped after they have been used to downdate the column norms, and R can be reconstituted in S when the algorithm terminates. At this time the one may go on to compute the first k columns of Q explicitly. Alternatively, one can leave the entire matrix Q in UTU form.

References

[1] Å. Björck. *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia, 1996.

[2] R. Schreiber and C. F. Van Loan. A storage efficient WY representation for products of Householder transformations. *SIAM Journal on Scientific and Statistical Computing*, 10:53–57, 1989.