Efficient rank calculation for matrices with a common submatrix

P.T. Eendebak*

April, 2016

Abstract

Rank calculation for matrices is important in many applications. In this article we descibe how to perform efficient rank calculations for matrices that share a common submatrix. We use a QR decomposition of the submatrix to improve the speed of the rank calculation of the matrices in the list.

1 Introduction

In this paper we analyse the calculation of the rank of a series of matrices A_i which have a number of columns in common. By using QR-decompositions of the common submatrices we can significantly speed up the rank calculations.

2 Theory

The rank of a matrix is defined as the maximum number of linearly independent column vectors of the matrix [Lang, 1993]. Most methods to calculate the rank of a matrix are based on reduction to normal form (e.g. row echelon form) or rank-revealing decompositions.

For example, the rank of a matrix is equal to the number of non-zero eigenvalues in a singular value decomposition of the matrix. This makes it possible to calculate the rank of a matrix by performing a SVD (singular value decomposition). Due to rounding errors the eigenvalues with absolute value smaller than a threshold (typically of the order of 10^{-14}) are counted as zero. An alternatative method is to calculate the QR decomposition of a matrix. See [Eig, 2016] for a list of rank revealing decompositions.

Most rank calculations are computationally expensive because the decompositions are expensive to calculate. For a matrix of size $m \times n$ the complexity of a QR decompostion (using Householder QT decomposition) is of order $2mn^2$.

Rank calculations for matrices with a common submatrix Let A_i be a series of matrices of the form $[AB_i]$ and suppose we want to calculate the rank of each of these matrices.

First perform a QR decomposition on the submatrix A:

$$AP = QR, \tag{1}$$

with P a permutation matrix, Q an orthogonal matrix and R an upper triangular matrix. The permutation matrix P is not stricly necessary, but allows for numerically more stable decompositions. We denote the rank of the matrix A (which can be determined by the number of non-zero diagonal elements of R) as $\text{rank}(R) = r_0$.

For each matrix A_i we calculate

$$X_i = Q^{-1} A_i P^{-1} = Q^T A_i P^T.$$

The rank of X_i is equal to the rank of A_i bacause both Q and P are invertible. Due to the QR decomposition, the matrix X_i is has the first r_0 in upper triangular form. See Figure 1 for an illustration

*Corresponding author. E-mail: pieter.eendebak@gmail.com

Data: List of arrays A_i of size $n \times m$, integer s
Result: Matrix rank for all of the arrays
Sort the list of arrays;
Calculate the QR decomposition of $S = A_0[:, 1:s]$;
for *Each array A_i in the list* **do**
    if $A_i[1:s] \,!= S$ **then**
        Set S equal to $A_i[1:s]$;
        Calculate the QR decomposition of S;
        Determine the rank r_0 of S;
    end
    Calculate $X_i = Q^T A_i P^T$;
    Let Y_i be the submatrix of X_i obtained by removing the first r_0 rows and first r_0 columns.;
    Calculate the rank of A_i as $r_0 + \mathrm{rank}(Y_i)$;
end

Algorithm 1: Rank calculation for a list of arrays.

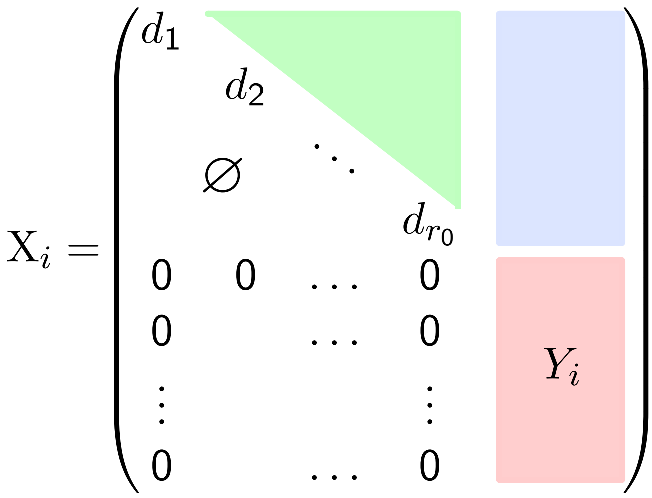

Figure 1: Shape of matrix X_i.

of the matrix X_i. Next, take the submatrix Y_i of X_i which is obtained by dropping the first r_0 rows and the first r_0 columns

$$Y_i = X_i[r_0 :, r_0 :]. \tag{2}$$

Then

$$\mathrm{rank}(A_i) = \mathrm{rank}(X_i) = \mathrm{rank}(A) + \mathrm{rank}(Y_i) = r_0 + \mathrm{rank}(Y_i). \tag{3}$$

Instead of calculating the rank of A_i directly we can determine the rank if Y_i, which is much smaller in size than the full matrix A_i.

Method To efficiently calculate the rank of list of matrices we use the algorithm described in Algorithm 1. The performance of the algorithm depends on how often the submatrix $X_i[1:s]$ is equal to the submatrix S for which the QR-decomposition has been calculated. It also depends on the values n, m and s.

3 Results

We analyse the performance of our method on a set of arrays with 64 rows and 15 columns. This set is a subset of the even-odd desgins $OA(64, 3, 2^{15})$.

Method	Marrays /s
SVD	353.8
LU decomposition	3042.4
QR decomposition	3547.619
QR-cache2	10321.493
QR-cache4	9750.001
QR-cache3	9557.565

Table 1: Results for rank calculations of second order interaction matrices of orthogonal arrays in $\mathrm{OA}(64; 3; 2^{15})$.

We compute the computation times for five different methods to calculate the rank. All methods have been implemented in C++ with the Eigen [Guennebaud et al., 2010] library. The code is available in the Orthogonal Array Package [Eendebak, 2013] (example code in `oaranktest.cpp`).

- **SVD** Calculate the SVD of the matrix and count the non-zero eigenvalues.
- **LU** Calculate the LU-decomposition of the matrix using Eigen.
- **QR** Calculate the QR-decomposition of the matrix.
- **QR-sub2** Calculate the rank using the algorithm in Algorithm 1 with parameter $s = m - 2$.
- **QR-sub3** Calculate the rank using the algorithm in Algorithm 1 with parameter $s = m - 3$.
- **QR-sub4** Calculate the rank using the algorithm in Algorithm 1 with parameter $s = m - 4$.

The resulting computation speeds are listed in Table 1. The computation speed is given in million arrays per second. The calculations are done on a Intel Core i7-4710MQ processor.

4 Discussion

Of the the methods that do not use the common submatrices the QR decomposition is the fastest. Our method using the common substructure of the matrices is a factor 2.5–3 faster. This is possible because the arrays in the dataset often only differ in the last few columns. The parameter s is a trade-off between beter efficiency for calculating the rank (higher s) and preventing recalculation of the QR decompostion of S (lower s). For this particular dataset the value of s is not very critical.

For other datasets the improvement in calculation speed may be better or worse, depending on the characteristics of the dataset.

References

[Eig, 2016] (2016). Catalogue of decompositions offered by eigen.

[Eendebak, 2013] Eendebak, P. T. (2013). The Orthogonal Array package. Technical report.

[Guennebaud et al., 2010] Guennebaud, G., Jacob, B., et al. (2010). Eigen v3. http://eigen.tuxfamily.org.

[Lang, 1993] Lang, S. (1993). *Algebra*. Addison-Wesley, Menlo Park Cal.