Adventures in Supercomputing with R

Lecture 6: More BLAS and Factorizations

George Ostrouchov

Oak Ridge National Laboratory and University of Tennessee

2022/02/20 (updated: 2022-03-23)

Exercise 4 Discussion

- Please resend my invitation to your ASwR GitHub repository
  - They expire after 7 days and I did not accept some in time
  - Consequently the deadline is extended through Thursday, March 24.
- I did see several team solutions
- Best time so far is ~18.5 seconds
- We can address remaining GitHub access issues after today's lecture

Matrix Computation

- Last time we looked at benchmarks for
  - `lm()`, `qr()`, `prcomp()`, `princomp()`, `crossprod()`, and `%*%`
- The first four are computed with matrix decomposition
- **Why do matrix decomposition? Seems like a lot of work!**

Why Matrix Decompositions?

Consider a standard regression problem

$$y = X\beta + \epsilon,$$

where X is an $n \times p$ design matrix (data matrix), y is a response vector, and ϵ is an error vector.

If we use least squares to slove for β, we have

$$\hat{\beta} = \arg\min_{\beta} \|y - X\beta\|_2.$$

The *normal equations* are usually shown as a linear system to solve for β

$$X^T X\beta = X^T y$$

and the solution is $\hat{\beta} = (X^T X)^{-1} X^T y$, when X is full rank.

How do we compute this? Is it fast? Is it accurate?

Why Matrix Decompositions?

If a matrix inverse is not your final result, there is a cheaper way to get there

- Solving $Ax = b$ for x by LU factorization is cheaper than using A^{-1}
  - Factor $A = LU$, then solve two triangular systems using back substitution: $Ly = b$ and $Ux = y$
  - Solution by LU decomposition is under half the work
- If A is symmetric and positive definite, as in $A = X^T X$, we have Cholesky factorization $A = R^T R$, where R is upper triangular
  - Another factor of 2 cheaper

Why Matrix Decompositions?

What about accuracy?

- The *condition number* of a matrix A is $\kappa(A) = \frac{\sigma_{\max}}{\sigma_{\min}} \geq 1$, assuming full rank, under L_2 norm
  - When A is viewed as a projection, it is a ratio of maximum to minimum stretching
  - In $Ax = b$, a bound on changes in x relative to changes in b
  - For least squares, that's change in estimate $\hat{\beta}$ relative to change in response y
  - Errors in finite representation of y can be magnified by the condition number

- Forming $X^T X$ squares the condition number

- We can use decompositions of X instead of $X^T X$ for solving the least squares problem

Surprises in Finite Representation

```
x = float::as.float(1/1:1e8)  # reducing to 32 bit representation
sum(x)                        # summing forward
```

```
## # A float32 vector: 1
## [1] 15.404
```

```
sum(rev(x))                   # summing backward
```

```
## # A float32 vector: 1
## [1] 18.808
```

```
5*.Machine$double.eps          # back to 64 bit
```

```
## [1] 1.110223e-15
```

```
(2 + 5*.Machine$double.eps) - 2  # cancellation example
```

```
## [1] 8.881784e-16
```

Why Matrix Decompositions?

- Decompositions introduce zeros to form a *triangular matrix* for a back substitution solution

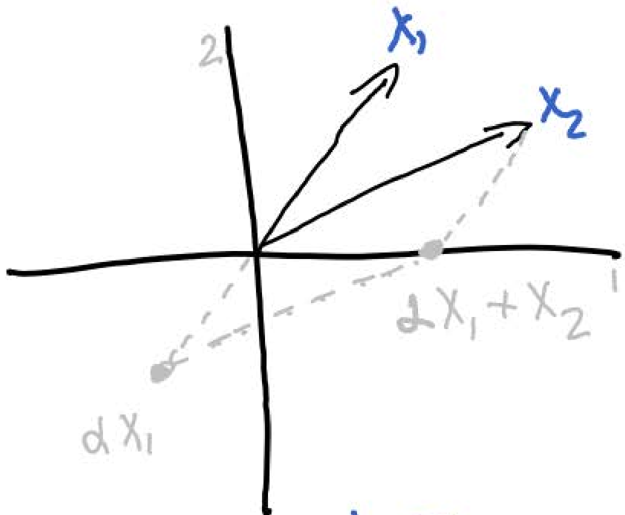
- $A = LU$ and $A = R^T R$ are fast for $A = X^T X$

  - Use *Gaussian elimination* (linear combination of rows) to introduce zeros

- Work with $n \times p$ matrix X to avoid squaring conditon number

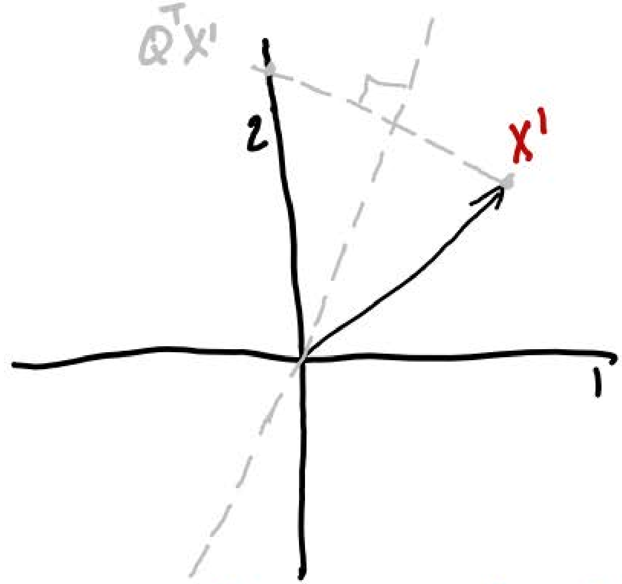
- $X = QR$, orthogonal Q and an upper triangular R
  - $Q^T Q = I_p$ and $QQ^T = I_n$ for an $n \times p$ matrix X
    - Usually Householder reflections
  - Orthogonal matrices have condition number 1: they do not magnify error

Gaussian elimination



row vectors

Householder reflection



column vectors

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x^1 & x^2 \end{bmatrix} \qquad U \begin{bmatrix} x & x \\ 0 & x \end{bmatrix} \qquad R \begin{bmatrix} x & x \\ 0 & x \end{bmatrix}$$

$$A = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} \qquad L \begin{bmatrix} 1 & 0 \\ -\alpha & 1 \end{bmatrix} \qquad Q \begin{bmatrix} x & x \\ x & x \end{bmatrix}$$

Using QR in Least Squares

Begin with $\hat{\beta} = \arg\min_{\beta} \|y - X\beta\|_2$.

Substitute $X = QR$ to get $\hat{\beta} = \arg\min_{\beta} \|y - QR\beta\|_2$.

Now, note that $\|Q\|_2 = 1$ and $Q^T Q = I_p$, so that $\hat{\beta} = \arg\min_{\beta} \|Q^T y - R\beta\|_2$.

So the the minumum is given by solving the triangular system $Q^T y = R\hat{\beta}$ for $\hat{\beta}$ by back substitution.

The `lm()` function and many other model fitting functions in R use the QR decomposition.

The Singular Value Decomposition

$$X = UDV^T$$

For $n \geq p$,
U is an $n \times p$ orthogonal matrix of left singular vectors
V is a $p \times p$ orthogonal matrix of right singular vectors
D is a $p \times p$ diagonal matrix of singular values

Exercise: show how to use this decomposition to solve the least squares problem (Hint: follow QR ideas):

$$\hat{\beta} = \arg\min_{\beta} \|y - X\beta\|_2.$$

Truncated SVD as Regression Basis Vectors

Suppose we have n images, each with p pixel values. The well known MNIST data set of digit images is an example with $n = 60\,000$ and $p = 784\ (28 \times 28)$.

Let A be the matrix of n_A images of a single digit, the pixel values of each image as a column. That is, we partition the digits into separate matrices, like A, and consider using them for classification of new images.

The SVD of $A = UDV^T$. If u_i and v_i are the columns of U and V, respectively, then

$$A = \sum_{i=1}^{p} d_i u_i v_i^T.$$

and image j in column $a_j = \sum_{i=1}^{p} (d_i v_{ij}) u_i$.

From matrix approximation, we know that this SVD can be truncated to some $k \ll p$ components and still represent each image well.

Truncated SVD as Regression Basis Vectors

For some $k \ll p$, we have $x_j = \sum_{i=1}^{k} (d_i v_{ij}) u_i$.

The u_i are basis functions constructed from data, a set of orthogonalized "images", which are the regressors and the $d_i v_{ij}$ are the regression coefficients.

We can now look at classification of a new image of a digit by regressing it onto each of the 10 digit bases and classifying it into the category that fits best.

The tuning parameter k can be optimized with crossvalidation.

References

Trefethen, L. N., Bau, D. (1997). Numerical Linear Algebra. SIAM. ISBN: 0898713617

- Great book for intuition, although fairly advanced

Golub, Gene H. and van Loan, Charles F.. Matrix Computations. Fourth : JHU Press, 2013.

- Encyclopedic book for algorithms and the theory behind them and error analysis