STAT 309: MATHEMATICAL COMPUTATIONS I
FALL 2015
LECTURE 8

1. QR and complete orthogonal factorization

- poor man's SVD
- can solve many problems on the SVD list using either of these factorizations
- but they are much cheaper to compute — there are direct algorithms for computing QR and complete orthogonal factorization in a finite number of arithmetic steps
- recall that SVD is spectral in nature — only iterative algorithms in general by Galois–Abel, although for any fixed precision (fixed number of decimal places), we can compute SVD in finitely many steps
- there are several versions of QR factorization
- version 1: for any $A \in \mathbb{C}^{m\times n}$ with $n \leq m$, there exist a unitary matrix $Q \in \mathbb{C}^{m\times m}$ (i.e., $Q^*Q = QQ^* = I_n$) and an upper-triangular matrix $R \in \mathbb{C}^{m\times n}$ (i.e., $r_{ij} = 0$ whenver $i > j$) such that
$$A = QR = Q\begin{bmatrix} R_1 \\ 0 \end{bmatrix} \tag{1.1}$$
  - $R_1 \in \mathbb{C}^{n\times n}$ is an upper-triangular square matrix in general
  - if A has full column rank, i.e., $\operatorname{rank}(A) = n$, then R_1 is nonsingular
  - this is called the *full* QR factorization of A
- version 2: for any $A \in \mathbb{C}^{m\times n}$ with $n \leq m$, there exist a unitary matrix $Q_1 \in \mathbb{C}^{m\times n}$ (i.e., $Q_1^*Q_1 = I_n$ but $Q_1Q_1^* \neq I_m$ unless $m = n$) and an upper-triangular square matrix $R_1 \in \mathbb{C}^{n\times n}$ such that
$$A = Q_1R_1 \tag{1.2}$$
  - R_1 here is in fact the same R_1 as in (1.1)
  - Q_1 is the first n columns of Q in (1.1), i.e., $Q = [Q_1, Q_2]$ where $Q_2 \in \mathbb{C}^{m\times(m-n)}$ is the last $m - n$ columns of Q
  - in fact we obtain (1.2) from (1.1) by simply multiplying out
$$A = QR = [Q_1, Q_2]\begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1R_1 + Q_20 = Q_1R_1$$
  - as before, if A has full column rank, i.e., $\operatorname{rank}(A) = n$, then R_1 is nonsingular
  - this is called the *reduced* QR factorization of A
- version 3: for any $A \in \mathbb{C}^{m\times n}$ with $\operatorname{rank}(A) = r$, there exist a permutation matrix $\Pi \in \mathbb{C}^{n\times n}$, a unitary matrix $Q \in \mathbb{C}^{m\times m}$, and a nonsingular, upper-triangular square matrix $R_1 \in \mathbb{C}^{r\times r}$ such that
$$A\Pi = Q\begin{bmatrix} R_1 & S \\ 0 & 0 \end{bmatrix} \tag{1.3}$$
  - $S \in \mathbb{C}^{r\times(n-r)}$ is just some matrix with no special properties
  - this is called the *rank-retaining* QR decomposition of A form

Date: October 20, 2015, version 1.0. Comments, bug reports: lekheng@galton.uchicago.edu.

– we may also write (1.3) as

$$A = QR\Pi^{\mathsf{T}} = Q \begin{bmatrix} R_1 & S \\ 0 & 0 \end{bmatrix} \Pi^{\mathsf{T}} \tag{1.4}$$

- version 4: for any $A \in \mathbb{C}^{m\times n}$ with $\operatorname{rank}(A) = r$, there exist a unitary matrix $Q \in \mathbb{C}^{m\times m}$, a unitary matrix $U \in \mathbb{C}^{n\times n}$, and a nonsingular, lower-triangular square matrix $L \in \mathbb{C}^{r\times r}$ such that

$$A = Q \begin{bmatrix} L & 0 \\ 0 & 0 \end{bmatrix} U^* \tag{1.5}$$

  – this is called the *complete orthogonal* factorization of A
  – it can be obtained from a full QR factorization of $\left[\begin{smallmatrix} R_1^* \\ S^* \end{smallmatrix}\right] \in \mathbb{C}^{m\times r}$, which has full column rank,

$$\begin{bmatrix} R_1^* \\ S^* \end{bmatrix} = Z \begin{bmatrix} R_2 \\ 0 \end{bmatrix} \tag{1.6}$$

  where $Z \in \mathbb{C}^{m\times m}$ is unitary and $R_2 \in \mathbb{C}^{r\times r}$ is nonsingular, upper-triangular square matrix
  – observe from (1.4) and (1.6) that

$$A = Q \begin{bmatrix} R_1 & S \\ 0 & 0 \end{bmatrix} \Pi^{\mathsf{T}} = Q \begin{bmatrix} R_2^* & 0 \\ 0 & 0 \end{bmatrix} Z^* \Pi^{\mathsf{T}} = Q \begin{bmatrix} L & 0 \\ 0 & 0 \end{bmatrix} U^*$$

  where we set $L = R_2^*$ and $U = \Pi Z$.
- note that for a matrix that is not of full column rank, a QR decomposition would necessarily mean either versions 3 or 4
- there are yet other variants of QR factorizations that can be obtained using essentially the same algorithms (Givens and Householder QR):

$$A = QR, \quad A = LQ, \quad A = RQ, \quad A = QL$$

  where Q is unitary, R is upper triangular, and L is lower triangular
- using such variants, we could for instance make the lower triangular matrix L in (1.5) an upper-triangular matrix instead
- the QR factorization is a generalization of the polar form of a complex number $a \in \mathbb{C}$,

$$a = re^{i\theta}$$

  to matrices

2. Aside: permutation matrices

- the permutation matrix Π in (1.3) comes from performing *column pivoting* in the algorithm
- recall that a permutation matrix is a simply the identity matrix with the rows and columns permuted, e.g.

$$\Pi = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{2.1}$$

- multiplying a matrix $A \in \mathbb{C}^{m\times n}$ by an $n \times n$ permutation matrix on the right, i.e., $A\Pi$, has the effect of permuting the *columns* of A according to precisely the way the columns of Π are permuted from the identity, e.g.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} c & a & b \\ f & d & e \\ i & g & h \end{bmatrix}$$

- multiplying a matrix $A \in \mathbb{C}^{m\times n}$ by an $m \times m$ permutation matrix on the left, i.e., ΠA, has the effect of permuting the *rows* of A according to precisely the way the rows of Π are permuted from the identity, e.g.

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} d & e & f \\ g & h & i \\ a & b & c \end{bmatrix}$$

- multiplying a square matrix $A \in \mathbb{C}^{n\times n}$ by an $n \times n$ permutation matrix on the left and its transpose on the right, i.e., $\Pi A \Pi^{\mathsf{T}}$, has the effect of permuting the *diagonal* of A — entries on the diagonal stays on the diagonal and entries off the diagonal stays off diagonal, e.g.

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}^{\mathsf{T}} = \begin{bmatrix} e & f & d \\ h & i & g \\ b & c & a \end{bmatrix}$$

  note that a, e, i stays on the diagonal as expected
- permutation matrices are always orthogonal (also unitary since it has real entries), i.e.

$$\Pi^{\mathsf{T}}\Pi = \Pi\Pi^{\mathsf{T}} = I$$

  or $\Pi^{-1} = \Pi^{\mathsf{T}} = \Pi^*$
- we don't store permutation matrices as matrices of floating point numbers, we store just the permutation, e.g. (2.1) can be stored as $3 \mapsto 1 \mapsto 2 \mapsto 3$ since it takes column 3 to column 1, column 1 to column 2, column 2 to column 3

3. existence and uniqueness of QR

- if $A \in \mathbb{C}^{m\times n}$ has full column rank, i.e., $\operatorname{rank}(A) = n \leq m$, then we will show existence and (some kind of) uniqueness of its reduced QR factorization
- uniqueness is easy if $m = n$
  - suppose
$$A = Q_1R_1 = Q_2R_2$$
    for $Q_1, Q_2 \in \mathbb{C}^{n\times n}$ are unitary and $R_1, R_2 \in \mathbb{C}^{n\times n}$ are nonsingular
  - then
$$Q_2^*Q_1 = R_2R_1^{-1}$$
  - note that the left-hand side is unitary and right hand side is upper-triangular
  - the only matrix that is both unitary and upper-triangular is a diagonal matrix of the form
$$D = \operatorname{diag}(e^{i\theta_1}, \dots, e^{i\theta_n})$$
  - so we get
$$Q_2 = Q_1D^*, \qquad R_2 = DR_1$$
  - QR factorization is unique up to such unimodular scaling
- more generally, we could also get uniqueness without requiring $m = n$ this follows from Gram–Schmidt, which we could also use to establish existence

4. Gram–Schmidt orthogonalization

- suppose $A \in \mathbb{C}^{n\times n}$ is square and full-rank
- so all the column vectors of A are linearly independent
- consider the QR factorization

$$A = \begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_n \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1 & \cdots & \mathbf{q}_n \end{bmatrix} \begin{bmatrix} r_{11} & \cdots & r_{1n} \\ & \ddots & \vdots \\ & & r_{nn} \end{bmatrix} = QR$$

- from this matrix equation, we get

$$\begin{aligned}\mathbf{a}_1 &= r_{11}\mathbf{q}_1\\ \mathbf{a}_2 &= r_{12}\mathbf{q}_1 + r_{12}\mathbf{q}_2\\ &\vdots\\ \mathbf{a}_n &= r_{1n}\mathbf{q}_1 + r_{2n}\mathbf{q}_2 + \cdots + r_{nn}\mathbf{q}_n\end{aligned}$$

- and from which we can deduce an algorithm
- first note that $\mathbf{a}_1 = r_{11}\mathbf{q}_1$, and so

$$r_{11} = \|\mathbf{a}_1\|_2, \quad \mathbf{q}_1 = \frac{1}{\|\mathbf{a}_1\|_2}\mathbf{a}_1$$

- next, from $\mathbf{a}_2 = r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2$ we get

$$r_{12} = \mathbf{q}_1^*\mathbf{a}_2, \quad r_{22} = \|\mathbf{a}_2 - r_{12}\mathbf{q}_1\|_2, \quad \mathbf{q}_2 = \frac{1}{r_{22}}(\mathbf{a}_2 - r_{12}\mathbf{q}_1)$$

- in general, we get

$$\mathbf{a}_k = \sum_{j=1}^{k} r_{jk}\mathbf{q}_j$$

- and hence

$$\mathbf{q}_k = \frac{1}{r_{kk}}\left(\mathbf{a}_k - \sum_{j=1}^{k-1} r_{jk}\mathbf{q}_j\right), \quad r_{jk} = \mathbf{q}_j^*\mathbf{a}_k$$

- note that $r_{kk} \neq 0$: since $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are linearly independent and so

$$\mathbf{a}_k \notin \operatorname{span}\{\mathbf{a}_1, \ldots, \mathbf{a}_{k-1}\} = \operatorname{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_{k-1}\}$$

  and so

$$\mathbf{a}_k - \sum_{j=1}^{k-1} r_{jk}\mathbf{q}_j \neq \mathbf{0}$$

  and so

$$r_{kk} = \left\|\mathbf{a}_k - \sum_{j=1}^{k-1} r_{jk}\mathbf{q}_j\right\|_2 \neq 0 \tag{4.1}$$

- this is the *Gram–Schmidt* algorithm, there are two ways to see it
  - given a list of linearly independent vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{C}^n$, it produces a list of orthogonormal vectors $\mathbf{q}_1, \ldots, \mathbf{q}_n$ that spans the same subspace
  - given a matrix $A \in \mathbb{C}^{n\times n}$ of full rank, it produces a QR factorization $A = QR$
- so we have established the existence of QR
- in fact, it is clear that if we started from a list of linearly independent vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{C}^m$ where $n \leq m$ or equivalently a matrix $A \in \mathbb{C}^{m\times n}$ of full column rank $\operatorname{rank}(A) = n \leq m$, the Gram–Schmidt algorithm would still produce a list of orthogonormal vectors $\mathbf{q}_1, \ldots, \mathbf{q}_n$ or equivalently a matrix $Q \in \mathbb{C}^{m\times n}$ with orthonormal columns
- the only difference is that the algorithm would terminate at step n when it runs out of input vectors
- note that this is a special QR factorization since $r_{kk} > 0$ for all $k = 1, \ldots, n$ (because r_{kk} is chosen to be a norm)
- in fact, requiring $r_{kk} > 0$ gives us uniqueness (not just uniqueness up to unimodular scaling)
- now what if $A \in \mathbb{C}^{m\times n}$ is not full rank, i.e., $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are not linearly independent
- in this case Gram–Schmidt could fail since r_{kk} in (4.1) can now be 0

- we need to modify Gram–Schmidt so that it finds a subset of $\mathbf{a}_1, \dots, \mathbf{a}_n$ that is linearly independent
- this is equivalent to finding a permutation matrix Π so that the first $r = \operatorname{rank}(A)$ columns of $A\Pi$ are linearly independent
- this can be done adaptively and corresponds to column pivoting
- we will discuss this later when we discuss Givens and Householder QR algorithms, which are what used in practice
- the truth is that Gram–Schmidt is really a lousy algorithm — it is numerically unstable
- for example, if $\mathbf{a}_1$ and $\mathbf{a}_2$ are almost parallel, then $\mathbf{a}_2 - r_{12}\mathbf{q}_1$ is almost zero and roundoff error becomes significant
- because of such numerical instability the computed $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_k$ gradually lose their orthogonality
- however it is not difficult to fix Gram–Schmidt by *reorthogonalization*, essentially by applying Gram–Schmidt a second time to the output of the first round of Gram–Schmidt $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_k$
- in exact arithmetic, $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_k$ is already orthogonal and applying Gram–Schmidt a second time has no effect
- but in the presence of rounding error, reorthogonalization has real effect — making the output of the second round orthogonal
- the nice thing is that there is no need to do a third round of Gram–Schmidt — twice suffices (for subtle reasons)

5. modified Gram–Schmidt algorithm

- we didn't discuss this in lectures but I'm adding this discussion of modified Gram–Schmidt, a way to improve the numerical stability of Gram–Schmidt
- note that $\mathbf{q}_k$ can be rewritten as

$$\mathbf{q}_k = \frac{1}{r_{kk}} \left(\mathbf{a}_k - \sum_{j=1}^{k-1} (\mathbf{q}_j^* \mathbf{a}_k) \mathbf{q}_j \right) = \frac{1}{r_{kk}} \left(\mathbf{a}_k - \sum_{j=1}^{k-1} \mathbf{q}_j \mathbf{q}_j^* \mathbf{a}_k \right) = \frac{1}{r_{kk}} \left(I - \sum_{j=1}^{k-1} \mathbf{q}_j \mathbf{q}_j^* \right) \mathbf{a}_k$$

- if we define $P_i = \mathbf{q}_i \mathbf{q}_i^* \in \mathbb{C}^{n \times n}$, then P_i is an *orthogonal projector* that satisfies $P_i^2 = P_i$ and $P_i P_j = 0$ if $i \neq j$
- we can write

$$\mathbf{q}_k = \frac{1}{r_{kk}} \left(I - \sum_{j=0}^{k-1} P_j \right) \mathbf{a}_k = \frac{1}{r_{kk}} \prod_{j=1}^{k-1} (I - P_j) \mathbf{a}_k$$

- although the classical Gram–Schmidt process is numerically unstable, the *modified Gram–Schmidt* method partially alleviates this difficulty
- note that

$$A = QR = \begin{bmatrix} r_{11}\mathbf{q}_1 & r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2 & \cdots \end{bmatrix}$$

- we define

$$A^{(k)} = \sum_{i=1}^{k-1} \mathbf{q}_i \mathbf{r}_i^{\mathsf{T}}, \quad \mathbf{r}_i^{\mathsf{T}} = \begin{bmatrix} r_{i1} & r_{i2} & \cdots & r_{ii} \end{bmatrix}$$

  which means

$$A - \sum_{i=1}^{k-1} \mathbf{q}_i \mathbf{r}_i^{\mathsf{T}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & A^{(k)} \end{bmatrix}$$

- if we write
$$A^{(k)} = \begin{bmatrix} \mathbf{z} & B \end{bmatrix}$$
then
$$r_{kk} = \|\mathbf{z}\|_2, \quad \mathbf{q}_k = \frac{1}{r_{kk}}\mathbf{z}$$
- we then compute
$$\begin{bmatrix} r_{k,k+1} & \cdots & r_{k,n} \end{bmatrix} = \mathbf{q}_k^{\mathsf{T}} B$$
which yields
$$A^{(k+1)} = B - \mathbf{q}_k \begin{bmatrix} r_{1k} & \cdots & r_{kk} \end{bmatrix}$$
- this process is numerically more stable than Gram–Schmidt although still not as good as Householder or Givens QR