

Math 515
Fall, 2008

Reading Assignment for September 14–20, 2008
These notes correspond to Chapter 11 in Trefethen and Bau.

1 Solving Full Rank Least Squares Problems

Consider the LS problem

$$\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_2,$$

where $A \in \mathcal{R}^{m \times n}$, $m \geq n$, and $\text{rank}(A) = n$.

1.1 QR Factorization and Full Rank LS Problem

To solve the full rank LS problem using the QR factorization, suppose

$$A = QR = Q \begin{bmatrix} R_1 \\ 0 \end{bmatrix},$$

where Q is an $m \times m$ orthogonal matrix and R_1 is an $n \times n$ upper triangular matrix. Note that because A has full column rank, R_1 is nonsingular, and if we define $\hat{\mathbf{b}} = Q^T \mathbf{b}$, then we can write:

$$Q^T A = R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}, \quad \text{and} \quad Q^T \mathbf{b} = \hat{\mathbf{b}} = \begin{bmatrix} \hat{\mathbf{b}}_1 \\ \hat{\mathbf{b}}_2 \end{bmatrix},$$

where we have partitioned $\hat{\mathbf{b}}$ so that $\hat{\mathbf{b}}_1 \in \mathcal{R}^n$ and $\hat{\mathbf{b}}_2 \in \mathcal{R}^{m-n}$. Using these relations we obtain:

$$\begin{aligned} \|\mathbf{b} - A\mathbf{x}\|_2^2 &= \|Q^T \mathbf{b} - Q^T A\mathbf{x}\|_2^2 \\ &= \left\| \begin{bmatrix} \hat{\mathbf{b}}_1 \\ \hat{\mathbf{b}}_2 \end{bmatrix} - \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \mathbf{x} \right\|_2^2 \\ &= \|\hat{\mathbf{b}}_1 - R_1 \mathbf{x}\|_2^2 + \|\hat{\mathbf{b}}_2\|_2^2. \end{aligned}$$

Since $\hat{\mathbf{b}}_2$ does not depend on \mathbf{x} , the norm is minimized when

$$\hat{\mathbf{b}}_1 - R_1 \mathbf{x} = \mathbf{0} \quad \Leftrightarrow \quad R_1 \mathbf{x} = \hat{\mathbf{b}}_1$$

That is, the LS solution is obtained by using backward substitution to solve the upper triangular linear system, $R_1 \mathbf{x} = \hat{\mathbf{b}}_1$. In MATLAB this might look like:

```
[Q, R] = qr(A);  
bhat = Q'*b;  
x = triu(R(1:n,1:n)) \ bhat(1:n,1);
```

Some remarks:

- `qr` is a built-in MATLAB function. For more information, see `doc qr`.

- `triu` is a built-in MATLAB function that takes the upper triangular part of the given matrix. There is a similar function `tril`.
- Although R is upper triangular, using `triu` makes sure that MATLAB knows it is exactly upper triangular. This may be needed if, due to roundoff errors, there are small entries, not exactly 0, below the diagonal.
- If the matrix is exactly upper triangular, then the backslash operator recognizes this, and efficiently uses backward substitution to solve the linear system.
- Actually, instead of the above three MATLAB statements, we could have simply done:

```
x = A \ b;
```

The backslash operator will recognize that A is rectangular, and use the QR factorization with the above three steps to compute \mathbf{x} . That is, `x = A \ b` is a short cut for doing the above three statements.

1.2 SVD and Full Rank LS Problem

To solve the full rank LS problem using the SVD, suppose

$$A = U\Sigma V^T = U \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} V^T$$

where U is an $m \times m$ orthogonal matrix, V is an $n \times n$ orthogonal matrix, and because A has full column rank, $\sigma_1 \geq \dots \geq \sigma_n > 0$. As with the QR factorization, we exploit the fact that norms are invariant under orthogonal transformation, so

$$\begin{aligned} \|\mathbf{b} - A\mathbf{x}\|_2^2 &= \|U^T\mathbf{b} - U^TAVV^T\mathbf{x}\|_2^2 \\ &= \|U^T\mathbf{b} - \Sigma V^T\mathbf{x}\|_2^2 \\ &= \|\hat{\mathbf{b}} - \Sigma\hat{\mathbf{x}}\|_2^2 \\ &= \left\| \begin{bmatrix} \hat{\mathbf{b}}_1 \\ \hat{\mathbf{b}}_2 \end{bmatrix} - \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} \hat{\mathbf{x}} \right\|_2^2 \\ &= \left\| \hat{\mathbf{b}}_1 - \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} \hat{\mathbf{x}} \right\|_2^2 + \|\hat{\mathbf{b}}_2\|_2^2 \end{aligned}$$

where $\hat{\mathbf{b}} = U^T\mathbf{b}$ and $\hat{\mathbf{x}} = V^T\mathbf{x}$. Since $\hat{\mathbf{b}}_2$ does not depend on \mathbf{x} , the norm is minimized when

$$\hat{\mathbf{b}}_1 - \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} \hat{\mathbf{x}} = \mathbf{0} \quad \Leftrightarrow \quad \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} \hat{\mathbf{x}} = \hat{\mathbf{b}}_1$$

This diagonal system involving $\hat{\mathbf{x}}$ is trivial to solve:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{b}_1/\sigma_1 \\ \hat{b}_2/\sigma_2 \\ \vdots \\ \hat{b}_n/\sigma_n \end{bmatrix}$$

and because $\hat{\mathbf{x}} = V^T \mathbf{x}$ and V is orthogonal, then we finally get \mathbf{x} by computing $\mathbf{x} = V\hat{\mathbf{x}}$. In MATLAB this might look like:

```
[U,S,V] = svd(A);
bhat = U'*b;
xhat = bhat ./ diag(S(1:n,1:n));
x = V*xhat;
```

Some remarks:

- The element-wise arithmetic operations, such as `./`, `.*`, `.^` are very useful. If you don't know this operations, see doc 'Arithmetic Operators' (the single quotes are needed).
- `diag` is a built-in MATLAB function. It can be used to create a diagonal matrix from a given vector, or (as is done here) to grab the diagonal entries of a matrix and put them into a vector.

1.3 Normal Equations and Full Rank LS Problem

To solve the full rank LS problem using the normal equations, recall that if A has full column rank, the normal equations of the LS problem $\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_2$ are given by:

$$A^T A \mathbf{x} = A^T \mathbf{b}$$

where $A^T A$ is symmetric and positive definite (spd). In general, the most efficient direct factorization method for spd matrices is the Cholesky factorization. If M is an spd matrix, then the Cholesky factorization is:

$$M = LL^T$$

where L is a lower triangular matrix with positive entries on the diagonal.

Note that if

$$A = QR = Q \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

and we force the diagonal entries of R_1 to be positive (we can always do this if $\text{rank}(A) = n$) then

$$A^T A = R^T Q^T Q R = \begin{bmatrix} R_1^T & 0 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = R_1^T R_1.$$

If we set $L = R_1^T$, and thus $L^T = R_1$, then we have the Cholesky factorization of $A^T A$. That is, the QR factorization of A can be used to obtain the Cholesky factorization of $A^T A$.

1.4 Final Remarks on Full Rank LS Problems

- For dense matrices $A \in \mathcal{R}^{m \times n}$ with no special structure, the full rank LS problem is generally solved using the QR factorization. The cost is $O(mn^2)$ arithmetic operations (multiplications, divisions, additions, subtractions).
- Computing the SVD is more expensive than computing the QR factorization. In particular, if the full SVD is computed, the cost is $O(m^2n)$ arithmetic operations. If the full factorization is not needed, and all we need is the LS solution \mathbf{x} , then some savings can be made, reducing the cost to $O(mn^2)$, but the hidden constant in the $O(\cdot)$ is larger than it is for the QR factorization.
- The normal equations system is rarely used to solve LS problems because $A^T A$ has a larger condition number than A . In particular, you should be able to show that $\kappa_2(A^T A) = (\kappa_2(A))^2$. Thus the normal equations approach is more sensitive to errors in the data. Furthermore, finite precision arithmetic can result in loss of information when explicitly forming the matrix $A^T A$.
- If the matrix is very large and sparse (i.e., has lots of zero entries) then direct factorization methods may be prohibitively expensive. In this case we usually have to use iterative methods; we'll discuss iterative methods later this semester.
- If the matrix is very large, and has some special structure, it may be possible to exploit the structure to get more efficient methods than using the standard approaches outlined in these notes. What can be done to exploit the structure to get more efficient methods is problem dependent – usually you have to be clever and draw upon your knowledge of numerical linear algebra. Your second homework assignment is a good illustration.
- We close this section with a couple of theoretical results:

Theorem. If $A \in \mathcal{R}^{m \times n}$, $\text{rank}(A) = n$, and

$$A = QR = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

then

$$\begin{aligned} \text{range}(A) &= \text{range}(Q_1) \\ \text{range}(A)^\perp &= \text{range}(Q_2). \end{aligned}$$

Theorem. If $A \in \mathcal{R}^{m \times n}$, $\text{rank}(A) = n$, and

$$A = QR = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

then the reduced QR factorization, $A = Q_1 R_1$, is unique, where Q_1 has orthonormal columns and R_1 is upper triangular with **positive** entries on the diagonal.

2 Rank Deficient Least Squares Problems

In this section we consider solving an LS problem where $A \in \mathcal{R}^{m \times n}$, $m \geq n$, and $\text{rank}(A) < n$; that is, A is rank deficient. If we try to compute a QR factorization in this case, we will obtain at least one diagonal component of R satisfying $r_{ii} = 0$, and the solution scheme used for the full rank problem breaks down. We consider two approaches: QR with column pivoting, and SVD.

2.1 QR with Column Pivoting

Here we compute

$$A\Pi = QR = Q \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix},$$

where Π is a permutation matrix that does column pivoting, and R_{11} is an $r \times r$ nonsingular upper triangular matrix, where $r = \text{rank}(A)$. We outline how this factorization is computed:

Step 1.

- Find the column of A with largest 2-norm.
- Swap with first column.
- Perform QR factorization step to reduce the (new) first column to

$$\begin{bmatrix} \alpha_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Note that $|\alpha_1| = \|\text{largest column of } A\|_2$, which means $|\alpha_1|$ is made as large as possible. In particular, $\alpha_1 \neq 0$, unless $A \equiv 0$.

Step 2.

- Overwrite A with

$$A \leftarrow Q_1^T A \Pi_1 = \left[\begin{array}{c|ccc} \times & \times & \cdots & \times \\ \hline 0 & & & \\ \vdots & & & \\ 0 & & & \end{array} \right] A_{22}^{(1)}$$

where \times denotes a possible nonzero entry in the matrix.

- Find the column in $A_{22}^{(1)}$ with largest 2-norm.
- Swap second column of A with this column.
- Perform a QR factorization step to reduce the first column of $A_{22}^{(1)}$ to

$$\begin{bmatrix} \alpha_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Step 2.

- Overwrite A with

$$A \leftarrow Q_2^T (Q_1^T A \Pi_1) \Pi_2 = \left[\begin{array}{cc|ccc} \times & \times & \times & \cdots & \times \\ 0 & 0 & \times & \cdots & \times \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & & \\ 0 & 0 & & & A_{22}^{(1)} \end{array} \right]$$

\vdots continue in this way until

Step k .

- Overwrite A with

$$A \leftarrow Q_{k-1}^T \cdots Q_2^T Q_1^T A \Pi_1 \Pi_2 \cdots \Pi_{k-1} = \left[\begin{array}{cc} A_{11}^{(k-1)} & A_{12}^{(k-1)} \\ 0 & A_{22}^{(k-1)} \end{array} \right]$$

where $A_{11}^{(k-1)}$ is upper triangular, with nonzero entries on the diagonal.

- Find column of $A_{22}^{(k-1)}$ with largest 2-norm.
- Swap this column of A with the k th column.
- Perform a QR factorization step to reduce the first column of $A_{22}^{(k-1)}$ to

$$\begin{bmatrix} \alpha_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Stop when the largest column of $A_{22}^{(k)}$ has 2-norm equal to 0. That is, $A_{22}^{(k)} \equiv 0$.

Some remarks:

1. Due to roundoff errors, it is not likely that $A_{22}^{(k)}$ will be exactly 0 for any $k = 1, 2, \dots, n$. Thus we need to choose a tolerance, `tol`, such that $\|A_{22}^{(k)}\| \leq \text{tol}$ implies that $A_{22}^{(k)}$ is “numerically” 0.
2. Extra cost of pivoting: With proper implementation (using a clever updating of the column norms) the extra cost needed to perform this pivoting in the QR factorization is $O(mn)$. This is relatively inexpensive compared to the overall $O(mn^2)$.

2.2 Solving Rank Deficient LS Problem using $A\Pi = QR$

Let $A \in \mathcal{R}^{m \times n}$, $m \geq n$, $\text{rank}(A) = r < n$, and suppose we have computed

$$A\Pi = QR$$

where

- Π is a permutation matrix,
- Q is orthogonal,
- $R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}$,
- R_{11} is upper triangular and nonsingular.

We want to find a vector \mathbf{x} to minimize $\|\mathbf{b} - A\mathbf{x}\|_2^2$. To do this, note that for any $\mathbf{x} \in \mathcal{R}^n$,

$$\|\mathbf{b} - A\mathbf{x}\|_2^2 = \|\mathbf{b} - A\Pi\Pi^T\mathbf{x}\|_2^2 = \|Q^T\mathbf{b} - Q^T A\Pi\Pi^T\mathbf{x}\|_2^2$$

Note that $Q^T A\Pi = R$, and define subvectors $\mathbf{r} \in \mathcal{R}^r$, $\mathbf{d} \in \mathcal{R}^{m-r}$, $\mathbf{y} \in \mathcal{R}^r$ and $\mathbf{z} \in \mathcal{R}^{n-r}$ as

$$Q^T\mathbf{b} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix}, \quad \Pi^T\mathbf{x} = \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix}.$$

Then

$$\begin{aligned} \|\mathbf{b} - A\mathbf{x}\|_2^2 &= \left\| \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix} - \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} \right\|_2^2 \\ &= \left\| \begin{bmatrix} \mathbf{c} - R_{11}\mathbf{y} - R_{12}\mathbf{z} \\ \mathbf{d} \end{bmatrix} \right\|_2^2 \\ &= \|\mathbf{c} - R_{11}\mathbf{y} - R_{12}\mathbf{z}\|_2^2 + \|\mathbf{d}\|_2^2. \end{aligned}$$

The norm will be minimized if we can find \mathbf{y} and \mathbf{z} such that

$$\|\mathbf{c} - R_{11}\mathbf{y} - R_{12}\mathbf{z}\|_2^2 + \|\mathbf{d}\|_2^2 = 0$$

To do this, let $\mathbf{z} \in \mathcal{R}^{n-r}$ be arbitrary. Then we want to find \mathbf{y} such that

$$R_{11}\mathbf{y} = \mathbf{c} - R_{12}\mathbf{z}.$$

Since R_{11} is a nonsingular upper triangular matrix, we can use backward substitution to solve for \mathbf{y} . Then a solution of the rank deficient LS problem is:

$$\mathbf{x} = \Pi \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix}.$$

Some remarks:

1. Since $\mathbf{z} \in \mathcal{R}^{n-r}$ is arbitrary, there are infinitely many solutions to the rank deficient LS problem.
2. One easy (and commonly used) choice for \mathbf{z} is $\mathbf{z} \equiv \mathbf{0}$.

2.3 Solving Rank Deficient LS Problem using SVD

Recall that if $A \in \mathcal{R}^{m \times n}$, $m \geq n$, $r = \text{rank}(A)$, then the SVD of A is

$$A = U\Sigma V^T$$

where $U \in \mathcal{R}^{m \times m}$ and $V \in \mathcal{R}^{n \times n}$ are orthogonal matrices and

$$\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_n & \\ & & & \end{bmatrix} \in \mathcal{R}^{m \times n}$$

with

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_n = 0.$$

Computing the SVD is more expensive than computing the QR factorization, but the SVD does give more information about the linear system.

To solve the LS problem using the SVD, observe that

$$\begin{aligned} \|\mathbf{b} - A\mathbf{x}\|_2^2 &= \|U^T\mathbf{b} - U^TAVV^T\mathbf{x}\|_2^2 \\ &= \|U^T\mathbf{b} - \Sigma\mathbf{y}\|_2^2 \end{aligned}$$

where we set $\mathbf{y} = V^T\mathbf{x}$. Letting $U = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_r & \dots & \mathbf{u}_m \end{bmatrix}$, where $\mathbf{u}_j = j$ th column of U , then

$$\begin{aligned} \|\mathbf{b} - A\mathbf{x}\|_2^2 &= \|U^T\mathbf{b} - \Sigma\mathbf{y}\|_2^2 \\ &= \left\| \begin{bmatrix} \mathbf{u}_1^T\mathbf{b} \\ \vdots \\ \mathbf{u}_r^T\mathbf{b} \\ \mathbf{u}_{r+1}^T\mathbf{b} \\ \vdots \\ \mathbf{u}_m^T\mathbf{b} \end{bmatrix} - \begin{bmatrix} \sigma_1 y_1 \\ \vdots \\ \sigma_r y_r \\ 0 \\ \vdots \\ 0 \end{bmatrix} \right\|_2^2 \\ &= \left\| \begin{bmatrix} \mathbf{u}_1^T\mathbf{b} - \sigma_1 y_1 \\ \vdots \\ \mathbf{u}_r^T\mathbf{b} - \sigma_r y_r \\ \mathbf{u}_{r+1}^T\mathbf{b} \\ \vdots \\ \mathbf{u}_m^T\mathbf{b} \end{bmatrix} \right\|_2^2 \\ &= \sum_{i=1}^r (\mathbf{u}_i^T\mathbf{b} - \sigma_i y_i)^2 + \sum_{i=r+1}^m (\mathbf{u}_i^T\mathbf{b})^2 \end{aligned}$$

The second summation term does not depend on \mathbf{x} , so $\|\mathbf{b} - A\mathbf{x}\|_2^2$ is minimized if we can find $\mathbf{y} = V^T\mathbf{x} \in \mathcal{R}^n$ such that

$$\sum_{i=1}^r (\mathbf{u}_i^T\mathbf{b} - \sigma_i y_i)^2 = 0 \quad \Rightarrow \quad \mathbf{u}_i^T\mathbf{b} - \sigma_i y_i = 0 \quad i = 1, 2, \dots, r.$$

That is, we set

$$y_i = \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i}, \quad i = 1, 2, \dots, r.$$

Note that the vector \mathbf{y} has n elements, but the above relation only specifies the first $r < n$ of these. The remaining values, y_{r+1}, \dots, y_n are arbitrary. Similar to the rank revealing QR approach, an easy choice is to set these arbitrary values to 0. If we do this, then we obtain an LS solution:

$$\begin{aligned} \mathbf{x}_{\text{LS}} = V\mathbf{y} &= \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_r & \mathbf{v}_{r+1} & \cdots & \mathbf{v}_n \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_r \\ y_{r+1} \\ \vdots \\ y_n \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_r & \mathbf{v}_{r+1} & \cdots & \mathbf{v}_n \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_r \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &= \sum_{i=1}^r y_i \mathbf{v}_i \\ &= \sum_{i=1}^r \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i \end{aligned}$$

Remarks:

1. Since y_{r+1}, \dots, y_n are arbitrary, there are infinitely many solutions to the rank deficient LS problem. Setting (as we did above) $y_{r+1} = \dots = y_n = 0$ gives the LS solution, \mathbf{x}_{LS} with minimal 2-norm. That is, $\|\mathbf{x}_{\text{LS}}\|_2$ is less than or equal to the 2-norm of any solution of the rank deficient LS problem.
2. In matrix notation, \mathbf{x}_{LS} can be written as follows:
 - Given the SVD, $A = U\Sigma V^T$.
 - Let $A^\dagger = V\Sigma^\dagger U^T$, where

$$\Sigma^\dagger = \begin{bmatrix} \frac{1}{\sigma_1} & & & & & & \\ & \ddots & & & & & \\ & & \frac{1}{\sigma_r} & & & & \\ & & & 0 & & & \\ & & & & \ddots & & \\ & & & & & 0 & \end{bmatrix} \in \mathcal{R}^{n \times m}$$

- Then it is not difficult to verify that

$$\mathbf{x}_{\text{LS}} = \sum_{i=1}^r \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i = A^\dagger \mathbf{b}$$

- The matrix $A^\dagger = V \Sigma^\dagger U^T$ is called the “pseudo-inverse” of A .
- If A has full column rank, then

$$A^\dagger = (A^T A)^{-1} A^T$$

Can you prove this?

- If A is square and nonsingular, then $A^\dagger = A^{-1}$. Can you prove this?