Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

3

Linear Least Squares Problems

3.1. Introduction

Given an m-by-n matrix A and an m-by-1 vector b, the *linear least squares problem* is to find an n-by-1 vector x minimizing $\|Ax - b\|_2$. If $m = n$ and A is nonsingular, the answer is simply $x = A^{-1}b$. But if $m > n$ so that we have more equations than unknowns, the problem is called *overdetermined*, and generally no x satisfies $Ax = b$ exactly. One occasionally encounters the *underdetermined* problem, where $m < n$, but we will concentrate on the more common overdetermined case.

This chapter is organized as follows. The rest of this introduction describes three applications of least squares problems, to *curve fitting*, to *statistical modeling* of noisy data, and to *geodetic modeling*. Section 3.2 discusses three standard ways to solve the least squares problem: the *normal equations*, the *QR decomposition*, and the *singular value decomposition (SVD)*. We will frequently use the SVD as a tool in later chapters, so we derive several of its properties (although algorithms for the SVD are left to Chapter 5). Section 3.3 discusses perturbation theory for least squares problems, and section 3.4 discusses the implementation details and roundoff error analysis of our main method, QR decomposition. The roundoff analysis applies to many algorithms using orthogonal matrices, including many algorithms for eigenvalues and the SVD in Chapters 4 and 5. Section 3.5 discusses the particularly ill-conditioned situation of rank-deficient least squares problem and how to solve them accurately. Section 3.7 and the questions at the end of the chapter give pointers to other kinds of least squares problems and to software for sparse problems.

EXAMPLE 3.1. A typical application of least squares is *curve fitting*. Suppose that we have m pairs of numbers $(y_1, b_1), \ldots, (y_m, b_m)$ and that we want to find the "best" cubic polynomial fit to b_i as a function of y_i. This means finding polynomial coefficients $x_1, \ldots, x_4$ so that the polynomial $p(y) = \sum_{j=1}^{4} x_j y^{j-1}$ minimizes the residual $r_i \equiv p(y_i) - b_i$ for $i = 1$ to m. We can also write this as

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

minimizing

$$
\begin{aligned}
r \equiv \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix} &= \begin{bmatrix} p(y_1) \\ p(y_2) \\ \vdots \\ p(y_m) \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \\
&= \begin{bmatrix} 1 & y_1 & y_1^2 & y_1^3 \\ 1 & y_2 & y_2^2 & y_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & y_m & y_m^2 & y_m^3 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \\
&\equiv A \cdot x - b,
\end{aligned}
$$

where r and b are m-by-1, A is m-by-4, and x is 4-by-1. To minimize r, we could choose any norm, such as $\|r\|_\infty$, $\|r\|_1$, or $\|r\|_2$. The last one, which corresponds to minimizing the sum of the squared residuals $\sum_{i=1}^m r_i^2$, is a *linear least squares problem.*

Figure 3.1 shows an example, where we fit polynomials of increasing degree to the smooth function $b = \sin(\pi y/5) + y/5$ at the 23 points $y = -5, -4.5, -4, \ldots, 5.5, 6$. The left side of Figure 3.1 plots the data points as circles, and four different approximating polynomials of degrees 1, 3, 6, and 19. The right side of Figure 3.1 plots the residual norm $\|r\|_2$ versus degree for degrees from 1 to 20. Note that as the degree increases from 1 to 17, the residual norm decreases. We expect this behavior, since increasing the polynomial degree should let us fit the data better.

But when we reach degree 18, the residual norm suddenly increases dramatically. We can see how erratic the plot of the degree 19 polynomial is on the left (the blue line). This is due to ill-conditioning, as we will later see. Typically, one does polynomial fitting only with relatively low degree polynomials, avoiding ill-conditioning [61]. Polynomial fitting is available as the function `polyfit` in Matlab.

Here is an alternative to polynomial fitting. More generally, one has a set of independent functions $f_1(y), \ldots, f_n(y)$ from $\mathbb{R}^k$ to $\mathbb{R}$ and a set of points $(y_1, b_1), \ldots, (y_m, b_m)$ with $y_i \in \mathbb{R}^k$ and $b_i \in \mathbb{R}$, and one wishes to find a best fit to these points of the form $b = \sum_{j=1}^n x_j f_j(y)$. In other words one wants to choose $x = [x_1, \ldots, x_n]^T$ to minimize the residuals $r_i \equiv \sum_{j=1}^n x_j f_j(y_i) - b_i$ for $1 \le i \le m$. Letting $a_{ij} = f_j(y_i)$, we can write this as $r = Ax - b$, where A is m-by-n, x is n-by-1, and b and r are m-by-1. A good choice of basis functions $f_i(y)$ can lead to better fits and less ill-conditioned systems than using polynomials [33, 84, 168]. $\diamond$

EXAMPLE 3.2. In statistical modeling, one often wishes to estimate certain parameters x_j based on some observations, where the observations are contaminated by noise. For example, suppose that one wishes to predict the college grade point average (GPA) (b) of freshman applicants based on their

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

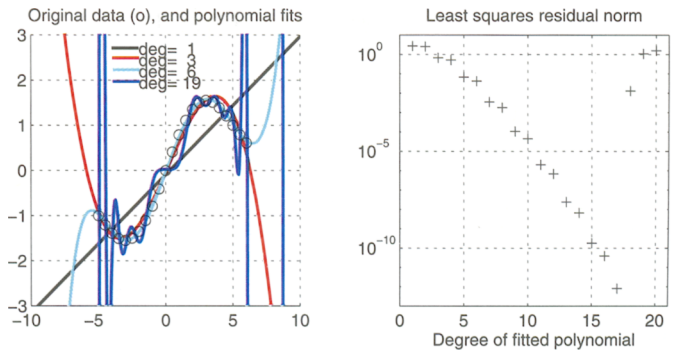


Fig. 3.1. *Polynomial fit to curve* $b = \sin(\pi y/5) + y/5$ *and residual norms.*

high school GPA (a_1) and two Scholastic Aptitude Test scores, verbal (a_2) and quantitative (a_3), as part of the college admissions process. Based on past data from admitted freshmen one can construct a *linear model* of the form $b = \sum_{j=1}^{3} a_j x_j$. The observations are a_{i1}, a_{i2}, a_{i3}, and b_i, one set for each of the m students in the database. Thus, one wants to minimize

$$r \equiv \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \equiv A \cdot x - b,$$

which we can do as a least squares problem.

Here is a statistical justification for least squares, which is called *linear regression* by statisticians: assume that the a_i are known exactly so that only b has noise in it, and that the noise in each b_i is independent and normally distributed with 0 mean and the same standard deviation σ. Let x be the solution of the least squares problem and x_T be the true value of the parameters. Then x is called a *maximum-likelihood estimate* of x_T, and the error $x - x_T$ is normally distributed, with zero mean in each component and *covariance matrix* $\sigma^2(A^TA)^{-1}$. We will see the matrix $(A^TA)^{-1}$ again below when we solve the least squares problem using the normal equations. For more details on the connection to statistics,[15] see, for example, [33, 259]. ◇

EXAMPLE 3.3. The least squares problem was first posed and formulated by Gauss to solve a practical problem for the German government. There are important economic and legal reasons to know exactly where the boundaries lie between plots of land owned by different people. Surveyors would go out and try to establish these boundaries, measuring certain angles and distances

[15]The standard notation in statistics differs from linear algebra: statisticians write $X\beta = y$ instead of $Ax = b$.

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

and then triangulating from known landmarks. As time passed, it became necessary to improve the accuracy to which the locations of the landmarks were known. So the surveyors of the day went out and remeasured many angles and distances between landmarks, and it fell to Gauss to figure out how to take these more accurate measurements and update the government database of locations. For this he invented least squares, as we will explain shortly [33].

The problem that Gauss solved did not go away and must be periodically revisited. In 1974 the US National Geodetic Survey undertook to update the US geodetic database, which consisted of about 700,000 points. The motivations had grown to include supplying accurate enough data for civil engineers and regional planners to plan construction projects and for geophysicists to study the motion of tectonic plates in the earth's crust (which can move up to 5 cm per year). The corresponding least squares problem was the largest ever solved at the time: about 2.5 million equations in 400,000 unknowns. It was also very sparse, which made it tractable on the computers available in 1978, when the computation was done [164].

Now we briefly discuss the formulation of this problem. It is actually nonlinear and is solved by approximating it by a sequence of linear ones, each of which is a linear least squares problem. The data base consists of a list of points (landmarks), each labeled by location: latitude, longitude, and possibly elevation. For simplicity of exposition, we assume that the earth is flat and suppose that each point i is labeled by linear coordinates $z_i = (x_i, y_i)^T$. For each point we wish to compute a correction $\delta z_i = (\delta x_i, \delta y_i)^T$ so that the corrected location $z_i' = (x_i', y_i')^T = z_i + \delta z_i$ more nearly matches the new, more accurate measurements. These measurements include both distances between selected pairs of points and angles between the line segment from point i to j and i to k (see Figure 3.2). To see how to turn these new measurements into constraints, consider the triangle in Figure 3.2. The corners are labeled by their (corrected) locations, and the angles θ and edge lengths L are also shown. From this data, it is easy to write down constraints based on simple trigonometric identities. For example, an accurate measurement of θ_i leads to the constraint

$$\cos^2 \theta_i = \frac{[(z_j' - z_i')^T (z_k' - z_i')]^2}{(z_j' - z_i')^T (z_j' - z_i') \cdot (z_k' - z_i')^T (z_k' - z_i')},$$

where we have expressed $\cos\theta_i$ in terms of dot products of certain sides of the triangle. If we assume that δz_i is small compared to z_i, then we can linearize this constraint as follows: multiply through by the denominator of the fraction, multiply out all the terms to get a quartic polynomial in all the "δ-variables" (like δx_i), and throw away all terms containing more than one δ-variable as a factor. This yields an equation in which all δ-variables appear linearly. If we collect all these linear constraints from all the new angle and distance measurements together, we get an overdetermined linear system of

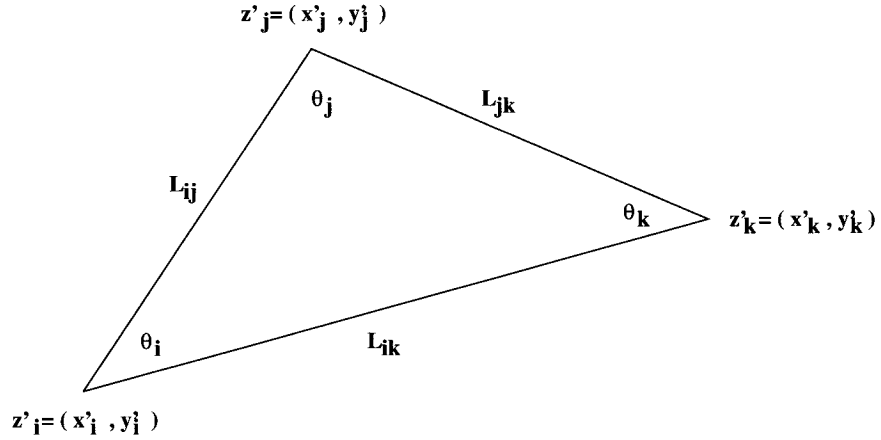


Fig. 3.2. *Constraints in updating a geodetic database.*

equations for all the δ-variables. We wish to find the smallest corrections, i.e., the smallest values of δx_i, etc., that most nearly satisfy these constraints. This is a least squares problem. $\diamond$

Later, after we introduce more machinery, we will also show how *image compression* can be interpreted as a least squares problem (see Example 3.4).

3.2. Matrix Factorizations That Solve the Linear Least Squares Problem

The linear least squares problem has several explicit solutions that we now discuss:

1. normal equations,
2. QR decomposition,
3. SVD,
4. transformation to a linear system (see Question 3.3).

The first method is the fastest but least accurate; it is adequate when the condition number is small. The second method is the standard one and costs up to twice as much as the first method. The third method is of most use on an ill-conditioned problem, i.e., when A is not of full rank; it is several times more expensive again. The last method lets us do iterative refinement to improve the solution when the problem is ill-conditioned. All methods but the third can be adapted to deal efficiently with sparse matrices [33]. We will discuss each solution in turn. We assume initially for methods 1 and 2 that A has full column rank n.

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

3.2.1. Normal Equations

To derive the *normal equations*, we look for the x where the gradient of $\|Ax - b\|_2^2 = (Ax - b)^T(Ax - b)$ vanishes. So we want

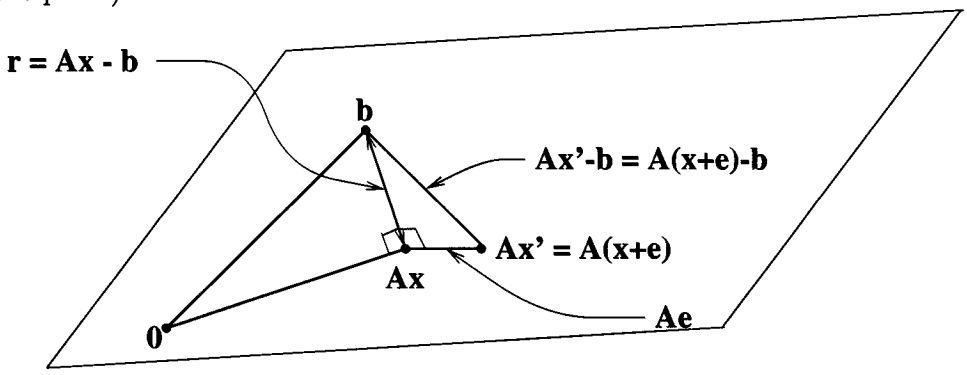
$$\begin{aligned} 0 &= \lim_{e \to 0} \frac{(A(x+e) - b)^T(A(x+e) - b) - (Ax - b)^T(Ax - b)}{\|e\|_2} \\ &= \lim_{e \to 0} \frac{2e^T(A^TAx - A^Tb) + e^TA^TAe}{\|e\|_2}. \end{aligned}$$

The second term $\frac{|e^TA^TAe|}{\|e\|_2} \leq \frac{\|A\|_2^2\|e\|_2^2}{\|e\|_2} = \|A\|_2^2\|e\|_2$ approaches 0 as e goes to 0, so the factor $A^TAx - A^Tb$ in the first term must also be zero, or $A^TAx = A^Tb$. This is a system of n linear equations in n unknowns, the normal equations.

Why is $x = (A^TA)^{-1}A^Tb$ the minimizer of $\|Ax - b\|_2^2$? We can note that the Hessian A^TA is positive definite, which means that the function is strictly convex and any critical point is a global minimum. Or we can complete the square by writing $x' = x + e$ and simplifying

$$\begin{aligned} (Ax' - b)^T(Ax' - b) &= (Ae + Ax - b)^T(Ae + Ax - b) \\ &= (Ae)^T(Ae) + (Ax - b)^T(Ax - b) \\ &\quad +2(Ae)^T(Ax - b) \\ &= \|Ae\|_2^2 + \|Ax - b\|_2^2 + 2e^T(A^TAx - A^Tb) \\ &= \|Ae\|_2^2 + \|Ax - b\|_2^2. \end{aligned}$$

This is clearly minimized by $e = 0$. This is just the Pythagorean theorem, since the residual $r = Ax - b$ is orthogonal to the space spanned by the columns of A, i.e., $0 = A^Tr = A^TAx - A^Tb$ as illustrated below (the plane shown is the span of the column vectors of A so that Ax, Ae, and $Ax' = A(x + e)$ all lie in the plane):



Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Since A^TA is symmetric and positive definite, we can use the Cholesky decomposition to solve the normal equations. The total cost of computing A^TA, A^Tb, and the Cholesky decomposition is $n^2m + \frac{1}{3}n^3 + O(n^2)$ flops. Since $m \geq n$, the n^2m cost of forming A^TA dominates the cost.

3.2.2. QR Decomposition

THEOREM 3.1. QR decomposition. *Let A be m-by-n with $m \geq n$. Suppose that A has full column rank. Then there exist a unique m-by-n orthogonal matrix Q ($Q^TQ = I_n$) and a unique n-by-n upper triangular matrix R with positive diagonals $r_{ii} > 0$ such that $A = QR$.*

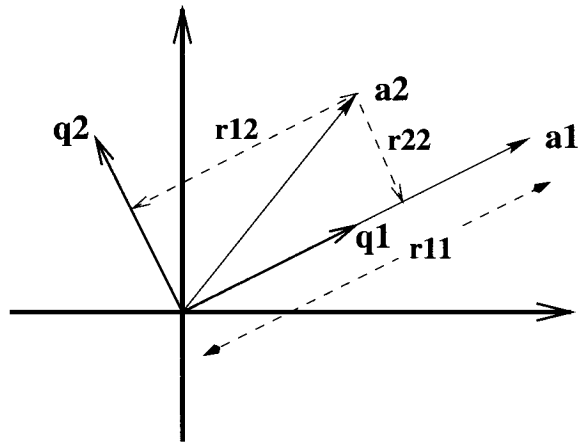
Proof. We give two proofs of this theorem. First, this theorem is a restatement of the Gram–Schmidt orthogonalization process [139]. If we apply Gram–Schmidt to the columns a_i of $A = [a_1, a_2, \ldots, a_n]$ from left to right, we get a sequence of orthonormal vectors q_1 through q_n spanning the same space: these orthogonal vectors are the columns of Q. Gram–Schmidt also computes coefficients $r_{ji} = q_j^T a_i$ expressing each column a_i as a linear combination of q_1 through q_i: $a_i = \sum_{j=1}^{i} r_{ji} q_j$. The r_{ji} are just the entries of R.

ALGORITHM 3.1. *The classical Gram–Schmidt (CGS) and modified Gram–Schmidt (MGS) Algorithms for factoring $A = QR$:*

for $i = 1$ *to* n /* *compute ith columns of* Q *and* R */
  $q_i = a_i$
  for $j = 1$ *to* $i - 1$ /* *subtract component in* q_j *direction from* a_i */
    $\begin{cases} r_{ji} = q_j^T a_i & \text{CGS} \\ r_{ji} = q_j^T q_i & \text{MGS} \end{cases}$
    $q_i = q_i - r_{ji} q_j$
  end for
  $r_{ii} = \|q_i\|_2$
  if $r_{ii} = 0$ /* a_i *is linearly dependent on* $a_1, \ldots, a_{i-1}$ */
    quit
  end if
  $q_i = q_i / r_{ii}$
end for

We leave it as an exercise to show that the two formulas for r_{ji} in the algorithm are mathematically equivalent (see Question 3.1). If A has full column rank, r_{ii} will not be zero. The following figure illustrates Gram–Schmidt when A is 2-by-2:

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php



The second proof of this theorem will use Algorithm 3.2, which we present in section 3.4.1. □

Unfortunately, CGS is numerically unstable in floating point arithmetic when the columns of A are nearly linearly dependent. MGS is more stable and will be used in algorithms later in this book but may still result in Q being far from orthogonal ($\|Q^TQ - I\|$ being far larger than ε) when A is ill-conditioned [31, 32, 33, 149]. Algorithm 3.2 in section 3.4.1 is a stable alternative algorithm for factoring $A = QR$. See Question 3.2.

We will derive the formula for the x that minimizes $\|Ax - b\|_2$ using the decomposition $A = QR$ in three slightly different ways. First, we can always choose $m-n$ more orthonormal vectors $\tilde{Q}$ so that $[Q, \tilde{Q}]$ is a square orthogonal matrix (for example, we can choose any $m - n$ more independent vectors $\tilde{X}$ that we want and then apply Algorithm 3.1 to the n-by-n nonsingular matrix $[Q, \tilde{X}]$). Then

$$\begin{aligned}
\|Ax - b\|_2^2 &= \|[Q, \tilde{Q}]^T(Ax - b)\|_2^2 \quad \text{by part 4 of Lemma 1.7} \\
&= \left\| \begin{bmatrix} Q^T \\ \tilde{Q}^T \end{bmatrix} (QRx - b) \right\|_2^2 \\
&= \left\| \begin{bmatrix} I^{n\times n} \\ O^{(m-n)\times n} \end{bmatrix} Rx - \begin{bmatrix} Q^Tb \\ \tilde{Q}^Tb \end{bmatrix} \right\|_2^2 \\
&= \left\| \begin{bmatrix} Rx - Q^Tb \\ -\tilde{Q}^Tb \end{bmatrix} \right\|_2^2 \\
&= \|Rx - Q^Tb\|_2^2 + \|\tilde{Q}^Tb\|_2^2 \\
&\geq \|\tilde{Q}^Tb\|_2^2.
\end{aligned}$$

We can solve $Rx - Q^Tb = 0$ for x, since A and R have the same rank, n, and so R is nonsingular. Then $x = R^{-1}Q^Tb$, and the minimum value of $\|Ax - b\|_2$ is $\|\tilde{Q}^Tb\|_2$.

Here is a second, slightly different derivation that does not use the matrix

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

$\tilde{Q}$. Rewrite $Ax - b$ as

$$\begin{aligned} Ax - b &= QRx - b = QRx - (QQ^T + I - QQ^T)b \\ &= Q(Rx - Q^Tb) - (I - QQ^T)b. \end{aligned}$$

Note that the vectors $Q(Rx - Q^Tb)$ and $(I - QQ^T)b$ are orthogonal, because $(Q(Rx - Q^Tb))^T((I - QQ^T)b) = (Rx - Q^Tb)^T[Q^T(I - QQ^T)]b = (Rx - Q^Tb)^T[0]b = 0$. Therefore, by the Pythagorean theorem,

$$\begin{aligned} \|Ax - b\|_2^2 &= \|Q(Rx - Q^Tb)\|_2^2 + \|(I - QQ^T)b\|_2^2 \\ &= \|Rx - Q^Tb\|_2^2 + \|(I - QQ^T)b\|_2^2, \end{aligned}$$

where we have used part 4 of Lemma 1.7 in the form $\|Qy\|_2^2 = \|y\|_2^2$. This sum of squares is minimized when the first term is zero, i.e., $x = R^{-1}Q^Tb$.

Finally, here is a third derivation that starts from the normal equations solution:

$$\begin{aligned} x &= (A^TA)^{-1}A^Tb \\ &= (R^TQ^TQR)^{-1}R^TQ^Tb = (R^TR)^{-1}R^TQ^Tb \\ &= R^{-1}R^{-T}R^TQ^Tb = R^{-1}Q^Tb. \end{aligned}$$

Later we will show that the cost of this decomposition and subsequent least squares solution is $2n^2m - \frac{2}{3}n^3$, about twice the cost of the normal equations if $m \gg n$ and about the same if $m = n$.

3.2.3. Singular Value Decomposition

The SVD is a very important decomposition which is used for many purposes other than solving least squares problems.

THEOREM 3.2. SVD. *Let A be an arbitrary m-by-n matrix with $m \geq n$. Then we can write $A = U\Sigma V^T$, where U is m-by-n and satisfies $U^TU = I$, V is n-by-n and satisfies $V^TV = I$, and $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$, where $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$. The columns $u_1, \ldots, u_n$ of U are called* left singular vectors. *The columns $v_1, \ldots, v_n$ of V are called* right singular vectors. *The σ_i are called* singular values. *(If $m < n$, the SVD is defined by considering A^T.)*

A geometric restatement of this theorem is as follows. Given any m-by-n matrix A, think of it as mapping a vector $x \in \mathbb{R}^n$ to a vector $y = Ax \in \mathbb{R}^m$. Then we can choose one orthogonal coordinate system for $\mathbb{R}^n$ (where the unit axes are the columns of V) and another orthogonal coordinate system for $\mathbb{R}^m$ (where the units axes are the columns of U) such that A is diagonal (Σ), i.e., maps a vector $x = \sum_{i=1}^n \beta_i v_i$ to $y = Ax = \sum_{i=1}^n \sigma_i \beta_i u_i$. In other words, any matrix is diagonal, provided that we pick appropriate orthogonal coordinate systems for its domain and range.

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Proof of Theorem 3.2. We use induction on m and n: we assume that the SVD exists for $(m-1)$-by-$(n-1)$ matrices and prove it for m-by-n. We assume $A \neq 0$; otherwise we can take $\Sigma = 0$ and let U and V be arbitrary orthogonal matrices.

The basic step occurs when $n = 1$ (since $m \geq n$). We write $A = U\Sigma V^T$ with $U = A/\|A\|_2$, $\Sigma = \|A\|_2$, and $V = 1$.

For the induction step, choose v so $\|v\|_2 = 1$ and $\|A\|_2 = \|Av\|_2 > 0$. Such a v exists by the definition of $\|A\|_2 = \max_{\|v\|_2=1} \|Av\|_2$. Let $u = \frac{Av}{\|Av\|_2}$, which is a unit vector. Choose $\tilde{U}$ and $\tilde{V}$ so that $U = [u, \tilde{U}]$ is an m-by-m orthogonal matrix, and $V = [v, \tilde{V}]$ is an n-by-n orthogonal matrix. Now write

$$U^T A V = \begin{bmatrix} u^T \\ \tilde{U}^T \end{bmatrix} \cdot A \cdot \begin{bmatrix} v & \tilde{V} \end{bmatrix} = \begin{bmatrix} u^T Av & u^T A\tilde{V} \\ \tilde{U}^T Av & \tilde{U}^T A\tilde{V} \end{bmatrix}.$$

Then

$$u^T Av = \frac{(Av)^T(Av)}{\|Av\|_2} = \frac{\|Av\|_2^2}{\|Av\|_2} = \|Av\|_2 = \|A\|_2 \equiv \sigma$$

and $\tilde{U}^T Av = \tilde{U}^T u \|Av\|_2 = 0$. We claim $u^T A\tilde{V} = 0$ too because otherwise $\sigma = \|A\|_2 = \|U^T AV\|_2 \geq \|[1, 0, \ldots, 0]U^T AV\|_2 = \|[\sigma | u^T A\tilde{V}]\|_2 > \sigma$, a contradiction. (We have used part 7 of Lemma 1.7.)

So $U^T AV = \begin{bmatrix} \sigma & 0 \\ 0 & \tilde{U}^T A\tilde{V} \end{bmatrix} = \begin{bmatrix} \sigma & 0 \\ 0 & \tilde{A} \end{bmatrix}$. We may now apply the induction hypothesis to $\tilde{A}$ to get $\tilde{A} = U_1 \Sigma_1 V_1^T$, where U_1 is $(m-1)$-by-$(n-1)$, Σ_1 is $(n-1)$-by-$(n-1)$, and V_1 is $(n-1)$-by-$(n-1)$. So

$$U^T AV = \begin{bmatrix} \sigma & 0 \\ 0 & U_1 \Sigma_1 V_1^T \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & U_1 \end{bmatrix} \begin{bmatrix} \sigma & 0 \\ 0 & \Sigma_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & V_1 \end{bmatrix}^T$$

or

$$A = \left(U \begin{bmatrix} 1 & 0 \\ 0 & U_1 \end{bmatrix} \right) \begin{bmatrix} \sigma & 0 \\ 0 & \Sigma_1 \end{bmatrix} \left(V \begin{bmatrix} 1 & 0 \\ 0 & V_1 \end{bmatrix} \right)^T,$$

which is our desired decomposition. □

The SVD has a large number of important algebraic and geometric properties, the most important of which we state here.

THEOREM 3.3. *Let $A = U\Sigma V^T$ be the SVD of the m-by-n matrix A, where $m \geq n$. (There are analogous results for $m < n$.)*

1. *Suppose that A is symmetric, with eigenvalues λ_i and orthonormal eigenvectors u_i. In other words $A = U\Lambda U^T$ is an eigendecomposition of A, with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, $U = [u_1, \ldots, u_n]$, and $UU^T = I$. Then an SVD of A is $A = U\Sigma V^T$, where $\sigma_i = |\lambda_i|$ and $v_i = \mathrm{sign}(\lambda_i) u_i$, where* $\mathrm{sign}(0) = 1$.

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

2. *The eigenvalues of the symmetric matrix* A^TA *are* σ_i^2. *The right singular vectors* v_i *are corresponding orthonormal eigenvectors.*

3. *The eigenvalues of the symmetric matrix* AA^T *are* σ_i^2 *and* $m-n$ *zeroes. The left singular vectors* u_i *are corresponding orthonormal eigenvectors for the eigenvalues* σ_i^2. *One can take any* $m-n$ *other orthogonal vectors as eigenvectors for the eigenvalue* 0.

4. *Let* $H = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$, *where* A *is square and* $A = U\Sigma V^T$ *is the SVD of* A. *Let* $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$, $U = [u_1, \ldots, u_n]$, *and* $V = [v_1, \ldots, v_n]$. *Then the* $2n$ *eigenvalues of* H *are* $\pm\sigma_i$, *with corresponding unit eigenvectors* $\frac{1}{\sqrt{2}}\begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}$.

5. *If* A *has full rank, the solution of* $\min_x \|Ax - b\|_2$ *is* $x = V\Sigma^{-1}U^Tb$.

6. $\|A\|_2 = \sigma_1$. *If* A *is square and nonsingular, then* $\|A^{-1}\|_2^{-1} = \sigma_n$ *and* $\|A\|_2 \cdot \|A^{-1}\|_2 = \frac{\sigma_1}{\sigma_n}$.

7. *Suppose* $\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0$. *Then the rank of* A *is* r. *The null space of* A, *i.e., the subspace of vectors* v *such that* $Av = 0$, *is the space spanned by columns* $r+1$ *through* n *of* V: $\operatorname{span}(v_{r+1}, \ldots, v_n)$. *The range space of* A, *the subspace of vectors of the form* Aw *for all* w, *is the space spanned by columns* 1 *through* r *of* U: $\operatorname{span}(u_1, \ldots, u_r)$.

8. *Let* S^{n-1} *be the unit sphere in* $\mathbb{R}^n$: $S^{n-1} = \{x \in \mathbb{R}^n : \|x\|_2 = 1\}$. *Let* $A \cdot S^{n-1}$ *be the image of* S^{n-1} *under* A: $A \cdot S^{n-1} = \{Ax : x \in \mathbb{R}^n \text{ and } \|x\|_2 = 1\}$. *Then* $A \cdot S^{n-1}$ *is an ellipsoid centered at the origin of* $\mathbb{R}^m$, *with principal axes* $\sigma_i u_i$.

9. *Write* $V = [v_1, v_2, \ldots, v_n]$ *and* $U = [u_1, u_2, \ldots, u_n]$, *so* $A = U\Sigma V^T = \sum_{i=1}^n \sigma_i u_i v_i^T$ *(a sum of rank-1 matrices). Then a matrix of rank* $k < n$ *closest to* A *(measured with* $\|\cdot\|_2$*) is* $A_k = \sum_{i=1}^k \sigma_i u_i v_i^T$, *and* $\|A - A_k\|_2 = \sigma_{k+1}$. *We may also write* $A_k = U\Sigma_k V^T$, *where* $\Sigma_k = \operatorname{diag}(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)$.

Proof.

1. This is true by the definition of the SVD.

2. $A^TA = V\Sigma U^TU\Sigma V^T = V\Sigma^2V^T$. This is an eigendecomposition of A^TA, with the columns of V the eigenvectors and the diagonal entries of Σ^2 the eigenvalues.

3. Choose an m-by-$(m-n)$ matrix $\tilde{U}$ so that $[U, \tilde{U}]$ is square and orthogonal. Then write

$$AA^T = U\Sigma V^TV\Sigma U^T = U\Sigma^2U^T = \left[U, \tilde{U}\right] \begin{bmatrix} \Sigma^2 & 0 \\ 0 & 0 \end{bmatrix} \left[U, \tilde{U}\right]^T.$$

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

This is an eigendecomposition of AA^T.

4. See Question 3.14.

5. $\|Ax - b\|_2^2 = \|U\Sigma V^T x - b\|_2^2$. Since A has full rank, so does Σ, and thus Σ is invertible. Now let $[U, \tilde{U}]$ be square and orthogonal as above so

$$\begin{aligned} \|U\Sigma V^T x - b\|_2^2 &= \left\| \begin{bmatrix} U^T \\ \tilde{U}^T \end{bmatrix} (U\Sigma V^T x - b) \right\|_2^2 \\ &= \left\| \begin{bmatrix} \Sigma V^T x - U^T b \\ -\tilde{U}^T b \end{bmatrix} \right\|_2^2 \\ &= \|\Sigma V^T x - U^T b\|_2^2 + \|\tilde{U}^T b\|_2^2. \end{aligned}$$

This is minimized by making the first term zero, i.e., $x = V\Sigma^{-1}U^T b$.

6. It is clear from its definition that the two-norm of a diagonal matrix is the largest absolute entry on its diagonal. Thus, by part 3 of Lemma 1.7, $\|A\|_2 = \|U^T AV\|_2 = \|\Sigma\|_2 = \sigma_1$ and $\|A^{-1}\|_2 = \|V^T A^{-1} U\|_2 = \|\Sigma^{-1}\|_2 = \sigma_n^{-1}$.

7. Again choose an m-by-$(m-n)$ matrix $\tilde{U}$ so that the m-by-m matrix $\hat{U} = [U, \tilde{U}]$ is orthogonal. Since $\hat{U}$ and V are nonsingular, A and $\hat{U}^T AV = \begin{bmatrix} \Sigma^{n\times n} \\ 0^{(m-n)\times n} \end{bmatrix} \equiv \hat{\Sigma}$ have the same rank—namely, r—by our assumption about Σ. Also, v is in the null space of A if and only if $V^T v$ is in the null space of $\hat{U}^T AV = \hat{\Sigma}$, since $Av = 0$ if and only if $\hat{U}^T AV(V^T v) = 0$. But the null space of $\hat{\Sigma}$ is clearly spanned by columns $r+1$ through n of the n-by-n identity matrix I_n, so the null space of A is spanned by V times these columns, i.e., v_{r+1} through v_n. A similar argument shows that the range space of A is the same as $\hat{U}$ times the range space of $\hat{U}^T AV = \hat{\Sigma}$, i.e., $\hat{U}$ times the first r columns of I_m, or u_1 through u_r.

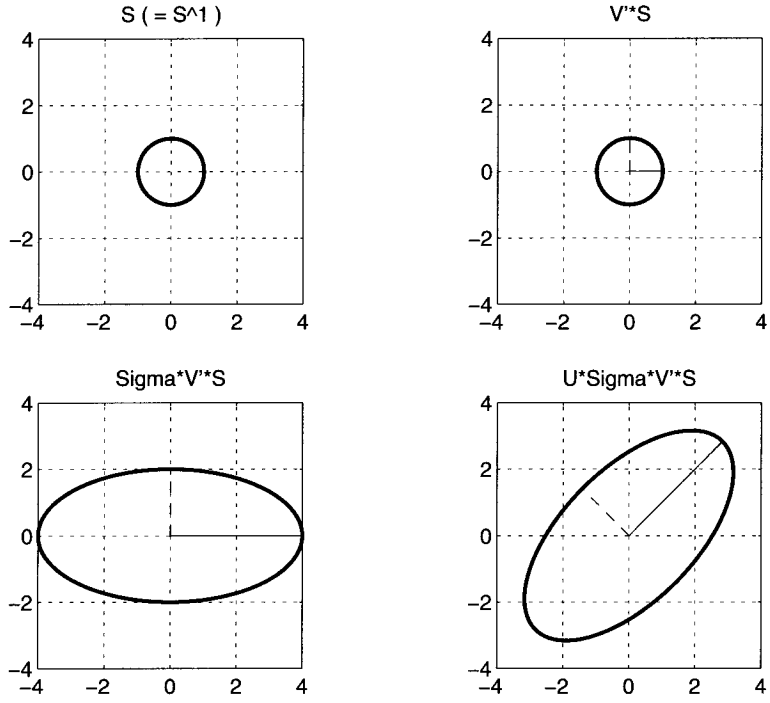
8. We "build" the set $A \cdot S^{n-1}$ by multiplying by one factor of $A = U\Sigma V^T$ at a time. The figure below illustrates what happens when

$$\begin{aligned} A &= \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \\ &= \begin{bmatrix} 2^{-1/2} & -2^{-1/2} \\ 2^{-1/2} & 2^{-1/2} \end{bmatrix} \cdot \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \cdot \begin{bmatrix} 2^{-1/2} & -2^{-1/2} \\ 2^{-1/2} & 2^{-1/2} \end{bmatrix}^T \\ &\equiv U\Sigma V^T. \end{aligned}$$

Assume for simplicity that A is square and nonsingular. Since V is orthogonal and so maps unit vectors to other unit vectors, $V^T \cdot S^{n-1} = S^{n-1}$. Next, since $v \in S^{n-1}$ if and only if $\|v\|_2 = 1$, $w \in \Sigma S^{n-1}$ if and only if $\|\Sigma^{-1} w\|_2 = 1$ or $\sum_{i=1}^n (w_i/\sigma_i)^2 = 1$. This defines an ellipsoid with

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

principal axes $\sigma_i e_i$, where e_i is the ith column of the identity matrix. Finally, multiplying each $w = \Sigma v$ by U just rotates the ellipse so that each e_i becomes u_i, the ith column of U.



9. A_k has rank k by construction and

$$\|A - A_k\|_2 = \left\| \sum_{i=k+1}^{n} \sigma_i u_i v_i^T \right\| = \left\| U \begin{bmatrix} 0 & & & \\ & \sigma_{k+1} & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} V^T \right\|_2 = \sigma_{k+1}.$$

It remains to show that there is no closer rank k matrix to A. Let B be any rank k matrix, so its null space has dimension $n - k$. The space spanned by $\{v_1, \ldots, v_{k+1}\}$ has dimension $k + 1$. Since the sum of their dimensions is $(n - k) + (k + 1) > n$, these two spaces must overlap. Let h be a unit vector in their intersection. Then

$$\begin{aligned} \|A - B\|_2^2 &\geq \|(A - B)h\|_2^2 = \|Ah\|_2^2 = \|U\Sigma V^T h\|_2^2 \\ &= \|\Sigma(V^T h)\|_2^2 \\ &\geq \sigma_{k+1}^2 \|V^T h\|_2^2 \\ &= \sigma_{k+1}^2. \quad \square \end{aligned}$$

EXAMPLE 3.4. We illustrate the last part of Theorem 3.3 by using it for *image compression*. In particular, we will illustrate it with low-rank approximations

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

of a clown. An m-by-n image is just an m-by-n matrix, where entry (i, j) is interpreted as the brightness of pixel (i, j). In other words, matrix entries ranging from 0 to 1 (say) are interpreted as pixels ranging from black (=0) through various shades of gray to white (=1). (Colors also are possible.) Rather than storing or transmitting all $m \cdot n$ matrix entries to represent the image, we often prefer to *compress* the image by storing many fewer numbers, from which we can still approximately reconstruct the original image. We may use Part 9 of Theorem 3.3 to do this, as we now illustrate.

Consider the image in Figure 3.3(a). This 320-by-200 pixel image corresponds to a 320-by-200 matrix A. Let $A = U\Sigma V^T$ be the SVD of A. Part 9 of Theorem 3.3 tells us that $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ is the best rank-k approximation of A, in the sense of minimizing $\|A - A_k\|_2 = \sigma_{k+1}$. Note that it only takes $m \cdot k + n \cdot k = (m + n) \cdot k$ words to store u_1 through u_k and $\sigma_1 v_1$ through $\sigma_k v_k$, from which we can reconstruct A_k. In contrast, it takes $m \cdot n$ words to store A (or A_k explicitly), which is much larger when k is small. So we will use A_k as our compressed image, stored using $(m + n) \cdot k$ words. The other images in Figure 3.3 show these approximations for various values of k, along with the relative errors σ_{k+1}/σ_1 and compression ratios $(m + n) \cdot k/(m \cdot n) = 520 \cdot k/64000 \approx k/123$.

k	Relative error $= \sigma_{k+1}/\sigma_k$	Compression ratio $= 520k/64000$
3	.155	.024
10	.077	.081
20	.040	.163

These images were produced by the following commands (the clown and other images are available in Matlab among the visualization demonstration files; check your local installation for location):

```
load clown.mat; [U,S,V]=svd(X); colormap('gray');
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)')
```

There are also many other, cheaper image-compression techniques available than the SVD [189, 152]. ◇

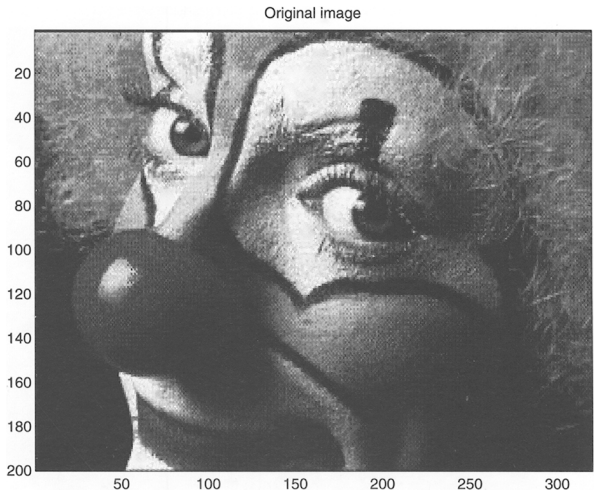
Later we will see that the cost of solving a least squares problem with the SVD is about the same as with QR when $m >> n$, and about $4n^2m - \frac{4}{3}n^3 + O(n^2)$ for smaller m. A precise comparison of the costs of QR and the SVD also depends on the machine being used. See section 3.6 for details.

DEFINITION 3.1. *Suppose that A is m-by-n with $m \geq n$ and has full rank, with $A = QR = U\Sigma V^T$ being A's QR decomposition and SVD, respectively. Then*

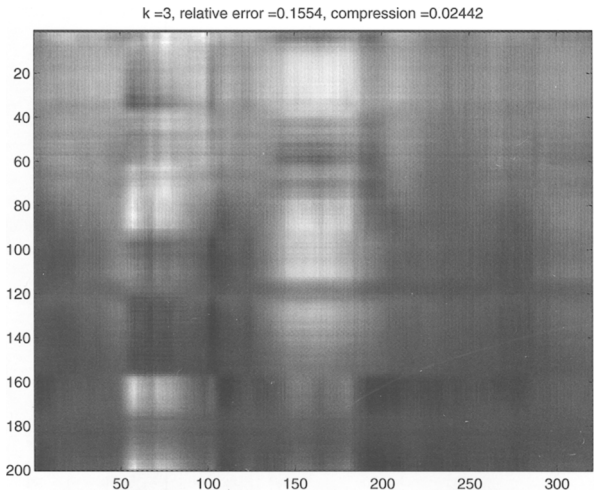
$$A^+ \equiv (A^T A)^{-1} A^T = R^{-1} Q^T = V \Sigma^{-1} U^T$$

is called the (Moore–Penrose) pseudoinverse *of A. If $m < n$, then $A^+ \equiv A^T (AA^T)^{-1}$.*

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php



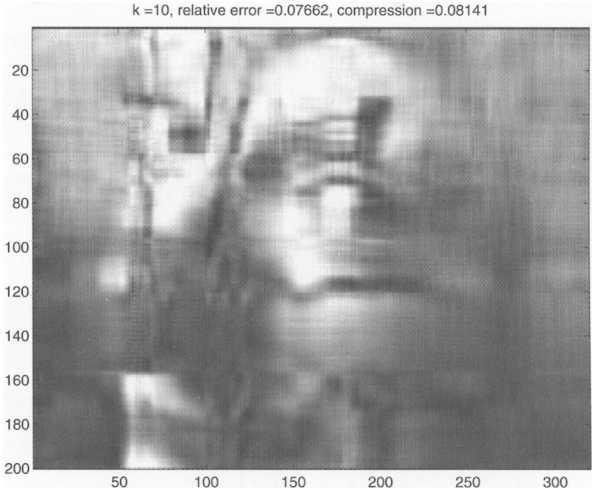
(a)



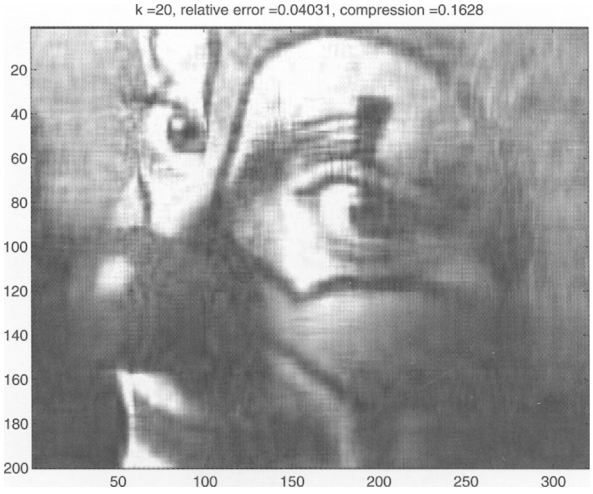
(b)

Fig. 3.3. *Image compression using the SVD.* (a) *Original image.* (b) *Rank* $k = 3$ *approximation.*

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php



(c)



(d)

Fig. 3.3. *Continued.* (c) *Rank* $k = 10$ *approximation.* (d) *Rank* $k = 20$ *approximation.*

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

The pseudoinverse lets us write the solution of the full-rank, overdetermined least squares problem as simply $x = A^+ b$. If A is square and full rank, this formula reduces to $x = A^{-1}b$ as expected. The pseudoinverse of A is computed as `pinv(A)` in Matlab. When A is not full rank, the Moore–Penrose pseudoinverse is given by Definition 3.2 in section 3.5.

3.3. Perturbation Theory for the Least Squares Problem

When A is not square, we define its condition number with respect to the 2-norm to be $\kappa_2(A) \equiv \sigma_{\max}(A)/\sigma_{\min}(A)$. This reduces to the usual condition number when A is square. The next theorem justifies this definition.

THEOREM 3.4. *Suppose that A is m-by-n with $m \geq n$ and has full rank. Suppose that x minimizes $\|Ax-b\|_2$. Let $r = Ax-b$ be the residual. Let $\tilde{x}$ minimize $\|(A+\delta A)\tilde{x} - (b+\delta b)\|_2$. Assume $\epsilon \equiv \max(\frac{\|\delta A\|_2}{\|A\|_2}, \frac{\|\delta b\|_2}{\|b\|_2}) < \frac{1}{\kappa_2(A)} = \frac{\sigma_{\min}(A)}{\sigma_{\max}(A)}$. Then*

$$\frac{\|\tilde{x} - x\|_2}{\|x\|_2} \leq \epsilon \cdot \left\{ \frac{2 \cdot \kappa_2(A)}{\cos\theta} + \tan\theta \cdot \kappa_2^2(A) \right\} + O(\epsilon^2) \equiv \epsilon \cdot \kappa_{LS} + O(\epsilon^2),$$

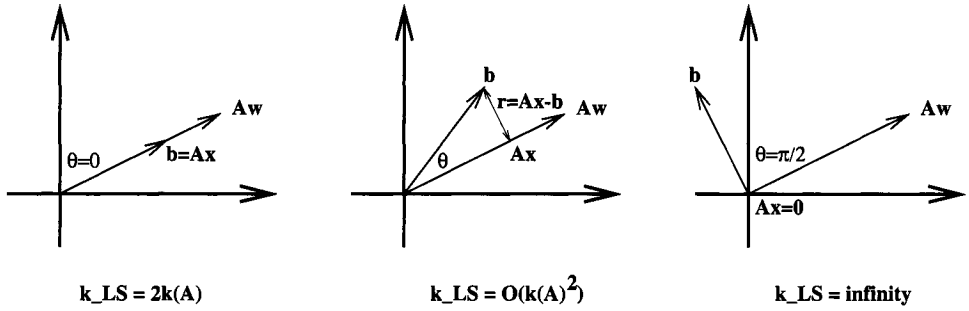
where $\sin\theta = \frac{\|r\|_2}{\|b\|_2}$. In other words, θ is the angle between the vectors b and Ax and measures whether the residual norm $\|r\|_2$ is large (near $\|b\|$) or small (near 0). κ_{LS} is the condition number for the least squares problem.

Sketch of Proof. Expand $\tilde{x} = \left((A+\delta A)^T(A+\delta A)\right)^{-1}(A+\delta A)^T(b+\delta b)$ in powers of δA and δb, and throw away all but the linear terms in δA and δb. □

We have assumed that $\epsilon \cdot \kappa_2(A) < 1$ for the same reason as in the derivation of bound (2.4) for the perturbed solution of the square linear system $Ax = b$: it guarantees that $A + \delta A$ has full rank so that $\tilde{x}$ is uniquely determined.

We may interpret this bound as follows. If θ is 0 or very small, then the residual is small and the effective condition number is about $2\kappa_2(A)$, much like ordinary linear equation solving. If θ is not small but not close to $\pi/2$, the residual is moderately large, and then the effective condition number can be much larger: $\kappa_2^2(A)$. If θ is close to $\pi/2$, so the true solution is nearly zero, then the effective condition number becomes unbounded even if $\kappa_2(A)$ is small. These three cases are illustrated below. The right-hand picture makes it easy to see why the condition number is infinite when $\theta = \pi/2$: in this case the solution $x = 0$, and almost any arbitrarily small change in A or b will yield a nonzero solution x, an "infinitely" large relative change.

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php



An alternative form for the bound in Theorem 3.4 that eliminates the $O(\epsilon^2)$ term is as follows [258, 149] (here $\tilde{r}$ is the perturbed residual $\tilde{r} = (A+\delta A)\tilde{x}) - (b+\delta b)$:

$$\begin{aligned} \frac{\|\tilde{x}-x\|_2}{\|x\|_2} &\leq \frac{\epsilon\kappa_2(A)}{1-\epsilon\kappa_2(A)}\left(2+(\kappa_2(A)+1)\frac{\|r\|_2}{\|A\|_2\|x\|_2}\right), \\ \frac{\|\tilde{r}-r\|_2}{\|r\|_2} &\leq (1+2\epsilon\kappa_2(A)). \end{aligned}$$

We will see that, properly implemented, both the QR decomposition and SVD are numerically stable; i.e., they yield a solution $\tilde{x}$ minimizing $\|(A+\delta A)\tilde{x}-(b+\delta b)\|_2$ with

$$\max\left(\frac{\|\delta A\|}{\|A\|}, \frac{\|\delta b\|}{\|b\|}\right) = O(\varepsilon).$$

We may combine this with the above perturbation bounds to get error bounds for the solution of the least squares problem, much as we did for linear equation solving.

The normal equations are not as accurate. Since they involve solving $(A^TA)x = A^Tb$, the accuracy depends on the condition number $\kappa_2(A^TA) = \kappa_2^2(A)$. Thus the error is always bounded by $\kappa_2^2(A)\varepsilon$, never just $\kappa_2(A)\varepsilon$. Therefore we expect that the normal equations can lose twice as many digits of accuracy as methods based on the QR decomposition and SVD.

Furthermore, solving the normal equations is *not* necessarily stable; i.e., the computed solution $\tilde{x}$ does not generally minimize $\|(A+\delta A)\tilde{x}-(b+\delta b)\|_2$ for small δA and δb. Still, when the condition number is small, we expect the normal equations to be about as accurate as the QR decomposition or SVD. Since the normal equations are the fastest way to solve the least squares problem, they are the method of choice when the matrix is well-conditioned.

We return to the problem of solving very ill-conditioned least squares problems in section 3.5.

3.4. Orthogonal Matrices

As we said in section 3.2.2, Gram–Schmidt orthogonalization (Algorithm 3.1) may not compute an orthogonal matrix Q when the vectors being orthogonal-

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

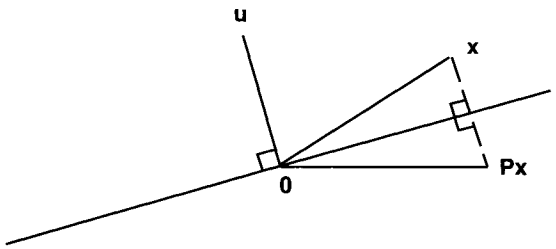
ized are nearly linearly dependent, so we cannot use it to compute the QR decomposition stably.

Instead, we base our algorithms on certain easily computable orthogonal matrices called *Householder reflections* and *Givens rotations*, which we can choose to introduce zeros into vectors that they multiply. Later we will show that any algorithm that uses these orthogonal matrices to introduce zeros is automatically stable. This error analysis will apply to our algorithms for the QR decomposition as well as many SVD and eigenvalue algorithms in Chapters 4 and 5.

Despite the possibility of nonorthogonal Q, the MGS algorithm has important uses in numerical linear algebra. (There is little use for its less stable version, CGS.) These uses include finding eigenvectors of symmetric tridiagonal matrices using bisection and inverse iteration (section 5.3.4) and the Arnoldi and Lanczos algorithms for reducing a matrix to certain "condensed" forms (sections 6.6.1, 6.6.6, and 7.4). Arnoldi and Lanczos algorithms are used as the basis of algorithms for solving sparse linear systems and finding eigenvalues of sparse matrices. MGS can also be modified to solve the least squares problem stably, but Q may still be far from orthogonal [33].

3.4.1. Householder Transformations

A Householder transformation (or reflection) is a matrix of the form $P = I - 2uu^T$ where $\|u\|_2 = 1$. It is easy to see that $P = P^T$ and $PP^T = (I - 2uu^T)(I - 2uu^T) = I - 4uu^T + 4uu^Tuu^T = I$, so P is a symmetric, orthogonal matrix. It is called a reflection because Px is reflection of x in the plane through 0 perpendicular to u.



Given a vector x, it is easy to find a Householder reflection $P = I - 2uu^T$ to zero out all but the first entry of x: $Px = [c, 0, \ldots, 0]^T = c \cdot e_1$. We do this as follows. Write $Px = x - 2u(u^Tx) = c \cdot e_1$ so that $u = \frac{1}{2(u^Tx)}(x - ce_1)$; i.e., u is a linear combination of x and e_1. Since $\|x\|_2 = \|Px\|_2 = |c|$, u must be parallel to the vector $\tilde{u} = x \pm \|x\|_2 e_1$, and so $u = \tilde{u}/\|\tilde{u}\|_2$. One can verify that either choice of sign yields a u satisfying $Px = ce_1$, as long as $\tilde{u} \neq 0$. We will use $\tilde{u} = x + \text{sign}(x_1)e_1$, since this means that there is no cancellation in

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

computing the first component of $\tilde{u}$. In summary, we get

$$\tilde{u} = \begin{bmatrix} x_1 + \text{sign}(x_1) \cdot \|x\|_2 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{with } u = \frac{\tilde{u}}{\|\tilde{u}\|_2}.$$

We write this as $u = \text{House}(x)$. (In practice, we can store $\tilde{u}$ instead of u to save the work of computing u, and use the formula $P = I - (2/\|\tilde{u}\|_2^2)\tilde{u}\tilde{u}^T$ instead of $P = I - 2uu^T$.)

EXAMPLE 3.5. We show how to compute the QR decomposition of a 5-by-4 matrix A using Householder transformations. This example will make the pattern for general m-by-n matrices evident. In the matrices below, P_i is a 5-by-5 orthogonal matrix, x denotes a generic nonzero entry, and o denotes a zero entry.

1. Choose P_1 so $$A_1 \equiv P_1 A = \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & x & x & x \\ o & x & x & x \\ o & x & x & x \end{bmatrix}.$$

2. Choose $P_2 = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & P_2' \end{array}\right]$ so $$A_2 \equiv P_2 A_1 = \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & x & x \\ o & o & x & x \end{bmatrix}.$$

3. Choose $P_3 = \left[\begin{array}{cc|c} 1 & & 0 \\ & 1 & \\ \hline 0 & & P_3' \end{array}\right]$ so $$A_3 \equiv P_3 A_2 = \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & o & x \\ o & o & o & x \end{bmatrix}.$$

4. Choose $P_4 = \left[\begin{array}{ccc|c} 1 & & & 0 \\ & 1 & & \\ & & 1 & \\ \hline 0 & & & P_4' \end{array}\right]$ so $$A_4 \equiv P_4 A_3 = \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & o & x \\ o & o & o & o \end{bmatrix}.$$

Here, we have chosen a Householder matrix P_i' to zero out the subdiagonal entries in column i; this does not disturb the zeros already introduced in previous columns.

Let us call the final 5-by-4 upper triangular matrix $\tilde{R} \equiv A_4$. Then $A = P_1^T P_2^T P_3^T P_4^T \tilde{R} = QR$, where Q is the first four columns of $P_1^T P_2^T P_3^T P_4^T = P_1 P_2 P_3 P_4$ (since all P_i are symmetric) and R is the first four rows of $\tilde{R}$. $\diamond$

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Here is the general algorithm for QR decomposition using Householder transformations.

ALGORITHM 3.2. *QR factorization using Householder reflections:*

for $i = 1$ *to* $\min(m-1, n)$
  $u_i = \text{House}(A(i:m, i))$
  $P_i' = I - 2u_i u_i^T$
  $A(i:m, i:n) = P_i' A(i:m, i:n)$
end for

Here are some more implementation details. We never need to form P_i explicitly but just multiply

$$(I - 2u_i u_i^T)A(i:m, i:n) = A(i:m, i:n) - 2u_i(u_i^T A(i:m, i:n)),$$

which costs less. To store P_i, we need only u_i, or $\tilde{u}_i$ and $\|\tilde{u}_i\|$. These can be stored in column i of A; in fact it need not be changed! Thus QR can be "overwritten" on A, where Q is stored in factored form $P_1 \cdots P_{n-1}$, and P_i is stored as $\tilde{u}_i$ below the diagonal in column i of A. (We need an extra array of length n for the top entry of $\tilde{u}_i$, since the diagonal entry is occupied by R_{ii}.)

Recall that to solve the least squares problem $\min \|Ax - b\|_2$ using $A = QR$, we need to compute $Q^T b$. This is done as follows: $Q^T b = P_n P_{n-1} \cdots P_1 b$, so we need only keep multiplying b by $P_1, P_2, \ldots, P_n$:

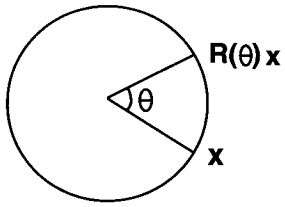
for $i = 1$ *to* n
  $\gamma = -2 \cdot u_i^T b(i:m)$
  $b(i:m) = b(i:m) + \gamma u_i$
end for

The cost is n dot products $\gamma = -2 \cdot u_i^T b$ and n "saxpys" $b + \gamma u_i$. The cost of computing $A = QR$ this way is $2n^2 m - \frac{2}{3}n^3$, and the subsequent cost of solving the least squares problem given QR is just an additional $O(mn)$.

The LAPACK routine for solving the least squares problem using QR is `sgels`. Just as Gaussian elimination can be reorganized to use matrix-matrix multiplication and other Level 3 BLAS (see section 2.6), the same can be done for the QR decomposition; see Question 3.17. In Matlab, if the m-by-n matrix A has more rows than columns and b is m by 1, `A\b` solves the least squares problem. The QR decomposition itself is also available via [Q,R]=qr(A).

3.4.2. Givens Rotations

A Givens rotation $R(\theta) \equiv \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ rotates any vector $x \in \mathbb{R}^2$ counterclockwise by θ:



We also need to define the Givens rotation by θ in coordinates i and j:

$$R(i,j,\theta) \equiv \begin{array}{c} \\ i \\ \\ j \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & i & & j & \end{array} \\ \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \ddots & & & & \\ & & & \cos\theta & & -\sin\theta & \\ & & & & \ddots & & \\ & & & \sin\theta & & \cos\theta & \\ & & & & & & \ddots \\ & & & & & & & 1 \\ & & & & & & & & 1 \end{bmatrix} \end{array}.$$

Given x, i, and j, we can zero out x_j by choosing $\cos\theta$ and $\sin\theta$ so that

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} = \begin{bmatrix} \sqrt{x_i^2 + x_j^2} \\ 0 \end{bmatrix}$$

or $\cos\theta = \frac{x_i}{\sqrt{x_i^2+x_j^2}}$ and $\sin\theta = \frac{-x_j}{\sqrt{x_i^2+x_j^2}}$.

The QR algorithm using Givens rotations is analogous to using Householder reflections, but when zeroing out column i, we zero it out one entry at a time (bottom to top, say).

EXAMPLE 3.6. We illustrate two intermediate steps in computing the QR decomposition of a 5-by-4 matrix using Givens rotations. To progress from

$$\begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & x & x \\ o & o & x & x \end{bmatrix} \quad \text{to} \quad \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & o & x \\ o & o & o & x \end{bmatrix}$$

we multiply

$$\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & c & -s \\ & & & s & c \end{bmatrix} \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & x & x \\ o & o & x & x \end{bmatrix} = \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & x & x \\ o & o & o & x \end{bmatrix}$$

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

and

$$\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & c' & -s' & \\ & & s' & c' & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & x & x \\ o & o & o & x \end{bmatrix} = \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & o & x & x \\ o & o & o & x \\ o & o & o & x \end{bmatrix}. \quad \diamond$$

The cost of the QR decomposition using Givens rotations is twice the cost of using Householder reflections. We will need Givens rotations for other applications later.

Here are some implementation details. Just as we overwrote A with Q and R when using Householder reflections, we can do the same with Givens rotations. We use the same trick, storing the information describing the transformation in the entries zeroed out. Since a Givens rotation zeros out just one entry, we must store the information about the rotation there. We do this as follows. Let $s = \sin\theta$ and $c = \cos\theta$. If $|s| < |c|$, store $s \cdot \text{sign}(c)$ and otherwise store $\frac{\text{sign}(s)}{c}$. To recover s and c from the stored value (call it p) we do the following: if $|p| < 1$, then $s = p$ and $c = \sqrt{1 - s^2}$; otherwise $c = \frac{1}{p}$ and $s = \sqrt{1 - c^2}$. The reason we do not just store s and compute $c = \sqrt{1 - s^2}$ is that when s is close to 1, c would be inaccurately reconstructed. Note also that we may recover either s and c or $-s$ and $-c$; this is adequate in practice.

There is also a way to apply a sequence of Givens rotations while performing fewer floating point operations than described above. These are called *fast Givens rotations* [7, 8, 33]. Since they are still slower than Householder reflections for the purposes of computing the QR factorization, we will not consider them further.

3.4.3. Roundoff Error Analysis for Orthogonal Matrices

This analysis proves backward stability for the QR decomposition and for many of the algorithms for eigenvalues and singular values that we will discuss.

LEMMA 3.1. *Let P be an exact Householder (or Givens) transformation, and $\tilde{P}$ be its floating point approximation. Then*

$$\text{fl}(\tilde{P}A) = P(A + E) \quad \|E\|_2 = O(\varepsilon) \cdot \|A\|_2$$

and

$$\text{fl}(A\tilde{P}) = (A + F)P \quad \|F\|_2 = O(\varepsilon) \cdot \|A\|_2.$$

Sketch of Proof. Apply the usual formula $\text{fl}(a \odot b) = (a \odot b)(1 + \varepsilon)$ to the formulas for computing and applying $\tilde{P}$. See Question 3.16. □

In words, this says that applying a single orthogonal matrix is backward stable.

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

THEOREM 3.5. *Consider applying a sequence of orthogonal transformations to A_0. Then the computed product is an exact orthogonal transformation of $A_0 + \delta A$, where $\|\delta A\|_2 = O(\varepsilon)\|A\|_2$. In other words, the entire computation is backward stable:*

$$\mathrm{fl}(\tilde{P}_j \tilde{P}_{j-1} \cdots \tilde{P}_1 A_0 \tilde{Q}_1 \tilde{Q}_2 \cdots \tilde{Q}_j) = P_j \cdots P_1 (A_0 + E) Q_1 \cdots Q_j$$

with $\|E\|_2 = j \cdot O(\varepsilon) \cdot \|A\|_2$. Here, as in Lemma 3.1, $\tilde{P}_i$ and $\tilde{Q}_i$ are floating point orthogonal matrices and P_i and Q_i are exact orthogonal matrices.

Proof. Let $\bar{P}_j \equiv P_j \cdots P_1$ and $\bar{Q}_j \equiv Q_1 \cdots Q_j$. We wish to show that $A_j \equiv \mathrm{fl}(\tilde{P}_j A_{j-1} \tilde{Q}_j) = \bar{P}_j (A + E_j) \bar{Q}_j$ for some $\|E_j\|_2 = jO(\varepsilon)\|A\|_2$. We use Lemma 3.1 recursively. The result is vacuously true for $j = 0$. Now assume that the result is true for $j - 1$. Then we compute

$$\begin{aligned} B &= \mathrm{fl}(\tilde{P}_j A_{j-1}) \\ &= P_j (A_{j-1} + E') \text{ by Lemma 3.1} \\ &= P_j (\bar{P}_{j-1}(A + E_{j-1})\bar{Q}_{j-1} + E') \text{ by induction} \\ &= \bar{P}_j (A + E_{j-1} + \bar{P}_{j-1}^T E' \bar{Q}_{j-1}^T)\bar{Q}_{j-1} \\ &\equiv \bar{P}_j (A + E'')\bar{Q}_{j-1}, \end{aligned}$$

where

$$\begin{aligned} \|E''\|_2 &= \|E_{j-1} + \bar{P}_{j-1}^T E' \bar{Q}_{j-1}^T\|_2 \le \|E_{j-1}\|_2 + \|\bar{P}_{j-1}^T E' \bar{Q}_{j-1}^T\|_2 \\ &= \|E_{j-1}\|_2 + \|E'\|_2 \\ &= jO(\varepsilon)\|A\|_2 \end{aligned}$$

since $\|E_{j-1}\|_2 = (j-1)O(\varepsilon)\|A\|_2$ and $\|E'\|_2 = O(\varepsilon)\|A\|_2$. Postmultiplication by $\tilde{Q}_j$ is handled in the same way. □

3.4.4. Why Orthogonal Matrices?

Let us consider how the error would grow if we were to multiply by a sequence of *nonorthogonal* matrices in Theorem 3.5 instead of orthogonal matrices. Let X be the exact nonorthogonal transformation and $\tilde{X}$ be its floating point approximation. Then the usual floating point error analysis of matrix multiplication tells us that

$$\mathrm{fl}(\tilde{X}A) = XA + E = X(A + X^{-1}E) \equiv X(A + F),$$

where $\|E\|_2 \le O(\varepsilon)\|X\|_2 \cdot \|A\|_2$ and so $\|F\|_2 \le \|X^{-1}\|_2 \cdot \|E\|_2 \le O(\varepsilon) \cdot \kappa_2(X) \cdot \|A\|_2$.

So the error $\|E\|_2$ is magnified by the condition number $\kappa_2(X) \ge 1$. In a larger product $\tilde{X}_k \cdots \tilde{X}_1 A \tilde{Y}_1 \cdots \tilde{Y}_k$ the error would be magnified by $\prod_i \kappa_2(X_i) \cdot \kappa_2(Y_i)$. This factor is minimized if and only if all X_i and Y_i are orthogonal (or scalar multiples of orthogonal matrices), in which case the factor is one.

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

3.5. Rank-Deficient Least Squares Problems

So far we have assumed that A has full rank when minimizing $\|Ax - b\|_2$. What happens when A is rank deficient or "close" to rank deficient? Such problems arise in practice in many ways, such as extracting signals from noisy data, solution of some integral equations, digital image restoration, computing inverse Laplace transforms, and so on [141, 142]. These problems are very ill-conditioned, so we will need to impose extra conditions on their solutions to make them well-conditioned. Making an ill-conditioned problem well-conditioned by imposing extra conditions on the solution is called *regularization* and is also done in other fields of numerical analysis when ill-conditioned problems arise.

For example, the next proposition shows that if A is exactly rank deficient, then the least squares solution is not even unique.

PROPOSITION 3.1. *Let A be m-by-n with $m \geq n$ and rank $A = r < n$. Then there is an $n - r$ dimensional set of vectors x that minimize $\|Ax - b\|_2$.*

Proof. Let $Az = 0$. Then if x minimizes $\|Ax - b\|_2$, so does $x + z$. □

Because of roundoff in the entries of A, or roundoff during the computation, it is most often the case that A will have one or more very small computed singular values, rather than some exactly zero singular values. The next proposition shows that in this case, the unique solution is likely to be very large and is certainly very sensitive to error in the right-hand side b (see also Theorem 3.4).

PROPOSITION 3.2. *Let $\sigma_{\min} = \sigma_{\min}(A)$, the smallest singular value of A. Assume $\sigma_{\min} > 0$. Then*

1. *if x minimizes $\|Ax - b\|_2$, then $\|x\|_2 \geq |u_n^T b|/\sigma_{\min}$, where u_n is the last column of U in $A = U\Sigma V^T$.*

2. *changing b to $b + \delta b$ can change x to $x + \delta x$, where $\|\delta x\|_2$ is as large as $\|\delta b\|_2/\sigma_{\min}$.*

In other words, if A is nearly rank deficient ($\sigma_{\min}$ is small), then the solution x is ill-conditioned and possibly very large.

Proof. For part 1, $x = A^+ b = V\Sigma^{-1}U^T b$, so $\|x\|_2 = \|\Sigma^{-1}U^T b\|_2 \geq |(\Sigma^{-1}U^T b)_n| = |u_n^T b|/\sigma_{\min}$. For part 2, choose δb parallel to u_n. □

We begin our discussion of regularization by showing how to regularize an *exactly* rank-deficient least squares problem: Suppose A is m-by-n with rank $r < n$. Within the $(n - r)$-dimensional solution space, we will look for the unique solution of smallest norm. This solution is characterized by the following proposition.

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

PROPOSITION 3.3. *When A is exactly singular, the x that minimize $\|Ax-b\|_2$ can be characterized as follows. Let $A = U\Sigma V^T$ have rank $r < n$, and write the SVD of A as*

$$A = [U_1, U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} [V_1, V_2]^T = U_1 \Sigma_1 V_1^T, \tag{3.1}$$

where Σ_1 is $r \times r$ and nonsingular and U_1 and V_1 have r columns. Let $\sigma = \sigma_{\min}(\Sigma_1)$, the smallest nonzero *singular value of A. Then*

1. *all solutions x can be written $x = V_1\Sigma_1^{-1}U_1^T b + V_2 z$, z an arbitrary vector.*
2. *the solution x has minimal norm $\|x\|_2$ precisely when $z = 0$, in which case $x = V_1\Sigma_1^{-1}U_1^T b$ and $\|x\|_2 \le \|b\|_2/\sigma$.*
3. *changing b to $b + \delta b$ can change the minimal norm solution x by at most $\|\delta b\|_2/\sigma$.*

In other words, the norm and condition number of the unique minimal norm solution x depend on the smallest nonzero *singular value of A.*

Proof. Choose $\tilde{U}$ so $[U, \tilde{U}] = [U_1, U_2, \tilde{U}]$ is an $m \times m$ orthogonal matrix. Then

$$\begin{aligned} \|Ax-b\|_2^2 &= \|[U,\tilde{U}]^T(Ax-b)\|_2^2 \\ &= \left\| \begin{bmatrix} U_1^T \\ U_2^T \\ \tilde{U}^T \end{bmatrix} (U_1\Sigma_1 V_1^T x - b) \right\|_2^2 \\ &= \left\| \begin{bmatrix} \Sigma_1 V_1^T x - U_1^T b \\ U_2^T b \\ \tilde{U}^T b \end{bmatrix} \right\|_2^2 \\ &= \|\Sigma_1 V_1^T x - U_1^T b\|_2^2 + \|U_2^T b\|_2^2 + \|\tilde{U}^T b\|_2^2. \end{aligned}$$

1. $\|Ax-b\|_2$ is minimized when $\Sigma_1 V_1^T x = U_1^T b$, or $x = V_1\Sigma_1^{-1}U_1^T b + V_2 z$ since $V_1^T V_2 z = 0$ for all z.
2. Since the columns of V_1 and V_2 are mutually orthogonal, the Pythagorean theorem implies that $\|x\|_2^2 = \|V_1\Sigma_1^{-1}U_1^T b\|_2^2 + \|V_2 z\|_2^2$, and this is minimized by $z = 0$.
3. Changing b by δb changes x by at most $\|V_1\Sigma_1^{-1}U_1^T\delta b\|_2 \le \|\Sigma_1^{-1}\|_2\|\delta b\|_2 = \|\delta b\|_2/\sigma$. □

Proposition 3.3 tells us that the minimum norm solution x is unique and may be well-conditioned if the smallest *nonzero* singular value is not too small. This is key to a practical algorithm, discussed in the next section.

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

EXAMPLE 3.7. Suppose that we are doing medical research on the effect of a certain drug on blood sugar level. We collect data from each patient (numbered from $i = 1$ to m) by recording his or her initial blood sugar level ($a_{i,1}$), final blood sugar level (b_i), the amount of drug administered ($a_{i,2}$), and other medical quantities, including body weights on each day of a week-long treatment ($a_{i,3}$ through $a_{i,9}$). In total, there are $n < m$ medical quantities measured for each patient. Our goal is to predict b_i given $a_{i,1}$ through $a_{i,n}$, and we formulate this as the least squares problem $\min_x \|Ax - b\|_2$. We plan to use x to predict the final blood sugar level b_j of future patient j by computing the dot product $\sum_{k=1}^{n} a_{jk}x_k$.

Since people's weight generally does not change significantly from day to day, it is likely that columns 3 through 9 of matrix A, which contain the weights, are very similar. For the sake of argument, suppose that columns 3 and 4 are *identical* (which may be the case if the weights are rounded to the nearest pound). This means that matrix A is rank deficient and that $x_0 = [0, 0, 1, -1, 0, \ldots, 0]^T$ is a right null vector of A. So if x is a (minimum norm) solution of the least squares problem $\min_x \|Ax - b\|_2$, then $x + \beta x_0$ is also a (nonminimum norm) solution for *any* scalar β, including, say, $\beta = 0$ and $\beta = 10^6$. Is there any reason to prefer one value of β over another? The value 10^6 is clearly not a good one, since future patient j, who gains one pound between days 1 and 2, will have that difference of one pound multiplied by 10^6 in the predictor $\sum_{k=1}^{n} a_{jk}x_k$ of final blood sugar level. It is much more reasonable to choose $\beta = 0$, corresponding to the minimum norm solution x. ◇

For further justification of using the minimum norm solution for rank-deficient problems, see [141, 142].

When A is square and nonsingular, the unique solution of $Ax = b$ is of course $b = A^{-1}x$. If A has more rows than columns and is possibly rank-deficient, the unique minimum-norm least squares solution may be similarly written $b = A^{+}b$, where the *Moore–Penrose pseudoinverse* A^{+} is defined as follows.

DEFINITION 3.2. *(Moore–Penrose pseudoinverse A^{+} for possibly rank-deficient A)*

Let $A = U\Sigma V^T = U_1\Sigma_1 V_1^T$ as in equation (3.1). Then $A^{+} \equiv V_1\Sigma_1^{-1}U_1^T$. This is also written $A^{+} = V^T\Sigma^{+}U$, where $\Sigma^{+} = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix}^{+} = \begin{bmatrix} \Sigma_1^{-1} & 0 \\ 0 & 0 \end{bmatrix}$.

So the solution of the least squares problem is always $x = A^{+}b$, and when A is rank deficient, x has minimum norm.

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

3.5.1. Solving Rank-Deficient Least Squares Problems Using the SVD

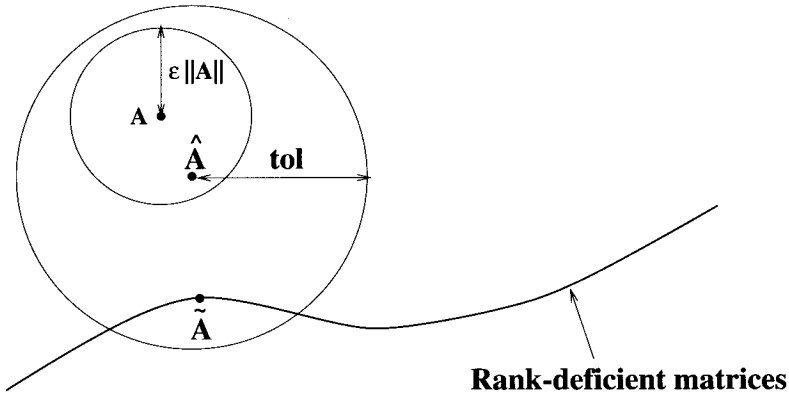
Our goal is to compute the minimum norm solution x, despite roundoff. In the last section, we saw that the minimal norm solution was unique and had a condition number depending on the smallest nonzero singular value. Therefore, computing the minimum norm solution requires knowing the smallest nonzero singular value and hence also the rank of A. The main difficulty is that the rank of a matrix changes discontinuously as a function of the matrix.

For example, the 2-by-2 matrix $A = \mathrm{diag}(1, 0)$ is exactly singular, and its smallest nonzero singular value is $\sigma = 1$. As described in Proposition 3.3, the minimum norm least squares solution to $\min_x \|Ax - b\|_2$ with $b = [1, 1]^T$ is $x = [1, 0]^T$, with condition number $1/\sigma = 1$. But if we make an arbitrarily tiny perturbation to get $\hat{A} = \mathrm{diag}(1, \epsilon)$, then σ drops to ϵ and $x = [1, 1/\epsilon]^T$ becomes enormous, as does its condition number $1/\epsilon$. In general, roundoff will make such tiny perturbations, of magnitude $O(\varepsilon)\|A\|_2$. As we just saw, this can increase the condition number from $1/\sigma$ to $1/\varepsilon$.

We deal with this discontinuity algorithmically as follows. In general each computed singular value $\hat{\sigma}_i$ satisfies $|\hat{\sigma}_i - \sigma_i| \leq O(\varepsilon)\|A\|_2$. This is a consequence of backward stability: the computed SVD will be the exact SVD of a slightly different matrix: $\hat{A} = \hat{U}\hat{\Sigma}\hat{V}^T = A + \delta A$, with $\|\delta A\| = O(\varepsilon) \cdot \|A\|$. (This is discussed in detail in Chapter 5.) This means that any $\hat{\sigma}_i \leq O(\varepsilon)\|A\|_2$ can be treated as zero, because roundoff makes it indistinguishable from 0. In the above 2-by-2 example, this means we would set the ϵ in $\hat{A}$ to zero before solving the least squares problem. This would raise the smallest nonzero singular value from ϵ to 1 and correspondingly decrease the condition number from $1/\epsilon$ to $1/\sigma = 1$.

More generally, let tol be a user-supplied measure of uncertainty in the data A. Roundoff implies that $\mathrm{tol} \geq \varepsilon \cdot \|A\|$, but it may be larger, depending on the source of the data in A. Now set $\tilde{\sigma}_i = \hat{\sigma}_i$ if $\hat{\sigma}_i > \mathrm{tol}$, and $\tilde{\sigma}_i = 0$ otherwise. Let $\tilde{\Sigma} = \mathrm{diag}(\tilde{\sigma}_i)$. We call $\hat{U}\tilde{\Sigma}\hat{V}^T$ the *truncated SVD* of A, because we have set singular values smaller than tol to zero. Now we solve the least squares problem using the truncated SVD instead of the original SVD. This is justified since $\|\hat{U}\tilde{\Sigma}\hat{V}^T - \hat{U}\hat{\Sigma}\hat{V}^T\|_2 = \|\hat{U}(\tilde{\Sigma} - \hat{\Sigma})\hat{V}^T\|_2 < \mathrm{tol}$; i.e., the change in A caused by changing each $\hat{\sigma}_i$ to $\tilde{\sigma}_i$ is less than the user's inherent uncertainty in the data. The motivation for using $\tilde{\Sigma}$ instead of $\hat{\Sigma}$ is that of all matrices within distance tol of $\hat{\Sigma}$, $\tilde{\Sigma}$ maximizes the smallest nonzero singular value σ. In other words, it minimizes both the norm of the minimum norm least squares solution x and its condition number. The picture below illustrates the geometric relationships among the input matrix A, $\hat{A} = \hat{U}\hat{\Sigma}\hat{V}^T$, and $\tilde{A} = \hat{U}\tilde{\Sigma}\hat{V}^T$, where we we think of each matrix as a point in Euclidean space $\mathbb{R}^{m \cdot n}$. In this space, the rank-deficient matrices form a surface, as shown below:

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php



EXAMPLE 3.8. We illustrate the above procedure on two 20-by-10 rank-deficient matrices A_1 (of rank $r_1 = 5$) and A_2 (of rank $r_2 = 7$). We write the SVDs of either A_1 or A_2 as $A_i = U_i \Sigma_i V_i^T$, where the common dimension of U_i, Σ_i, and V_i is the rank r_i of A_i; this is the same notation as in Proposition 3.3. The r_i nonzero singular values of A_i (singular values of Σ_i) are shown as red x's in Figure 3.4 (for A_1) and Figure 3.5 (for A_2). Note that A_1 in Figure 3.4 has five large nonzero singular values (all slightly exceeding 1 and so plotted on top of one another, on the right edge the graph), whereas the seven nonzero singular values of A_2 in Figure 3.5 range down to $1.2 \cdot 10^{-9} \approx$ tol.

We then choose an r_i-dimensional vector x_i', and let $x_i = V_i x_i'$ and $b_i = A_i x_i = U_i \Sigma_i x_i'$, so x_i is the exact minimum norm solution minimizing $\|A_i x_i - b_i\|_2$. Then we consider a sequence of perturbed problems $A_i + \delta A$, where the perturbation δA is chosen randomly to have a range of norms, and solve the least squares problems $\|(A_i + \delta A)y_i - b_i\|_2$ using the truncated least squares procedure with tol $= 10^{-9}$. The blue lines in Figures 3.4 and 3.5 plot the computed rank of $A_i + \delta A$ (number of computed singular values exceeding tol $= 10^{-9}$) versus $\|\delta A\|_2$ (in the top graphs), and the error $\|y_i - x_i\|_2 / \|x_i\|_2$ (in the bottom graphs). The Matlab code for producing these figures is in HOMEPAGE/Matlab/RankDeficient.m.

The simplest case is in Figure 3.4, so we consider it first. $A_1 + \delta A$ will have five singular values near or slightly exceeding 1 and the other five equal to $\|\delta A\|_2$ or less. For $\|\delta A\|_2 <$ tol, the computed rank of $A_1 + \delta A$ stays the same as that of A_1, namely, 5. The error also increases slowly from near machine epsilon ($\approx 10^{-16}$) to about 10^{-10} near $\|\delta A\|_2 =$ tol, and then both the rank and the error jump, to 10 and 1, respectively, for larger $\|\delta A\|_2$. This is consistent with our analysis in Proposition 3.3, which says that the condition number is the reciprocal of the smallest nonzero singular value, i.e., the smallest singular value exceeding tol. For $\|\delta A\|_2 <$ tol, this smallest nonzero singular value is near to, or slightly exceeds, 1. Therefore Proposition 3.3 predicts an error of $\|\delta A\|_2 / O(1) = \|\delta A\|_2$. This well-conditioned situation is confirmed by the small error plotted to the left of $\|\delta A\|_2 =$ tol in the bottom graph of Figure 3.4. On the other hand, when $\|\delta A\|_2 >$ tol, then the smallest nonzero

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

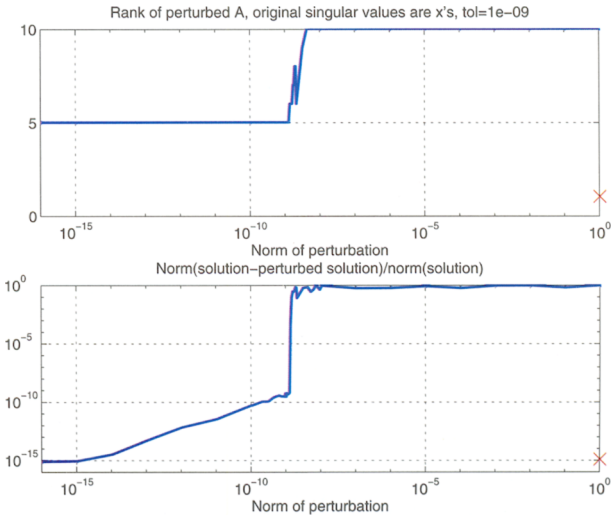


Fig. 3.4. *Graph of truncated least squares solution of* $\min_{y_1} \|(A_1 + \delta A)y_1 - b_1\|_2$, *using* tol $= 10^{-9}$. *The singular values of* A_1 *are shown as red x's. The norm* $\|\delta A\|_2$ *is the horizontal axis. The top graph plots the rank of* $A_1 + \delta A$, *i.e., the numbers of singular values exceeding* tol. *The bottom graph plots* $\|y_1 - x_1\|_2/\|x_1\|_2$, *where* x_1 *is the solution with* $\delta A = 0$.

singular value is $O(\|\delta A\|_2)$, which is quite small, causing the error to jump to $\|\delta A\|_2/O(\|\delta A\|_2) = O(1)$, as shown to the right of $\|\delta A\|_2 =$ tol in the bottom graph of Figure 3.4.

In Figure 3.5, the nonzero singular values of A_2 are also shown as red x's; the smallest one, $1.2 \cdot 10^{-9}$, is just larger than tol. So the predicted error when $\|\delta A\|_2 <$ tol is $\|\delta A\|_2/10^{-9}$, which grows to $O(1)$ when $\|\delta A\|_2 =$ tol. This is confirmed by the bottom graph in Figure 3.5. $\diamond$

3.5.2. Solving Rank-Deficient Least Squares Problems Using QR with Pivoting

A cheaper but sometimes less accurate alternative to the SVD is *QR with pivoting*. In exact arithmetic, if A had rank $r < n$ and its first r columns were independent, then its QR decomposition would look like

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

where R_{11} is r-by-r and nonsingular and R_{12} is r-by-$(n-r)$. With roundoff, we might hope to compute

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \\ 0 & 0 \end{bmatrix}$$

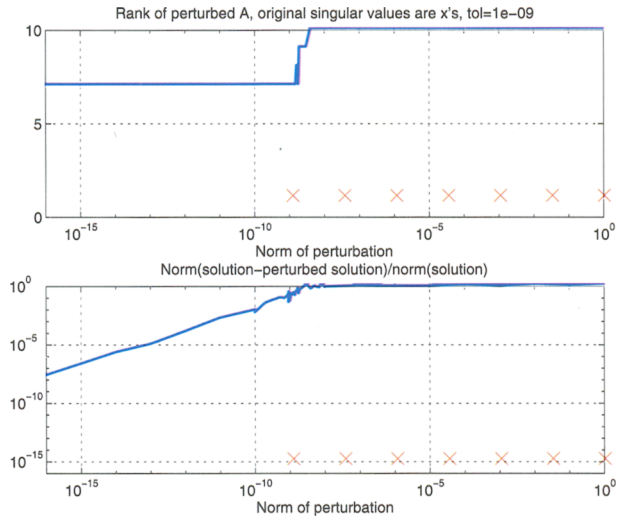


Fig. 3.5. *Graph of truncated least squares solution of* $\min_{y_2} \|(A_2 + \delta A)y_2 - b_2\|_2$, *using* tol $= 10^{-9}$. *The singular values of* A_2 *are shown as red* x*'s. The norm* $\|\delta A\|_2$ *is the horizontal axis. The top graph plots the rank of* $A_2 + \delta A$, *i.e., the numbers of singular values exceeding* tol. *The bottom graph plots* $\|y_2 - x_2\|_2/\|x_2\|_2$, *where* x_2 *is the solution with* $\delta A = 0$.

with $\|R_{22}\|_2$ very small, on the order of $\varepsilon\|A\|_2$. In this case we could just set $R_{22} = 0$ and minimize $\|Ax-b\|_2$ as follows: let $[Q, \tilde{Q}]$ be square and orthogonal so that

$$\begin{aligned} \|Ax - b\|_2^2 &= \left\| \begin{bmatrix} Q^T \\ \tilde{Q}^T \end{bmatrix} (Ax - b) \right\|_2^2 = \left\| \begin{bmatrix} Rx - Q^T b \\ -\tilde{Q}^T b \end{bmatrix} \right\|_2^2 \\ &= \|Rx - Q^T b\|_2^2 + \|\tilde{Q}^T b\|_2^2. \end{aligned}$$

Write $Q = [Q_1, Q_2]$ and $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ conformally with $R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}$ so that

$$\|Ax - b\|_2^2 = \|R_{11}x_1 + R_{12}x_2 - Q_1^T b\|_2^2 + \|Q_2^T b\|_2^2 + \|\tilde{Q}^T b\|_2^2$$

is minimized by choosing $x = \begin{bmatrix} R_{11}^{-1}(Q_1^T b - R_{12}x_2) \\ x_2 \end{bmatrix}$ for any x_2. Note that the choice $x_2 = 0$ does not necessarily minimize $\|x\|_2$, but it is a reasonable choice, especially if R_{11} is well-conditioned and $R_{11}^{-1}R_{12}$ is small.

Unfortunately, this method is not reliable since R may be nearly rank deficient even if no R_{22} is small. For example, the n-by-n bidiagonal matrix

$$A = \begin{bmatrix} \frac{1}{2} & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \frac{1}{2} \end{bmatrix}$$

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

has $\sigma_{\min}(A) \approx 2^{-n}$, but $A = Q \cdot R$ with $Q = I$ and $R = A$, and no R_{22} is small.

To deal with this failure to recognize rank deficiency, we may do *QR with column pivoting.* This means that we factorize $AP = QR$, P being a permutation matrix. This idea is that at step i (which ranges from 1 to n, the number of columns) we select from the unfinished part of A (columns i to n and rows i to m) the column of largest norm and exchange it with the ith column. We then proceed to compute the usual Householder transformation to zero out column i in entries $i+1$ to m. This pivoting strategy attempts to keep R_{11} as well-conditioned as possible and R_{22} as small as possible.

EXAMPLE 3.9. If we compute the QR decomposition with column pivoting to the last example (.5 on the diagonal and 1 on the superdiagonal) with $n = 11$, we get $R_{11,11} = 4.23 \cdot 10^{-4}$, a reasonable approximation to $\sigma_{\min}(A) = 3.66 \cdot 10^{-4}$. Note that $R_{nn} \geq \sigma_{\min}(A)$ since $\sigma_{\min}(A)$ is the norm of the smallest perturbation that can lower the rank, and setting R_{nn} to 0 lowers rank. ◇

One can show only $\frac{R_{nn}}{\sigma_{\min}(A)} \lesssim 2^n$, but usually R_{nn} is a reasonable approximation to $\sigma_{\min}(A)$. The worst case, however, is as bad as worst-case pivot growth in GEPP.

More sophisticated pivoting schemes than QR with column pivoting, called *rank-revealing* QR algorithms, have been a subject of much recent study. Rank-revealing QR algorithms that detect rank more reliably and sometimes also faster than QR with column pivoting have been developed [28, 30, 48, 50, 109, 126, 128, 150, 196, 236]. We discuss them further in the next section.

QR decomposition with column pivoting is available as subroutine `sgeqpf` in LAPACK. LAPACK also has several similar factorizations available: RQ (`sgerqf`), LQ (`sgelqf`), and QL (`sgeqlf`). Future LAPACK releases will contain improved versions of QR.

3.6. Performance Comparison of Methods for Solving Least Squares Problems

What is the fastest algorithm for solving dense least squares problems? As discussed in section 3.2, solving the normal equations is fastest, followed by QR and the SVD. If A is quite well-conditioned, then the normal equations are about as accurate as the other methods, so even though the normal equations are not numerically stable, they may be used as well. When A is not well-conditioned but far from rank deficient, we should use QR.

Since the design of fast algorithms for rank-deficient least squares problems is a current research area, it is difficult to recommend a single algorithm to use. We summarize a recent study [206] that compared the performance of several algorithms, comparing them to the fastest stable algorithm for the non–rank-deficient case: QR without pivoting, implemented using Householder transformations as described in section 3.4.1, with memory hierarchy optimizations

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

described in Question 3.17. These comparisons were made in double precision arithmetic on an IBM RS6000/590. Included in the comparison were the rank-revealing QR algorithms mentioned in section 3.5.2 and various implementations of the SVD (see section 5.4). Matrices of various sizes and with various singular value distributions were tested. We present results for two singular value distributions:

Type 1: random matrices, where each entry is uniformly distributed from -1 to 1;

Type 2: matrices with singular values distributed geometrically from 1 to ε (in other words, the ith singular value is γ^i, where γ is chosen so that $\gamma^n = \varepsilon$).

Type 1 matrices are generally well-conditioned, and Type 2 matrices are rank-deficient. We tested small square matrices ($n = m = 20$) and large square matrices ($m = n = 1600$). We tested square matrices because if m is sufficiently greater than n in the m-by-n matrix A, it is cheaper to do a QR decomposition as a "preprocessing step" and then perform rank-revealing QR or the SVD on R. (This is done in LAPACK.) If $m \gg n$, then the initial QR decomposition dominates the the cost of the subsequent operations on the n-by-n matrix R, and all the algorithms cost about the same.

The fastest version of rank-revealing QR was that of [30, 196]. On Type 1 matrices, this algorithm ranged from 3.2 times slower than QR without pivoting for $n = m = 20$ to just 1.1 times slower for $n = m = 1600$. On Type 2 matrices, it ranged from 2.3 times slower (for $n = m = 20$) to 1.2 times slower (for $n = m = 1600$). In contrast, the current LAPACK algorithm, `dgeqpf`, was 2 times to 2.5 times slower for both matrix types.

The fastest version of the SVD was the one in [58], although one based on divide-and-conquer (see section 5.3.3) was about equally fast for $n = m = 1600$. (The one based on divide-and-conquer also used much less memory.) For Type 1 matrices, the SVD algorithm was 7.8 times slower (for $n = m = 20$) to 3.3 times slower (for $n = m = 1600$). For Type 2 matrices, the SVD algorithm was 3.5 times slower (for $n = m = 20$) to 3.0 times slower (for $n = m = 1600$). In contrast, the current LAPACK algorithm, `dgelss`, ranged from 4 times slower (for Type 2 matrices with $n = m = 20$) to 97 times slower (for Type 1 matrices with $n = m = 1600$). This enormous slowdown is apparently due to memory hierarchy effects.

Thus, we see that there is a tradeoff between reliability and speed in solving rank-deficient least squares problems: QR without pivoting is fastest but least reliable, the SVD is slowest but most reliable, and rank-revealing QR is in-between. If $m \gg n$, all algorithms cost about the same. The choice of algorithm depends on the relative importance of speed and reliability to the user.

Future LAPACK releases will contain improved versions of both rank-revealing QR and SVD algorithms for the least squares problem.

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

3.7. References and Other Topics for Chapter 3

The best recent reference on least squares problems is [33], which also discusses variations on the basic problem discussed here (such as constrained, weighted, and updating least squares), different ways to regularize rank-deficient problems, and software for sparse least squares problems. See also chapter 5 of [121] and [168]. Perturbation theory and error bounds for the least squares solution are discussed in detail in [149]. Rank-revealing QR decompositions are discussed in [28, 30, 48, 50, 126, 150, 196, 206, 236]. In particular, these papers examine the tradeoff between cost and accuracy in rank determination, and in [206] there is a comprehensive performance comparison of the available methods for rank-deficient least squares problems.

3.8. Questions for Chapter 3

QUESTION 3.1. *(Easy)* Show that the two variations of Algorithm 3.1, CGS and MGS, are mathematically equivalent by showing that the two formulas for r_{ji} yield the same results in exact arithmetic.

QUESTION 3.2. *(Easy)* This question will illustrate the difference in numerical stability among three algorithms for computing the QR factorization of a matrix: Householder QR (Algorithm 3.2), CGS (Algorithm 3.1), and MGS (Algorithm 3.1). Obtain the Matlab program QRStability.m from HOMEPAGE/Matlab/QRStability.m. This program generates random matrices with user-specified dimensions `m` and `n` and condition number `cnd`, computes their QR decomposition using the three algorithms, and measures the accuracy of the results. It does this with the *residual* $\|A - Q \cdot R\|/\|A\|$, which should be around machine epsilon ε for a stable algorithm, and the *orthogonality of* Q $\|Q^T \cdot Q - I\|$, which should also be around ε. Run this program for small matrix dimensions (such as `m`$= 6$ and `n`$= 4$), modest numbers of random matrices (`samples`$= 20$), and condition numbers ranging from `cnd`$= 1$ up to `cnd`$= 10^{15}$. Describe what you see. Which algorithms are more stable than others? See if you can describe how large $\|Q^T \cdot Q - I\|$ can be as a function of choice of algorithm, `cnd` and ε.

QUESTION 3.3. *(Medium; Hard)* Let A be m-by-n, $m \geq n$, and have full rank.

1. *(Medium)* Show that $\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \cdot \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$ has a solution where x minimizes $\|Ax - b\|_2$. One reason for this formulation is that we can apply iterative refinement to this linear system if we want a more accurate answer (see section 2.5).

2. *(Medium)* What is the condition number of the coefficient matrix in terms of the singular values of A? Hint: Use the SVD of A.

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

3. *(Medium)* Give an explicit expression for the inverse of the coefficient matrix, as a block 2-by-2 matrix. Hint: Use 2-by-2 block Gaussian elimination. Where have we previously seen the (2,1) block entry?

4. *(Hard)* Show how to use the QR decomposition of A to implement an iterative refinement algorithm to improve the accuracy of x.

QUESTION 3.4. *(Medium) Weighted least squares:* If some components of $Ax - b$ are more important than others, we can weight them with a scale factor d_i and solve the weighted least squares problem $\min \|D(Ax-b)\|_2$ instead, where D has diagonal entries d_i. More generally, recall that if C is symmetric positive definite, then $\|x\|_C \equiv (x^T C x)^{1/2}$ is a norm, and we can consider minimizing $\|Ax - b\|_C$. Derive the normal equations for this problem, as well as the formulation corresponding to the previous question.

QUESTION 3.5. *(Medium; Z. Bai)* Let $A \in \mathbb{R}^{n\times n}$ be positive definite. Two vectors u_1 and u_2 are called A-orthogonal if $u_1^T A u_2 = 0$. If $U \in \mathbb{R}^{n\times r}$ and $U^T A U = I$, then the columns of U are said to be A-orthonormal. Show that every subspace has an A-orthonormal basis.

QUESTION 3.6. *(Easy; Z. Bai)* Let A have the form

$$A = \begin{bmatrix} R \\ S \end{bmatrix},$$

where R is n-by-n and upper triangular, and S is m-by-n and dense. Describe an algorithm using Householder transformations for reducing A to upper triangular form. Your algorithm should not "fill in" the zeros in R and thus require fewer operations than would Algorithm 3.2 applied to A.

QUESTION 3.7. *(Medium; Z. Bai)* If $A = R + uv^T$, where R is an upper triangular matrix, and u and v are column vectors, describe an efficient algorithm to compute the QR decomposition of A. Hint: Using Givens rotations, your algorithm should take $O(n^2)$ operations. In contrast, Algorithm 3.2 would take $O(n^3)$ operations.

QUESTION 3.8. *(Medium; Z. Bai)* Let $x \in \mathbb{R}^n$ and let P be a Householder matrix such that $Px = \pm\|x\|_2 e_1$. Let $G_{1,2}, \ldots, G_{n-1,n}$ be Givens rotations, and let $Q = G_{1,2} \cdots G_{n-1,n}$. Suppose $Qx = \pm\|x\|_2 e_1$. Must P equal Q? (You need to give a proof or a counterexample.)

QUESTION 3.9. *(Easy; Z. Bai)* Let A be m-by-n, with SVD $A = U\Sigma V^T$. Compute the SVDs of the following matrices in terms of U, Σ, and V:

1. $(A^T A)^{-1}$,
2. $(A^T A)^{-1} A^T$,

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

3. $A(A^TA)^{-1}$,

4. $A(A^TA)^{-1}A^T$.

QUESTION 3.10. *(Medium; R. Schreiber)* Let A_k be a best rank-k approximation of the matrix A, as defined in Part 9 of Theorem 3.3. Let σ_i be the ith singular value of A. Show that A_k is unique if $\sigma_k > \sigma_{k+1}$.

QUESTION 3.11. *(Easy; Z. Bai)* Let A be m-by-n. Show that $X = A^+$ (the Moore–Penrose pseudoinverse) minimizes $\|AX - I\|_F$ over all n-by-m matrices X. What is the value of this minimum?

QUESTION 3.12. *(Medium; Z. Bai)* Let A, B, and C be matrices with dimensions such that the product A^TCB^T is well defined. Let $\mathcal{X}$ be the set of matrices X minimizing $\|AXB - C\|_F$, and let X_0 be the unique member of $\mathcal{X}$ minimizing $\|X\|_F$. Show that $X_0 = A^+CB^+$. Hint: Use the SVDs of A and B.

QUESTION 3.13. *(Medium; Z. Bai)* Show that the Moore–Penrose pseudoinverse of A satisfies the following identities:

$$\begin{aligned} AA^+A &= A, \\ A^+AA^+ &= A^+, \\ A^+A &= (A^+A)^T, \\ AA^+ &= (AA^+)^T. \end{aligned}$$

QUESTION 3.14. *(Medium)* Prove part 4 of Theorem 3.3: Let $H = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$, where A is square and $A = U\Sigma V^T$ is its SVD. Let $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$, $U = [u_1, \ldots, u_n]$, and $V = [v_1, \ldots, v_n]$. Prove that the $2n$ eigenvalues of H are $\pm\sigma_i$, with corresponding unit eigenvectors $\frac{1}{\sqrt{2}}\begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}$. Extend to the case of rectangular A.

QUESTION 3.15. *(Medium)* Let A be m-by-n, $m < n$, and of full rank. Then $\min \|Ax - b\|_2$ is called an *underdetermined least squares problem.* Show that the solution is an $(n - m)$-dimensional set. Show how to compute the unique minimum norm solution using appropriately modified normal equations, QR decomposition, and SVD.

QUESTION 3.16. *(Medium)* Prove Lemma 3.1.

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

QUESTION 3.17. *(Hard)* In section 2.6.3, we showed how to reorganize Gaussian elimination to perform Level 2 BLAS and Level 3 BLAS at each step in order to exploit the higher speed of these operations. In this problem, we will show how to apply a sequence of Householder transformations using Level 2 and Level 3 BLAS.

1. Let $u_1, \ldots, u_b$ be a sequence of vectors of dimension n, where $\|u_i\|_2 = 1$ and the first $i-1$ components of u_i are zero. Let $P = P_b \cdot P_{b-1} \cdots P_1$, where $P_i = I - 2u_i u_i^T$ is a Householder transformation. Show that there is a b-by-b lower triangular matrix T such that $P = I - UTU^T$, where $U = [u_1, \ldots, u_b]$. In particular, provide an algorithm for computing the entries of T. This identity shows that we can replace multiplication by b Householder transformations P_1 through P_b by three matrix multiplications by U, T, and U^T (plus the cost of computing T).

2. Let House(x) be a function of the vector x which returns a unit vector u such that $(I - 2uu^T)x = \|x\|_2 e_1$; we showed how to implement House(x) in section 3.4. Then Algorithm 3.2 for computing the QR decomposition of the m-by-n matrix A may be written as

   for $i = 1 : m$
   $u_i = \text{House}(A(i : m, i))$
   $P_i = I - 2u_i u_i^T$
   $A(i : m, i : n) = P_i A(i : m, i : n)$
   endfor

   Show how to implement this in terms of the Level 2 BLAS in an efficient way (in particular, matrix-vector multiplications and rank-1 updates). What is the floating point operation count? (Just the high-order terms in n and m are enough.) It is sufficient to write a short program in the same notation as above (although trying it in Matlab and comparing with Matlab's own QR factorization are a good way to make sure that you are right!).

3. Using the results of step (1), show how to implement QR decomposition in terms of Level 3 BLAS. What is the operation count? This technique is used to accelerate the QR decomposition, just as we accelerated Gaussian elimination in section 2.6. It is used in the LAPACK routine `sgeqrf`.

QUESTION 3.18. *(Medium)* It is often of interest to solve *constrained least squares problems*, where the solution x must satisfy a linear or nonlinear constraint in addition to minimizing $\|Ax - b\|_2$. We consider one such problem here. Suppose that we want to choose x to minimize $\|Ax - b\|_2$ subject to the linear constraint $Cx = d$. Suppose also that A is m-by-n, C is p-by-n, and C has full rank. We also assume that $p \leq n$ (so $Cx = d$ is guaranteed to

Downloaded 01/10/18 to 152.23.53.139. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

be consistent) and $n \leq m + p$ (so the system is not underdetermined). Show that there is a unique solution under the assumption that $\begin{bmatrix} A \\ C \end{bmatrix}$ has full column rank. Show how to compute x using two QR decompositions and some matrix-vector multiplications and solving some triangular systems of equations. Hint: Look at LAPACK routine `sgglse` and its description in the LAPACK manual [10] (NETLIB/lapack/lug/lapack_lug.html).

QUESTION 3.19. *(Hard; Programming)* Write a program (in Matlab or any other language) to update a geodetic database using least squares, as described in Example 3.3. Take as input a set of "landmarks," their approximate coordinates (x_i, y_i), and a set of new angle measurements θ_j and distance measurements L_{ij}. The output should be corrections $(\delta x_i, \delta y_i)$ for each landmark, an error bound for the corrections, and a picture (triangulation) of the old and new landmarks.

QUESTION 3.20. *(Hard)* Prove Theorem 3.4.

QUESTION 3.21. *(Medium)* Redo Example 3.1, using a rank-deficient least squares technique from section 3.5.1. Does this improve the accuracy of the high-degree approximating polynomials?