QR Factorization and
Singular Value Decomposition

COS 323
Last time

- Solving non-linear least squares
  - Newton, Gauss-Newton, Levenberg-Marquardt methods
  - Intro to logistic regression
- Dealing with outliers and bad data:
  - Robust regression, least absolute deviation, and iteratively re-weighted least-squares
- Practical considerations
- Solving with Excel and Matlab
Today

- How do we solve least-squares...
  - without incurring condition-squaring effect of normal equations ($A^T Ax = A^T b$)
  - when $A$ is singular, “fat”, or otherwise poorly-specified?

- QR Factorization
  - Householder method

- Singular Value Decomposition

- Total least squares

- Practical notes
Review: Condition Number

- Cond(A) is function of A
- Cond(A) \( \geq 1 \), bigger is **bad**
- Measures how change in input is propagated to change in output

\[
\frac{\| \Delta x \|}{\| x \|} \leq \text{cond}(A) \frac{\| \Delta A \|}{\| A \|}
\]

- E.g., if \( \text{cond}(A) = 451 \) then can lose \( \log(451) = 2.65 \) digits of accuracy in \( x \), compared to precision of \( A \)
Normal Equations are Bad

\[
\frac{\| \Delta x \|}{\| x \|} \leq \text{cond}(A) \frac{\| \Delta A \|}{\| A \|}
\]

• Normal equations involves solving \( A^T A x = A^T b \)

• \( \text{cond}(A^T A) = [\text{cond}(A)]^2 \)

• E.g., if \( \text{cond}(A) = 451 \) then can lose \( \log(451^2) = 5.3 \) digits of accuracy, compared to precision of \( A \)
QR Decomposition
What if we didn’t have to use $A^T A$?

• Suppose we are “lucky”:

\[
\begin{bmatrix}
\# & \# & \ldots & \# \\
0 & \# & \# & \\
0 & 0 & \ddots & \\
0 & \ldots & 0 & \# \\
\vdots & \vdots & & \ddots \\
0 & 0 & \ldots & 0
\end{bmatrix} x \equiv \begin{bmatrix} \\
\# \\
\# \\
\# \\
\# \\
\# \\
\#
\end{bmatrix}
\]

\[
\begin{bmatrix}
R \\
O
\end{bmatrix} x = b
\]

• Upper triangular matrices are nice!
How to make $A$ upper-triangular?

- **Gaussian elimination?**
  - Applying elimination yields $MAx = Mb$
  - Want to find $x$ s.t. minimizes $||Mb-MAx||_2$
  - Problem: $||Mv||_2 \neq ||v||_2$ (i.e., $M$ might “stretch” a vector $v$)
  - Another problem: $M$ may stretch different vectors differently
    - i.e., $M$ does not preserve Euclidean norm
  - i.e., $x$ that minimizes $||Mb-MAx||$ may **not be same** $x$ that minimizes $Ax=b$
QR Factorization

- Can’t usually find $R$ such that $A = \begin{bmatrix} R \\ O \end{bmatrix}$

- Can find $Q$, $R$ such that $A = Q \begin{bmatrix} R \\ O \end{bmatrix}$, so $\begin{bmatrix} R \\ O \end{bmatrix} x = Q^T b$

- If $Q$ orthogonal, doesn’t change least-squares solution
  - $Q^T Q = I$, columns of $Q$ are orthonormal
  - i.e., $Q$ preserves Euclidean norm: $\|Qv\|_2 = \|v\|_2$
Goal of QR

\[ A = Q \begin{bmatrix} R \\ O \end{bmatrix} = Q \begin{bmatrix} \vdots & \vdots & \ldots & \vdots \\ 0 & \ddots & \vdots \\ \vdots & 0 & \ddots & \vdots \\ \vdots & \vdots & 0 & ? \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \]

- \( R \): \( nxn \), upper tri.
- \( O \): \( (m-n)xn \), all zeros
Reformulating Least Squares using QR

\[
\|r\|_2^2 = \|b - Ax\|_2^2
\]

\[
= \left\| b - Q \begin{bmatrix} R \\ O \end{bmatrix} x \right\|_2^2 = \left\| Q^T b - Q^T Q \begin{bmatrix} R \\ O \end{bmatrix} x \right\|_2^2 \quad \text{because } A = Q \begin{bmatrix} R \\ O \end{bmatrix}
\]

\[
= \left\| Q^T b - \begin{bmatrix} R \\ O \end{bmatrix} x \right\|_2^2 \quad \text{because } Q \text{ is orthogonal (} Q^T Q = I )
\]

\[
= \left\| c_1 - Rx + c_2 \right\|_2^2 \quad \text{if we call } Q^T b = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}
\]

\[
= \left\| c_1 - Rx \right\|_2^2 + \left\| c_2 \right\|_2^2
\]

\[
= \left\| c_2 \right\|_2^2 \quad \text{if we choose } x \text{ such that } Rx = c_1
\]
Householder Method for Computing
QR Decomposition
Orthogonalization for Factorization

\[ A = Q \begin{bmatrix} R \\ O \end{bmatrix} \]

- Rough idea:
  - For each i-th column of A, “zero out” rows i+1 and lower
  - Accomplish this by multiplying A with an orthogonal matrix \( H_i \)
  - Equivalently, apply an orthogonal transformation to the i-th column (e.g., rotation, reflection)
  - Q becomes product \( H_1 \ast \ldots \ast H_n \), R contains zero-ed out columns
Householder Transformation

- Accomplishes the critical sub-step of factorization:
  - Given any vector (e.g., a column of A), reflect it so that its last $p$ elements become 0.
  - Reflection preserves length (Euclidean norm)
Computing Householder

- if $a$ is the $k$-th column:

$$a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

$$v = \begin{bmatrix} 0 \\ a_2 \end{bmatrix} - \alpha e_k$$

where $\alpha = -\text{sign}(a_k)\|a_2\|_2$

apply $H$ to $a$ and columns to the right:

$$Hu = u - \left(2 \frac{v^T u}{v^T v}\right)v$$

(*with some shortcuts - see p124)

Exercise: Show $H$ is orthogonal ($H^TH=I$)
Outcome of Householder

\[ H_n \ldots H_1 A = \begin{bmatrix} R \\ O \end{bmatrix} \]

where \( Q^T = H_n \ldots H_1 \)

so \( Q = H_1 \ldots H_n \)

so \( A = Q \begin{bmatrix} R \\ O \end{bmatrix} \)
Review: Least Squares using QR

\[ \| r \|^2 = \| b - Ax \|^2 \]

\[ = \left\| b - Q \begin{bmatrix} R \\ O \end{bmatrix} x \right\|^2 = \left\| Q^T b - Q^T Q \begin{bmatrix} R \\ O \end{bmatrix} x \right\|^2 \]

because \( A = Q \begin{bmatrix} R \\ O \end{bmatrix} \)

\[ = \left\| Q^T b - \begin{bmatrix} R \\ O \end{bmatrix} x \right\|^2 \]

because \( Q \) is orthogonal (\( Q^T Q = I \))

\[ = \left\| c_1 - Rx + c_2 \right\|^2 \]

if we call \( Q^T b = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \)

\[ = \left\| c_1 - Rx \right\|^2 + \left\| c_2 \right\|^2 \]

\[ = \left\| c_2 \right\|^2 \]

if we choose \( x \) such that \( Rx = c_1 \)
Using Householder

- Iteratively compute $H_1$, $H_2$, … $H_n$ and apply to $A$ to get $R$
  - also apply to $b$ to get

$$Q^T b = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

- Solve for $Rx=c_1$ using back-substitution
Alternative Orthogonalization Methods

• **Givens:**
  – Don’t reflect; rotate instead
  – Introduces zeroes into A one at a time
  – More complicated implementation than Householder
  – Useful when matrix is sparse

• **Gram-Schmidt**
  – Iteratively express each new column vector as a linear combination of previous columns, plus some (normalized) orthogonal component
  – Conceptually nice, but suffers from subtractive cancellation
Singular Value Decomposition
**Motivation #1**

- Diagonal matrices are even nicer than triangular ones:

\[
\begin{bmatrix}
# & 0 & 0 & 0 & 0 \\
0 & # & 0 & 0 & 0 \\
0 & 0 & \ddots & 0 & 0 \\
0 & \cdots & \cdots & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \cdots & 0
\end{bmatrix}
\begin{bmatrix}
# \\
# \\
# \\
# \\
# \\
# 
\end{bmatrix}
\]

\[x \approx \begin{bmatrix}
# \\
# \\
# \\
# \\
# \\
# 
\end{bmatrix}\]
Motivation #2

- What if you have fewer data points than parameters in your function?
  - i.e., $A$ is “fat”
  - Intuitively, can’t do standard least squares
  - Recall that solution takes the form $A^T A x = A^T b$
  - When $A$ has more columns than rows, $A^T A$ is singular: can’t take its inverse, etc.
Motivation #3

• What if your data poorly constrains the function?
• Example: fitting to $y=ax^2+bx+c$
Underconstrained Least Squares

• Problem: if problem very close to singular, roundoff error can have a huge effect
  – Even on “well-determined” values!

• Can detect this:
  – Uncertainty proportional to covariance $C = (A^T A)^{-1}$
  – In other words, unstable if $A^T A$ has small values
  – More precisely, care if $x^T (A^T A) x$ is small for any $x$

• Idea: if part of solution unstable, set answer to 0
  – Avoid corrupting good parts of answer
Singular Value Decomposition (SVD)

- Handy mathematical technique that has application to many problems
- Given any $m \times n$ matrix $A$, algorithm to find matrices $U$, $V$, and $W$ such that
  \[ A = U W V^T \]
  - $U$ is $m \times n$ and orthonormal
  - $W$ is $n \times n$ and diagonal
  - $V$ is $n \times n$ and orthonormal
SVD

\[
\begin{pmatrix}
A
\end{pmatrix}
= \begin{pmatrix}
U \\
\end{pmatrix}
\begin{pmatrix}
w_1 & 0 & 0 \\
0 & \ddots & 0 \\
0 & 0 & w_n
\end{pmatrix}
\begin{pmatrix}
V
\end{pmatrix}^T
\]

- Treat as black box: code widely available
In Matlab: \([U,W,V]=\text{svd}(A,0)\)
The $w_i$ are called the singular values of $A$
• If $A$ is singular, some of the $w_i$ will be 0
• In general $\text{rank}(A) = \text{number of nonzero } w_i$
• SVD is mostly unique (up to permutation of singular values, or if some $w_i$ are equal)
SVD and Inverses

• Why is SVD so useful?
• Application #1: inverses

$$A^{-1} = (V^T)^{-1} W^{-1} U^{-1} = V W^{-1} U^T$$
  – Using fact that inverse = transpose for orthogonal matrices
  – Since $W$ is diagonal, $W^{-1}$ also diagonal with reciprocals of entries of $W$
SVD and the Pseudoinverse

- \( A^{-1} = (V^T)^{-1} W^{-1} U^{-1} = V W^{-1} U^T \)

- This fails when some \( w_i \) are 0
  - It’s *supposed* to fail – singular matrix
  - Happens when rectangular \( A \) is **rank deficient**

- Pseudoinverse: if \( w_i = 0 \), set \( 1/w_i \) to 0 (!)
  - “Closest” matrix to inverse
  - Defined for all (even non-square, singular, etc.) matrices
  - Equal to \((A^T A)^{-1} A^T\) if \(A^T A\) invertible
SVD and Condition Number

- Singular values used to compute Euclidean (spectral) norm for a matrix:

\[
\text{cond}(A) = \frac{\sigma_{\text{max}}(A)}{\sigma_{\text{min}}(A)}
\]
SVD and Least Squares

• Solving $Ax=b$ by least squares:
  
  $A^TAx = A^Tb \Rightarrow x = (A^TA)^{-1}A^Tb$

  • Replace with $A^+$: $x = A^+b$

  • Compute pseudoinverse using SVD
    
    – Lets you see if data is singular ($< n$ nonzero singular values)
    
    – Even if not singular, condition number tells you how stable the solution will be
    
    – Set $1/w_i$ to 0 if $w_i$ is small (even if not exactly 0)
SVD and Matrix Similarity

- One common definition for the norm of a matrix is the Frobenius norm:
  \[ \|A\|_F = \sum_i \sum_j a_{ij}^2 \]

- Frobenius norm can be computed from SVD
  \[ \|A\|_F = \sum_i w_i^2 \]

- Euclidean (spectral) norm can also be computed:
  \[ \|A\|_2 = \{ \max|\lambda| : \lambda \in \sigma(A) \} \]

- So changes to a matrix can be evaluated by looking at changes to singular values
Suppose you want to find best rank-\(k\) approximation to \(A\)

Answer: set all but the largest \(k\) singular values to zero

Can form compact representation by eliminating columns of \(U\) and \(V\) corresponding to zeroed \(w_i\)
SVD and Eigenvectors

- Let \( \mathbf{A} = \mathbf{U} \mathbf{W} \mathbf{V}^T \), and let \( \mathbf{x}_i \) be the \( i \)th column of \( \mathbf{V} \).

- Consider \( \mathbf{A}^T \mathbf{A} \mathbf{x}_i \):

\[
\mathbf{A}^T \mathbf{A} \mathbf{x}_i = \mathbf{V} \mathbf{W}^T \mathbf{U}^T \mathbf{U} \mathbf{W} \mathbf{V}^T \mathbf{x}_i = \mathbf{V} \mathbf{W}^2 \mathbf{V}^T \mathbf{x}_i = \mathbf{V} \begin{pmatrix} 1 \\ \vdots \\ 0 \end{pmatrix} = \mathbf{V} \begin{pmatrix} 0 \\ \vdots \\ w_i^2 \end{pmatrix} = w_i^2 \mathbf{x}_i
\]

- So elements of \( \mathbf{W} \) are \( \sqrt{\text{eigenvalues}} \) and columns of \( \mathbf{V} \) are eigenvectors of \( \mathbf{A}^T \mathbf{A} \).
Total Least Squares

- One final least squares application
- Fitting a line: vertical vs. perpendicular error
Total Least Squares

- Distance from point to line:
  \[ d_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \cdot \vec{n} - a \]
  where \( n \) is normal vector to line, \( a \) is a constant

- Minimize:
  \[ \chi^2 = \sum_i d_i^2 = \sum_i \left[ \begin{pmatrix} x_i \\ y_i \end{pmatrix} \cdot \vec{n} - a \right]^2 \]
Total Least Squares

• First, let’s pretend we know n, solve for a

\[ \chi^2 = \sum_i \left[ \left( \begin{array}{c} x_i \\ y_i \end{array} \right) \cdot \vec{n} - a \right]^2 \]

\[ a = \frac{1}{m} \sum_i \left( \begin{array}{c} x_i \\ y_i \end{array} \right) \cdot \vec{n} \]

• Then

\[ d_i = \left( \begin{array}{c} x_i \\ y_i \end{array} \right) \cdot \vec{n} - a = \left( \begin{array}{c} x_i - \frac{\sum x_i}{m} \\ y_i - \frac{\sum y_i}{m} \end{array} \right) \cdot \vec{n} \]
Total Least Squares

- So, let’s define

\[
\begin{pmatrix}
\tilde{x}_i \\
\tilde{y}_i
\end{pmatrix}
= \begin{pmatrix}
x_i - \frac{\sum x_i}{m} \\
y_i - \frac{\sum y_i}{m}
\end{pmatrix}
\]

and minimize

\[
\sum_i \left[ \left( \tilde{x}_i \cdot \tilde{n} \right)^2 \right]
\]
Total Least Squares

- Write as linear system

\[
\begin{pmatrix}
\tilde{x}_1 & \tilde{y}_1 \\
\tilde{x}_2 & \tilde{y}_2 \\
\tilde{x}_3 & \tilde{y}_3 \\
\vdots & \vdots
\end{pmatrix}
\begin{pmatrix}
  n_x \\
  n_y
\end{pmatrix} = \vec{0}
\]

- Have An=0
  - Problem: lots of n are solutions, including n=0
  - Standard least squares will, in fact, return n=0
Constrained Optimization

- Solution: constrain $n$ to be unit length
- So, try to minimize $|An|^2$ subject to $|n|^2=1$
  \[ \|A\vec{n}\|^2 = (A\vec{n})^T (A\vec{n}) = \vec{n}^T A^T A \vec{n} \]
- Expand in eigenvectors $e_i$ of $A^T A$:
  \[ \vec{n} = \mu_1 e_1 + \mu_2 e_2 \]
  \[ \vec{n}^T (A^T A) \vec{n} = \lambda_1 \mu_1^2 + \lambda_2 \mu_2^2 \]
  \[ \|\vec{n}\|^2 = \mu_1^2 + \mu_2^2 \]
where the $\lambda_i$ are eigenvalues of $A^T A$
Constrained Optimization

• To minimize \( \lambda_1 \mu_1^2 + \lambda_2 \mu_2^2 \) subject to \( \mu_1^2 + \mu_2^2 = 1 \)
set \( \mu_{\text{min}} = 1 \), all other \( \mu_i = 0 \)

• That is, \( n \) is eigenvector of \( A^T A \) with the smallest corresponding eigenvalue
Comparison of Least Squares Methods

- **Normal equations** \((A^\top Ax = A^\top b)\)
  - \(O(mn^2)\) (using Cholesky)
  - \(\text{cond}(A^\top A) = [\text{cond}(A)]^2\)
  - Cholesky fails if 
    \(\text{cond}(A) \approx 1/\sqrt{\text{machine eps}}\)

- **Householder**
  - Usually best orthogonalization method
  - \(O(mn^2 - n^3/3)\) operations
  - Relative error is best possible for least squares
  - Breaks if \(\text{cond}(A) \sim 1/\text{machine eps}\)

- **SVD**
  - Expensive: \(mn^2 + n^3\) with bad constant factor
  - Can handle rank-deficiency, near-singularity
  - Handy for many different things
Matlab functions

- **qr**: explicit QR factorization
- **svd**
- **A\b**: (‘\’ operator)
  - Performs least-squares if A is m-by-n
  - Uses QR decomposition
- **pinv**: pseudoinverse
- **rank**: Uses SVD to compute rank of a matrix