QR Factorization and Singular Value Decomposition

COS 323

Today

- How do we solve least-squares…
  - without incurring condition-squaring effect of normal equations ($A^TAx = A^Tb$)
  - 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) >= 1, bigger is **bad**
- Measures how change in input propagates to output: $$\frac{\| \Delta x \|}{\| x \|} \le cond(A) \frac{\| \Delta A \|}{\| A \|}$$
- E.g., if cond(A) = 451 then can lose log(451)= 2.65 digits of accuracy in x, compared to precision of A

Normal Equations are Bad

$$\frac{\|\Delta x\|}{\|x\|} \leq cond(A)\frac{\|\Delta A\|}{\|A\|}$$

- Normal equations involves solving $A^TAx = A^Tb$
- $cond(A^TA) = [cond(A)]^2$
- E.g., if 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":

$$\begin{bmatrix} \# & \# & \cdots & \# \\ 0 & \# & & \# \\ 0 & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & \# \\ 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} x \cong \begin{bmatrix} \# \\ \# \\ \# \\ \# \\ \# \\ \# \\ \# \end{bmatrix}$$

$$\begin{bmatrix} R \\ O \end{bmatrix} x = b$$

- 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 $||Mb\text{-}MAx||_2$
  - Problem: $||Mv||_2 \mathrel{!=} ||v||_2$ (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 $||Mb\text{-}MAx||$ **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 R and **orthogonal** Q such that

$$A = Q\begin{bmatrix} R \\ O \end{bmatrix}, \text{ so } \begin{bmatrix} R \\ O \end{bmatrix} x = Q^T b$$

- Doesn't change least-squares solution
  - $Q^TQ=I$, columns of Q are orthonormal
  - i.e., Q preserves Euclidean norm: $||Qv||_2=||v||_2$

Goal of QR

$$\underbrace{A}_{m\times n} = \underbrace{Q}_{m\times m}\underbrace{\begin{bmatrix} R \\ O \end{bmatrix}}_{m\times n} = Q\begin{bmatrix} ? & ? & \cdots & ? \\ 0 & \ddots & & \vdots \\ \vdots & 0 & \ddots & \vdots \\ \vdots & & 0 & ? \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

R: $n\times n$, upper tri.

O: $(m\text{-}n)\times n$, all zeros

Reformulating Least Squares using QR

$$\|r\|_2^2 = \|b - Ax\|_2^2$$

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

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

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

$$= \|c_1 - Rx\|_2^2 + \|c_2\|_2^2$$

$$= \|c_2\|_2^2$$ **if we choose x such that $Rx=c_1$**

Householder Method for Computing QR Decomposition

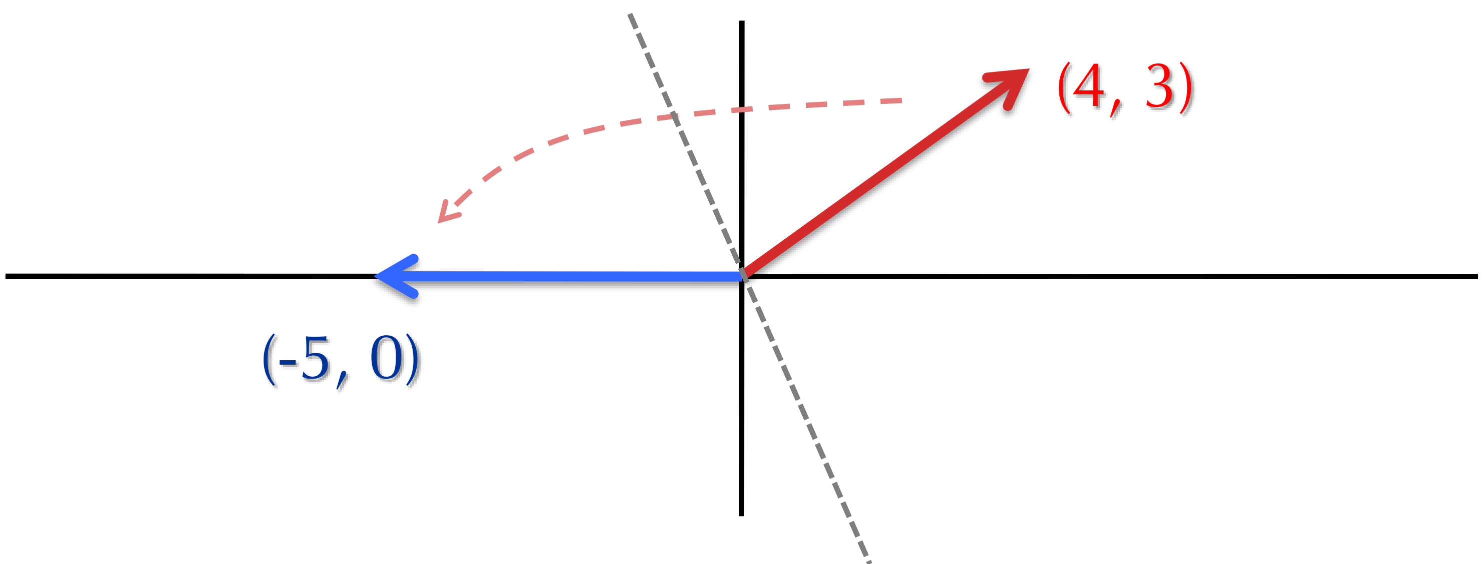
Orthogonalization for Factorization

$$A = Q\begin{bmatrix} R \\ O \end{bmatrix}$$

- 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 $H_1 * \ldots * H_n$, 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 $\text{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$ is the k^{th} column, with a_1 of height k-1

then let $v = \begin{bmatrix} 0 \\ a_2 \end{bmatrix} - \alpha e_k$ where $\alpha = -sign(a_k)\|a_2\|_2$

and construct $\mathbf{H} = \mathbf{I} - 2\dfrac{vv^{\mathrm{T}}}{v^{\mathrm{T}}v}$

Apply H to a and columns to the right:

$Ha = a - \left(2\dfrac{v^T a}{v^T v}\right)v$ (*with some shortcuts - see p.124)

Computing Householder – Example

Let $a = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}$, and find Householder transformation

that sets everything below first component to zero.

$$\text{Choose } v = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} - \alpha \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \text{ where } \alpha = -sign(2) \left\| \begin{matrix} 2 \\ 1 \\ 2 \end{matrix} \right\|_2$$

$$\text{so } v = \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix}, \text{ and } \mathbf{H}a = a - 2\frac{v^{\mathrm{T}}a}{v^{\mathrm{T}}v}v = \begin{bmatrix} -3 \\ 0 \\ 0 \end{bmatrix}$$

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 $\mathrm{A} = \mathrm{Q}\begin{bmatrix} R \\ O \end{bmatrix}$

Review: Least Squares using QR

$$\|r\|_2^2 = \|b - Ax\|_2^2$$

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

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

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

$$= \|c_1 - Rx\|_2^2 + \|c_2\|_2^2$$

$$= \|c_2\|_2^2$$ **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:

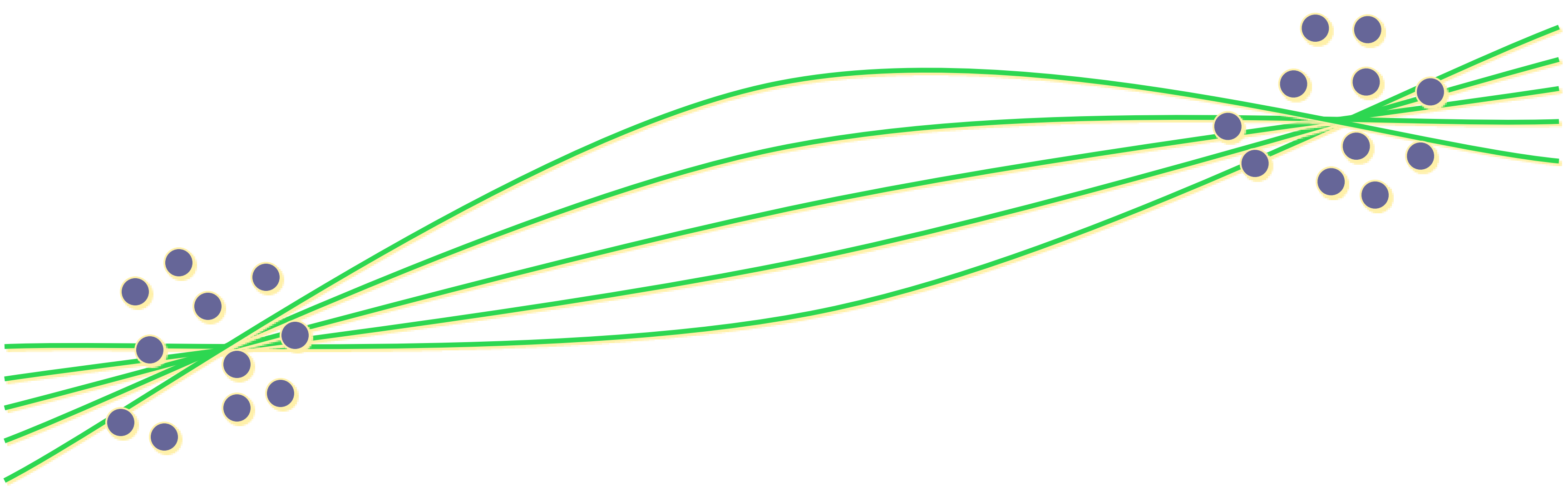
$$\begin{bmatrix} \# & 0 & 0 & 0 \\ 0 & \# & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \# \\ 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} x \cong \begin{bmatrix} \# \\ \# \\ \# \\ \# \\ \# \\ \# \\ \# \end{bmatrix}$$

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 $C = (A^TA)^{-1}$
  - In other words, unstable if A^TA has small values
  - More precisely, care if $x^T(A^TA)x$ 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 $\mathbf{A}$, algorithm to find matrices $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ such that

$$\mathbf{A} = \mathbf{U}\mathbf{W}\mathbf{V}^T$$

$\mathbf{U}$ is $m \times n$ and **orthonormal**

$\mathbf{W}$ is $n \times n$ and **diagonal**

$\mathbf{V}$ is $n \times n$ and **orthonormal**

SVD

$$\begin{pmatrix} \\ \\ & \mathbf{A} & \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \\ & \mathbf{U} & \\ \\ \\ \end{pmatrix} \begin{pmatrix} w_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_n \end{pmatrix} \begin{pmatrix} \\ & \mathbf{V} & \\ \\ \end{pmatrix}^{\mathrm{T}}$$

- Based on Householder reduction, QR decomposition, but treat as black box: code widely available
  e.g., in Matlab: `[U,W,V]=svd(A,0)`

SVD

- The w_i are called the singular values of **A**
- If **A** is singular, some of the w_i will be 0
- In general $rank(\mathbf{A})$ = 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
- $\mathbf{A}^{-1} = (\mathbf{V}^T)^{-1}\,\mathbf{W}^{-1}\,\mathbf{U}^{-1} = \mathbf{V}\,\mathbf{W}^{-1}\,\mathbf{U}^T$
  - Using fact that inverse = transpose for orthogonal matrices
  - Since $\mathbf{W}$ is diagonal, $\mathbf{W}^{-1}$ also diagonal with reciprocals of entries of $\mathbf{W}$

SVD and the Pseudoinverse

- $\mathbf{A}^{-1} = (\mathbf{V}^T)^{-1} \mathbf{W}^{-1} \mathbf{U}^{-1} = \mathbf{V} \mathbf{W}^{-1} \mathbf{U}^T$
- 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 $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ if $\mathbf{A}^T\mathbf{A}$ invertible

SVD and Condition Number

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

$$\mathrm{cond}(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$$

SVD and Least Squares

- Solving $\mathbf{Ax}=\mathbf{b}$ 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: $$\|\mathbf{A}\|_{\mathrm{F}} = \sum_i \sum_j a_{ij}^{\ 2}$$

- Frobenius norm can be computed from SVD
$$\|\mathbf{A}\|_{\mathrm{F}} = \sum_i w_i^{\ 2}$$
- Euclidean (spectral) norm can also be computed:
$$\|\mathbf{A}\|_2 = \{\max|\lambda| : \lambda \in \sigma(\mathbf{A})\}$$
- So changes to a matrix can be evaluated by looking at changes to singular values

SVD and Matrix Similarity

- 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

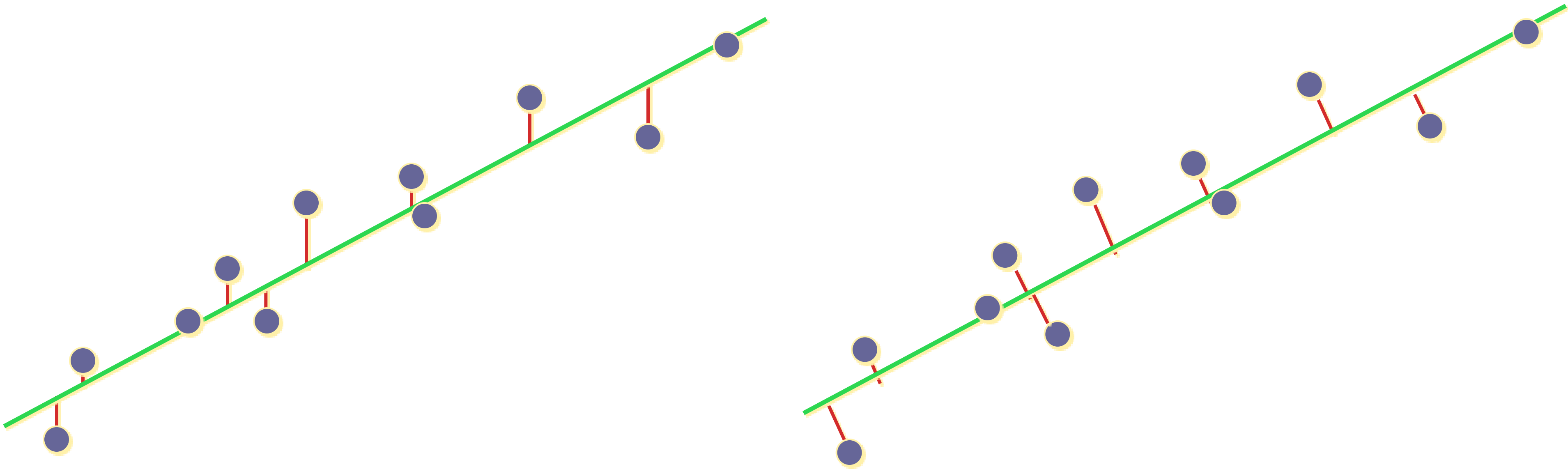
- Let $\mathbf{A}=\mathbf{UWV}^{\mathrm{T}}$, and let x_i be i^{th} column of $\mathbf{V}$
- Consider $\mathbf{A}^{\mathrm{T}}\mathbf{A}\, x_i$:

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}x_i = \mathbf{VW}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\mathbf{UWV}^{\mathrm{T}}x_i = \mathbf{VW}^2\mathbf{V}^{\mathrm{T}}x_i = \mathbf{VW}^2\begin{pmatrix}0\\ \vdots\\ 1\\ \vdots\\ 0\end{pmatrix} = \mathbf{V}\begin{pmatrix}0\\ \vdots\\ w_i^{\,2}\\ \vdots\\ 0\end{pmatrix} = w_i^{\,2}x_i$$

- So elements of $\mathbf{W}$ are sqrt(eigenvalues) and columns of $\mathbf{V}$ are eigenvectors of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$

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

$$\chi^2 = \sum_i \left[\begin{pmatrix} x_i \\ y_i \end{pmatrix} \cdot \vec{n} - a \right]^2$$

$$a = \frac{1}{m} \sum_i \begin{pmatrix} x_i \\ y_i \end{pmatrix} \cdot \vec{n}$$

- Then

$$d_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \cdot \vec{n} - a = \begin{pmatrix} x_i - \frac{\Sigma x_i}{m} \\ y_i - \frac{\Sigma y_i}{m} \end{pmatrix} \cdot \vec{n}$$

Total Least Squares

- So, let's define

$$\begin{pmatrix} \tilde{x}_i \\ \tilde{y}_i \end{pmatrix} = \begin{pmatrix} x_i - \frac{\Sigma x_i}{m} \\ y_i - \frac{\Sigma y_i}{m} \end{pmatrix}$$

  and minimize

$$\sum_i \left[\begin{pmatrix} \tilde{x}_i \\ \tilde{y}_i \end{pmatrix} \cdot \vec{n} \right]^2$$

Total Least Squares

- Write as linear system

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

- 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 $|n|^2=1$

$$\left\|\mathbf{A}\vec{n}\right\|^2 = \left(\mathbf{A}\vec{n}\right)^{\mathrm{T}}\left(\mathbf{A}\vec{n}\right) = \vec{n}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\,\vec{n}$$

- Expand in eigenvectors e_i of A^TA:

$$\vec{n} = \mu_1\mathbf{e}_1 + \mu_2\mathbf{e}_2$$

$$\vec{n}^{\mathrm{T}}\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)\vec{n} = \lambda_1\mu_1^2 + \lambda_2\mu_2^2$$

$$\left\|\vec{n}\right\|^2 = \mu_1^2 + \mu_2^2$$

where the λ_i are eigenvalues of A^TA

Constrained Optimization

- To minimize $\lambda_1\mu_1^2 + \lambda_2\mu_2^2$ subject to $\mu_1^2 + \mu_2^2 = 1$ set $\mu_{\mathrm{min}} = 1$, all other $\mu_{\mathrm{i}} = 0$
- That is, n is eigenvector of A^TA with the smallest corresponding eigenvalue

Comparison of Least Squares Methods

- **Normal equations** ($A^TAx = A^Tb$)
  - $O(mn^2)$ (using Cholesky)
  - $\text{cond}(A^TA)=[\text{cond}(A)]^2$
  - Cholesky fails if $\text{cond}(A) \sim 1/\text{sqrt}(\text{machine epsilon})$
- **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