Singular Value Decomposition

COS 323
Underconstrained Least Squares

• What if you have fewer data points than parameters in your function?
  – Intuitively, can’t do standard least squares
  – Recall that solution takes the form $A^TAx = A^Tb$
  – When $A$ has more columns than rows, $A^TA$ is singular: can’t take its inverse, etc.
Underconstrained Least Squares

- More subtle version: more data points than unknowns, but data poorly constrains 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 $A$, 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

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

- Treat as black box: code widely available

In Matlab: \([U, W, V] = \text{svd}(A, 0)\)
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 $\text{rank}(A) = \text{number of nonzero } w_i$
• 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 Inverses

- \( A^{-1} = (V^T)^{-1} W^{-1} U^{-1} = VW^{-1} U^T \)
- This fails when some \( w_i \) are 0
  - It’s supposed to fail – singular matrix
- 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 Least Squares

- Solving $Ax = b$ by least squares
- $x = \text{pseudoinverse}(A)$ times $b$
- Compute pseudoinverse using SVD
  - Lets you see if data is singular
  - Even if not singular, ratio of max to min singular values $(\lambda = \text{condition number})$ 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 Eigenvectors

- Let \( A = UVW^T \), and let \( x_i \) be \( i^{th} \) column of \( V \)
- Consider \( A^TA x_i \):

\[
A^TA x_i = VW^T U^T UWV^T x_i = VW^2 V^T x_i = VW^2 \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = V \begin{pmatrix} 0 \\ \vdots \\ w_i^2 \\ \vdots \\ 0 \end{pmatrix} = w_i^2 x_i
\]

- So elements of \( W \) are \( \sqrt{\text{eigenvalues}} \) and columns of \( V \) are eigenvectors of \( A^TA \)
  - What we wanted for robust least squares fitting!
SVD and Matrix Similarity

• One common definition for the norm of a matrix is the Frobenius norm:

\[ \|A\|_F = \sum \sum a_{ij}^2 \]  

• Frobenius norm can be computed from SVD

\[ \|A\|_F = \sum w_i^2 \]  

• 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 Matrix Similarity
SVD and PCA

• Principal Components Analysis (PCA): approximating a high-dimensional data set with a lower-dimensional subspace
SVD and PCA

• Data matrix with points as rows, take SVD
  – Subtract out mean (“whitening”)

• Columns of $V_k$ are principal components

• Value of $w_i$ gives importance of each component
PCA on Faces: “Eigenfaces”

For all except average,
“gray” = 0,
“white” > 0,
“black” < 0
Using PCA for Recognition

• Store each person as coefficients of projection onto first few principal components

\[ \text{image} = \sum_{i=0}^{i_{\text{max}}} a_i \text{Eigenface}_i \]

• Compute projections of target image, compare to database (“nearest neighbor classifier”)
Total Least Squares

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

• Distance from point to line:

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

where \( n \) is normal vector to line, \( a \) is a constant

• Minimize:

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

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

\[
\chi^2 = \sum_{i} \left[ \begin{pmatrix} x_i \\ y_i \end{pmatrix} \cdot \bar{n} - a \right]^2
\]

\[
a = \frac{1}{m} \sum_{i} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \cdot \bar{n}
\]

• Then

\[
d_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \cdot \bar{n} - a = \begin{pmatrix} x_i - \frac{\Sigma x_i}{m} \\ y_i - \frac{\Sigma y_i}{m} \end{pmatrix} \cdot \bar{n}
\]
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

\[
\sum_i \left[ \begin{pmatrix}
\tilde{x}_i \\
\tilde{y}_i
\end{pmatrix} \cdot \bar{n}
\end{pmatrix}^2
\]
Total Least Squares

• Write as linear system

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

• 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 $A^TA$ with the smallest corresponding eigenvalue