Singular Value Decomposition

Szymon Rusinkiewicz

COS 302, Fall 2020

PRINCETON UNIVERSITY

Singular Value Decomposition (SVD)

- Matrix *decomposition* that reveals structure
- Useful for:
  - Inverses, pseudoinverses
  - Stable least-squares, even for unconstrained problems
  - Matrix similarity and approximation
  - Dimensionality reduction and PCA
  - Orthogonalization
  - Constrained least squares and multidimensional scaling

Let's look at motivaion for these

Condition Number

- cond(***A***) is function of ***A***
- cond(***A***) >= 1, bigger is bad
- Measures how change in input propagates to output:

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

  – E.g., if cond(***A***) = 451 then can lose log(451)= 2.65 digits of accuracy in x, compared to precision of ***A***
- For matrices with real eigenvalues, cond(***A***) = $|\lambda_{max}| \,/\, |\lambda_{min}|$

Normal Equations are Bad

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

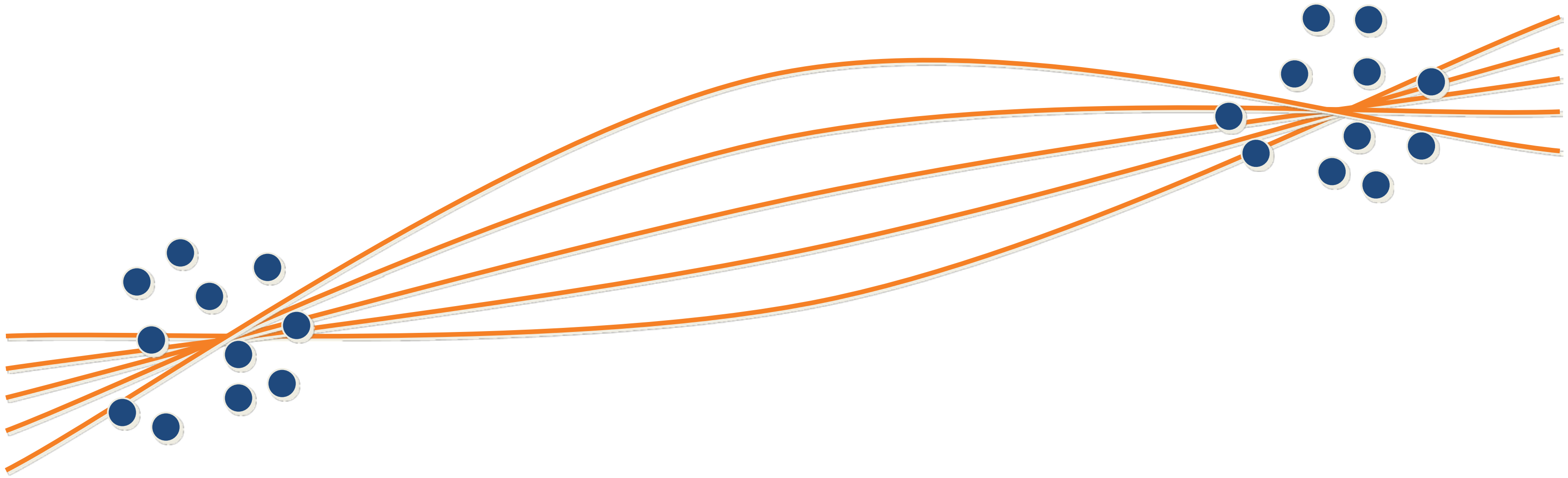
- Least squares using normal equations involves solving $\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{b}$
- $\mathrm{cond}(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}) = [\mathrm{cond}(\boldsymbol{A})]^2$
- E.g., if $\mathrm{cond}(\boldsymbol{A}) = 451$ then can lose $\log(451^2) = 5.3$ digits of accuracy, compared to precision of $\boldsymbol{A}$

Underconstrained Least Squares

- What if you have fewer data points than parameters in your function?
  - Intuitively, can't do standard least squares
  - Solution takes the form $\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{b}$
  - When $\boldsymbol{A}$ has more columns than rows, $\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}$ 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 $\boldsymbol{C} = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}$
  - In other words, unstable if $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ has small values
  - More precisely, care if $\boldsymbol{x}^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})\boldsymbol{x}$ is small for any $\boldsymbol{x}$
- 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 $\boldsymbol{A}$, algorithm to find matrices $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{W}$ with:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^{\mathrm{T}}$$

$\boldsymbol{U}$ is $m \times m$ and **orthonormal**

$\boldsymbol{W}$ is $m \times n$ and **zero except main diagonal**

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

- Won't derive algorithm – treat as black box (e.g., **`numpy.linalg.svd`**)

"Full" SVD

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

$m{\times}n \qquad m{\times}m \qquad m{\times}n \qquad n{\times}n$

```
u,w,vt = numpy.linalg.svd(a)
```

SVD

- Handwavy explanation: rotate to a basis where all the scaling and stretching of $\boldsymbol{A}$ is along coordinate axes
  - Should remind you of eigendecomposition (which would have $\boldsymbol{U} = \boldsymbol{V}$)
- The w_i are called the singular values of $\boldsymbol{A}$
- If $\boldsymbol{A}$ is singular, some of the w_i will be 0
- In general $rank(\boldsymbol{A}) = $ number of nonzero w_i
- SVD is mostly unique (up to permutation of singular values, or if some w_i are equal)
  - The w_i are conventionally returned in sorted order, largest to smallest

Singular Value Decomposition (SVD)

- If $m > n$, only n nonzero rows in $\boldsymbol{W}$, many useless columns in $\boldsymbol{U}$
- If $n > m$, only m nonzero columns in $\boldsymbol{W}$, many useless columns in $\boldsymbol{V}$
  - Define "compact" or "reduced" versions that omit all those zeroes

"Compact" SVD, if $m > n$

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

$m \times n$ $\qquad$ $m \times n$ $\qquad$ $n \times n$ $\qquad$ $n \times n$

```
u,w,vt = numpy.linalg.svd(a, full_matrices=False)
```

"Compact" SVD, if $n > m$

$$\Bigg[\quad \boldsymbol{A} \quad\Bigg] = \Bigg[\quad \boldsymbol{U} \quad\Bigg] \begin{bmatrix} w_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_m \end{bmatrix} \Bigg[\quad \boldsymbol{V} \quad\Bigg]^{\mathrm{T}}$$

$m \times n$ $\quad$ $m \times m$ $\quad$ $m \times m$ $\quad$ $m \times n$

```
u,w,vt = numpy.linalg.svd(a, full_matrices=False)
```

SVD and Inverses

- Why is SVD so useful?
- Application #1: inverses
- $\boldsymbol{A}^{-1} = (\boldsymbol{V}^{\mathrm{T}})^{-1}\,\boldsymbol{W}^{-1}\,\boldsymbol{U}^{-1} = \boldsymbol{V}\boldsymbol{W}^{-1}\,\boldsymbol{U}^{\mathrm{T}}$
  - Using fact that inverse = transpose for orthogonal matrices
  - Since $\boldsymbol{W}$ is diagonal, $\boldsymbol{W}^{-1}$ also diagonal with reciprocals of entries of $\boldsymbol{W}$

SVD and the Pseudoinverse

- $\boldsymbol{A}^{-1} = (\boldsymbol{V}^{\mathrm{T}})^{-1}\, \boldsymbol{W}^{-1}\, \boldsymbol{U}^{-1} \;=\; \boldsymbol{V}\boldsymbol{W}^{-1}\, \boldsymbol{U}^{\mathrm{T}}$
- This fails when some w_i are 0
  - It's *supposed* to fail – singular matrix
  - Happens when rectangular $\boldsymbol{A}$ is **rank deficient**
- Pseudoinverse $\boldsymbol{A}^{+}$: 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 $(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}$ if $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ invertible

SVD and Least Squares

- Solving $\boldsymbol{Ax}=\boldsymbol{b}$ by least squares:
- $\boldsymbol{A}^\mathrm{T}\boldsymbol{Ax} = \boldsymbol{A}^\mathrm{T}\boldsymbol{b} \ \rightarrow\ \mathbf{x} = (\boldsymbol{A}^\mathrm{T}\boldsymbol{A})^{-1}\boldsymbol{A}^\mathrm{T}\boldsymbol{b}$
- Replace with $\boldsymbol{A}^+$: $\boldsymbol{x} = \boldsymbol{A}^+\boldsymbol{b}$
  - Compute pseudoinverse using SVD
- Lets you see if data is singular ($< n$ nonzero singular values)
- Singular values tell you how stable the solution will be
  - Condition number = ratio of largest to smallest singular values
- For better stability, set $1/w_i$ to 0 if w_i is small (even if not exactly 0)
  - Accuracy / stability tradeoff? Not if that component was underconstrained…

SVD and Matrix Similarity

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

$$\|\mathbf{A}\|_F = \sum_i \sum_j a_{ij}{}^2$$

- Frobenius norm can be computed from SVD

$$\|\mathbf{A}\|_F = \sum_i w_i{}^2$$

- 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

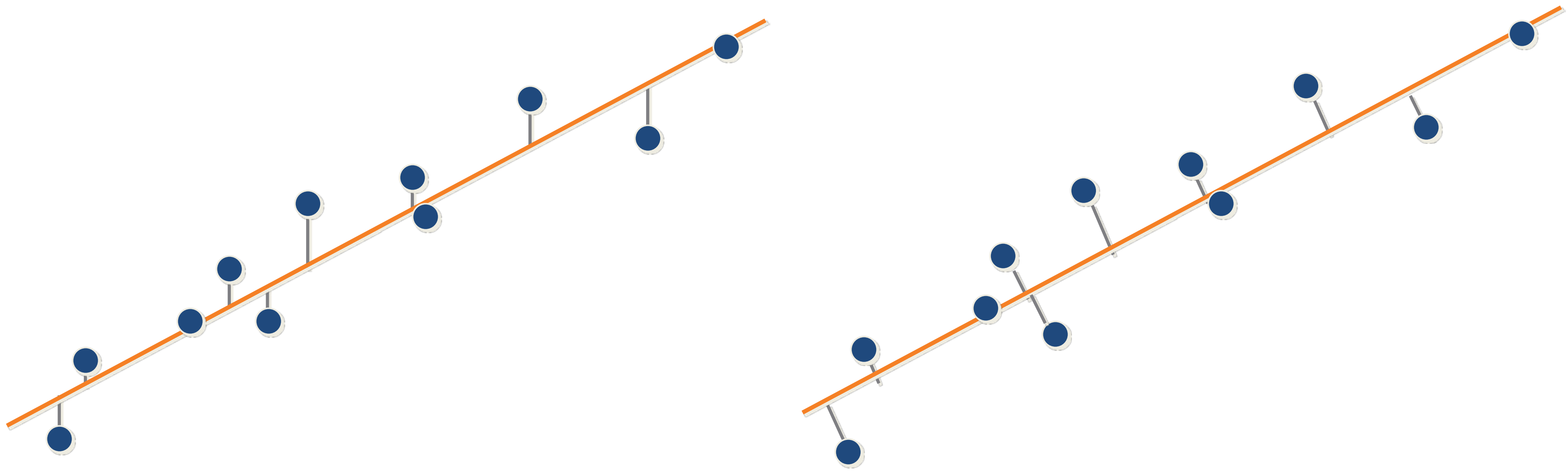
- Suppose you want to find best rank-k approximation to $\boldsymbol{A}$
- Answer: set all but the largest k singular values to zero
- Can form compact representation by eliminating columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ corresponding to zeroed w_i

SVD and Orthogonalization

- ***U*** and ***V*** are orthonormal, all stretching and scaling in ***W***
- The matrix $\boldsymbol{UV}^\mathrm{T}$ is the "closest" orthonormal matrix to ***A***
  - Yet another useful application of the matrix-approximation properties of SVD
  - Much more stable numerically than Graham-Schmidt orthogonalization

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 $\boldsymbol{An} = \mathbf{0}$
  - Problem: lots of n are solutions, including $\boldsymbol{n} = \mathbf{0}$
  - Standard least squares will, in fact, return $\boldsymbol{n} = \mathbf{0}$

Constrained Optimization

- Solution: constrain n to be unit length
- So, try to minimize $\|\boldsymbol{An}\|^2$ subject to $\|\boldsymbol{n}\|^2=1$

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

- Expand in eigenvectors $\mathbf{e}_i$ of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$:

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

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

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

where the λ_i are eigenvalues of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$

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, $\boldsymbol{n}$ is eigenvector of $\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}$ with the smallest corresponding eigenvalue

SVD and Eigenvectors

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

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

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

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, $\boldsymbol{n}$ is eigenvector of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ with
  the smallest corresponding eigenvalue

- That is, $\boldsymbol{n}$ is column of $\boldsymbol{V}$ corresponding to smallest singular value
  - Provides a solution to the total least squares problem
  - Also very related to PCA – next time