Linear Algebra and Least Squares Refresher

Arnold Kalmbach, Yewon Lee
CSC477 Tutorial #2
Sept 28, 2022

Outline

- Notation reminder, and a couple useful facts
- Intro to linear least squares (with an example)
- Singular Value Decomposition
    - Relationship to eigendecomposition

Notation Reminder - Inverse

$$AX = I \iff X = A^{-1}$$

- If $\mathbf{A}^{-1}$ exists, **A** is *nonsingular*.
- All the following are equivalent
  - **A** is nonsingular
  - det(**A**) != **0**
  - rank(**A**) = n
  - **Ax** = **0** has a unique solution **x** = **0**

Notation Reminder - Orthogonal Matrix

$$Q^T Q = Q Q^T = I$$

- Equivalently, if all the rows & columns are orthonormal ie

$$Q = [q_1, q_2, \ldots, q_n]$$

$$q_i^T q_j = \begin{Bmatrix} 1 & i = j \\ 0 & i \neq j \end{Bmatrix}$$

Notation Reminder - Vector Norms $||v||$

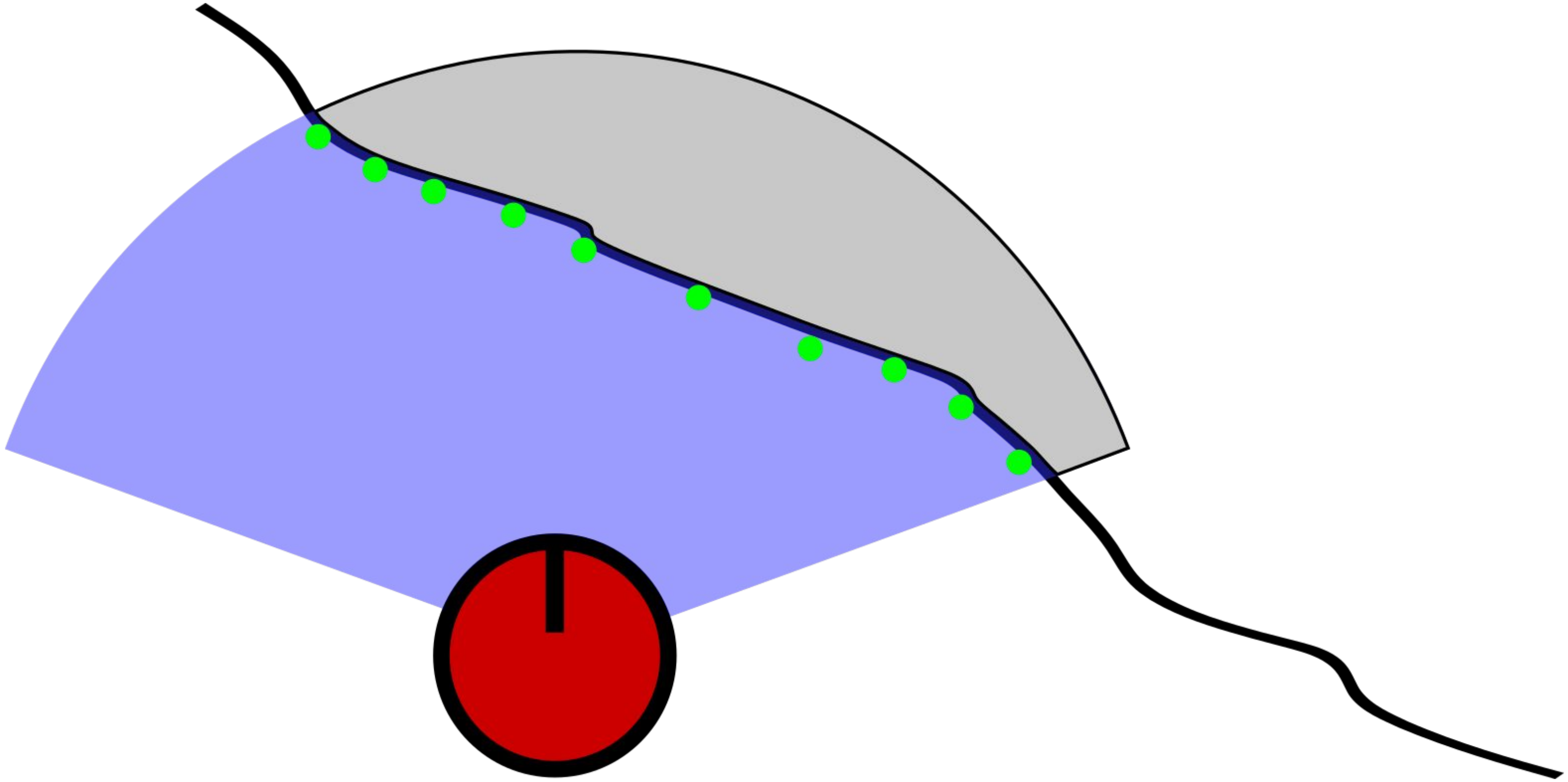
- P-Norms

$$||v||_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p}$$

- 1-Norm = Sum of elements, 2-Norm = Euclidean Distance, Infinity-Norm = Largest Element

Linear Least Squares by Example

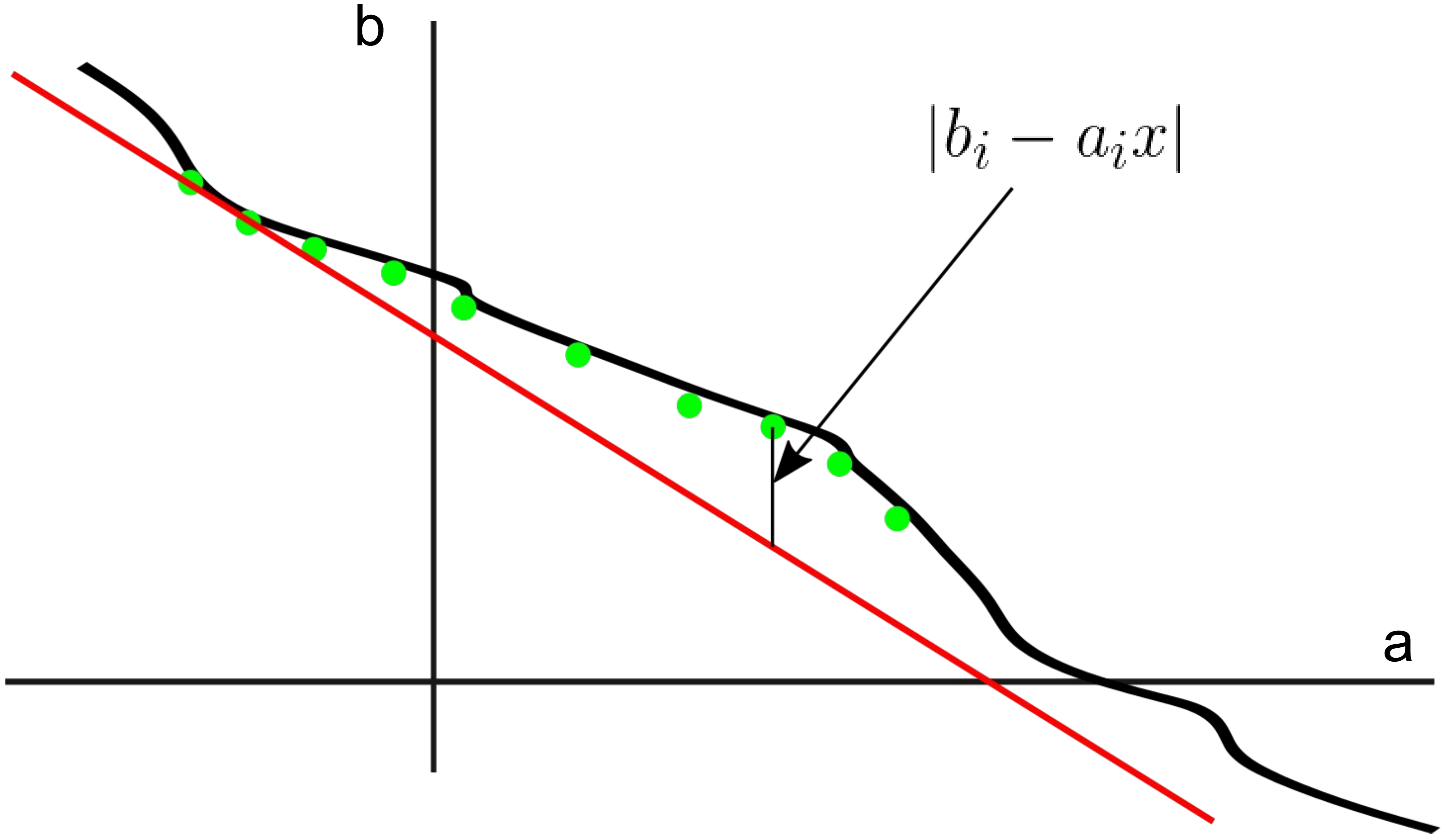
How to Estimate the Location of the Wall?



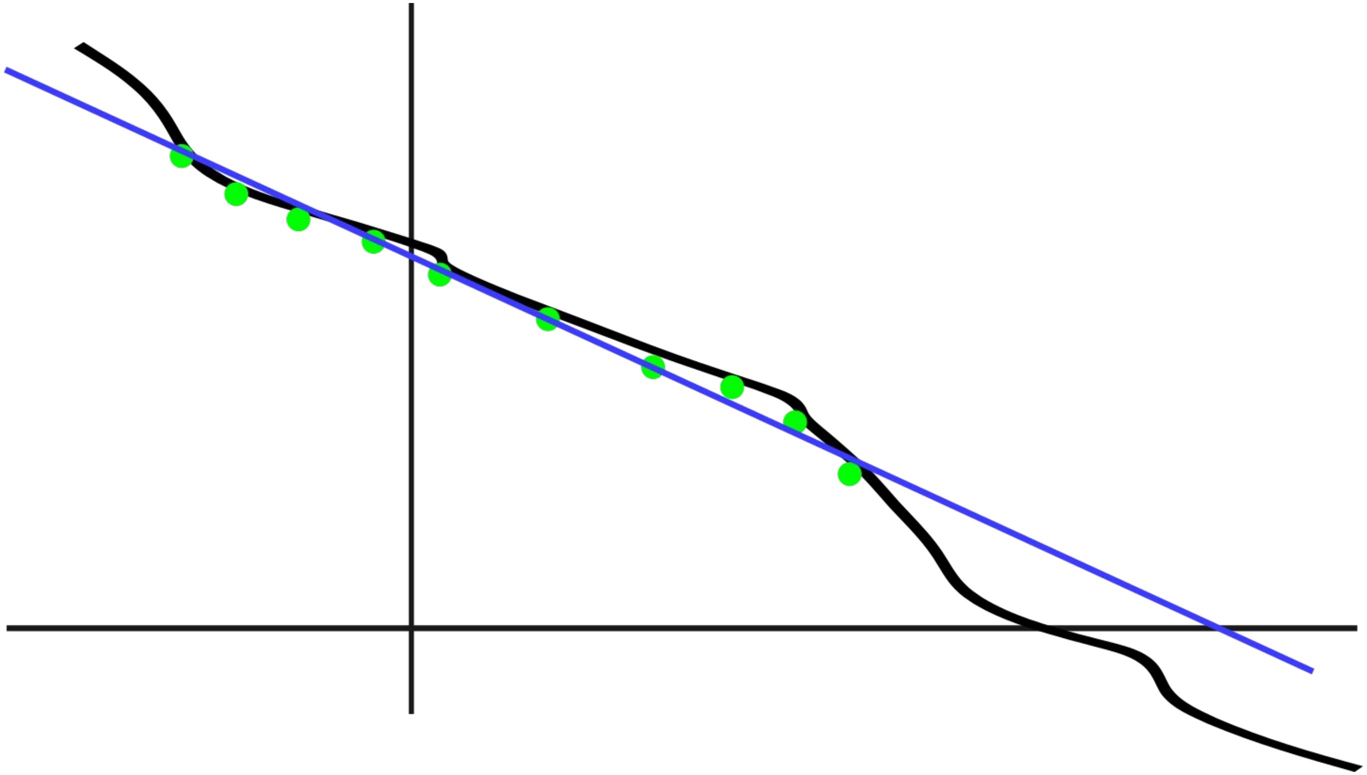
Idea: Fit a line to the sample points.

This is an **over-determined system**

Least Squares: Minimize Sum of Squared Error

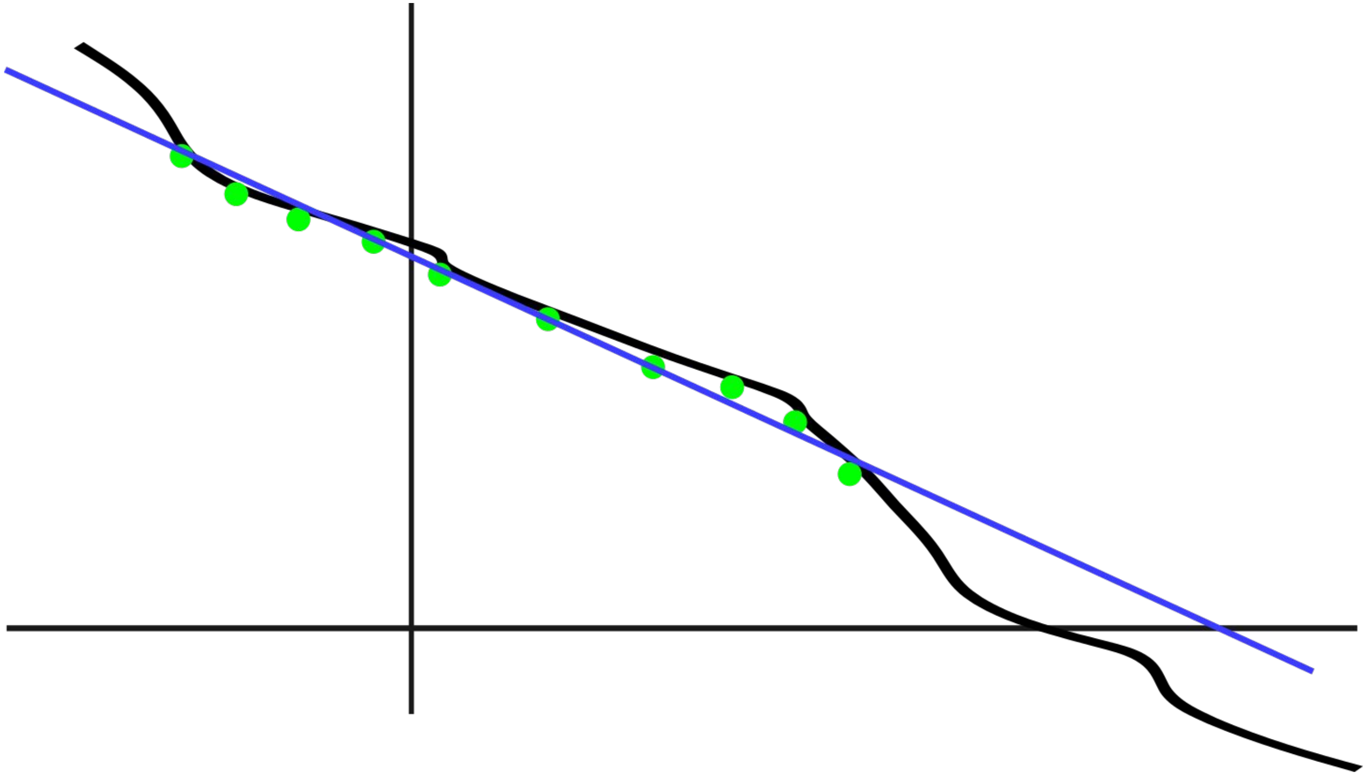


Least Squares: Minimize Sum of Squared Error



$$\hat{x} = \arg\min_{x} \left(\sum_{i=1}^{n} (b_i - a_i x)^2 \right)^{1/2}$$

Least Squares: Minimize Sum of Squared Error



$$\hat{x} = \arg\min_{x} \|b - Ax\|_2^2$$

Least Squares Criterion

Very general formulation: $\hat{x} = \arg\min_{x} \|b - h_x(A)\|_2^2$

Most common options (not covered much in this course):

1. Take gradient of $\overset{f(x)}{h(A)}$ wrt. x
2. Approximate h with a linear function

Can use Linear or nonlinear least squares to set up all kinds of modelling problems as optimization problems.

Least Squares: Solution

$$\nabla f(\hat{x}) = 2A^T(A\hat{x} - b) = 0$$

$$(A^T A)\hat{x} = A^T b$$

$$\hat{x} = \left(A^T A\right)^{-1} A^T b = A^\dagger b$$

If $b = Ax^\star + v, \quad v \sim \mathcal{N}(0, \sigma I)$ this produces the ***optimal*** estimator (BLUE)

To calculate with numpy:

- `numpy.linalg.pinv(A)`
- `x_hat = numpy.linalg.lstsq(A, b)`

Issues Computing the Solution

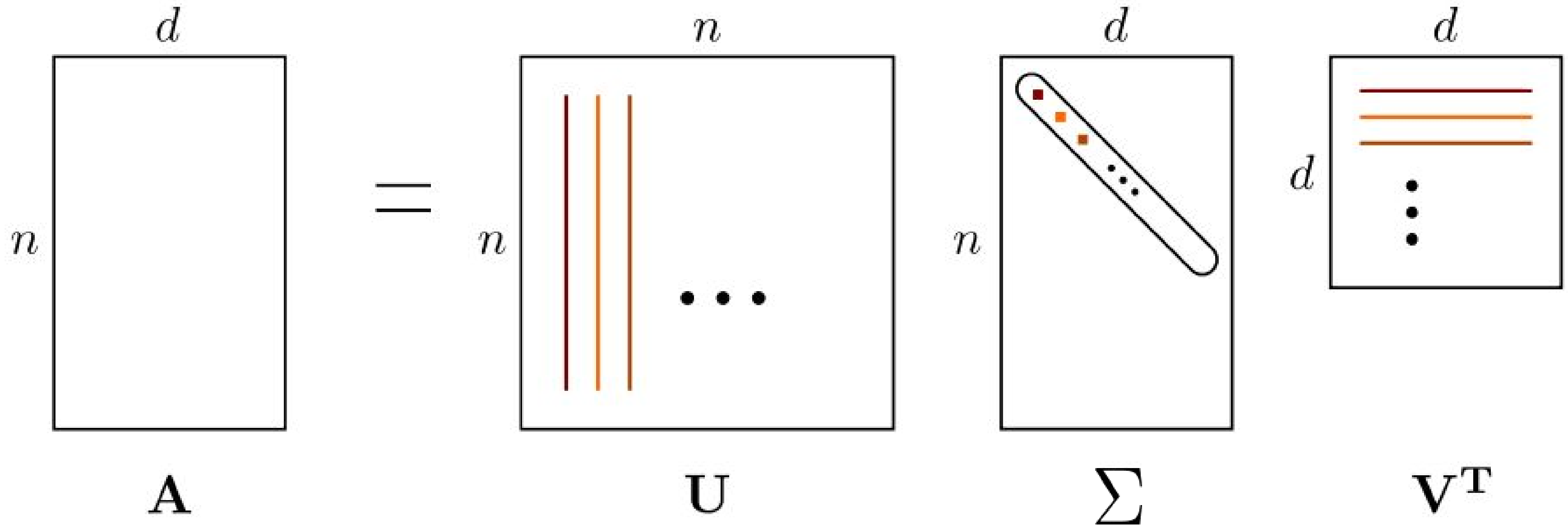
$$\hat{x} = \left(A^T A\right)^{-1} A^T b = A^{\dagger} b$$

Computing $(A^T A)^{-1}$ by Gaussian Elimination is numerically unstable and slow!

We can do better if we decompose A

- $A = LL^T$ (Cholesky Factorization, **skip**)
- $A = [Q_1 \ Q_2] \begin{bmatrix} R \\ 0 \end{bmatrix}$ (QR Factorization, **skip**)
- $A = U \Sigma V^T$ (**Singular Value Decomposition)**

Singular Value Decomposition (SVD)



Singular Value Decomposition (SVD)

$$A = U\Sigma V^T$$

Orthogonal	Diagonal (Squared eigenvalues)	Orthogonal

$\sum^{\dagger}$ = Reciprocal of each diagonal entry, transpose

Very useful fact:

$$A^T A = V\Sigma^{\dagger} U^T U \Sigma V^T = V(\Sigma^{\dagger}\Sigma)V^T$$

SVD Example

$$\overset{\text{nxd}}{A} = \overset{\text{nxn}}{U}\overset{\text{nxd}}{\Sigma}\overset{\text{dxd}}{V^T}$$

Σ has diagonal elements

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k > \sigma_{k+1} = \sigma_{k+2} = \ldots = 0$$

Where σ_i are the square root of the eigenvalues of $A^T A$ and $k = \text{rank}(A)$

$$A = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\det(A^T A - \lambda I) = 0 \Rightarrow \lambda_1 = 1,\ \text{rank}(A) = 1$$

$$\Sigma = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

SVD Example

$$\overset{\text{nxd}}{A} = \overset{\text{nxn}}{U}\,\overset{\text{nxd}}{\Sigma}\,\overset{\text{dxd}}{V^T}$$

$$V = [\underbrace{v_1 \; v_2 \; \dots \; v_k}_{\substack{\text{Normalized} \\ \text{eigenvectors of} \\ A^TA}} \; \underbrace{v_{k+1} \; \dots \; v_d}_{\substack{\text{Obtained from } A^TAv_j = 0 \\ \text{such that orthogonality of V is} \\ \text{satisfied}}}]$$

$$(A^TA - \lambda_1 I)v_1 = 0, \; \lambda_1 = 1$$

$$\begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} v_1 = 0$$

$$v_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$A^TAv_2 = 0$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} v_2 = 0$$

$$v_1 \cdot v_2 = 0$$

$$v_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\Rightarrow V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

SVD Example

$$\overset{\text{nxd}}{A} = \overset{\text{nxn}}{U}\,\overset{\text{nxd}}{\Sigma}\,\overset{\text{dxd}}{V^T}$$

$$U = [\underbrace{u_1\ u_2\ \dots\ u_k}_{\substack{\text{Normalized}\\ \text{eigenvectors of}\\ AA^T}}\ \underbrace{u_{k+1}\ \dots\ u_n}_{\substack{\text{Obtained from}\\ AA^T u_j = 0}}]$$

$$AA^T = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 1$$

$$\det(AA^T - \lambda I) = 0 \Rightarrow \lambda_1 = 1$$

$$AA^T u_1 = \lambda_1 u_1 \iff u_1 = u_1$$

$$u_1 = 1$$

$$U = 1$$

SVD and Solving Least Squares

$$A = U\Sigma V^T$$

If we can compute $A^\dagger = (A^T A)^{-1} A^T$ stably, we can solve LS problems.

Recall: $A^T A = V \left(\Sigma^\dagger \Sigma\right) V^T$

So $A^\dagger = V\Sigma^\dagger U^T$

To calculate with numpy

- `U,S,V = numpy.linalg.svd(A)`