Linear Algebra and Least Squares Refresher

Arnold Kalmbach

Comp 417 Tutorial #2

Jan. 20, 2017

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

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

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

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

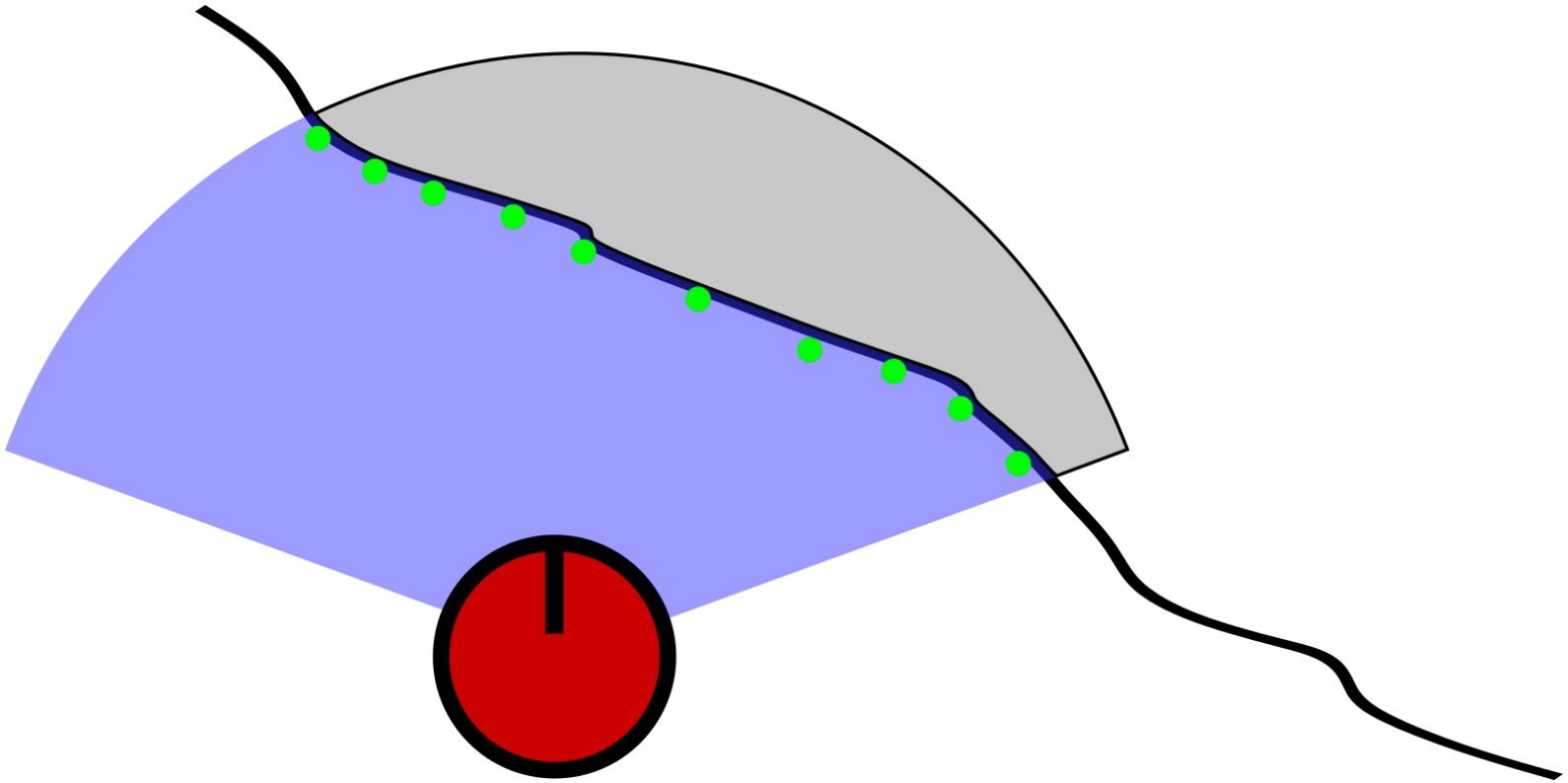
- P-Norms

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

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

Linear Least Squares by Example

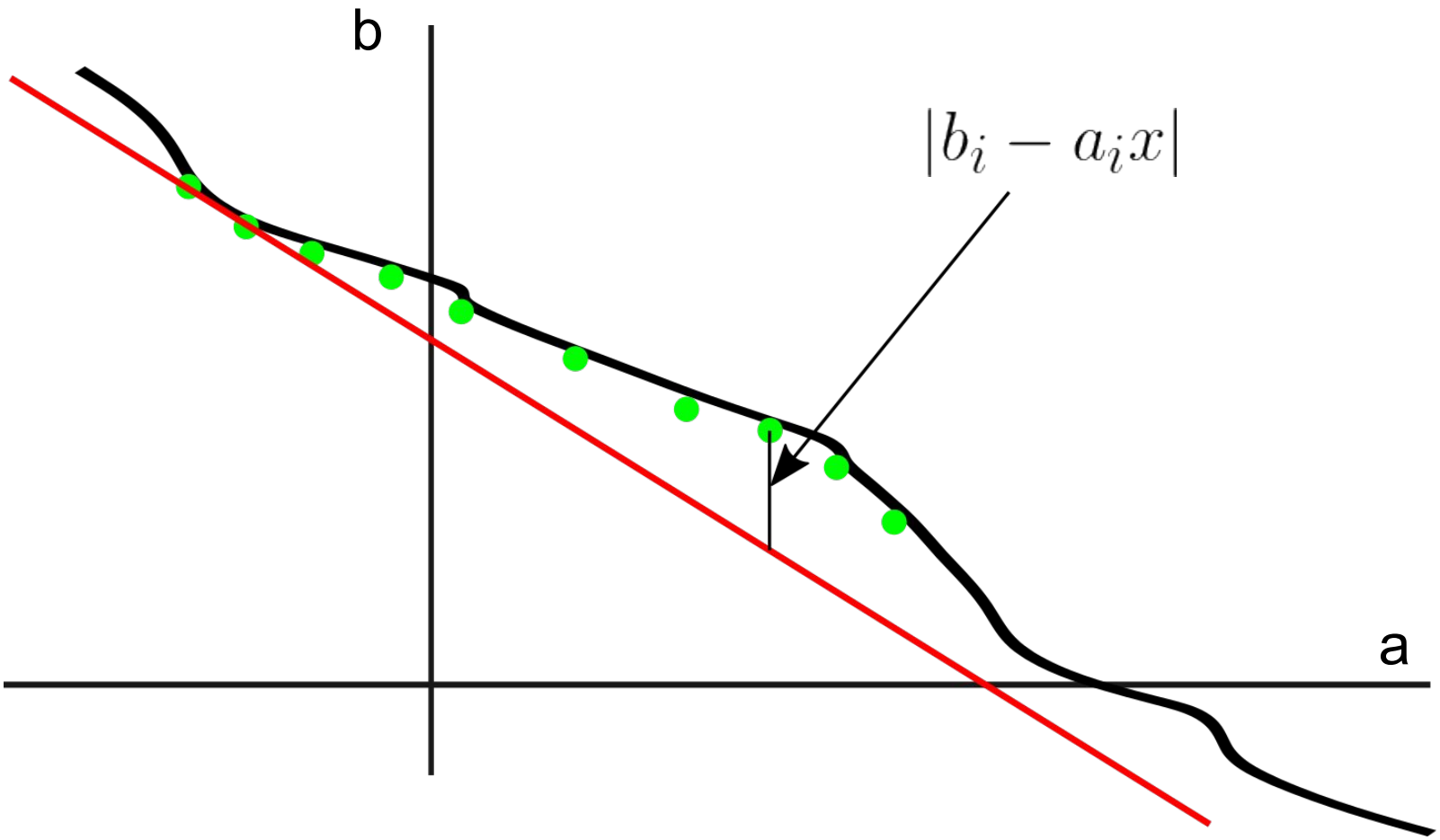
How to Estimate the Location of the Wall?



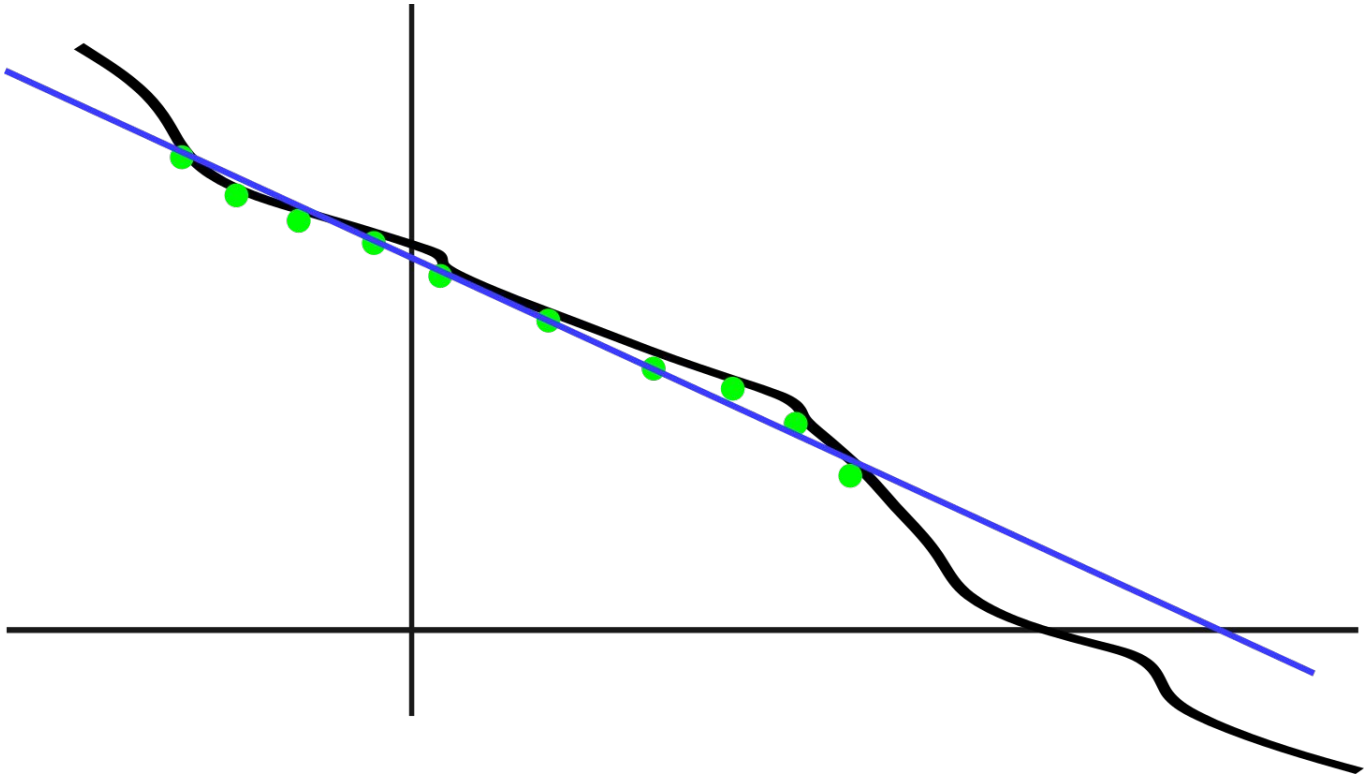
Idea: Fit a line to the sample points.

This is an **over-determined system**

Least Squares: Minimize Sum of Squared Error

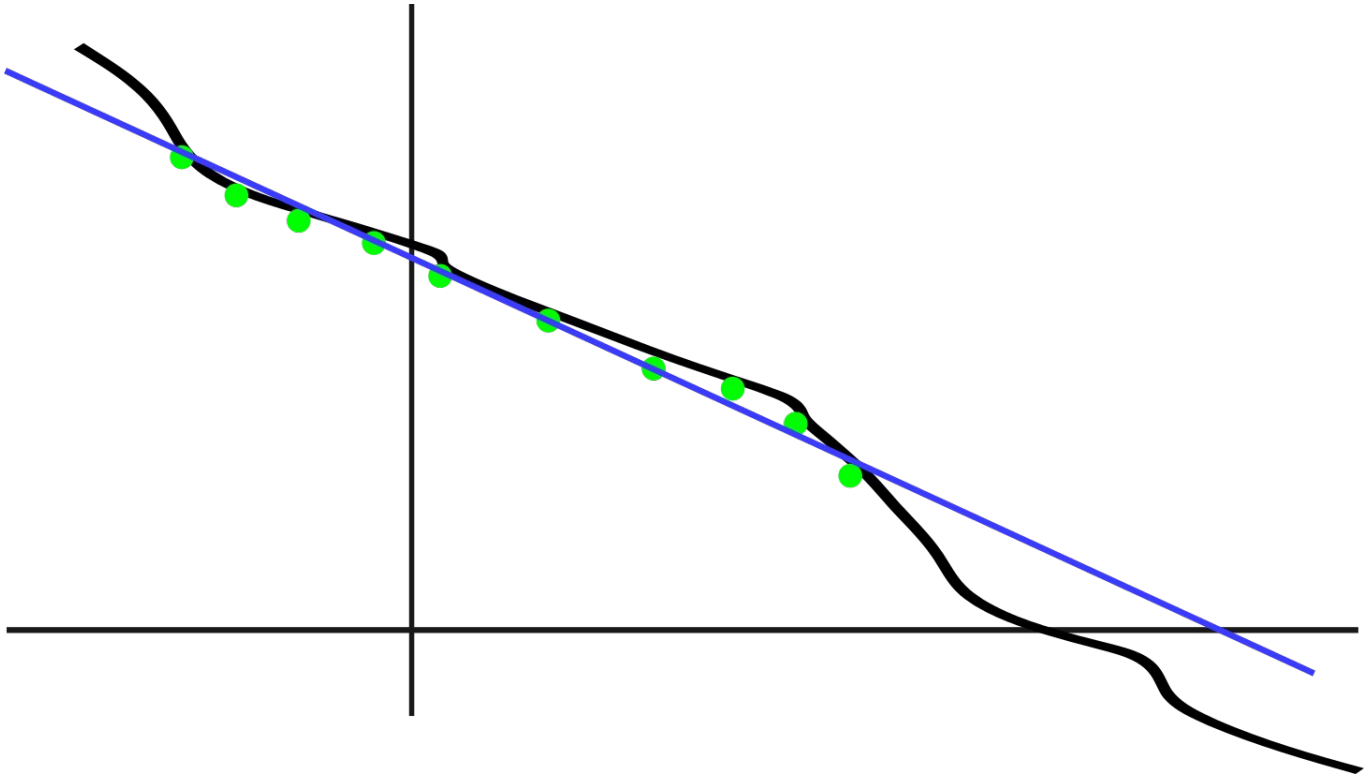


Least Squares: Minimize Sum of Squared Error



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

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 *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

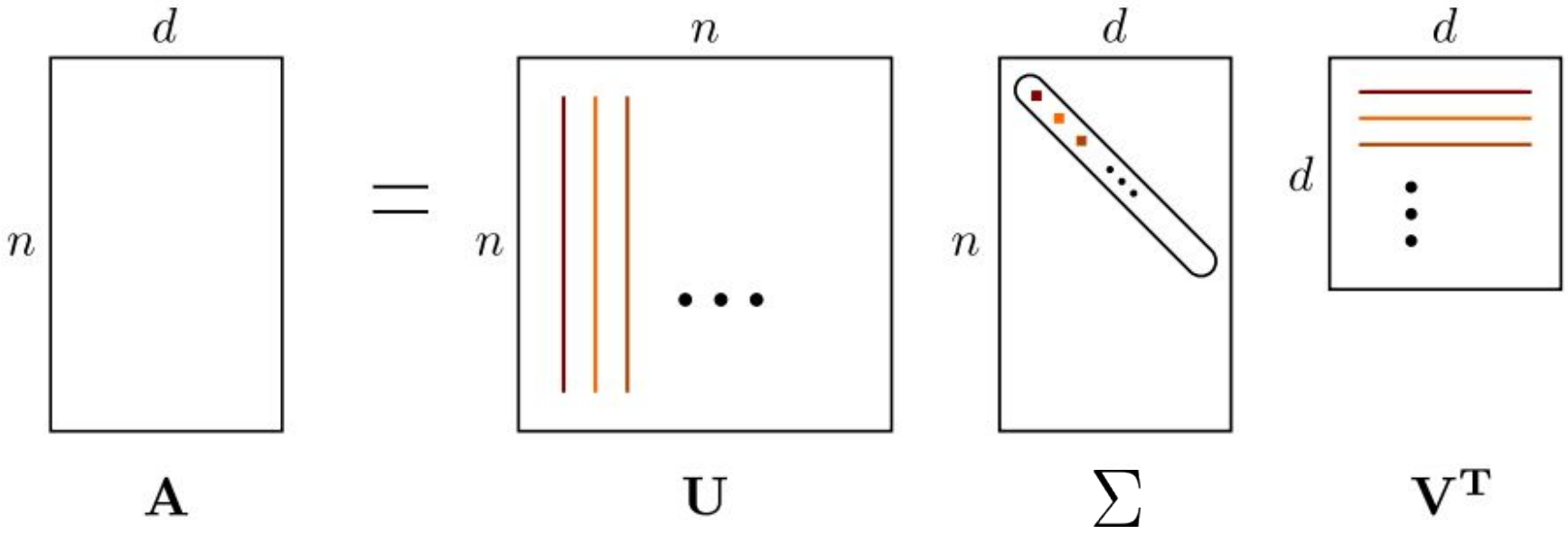
$$\hat{x} = \left(A^T A\right)^{-1} A^T b = A^{\dagger} b$$

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**)
- $A = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix}$ (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	Orthogonal
(Left Singular Vectors)	(Squared Eigenvalues)	(Right Singular Vectors)

$\Sigma^\dagger \triangleq$ 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 and Eigendecomposition $A = U\Sigma V^T$

Recall: For nonsingular A, eigenvectors *x* and eigenvalues lambda are the solutions to $Ax = \lambda x$

Or equivalently with *x* the rows of orthogonal matrix Q

$$AQ = Q\Lambda \leftrightarrow A = Q\Lambda Q^{-1}$$

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

So $\Sigma^\dagger \Sigma$ is the diagonal eigenvalue matrix for $A^T A$ and V is the eigenvector matrix
(similar for $AA^T, \Sigma\Sigma^\dagger, U$)

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)`