

# ECE 532 - lecture 9 - motivating the SVD

①

{ DEMO OF REGRESSION AND CROSS-VALIDATION IN MATLAB/excel }.

we'll motivate the SVD using  
examples drawn from

{ 1. sensitivity analysis  
2. compression }

ex 1: sensitivity in 1-D: suppose all numbers are subject to  $\pm 0.001$  error.

$$\text{solve } 2x=1 \Rightarrow 0.4993 \leq x \leq 0.5008$$

$$0.2x=1 \Rightarrow 4.97 \leq x \leq 5.03$$

$$0.02x=1 \Rightarrow 47.6 \leq x \leq 52.7 \quad (\text{should be } x=50)$$

$$0.002x=1 \Rightarrow 333 \leq x \leq 1001 \quad (\text{should be } x=500)$$

obvious problem! closer to zero  $\Rightarrow$  amplified error.

ex 2: sensitivity in 2-D (less obvious).

$$\text{solve } \begin{pmatrix} 1 & 1 \\ 1 & 1.001 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

$$\downarrow \begin{pmatrix} 1 & 1 & | & 2 \\ 0 & 0.001 & | & 0 \end{pmatrix} \Rightarrow \begin{cases} x=2 \\ y=0 \end{cases}$$

\* numbers aren't big!

$$\text{solve } \begin{pmatrix} 1 & 1 & | & 2 \\ 1 & 1.001 & | & 2.001 \end{pmatrix}$$

\* we didn't even touch the A matrix!

$$\downarrow \begin{pmatrix} 1 & 1 & | & 2 \\ 0 & 0.001 & | & 0.001 \end{pmatrix} \Rightarrow \begin{cases} x=1 \\ y=1 \end{cases}$$

\* related to the fact that columns of A are "almost" linearly dependent, i.e. A is "almost" singular.

(2)

Key issue: "rank" and "invertibility" are discrete concepts.

A matrix  $x$  is either invertible or it is not!

in reality, if the matrix is generated from data, then

★ all square matrices are invertible.

★ all  $m \times n$  matrices have rank  $\min(m, n)$ . (full rank).

the SVD, which we will learn about soon, gives us an understanding of "how close" a matrix is to being invertible. Or how close a matrix is to being rank  $r$  for some  $r$ .

low-rank matrices (or "almost low rank" matrices) are useful in compression.

Sample example: dense Netflix matrix  $A \in \mathbb{R}^{10^7 \times 10^4}$  (10M users, 10k movies)

$$\left\{ \begin{array}{l} \text{storage } \propto mn : 10^{11} \times 4 \text{ bytes } \approx 400 \text{ GB.} \end{array} \right.$$

$$\left\{ \begin{array}{l} \text{multiplication } y = Ax \propto mn : 10^{11} \text{ flop} / 10 \text{ Gflops } \approx 10 \text{ sec.} \end{array} \right.$$

If  $A = BC$  ( $A$  is approx rank  $r$ ) then if  $r=10$ ,

$$\left\{ \begin{array}{l} \text{storage } \propto mr + nr : (10^8 + 10^5) \times 4 \text{ bytes } \approx 0.4 \text{ GB.} \end{array} \right.$$

$$\left\{ \begin{array}{l} \text{multiplication } y = B(Cx) \propto mr + nr : (10^8 + 10^5) / 10 \text{ Gflops } \approx 10 \text{ ms.} \end{array} \right.$$

(1000x less space and 1000x faster).

(3)

Any matrix  $A \in \mathbb{R}^{m \times n}$  with  $\text{rank}(A) = r$  can be written

as  $A = \underbrace{BC}_{(m \times r)(r \times n)}$ . Why?  $r = \dim(\text{Range}(A))$ . if  $[b_1, \dots, b_r]$

is a basis for  $R(A)$ , then  $A = [a_1 \ a_2 \ \dots \ a_n]$ , each  $a_i \in R(A) = R(B)$ . So there exists some  $c_i \in \mathbb{R}^r$  such that

$a_i = Bc_i$  for each  $i$ . In other words;

$$\begin{aligned}[a_1 \ a_2 \ \dots \ a_n] &= [Bc_1 \ Bc_2 \ \dots \ Bc_n] \\ &= B[c_1 \ c_2 \ \dots \ c_n] \\ &= BC.\end{aligned}$$

The SVD gives us a direct and easy way of finding such a factorization.

\* Note: Other factorizations like this exist. For example, the LU decomposition, the QR factorization, or simply doing elementary row operations can produce factorizations as well. We will not talk about these other factorizations in class.

Fact: the rank of  $A$  is also the smallest  $r$  such that there exists  $B \in \mathbb{R}^{m \times r}$ ,  $C \in \mathbb{R}^{r \times n}$  with  $A = BC$ .

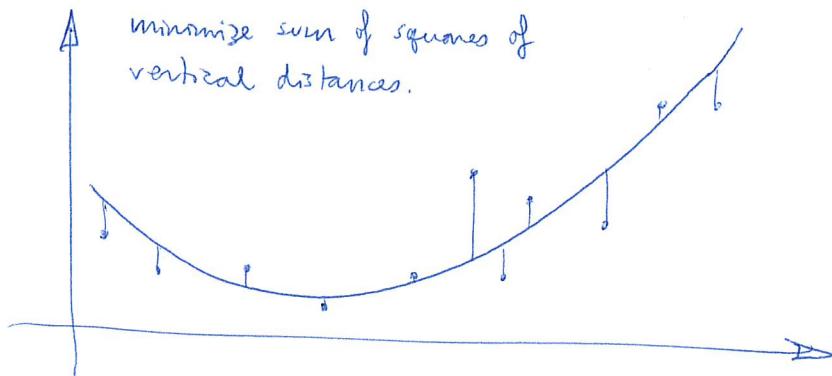
(4)

We saw that regression is a way of modeling a set of points as being approximately a linear combination of some set of basis functions.

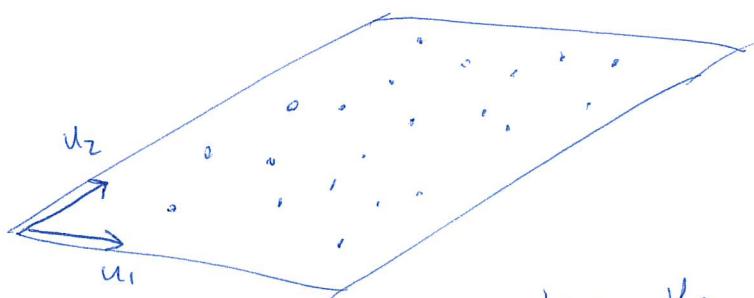
functions:

$$\{1, x, x^2\}$$

(quadratic fit)



Another possible model is if the points are "almost" linearly dependent, i.e. they almost lie in a subspace. This is related to the idea that the data matrix might be "almost low rank".



here, data are almost on a subspace. So we are claiming that each data point

$$x_i \approx \alpha_1^{(i)} u_1 + \alpha_2^{(i)} u_2$$

here, the  $u_1, u_2$  should be selected so that the perpendicular distance (projection of  $x_i$  onto  $\mathbb{R}[u_1, u_2]$ ) is as small as possible for all the points.

This is known as PCA  
(principal component analysis)

and is essentially the same as finding the SVD.

minimize sum of squares of perpendicular distances.

