EECS 16B : Module 3 / Lecture 4

Announcements:

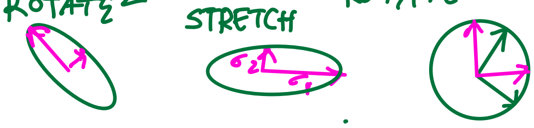
- 0.25 EC point for each lecture you attend for rest of the term.
- links.eecs16b.org/lecture-ec
- Lab Design Contest: See Ed post (EC opportunities!)

Last time:

- SVD: 3 forms
  - "full": $A = U \Sigma V^T$
  - "compact": $A = U_r \Sigma_r V_r^T$
  - "outer-product": $A = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^T$

- Geometry of SVD: $A\vec{x} = U \Sigma V^T \vec{x}$

ROTATE (U), STRETCH (Σ), ROTATE (V^T)



- Pseudoinverse: $A \in \mathbb{R}^{m \times n}$

  - $\underset{(m\times n)}{A} = U_r \underset{(r\times r)}{\Sigma_r} V_r^T \Rightarrow \underset{(n\times m)}{A^+} = V_r \underset{(r\times r)}{\Sigma_r^{-1}} U_r^T$
  - $A = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^T \Rightarrow A^+ = \sum_{i=1}^{r} \frac{1}{\sigma_i} \vec{v}_i \cdot \vec{u}_i^T$

    $r \le \min(m, n)$

  - $\boxed{Ax = y} \Rightarrow \boxed{x = A^+ y}$, holds for $\begin{cases} m < n \\ m > n \\ m = n \end{cases}$ and <u>any</u> rank (r)

$$A = U_r \Sigma_r V_r^T$$
$$A^+ = V_r \Sigma_r^{-1} U_r^T$$

<u>Last lecture:</u> $A\vec{x} = \vec{y}$, $A \in \mathbb{R}^{m \times n}$

$\vec{x} = A^+ \vec{y}$, where A^+ was defined using SVD of A, is:

- unique solution of $A\vec{x} = \vec{y}$ when A is square and invertible $(m = n = r)$ because $A^+ = A^{-1}$ in that case

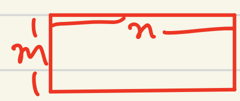
- LS solution when A is tall $(m > n)$. If full column rank also $(n = r)$, then

$$A^+ = (A^T A)^{-1} A^T, \circledast$$

which recovers LS solution studied before.

- Min. norm solution when A is wide $(n > m)$ and infinitely many solutions exist. If full row rank $(m = r)$,

$$A^+ = A^T (AA^T)^{-1}.$$



Thus,

$$\boxed{\vec{x}_{MN} = A^T (AA^T)^{-1} \vec{y}.}$$

$\circledast$

$$A = U_r \Sigma_r V_r^T \Rightarrow A = U_r \Sigma_r V^T \quad \left(\begin{matrix} V_r = V \\ \text{since } r = n \end{matrix}\right)$$

$$A^T = V \Sigma_r U_r^T \Rightarrow A^T A = V \Sigma_r \underbrace{U_r^T U_r}_{I_r} \Sigma_r V^T$$

$$= V \Sigma_r^2 V^T$$

$$\Rightarrow (A^T A)^{-1} = V \Sigma_r^{-2} V^T$$

$$\Rightarrow (A^T A)^{-1} A^T = (V \Sigma_r^{-2} V^T)(V \Sigma_r U_r^T) = V \Sigma_r^{-1} U_r^T$$

$$= A^+$$

$$\Rightarrow \vec{x}_{LS} = (A^T A)^{-1} A^T \vec{b} = A^+ \vec{b}$$

REMINDER!

If columns of Q are orthonormal,

- $Q^T Q = I$
- QQ^T is a projection onto $\text{Col}(Q)$

<u>TODAY:</u>

- PCA (Principal Component Analysis)

Why do we care about the SVD?

Suppose we have:

$$A_{m\times n} = U_{m\times m}\,\Big[\,\underbrace{\mathrm{diag}(100,\ 90,\ 55,\ 1,\ 0.5,\ 0.01,\ \ldots)}_{m}\ \Big|\ \underbrace{0}_{(n-m)}\,\Big]\,V^T_{n\times n}$$

If the underlying structure of the data matrix A is low-dimensional, then the SVD can help you "discover" this structure automatically.

→ also called unsupervised learning!

This is called **PRINCIPAL COMPONENT ANALYSIS** (PCA).

Dimensionality Reduction!

→ Fewer dimensions on data ≡ less to compute
→ " ≡ less to store
→ Redundancy removed "systematically"
→ Easier to visualize using 2D/3D plots
→ Better interpretability of data
→ Helps automatically "discover" the most significant features & skip the rest.

SVD/PCA has applications in many domains

<u>Healthcare</u>: • predicting patient's health & susceptability to diseases based on health risk factors.

<u>Biology</u> : • predicting which gene mutations are likely to cause cancer

<u>Retail</u>: • predicting which user will buy which product based on historical data

⋮

SVD / PCA

- data reduction technique
- data-driven generalization of the "Fourier transform" (FFT) tailored to specific problem
- used universally by big-tech companies!
  - Google : PageRank
  - FaceBook : Face Recognition
  - Netflix : Recommender systems (Netflix prize)
  - Amazon

    ⋮
  - $$$

- simple & interpretable
- scalable

Low-rank Approximation

Given a high rank matrix $A \in \mathbb{R}^{m \times n}$ with

$$r \approx \min\{m, n\},$$

find an approximation with rank $\ell \ll \min\{m, n\}$.

$$\underset{m\times n}{A} = \underset{m\times m}{U} \left[\begin{array}{c|c} \underbrace{\begin{matrix} 100 & & & & & & \\ & 90 & & & & & \\ & & 55 & & & & \\ & & & 1 & & & \\ & & & & 0.5 & & \\ & & & & & 0.01 & \\ & & & & & & \ddots \end{matrix}}_{m} & \underbrace{\begin{matrix} 0 \end{matrix}}_{(n-m)} \end{array}\right] \underset{n\times n}{V^T}$$

Suppose $m = n = r = 10{,}000$

$\ell = 10$

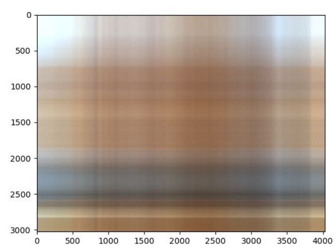
$$\boxed{A \approx \sum_{i=1}^{10} \sigma_i \vec{u}_i \vec{v}_i^{\,T}}$$

- A has 10^8 entries
- $\vec{u}_i, \vec{v}_i$ of dimension 10,000 each, so 20,000 total per outer-product SVD; 10 terms $\Rightarrow$ 200,000 versus 100,000,000

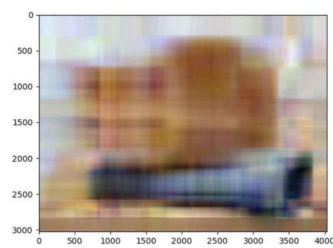
500 x savings!

Figure 5: The author's friend's cat Snyder.

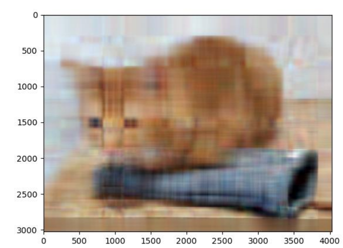
It can be represented as three matrices $A_R, A_G, A_B \in \mathbb{R}^{4032\times 3024}$ corresponding to R, G, and B of the image. We perform a rank-ℓ approximation $A_R = U_{R;\ell}\Sigma_{R;\ell}V_{R;\ell}^\top$, $A_G = U_{G;\ell}\Sigma_{G;\ell}V_{G;\ell}^\top$, $A_G = U_{G;\ell}\Sigma_{G;\ell}V_{G;\ell}^\top$, and then compose an image out of them, for different values of ℓ. The results are shown below.



(a) $\ell = 1$.



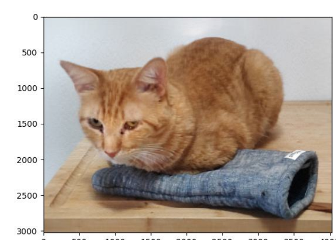
(b) $\ell = 5$.



(c) $\ell = 10$.

(d) $\ell = 25$.

(e) $\ell = 50$.



(f) $\ell = 100$.

By rank 100 approximation, the image is almost perfect. Now, the original image had $3 \times 4032 \times 3024 = 36578304$ entries; at rank 100, we have $3 \times 100 \times (4032 + 3024 + 1) = 2117100$ entries, so we need to store around 5% of the original image. Not bad!

See Appendix D for some code showing how these images were created.

6.2 PCA

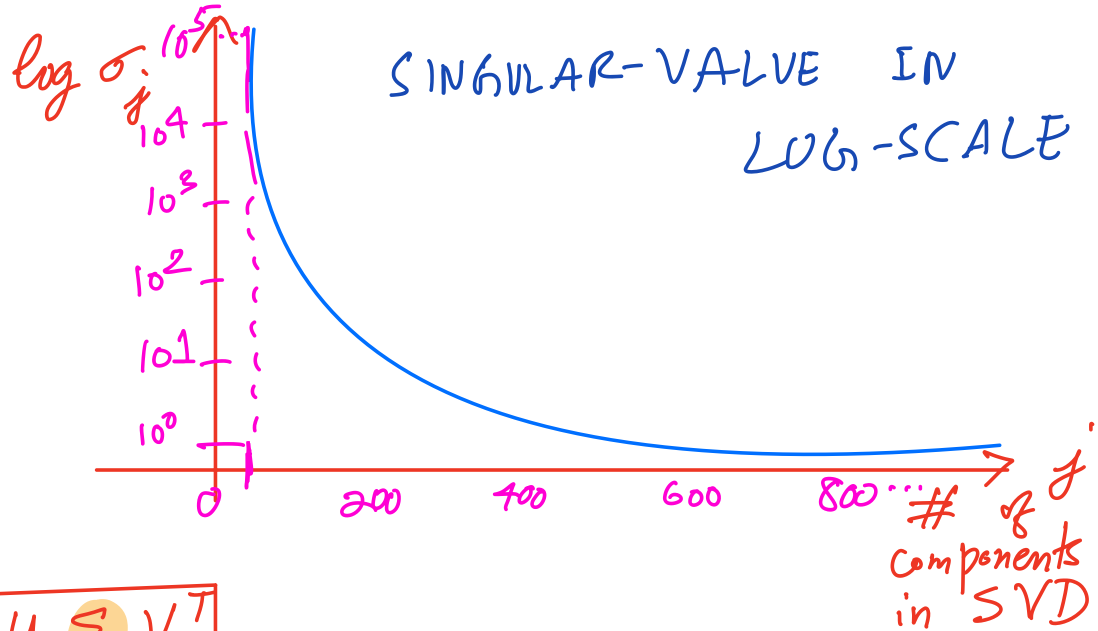
Sadly, no cats for this example.

Suppose we, as course staff, have m students in our class, and n assignments. Let $A \in \mathbb{R}^{m\times n}$ be a matrix, such that the i^{th} student's grade in the j^{th} assignment is A_{ij}.

- If we consider the assignments to be the data points, then A is a data matrix with column data.

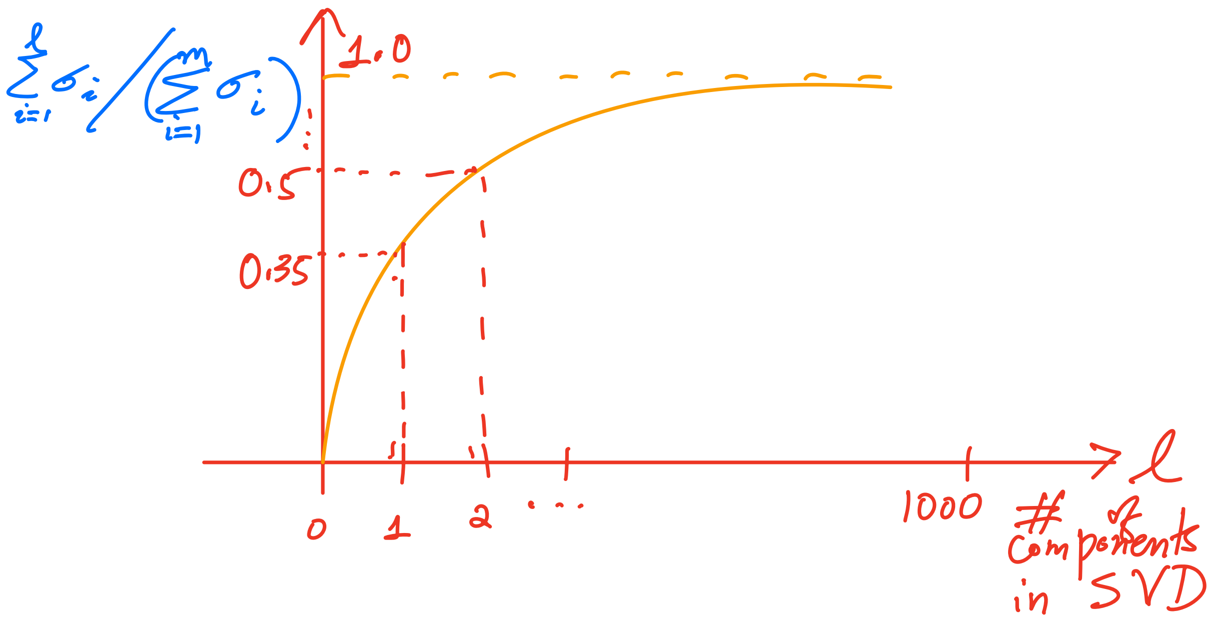
$$\boxed{A = U_r \Sigma_r V_r^T}$$

SINGULAR-VALUE IN LOG-SCALE



$$\boxed{A = U_r \Sigma_r V_r^T}$$

SINGULAR VALUE CUMULATIVE SUM



$$A_\ell = \sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^T$$

- Eckart-Young Theorem (Note 15) states that the SVD truncation above is more than a heuristic: A_ℓ gives the least possible deviation from A that is possible with a rank-ℓ matrix. More precisely,

A_ℓ above solves: $\min_{B \in \mathbb{R}^{m \times n}} \|A-B\|_F \rightarrow$ Frobenius norm

such that $\text{rank}(B)=\ell$.

$$\|X\|_F = \left\| \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ x_{m1} & \cdots & x_{mn} \end{pmatrix} \right\|_F = \left(x_{11}^2 + x_{12}^2 + \cdots + x_{mn}^2\right)$$

- Suppose we want the best rank-1 approximation to $A \in \mathbb{R}^{m \times n}$, then over all rank-1 matrices $B \in \mathbb{R}^{m \times n}$, the winner is the rank-1 SVD decomposition $(\sigma_1 \vec{u}_1 \vec{v}_1^T)$, where σ_1 is the first (largest) singular value of A and $\vec{u}_1, \vec{v}_1$ are the first singular vectors of A.

"Best" is in the sense of minimum Frobenius norm of error. $\|A-B\|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} |A_{ij} - B_{ij}|^2$

- Best Rank-2 approx. to A is $\sum_{i=1}^{2} \sigma_i \vec{u}_i \vec{v}_i^T$

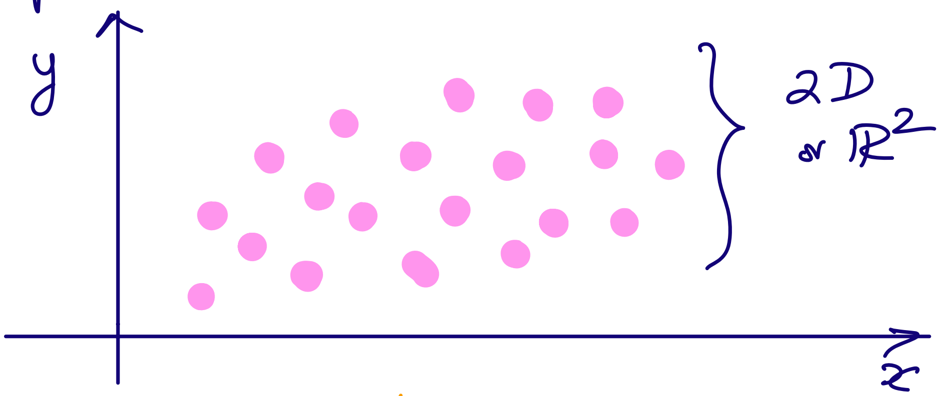
⋮

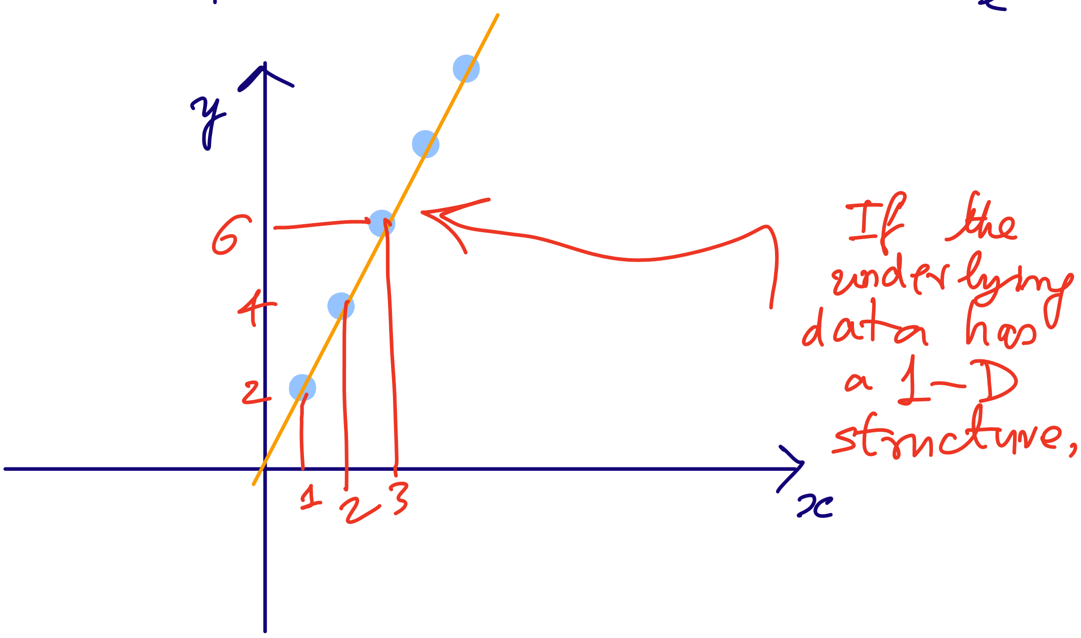
- Best Rank-ℓ approx. to A is $\sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^T$

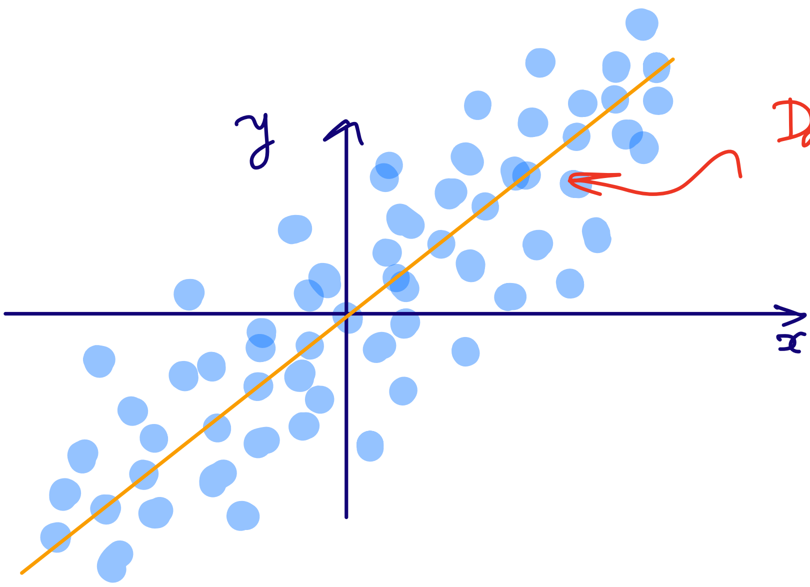
ordered from largest to smallest

PRINCIPAL COMPONENT ANALYSIS or PCA

"Principal Components," i.e. lower-dimensional structure in simple 2D data:







Data lies mostly on a 1-D space even though the "ambient dimension" is 2-D

Suppose we have 2-D points (x_i, y_i) as follows: $\{(1,2), (2,4), (3,6), (4,8), (5,10)\}$

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 6 & 8 & 10 \end{bmatrix}$$

(This is the same example matrix we have seen before in our study of the SVD!)

Ex.:

$$A = \begin{bmatrix} 1 & 2 & 4 & 5 \\ 2 & 4 & 8 & 10 \end{bmatrix}_{2\times4}$$

$$A = \underbrace{\begin{bmatrix} \frac{1}{\sqrt5} & -\frac{2}{\sqrt5} \\ \frac{2}{\sqrt5} & \frac{1}{\sqrt5} \end{bmatrix}}_{U_{2\times2}} \underbrace{\begin{bmatrix} \overset{\sigma_1}{230} & 0 & 0 & 0 \\ 0 & \overset{\sigma_2}{0} & 0 & 0 \end{bmatrix}}_{\Sigma_{2\times4}} \underbrace{\begin{bmatrix} \frac{1}{\sqrt{46}} & \frac{2}{23} & 2\sqrt{\frac{2}{23}} & \frac{5}{\sqrt{46}} \\ -\frac{5}{\sqrt{26}} & 0 & 0 & \frac{1}{\sqrt{26}} \\ \frac{-2}{\sqrt{273}} & 0 & \sqrt{\frac{13}{21}} & \frac{-10}{\sqrt{273}} \\ -\frac{1}{\sqrt{483}} & \sqrt{\frac{21}{23}} & \frac{-4}{\sqrt{483}} & \frac{-5}{\sqrt{483}} \end{bmatrix}}_{V^T_{4\times4}}$$

$\boxed{\text{("FULL" SVD)}}$

$$A = \overset{\vec{u}_1}{\begin{bmatrix} \frac{1}{\sqrt5} \\ \frac{2}{\sqrt5} \end{bmatrix}} \overset{\sigma_1}{\begin{bmatrix} \sqrt{230} \end{bmatrix}} \overset{\vec{v}_1^T}{\begin{bmatrix} \frac{1}{\sqrt{46}} & \sqrt{\frac{2}{23}} & 2\sqrt{\frac{2}{23}} & \frac{5}{\sqrt{46}} \end{bmatrix}}$$

$\boxed{\text{"COMPACT" SVD}}$

- $\vec{u}_1$ is a "basis" for the Col.space (A)
- $(x, y) \in \text{Col}(A)$ if $y = 2x$

Generalizing PCA concept

Movie recommendation problem (Netflix app.)

	U_1 User 1	U_2 User 2	U_3 User 3	...	U_{1000} User 1000
V_1 Video 1	75	80	20		50
V_2 Video 2	30	20	85		70
⋮				⋮ ⋮ ⋮ q_{ij}	
V_{100} Video 100	90	95	30		20

Q-matrix

q_{ij} : rating of user j for video i.

- 100 videos (movies) / 1000 users: users who rate (score) the movies.
- Goal: learn the "low-dimensional" structure underlying this "big" chunk of data

(In practice, there are millions of users & thousands of movies)

Goal: Understand the different "types" of movies
↳ use this to make recommendations to users.

Say every movie is represented by 4 attributes: (features)

SCORE for VIDEO i : $[\,a_i \quad b_i \quad c_i \quad d_i\,]$.

a_i: "action" level
b_i: "bloodiness" or horror level
c_i: "comedy" level
d_i: "drama" level

User j: "sensitivity" to components; i.e. how much user j "likes" or "dislikes" the movie attributes:

j : $[\,s_{aj} \quad s_{bj} \quad s_{cj} \quad s_{dj}\,]$

$$\boxed{q_{ij} = s_{aj}\cdot a_i + s_{bj}\, b_i + s_{cj}\cdot c_i + s_{dj}\cdot d_i}$$

e.g. $q_{ij} = 80\,(0.2) + 0\,(0.1) + 77\,(0) + 20\,(0.7) = \boxed{30}$

$\vec{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{100} \end{bmatrix}$ action scores of all videos $\in \mathbb{R}^{100}$

Similarly, for $\vec{b}, \vec{c}, \vec{d}$.

$$\vec{S_a} = \begin{bmatrix} s_{a_1} \\ s_{a_2} \\ \vdots \\ s_{a_{1000}} \end{bmatrix}$$

"action" sensitivity vector for users $\in \mathbb{R}^{1000}$

Similarly, for $\vec{S_b}$, $\vec{S_c}$, $\vec{S_d}$

Consider $\vec{a} \cdot \vec{S_a}^T$ $(100 \times 1)\ (1 \times 1000)$ "outer product"

$$= \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{100} \end{bmatrix} \begin{bmatrix} s_{a_1} & s_{a_2} & \cdots & s_{a_{1000}} \end{bmatrix}$$

$$= \begin{bmatrix} a_1 s_{a_1} & a_1 s_{a_2} & \cdots & a_1 s_{a_{1000}} \\ a_2 s_{a_1} & & & \vdots \\ \vdots & & & \\ a_{100} s_{a_1} & \cdots & & a_{100} s_{a_{1000}} \end{bmatrix}$$

$$Q = \vec{a} \cdot \vec{s_a}^T + \vec{b} \cdot \vec{s_b}^T + \vec{c} \cdot \vec{s_c}^T + \vec{d} \cdot \vec{s_d}^T$$

$$Q = U \Sigma V^T = \sum_{i=1}^{r} \sigma_i \vec{u_i} \vec{v_i}^T$$

r: rank of Q

The (k) principal components of matrix Q:

- along the columns: $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_k$
- along the rows: $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_k$

$$\begin{array}{cc} & \begin{array}{cccc} U_1 & U_2 & & U_{1000} \end{array} \\ Q = \begin{array}{c} V_1 \\ V_2 \\ \vdots \\ V_{100} \end{array} & \begin{bmatrix} | & | & & | \\ \vec{q}_1 & \vec{q}_2 & \cdots & \vec{q}_n \\ | & | & & | \end{bmatrix} \end{array}$$

$(m = 100, \; n = 1000)$

- Data is organized by columns:
  - i.e., each data point is a 100-dim. vector containing User j's ratings for the 100 movies $\{q_{1j}, q_{2j}, \ldots q_{100,j}\}$

Goal: Find the first principal component; that is, that vector (direction) that is "most informative" about the data.