Lecture 10 : Finding structure with randomness

- Singular values + vectors
- Randomized subspace iteration
- Randomly pivoted Cholesky

Singular values + vectors

- Every matrix $A \in \mathbb{R}^{L \times N}$ has a singular value decomposition

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

$$= (u_1 \; u_2 \; \cdots \; u_{L \wedge N}) \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_{L \wedge N} \end{pmatrix} (v_1 \; v_2 \; \cdots \; v_{L \wedge N})^T$$

$$= \sum_{i=1}^{L \wedge N} u_i \sigma_i v_i^T$$

- $u_1, u_2, \ldots, u_{L \wedge N}$ are orthonormal vectors in $\mathbb{R}^L$

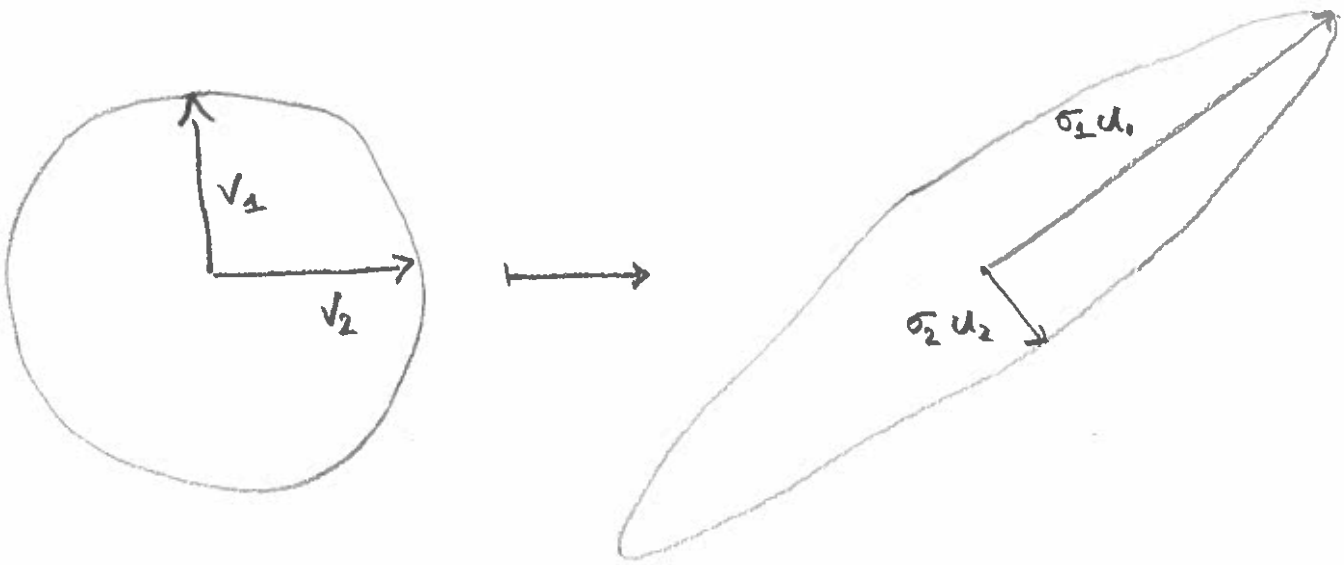
  $\langle u_i, u_j \rangle = \delta_{ij}$

- $v_1, v_2, \ldots, v_{L \wedge N}$ are orthonormal vectors in $\mathbb{R}^N$

  $\langle v_i, v_j \rangle = \delta_{ij}$

- $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{L \wedge N} \geq 0$ are ordered singular values

- Visually



- Singular vectors are stretched + rotated

Data science interpretation

↳ Top singular vectors represent dominant patterns in data ("principal components")

↳ Bottom singular vectors represent unwanted noise

⇒ we want to recover top singular vectors!

Q How?

A1 we can do np.linalg.svd(A) but this requires roughly $10 \min\{L, N\}$ passes through the data. If $L = N = 100,000$, then we need to pass through the matrix 1,000,000 times ∴

A2 we can use randomized low-rank matrix approximation + only pass through the matrix

100 - 1,000 times (and it's relatively easy to code) :)

Overview Combine (a) repeated matrix multiplications, (b) large blocks of 10-100 vectors, and (c) orthogonal projection.

(a) repeated matrix multiplications

- If $\vec{w} = \sum_{i=1}^{N} c_i \vec{v}_i$ is an input vector,

then

$$A\vec{w} = \sum_{i=1}^{NAL} (c_i \sigma_i)\, \vec{u}_i$$

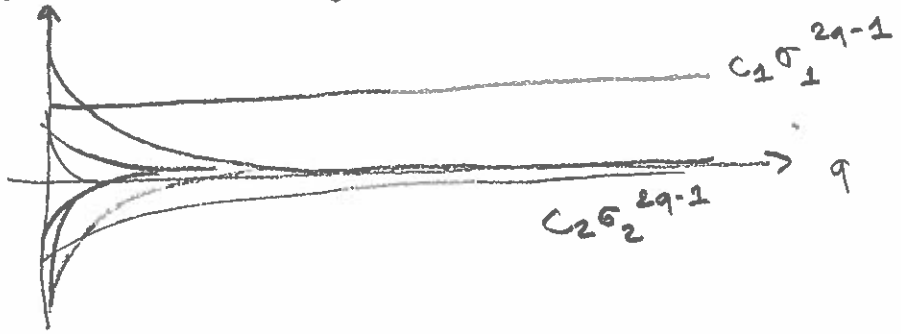
(c_i: stretch; $\vec{u}_i$: rotate)

and

$$(AA^T)^{q-1} A\vec{w} = \sum_{i=1}^{NAL} (c_i \sigma_i^{2q-1})\, \vec{u}_i$$

($(AA^T)^{q-1}$: repeated multiplications with A and A^T; σ_i^{2q-1}: lots of stretch; $\vec{u}_i$: rotate)

- Coefficients are geometrically changing, with rates determined by singular values



- Increasing overlap with top singular vectors as we increase q, even if coefficients c_i are completely random as long as $c_i \neq 0$.

(b) large blocks of vectors

- Set $\Omega = (\vec{\omega}_1 \ \vec{\omega}_2 \ \cdots \ \vec{\omega}_K) \in \mathbb{R}^{N \times K}$ where $10 \leq K \leq 100$ and $\vec{\omega}_i \sim$ iid $\mathcal{N}(0, I)$ for $1 \leq i \leq K$.

- It's fast to compute

$$\Omega \longmapsto A\Omega$$

and

$$\Omega \longmapsto (AA^T)^{q-1} A\Omega$$

on modern computers due to caching, multithreading, and parallelism

(c) orthogonal projection

- An orthogonal projection matrix $\Pi_M \in \mathbb{R}^{L \times L}$ where $M = (AA^T)^{q-1} A\Omega$ is defined as

$$\Pi_M = XX^T \quad \text{where} \quad X = \underset{\uparrow}{\text{orth}}(M)$$

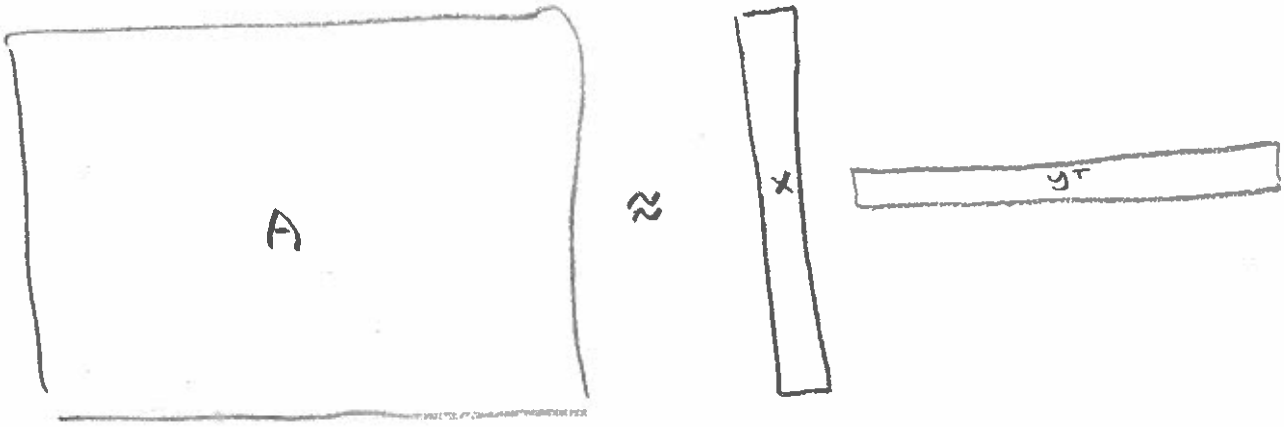
orthogonalize columns of M

- keeps all the components inside range(M),

annihilates all the components inside range(M)+

- we can build a low-rank approximation

$$\hat{A} = \Pi_m A = X X^T A = X Y^T, \quad \text{where} \quad Y = A^T X$$



$\Rightarrow$ $X \in \mathbb{R}^{L \times K}$ and $Y \in \mathbb{R}^{N \times K}$ are low-rank,
easy to store, easy to calculate singular values + vectors

Randomized subspace iteration

Input: matrix $A \in \mathbb{R}^{L \times N}$, block size K, iteration count q

Output: Orthogonal $U \in \mathbb{R}^{L \times K}$, orthogonal $V \in \mathbb{R}^{N \times K}$, diagonal $\Sigma \in \mathbb{R}^{K \times K}$ s.t. $\hat{A} = U \Sigma V^*$.

↖ Approximate singular vectors + values

1. Generate Gaussian $Y \in \mathbb{R}^{N \times K}$
2. For $i = 1, \ldots, q$:
   a) $X = AY$

b) $X = \text{orth}(Y)$

c) $Y = A^T X$

3. Calculate SVD $Y^T = \hat{U} \Sigma V^T$

4. $U = X\hat{U}$ □

Notes

- we can run this algorithm with $q = 1$. Then it's called "Randomized SVD" and works well for rapidly decaying singular values.

- we can <u>supercharge</u> the algorithm to produce an approximation

$$\hat{A} = \Pi_M A, \quad \text{where} \quad M = \underbrace{\left(A\Omega \;\; (AA^T)A\Omega \;\cdots\; (AA^T)^{q-1}A\Omega\right)}_{L \times (Kq) \text{ block matrix}}$$

Then it's called "Randomized block Krylov iteration" and works well for slowly decaying singular values.

Thm RSI satisfies

$$\mathbb{E}\,\|A - \hat{A}\|^2 \le \min_{r \le K-2} \sigma_{r+1}(A)^2 \left(1 + \frac{r}{K-r-1} \sum_{i=r+1}^{LAN} \left(\frac{\sigma_i(A)}{\sigma_{r+1}(A)}\right)^{4q-2}\right)^{\frac{1}{4q-2}}$$

Gets better with increasing K and q

Summary RSI (+ RSVD + RBKI) often provide a good approximation of A (and its singular values and vectors) using 10-1000 passes through the data.

Q Can we provide a good approximation of A without even looking at all its entries?

A1 No, definitely not. We might miss an important entry, e.g.

$$A = \begin{pmatrix} & & & & \\ & & & 1 & \\ & & & & \\ & & & & \end{pmatrix}$$

← How can we possibly find the nonzero entry?

A2 If $A \in \mathbb{R}^{N \times N}$ is positive semidefinite (psd) then yes maybe.

psd $\Rightarrow$ $\langle v, Av \rangle \geq 0 \quad \forall\, v \in \mathbb{R}^N$

$\Rightarrow$ $A_{ii} + 2\alpha A_{ij} + \alpha^2 A_{jj} \geq 0$ for $\alpha = \pm \left(A_{ii} / A_{jj} \right)^{1/2}$

$\Rightarrow$ $|A_{ij}| \leq A_{ii}^{1/2} A_{jj}^{1/2}$

Off-diagonal elements will be small if corresponding diagonal elements are small

$\Rightarrow$ Only need to search for biggish diagonal entries

Randomly pivoted Cholesky: intuition

1. Set the initial approximation $\hat{A}^{(0)} = 0$ and residual $A^{(0)} = A - \hat{A}^{(0)} = A$.

2. For $t = 1, \ldots, k$:

   a) Randomly sample $s_t \in \{1, \ldots, N\}$ with

   $$\mathbb{P}\{s_t = i\} = A^{(t-1)}_{ii} \,/\, \operatorname{diag} A^{(t-1)}_{ii}$$

   b) $$\hat{A}^{(t)} = \hat{A}^{(t-1)} + \frac{A^{(t-1)}(\cdot, s_t)\, A^{(t-1)}(s_t, \cdot)}{A^{(t-1)}(s_t, s_t)}$$

   c) $$A^{(t)} = A^{(t)} - \frac{A^{(t-1)}(\cdot, s_t)\, A^{(t-1)}(s_t, \cdot)}{A^{(t-1)}(s_t, s_t)}$$

⇒ Each step, we choose an index s_t randomly and update the approximation $\hat{A}^{(t)}$ and residual $A^{(t)}$ using the s_t column of $A^{(t-1)}$.

Ex $A^{(0)} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ $\begin{matrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{matrix}$ $\hat{A}^{(0)} = 0$

$s_1 = 3$

$$A^{(1)} = \begin{pmatrix} 0 & & & & \\ & 0 & & & \\ & & 0 & & \\ & & & 1 & 1 & 1 \\ & & & 1 & 2 & 1 \\ & & & 1 & 1 & 1 \end{pmatrix} \qquad \hat{A}^{(1)} = \begin{pmatrix} 1 & 1 & 1 & & & \\ 1 & 1 & 1 & & & \\ 1 & 1 & 1 & & & \\ & & & 0 & & \\ & & & & 0 & \\ & & & & & 0 \end{pmatrix}$$

$S_2 = 6$

$$A^{(2)} = \begin{pmatrix} 0 & & & & \\ & 0 & & & \\ & & 0 & & \\ & & & 0 & \\ & & & & 1 \\ & & & & & 0 \end{pmatrix} \qquad \hat{A}^{(2)} = \begin{pmatrix} 1 & 1 & 1 & & & \\ 1 & 1 & 1 & & & \\ 1 & 1 & 1 & & & \\ & & & 1 & 1 & 1 \\ & & & 1 & 1 & 1 \\ & & & 1 & 1 & 1 \end{pmatrix}$$

In general, we obtain a "Nyström approximation"

$$\hat{A} = A(:,S)\,\big[A(S,S)\big]^{+}\,A(S,:)$$

where $S = \{s_1, \dots, s_k\}$ are randomly selected columns

- Perfectly recovers the rows + columns in S
- Approximates the other rows and columns

Thm RPC satisfies

$$\mathbb{E}\,\mathrm{tr}\left(A - \hat{A}^{(t)}\right) \le 2 \sum_{i=r+1}^{N} \lambda_i(A) \quad \text{if} \quad t \ge r\left[1 + \log\left(\frac{\mathrm{tr}\,A}{\sum_{i=r+1}^{N} \lambda_i(A)}\right)\right]$$

Randomly pivoted Cholesky: fast implementation

Input: psd matrix $A \in \mathbb{R}^{N \times N}$, iteration count K

Output: factor matrix $F \in \mathbb{R}^{N \times K}$ s.t. $\hat{A} = FF^T$

1. Initialize $d = \text{diag}\, A$ and F s.t. $F_{ij} = 0$ for $1 \le i \le N$, $1 \le j \le N$.

2. For $t = 1, \ldots, K$:

   a) Sample $s_t \sim d / \sum_{i=1}^{N} d_i$

   b) Set $g = A(:, s_t) - F(:, 1:(t-1))\, F(s_t, 1:(t-1))^T$

   c) Set $F(:, t) = g / \sqrt{g_{s_t}}$

   d) Set $d = d - |F(:, t)|^2$ □

⇒ Only $(K+1)N$ entry look-ups and $\mathcal{O}(K^2 N)$ operations

- High-quality approximation if A has fast-decaying eigenvalues
- Especially useful for approximating Kernel matrices in machine learning since each entry evaluation is expensive.