COMS 4771
Correlation analysis (SVD/PCA)

Singular value decomposition (SVD)

- Perhaps the most important matrix decomposition / factorization
- Many applications in all areas of science, engineering, etc.
  - Solve best fitting subspace problem
  - Principal components analysis (PCA)
  - Interpret/understand linear transformations
  - Construct low-rank matrix approximations
  - Many others in domain-specific contexts
    - Graphics & molecular biology/chemistry: optimal rotation to align point sets
    - Information retrieval: recommender systems
    - Social sciences: discover close-knit communities in networks
    - Statistics/applied math: solve ill-posed linear inverse problems

1. Best fitting subspaces

Example from psychometrics

International Personality Items	Very Inaccurate -2	Moderately Inaccurate -1	Neither Accurate Nor Inaccurate 0	Moderately Accurate +1	Very Accurate +2
1. Am the life of the party.	O	O	O	O	O
2. Feel little concern for others.	O	O	O	O	O
3. Am always prepared.	O	O	O	O	O
4. Get stressed out easily.	O	O	O	O	O
5. Have a rich vocabulary.	O	O	O	O	O
6. Don't talk a lot.	O	O	O	O	O
7. Am interested in people.	O	O	O	O	O
8. Leave my belongings around.	O	O	O	O	O
9. Am relaxed most of the time.	O	O	O	O	O
10. Have difficulty understanding abstract ideas.	O	O	O	O	O
11. Feel comfortable around people.	O	O	O	O	O
12. Insult people.	O	O	O	O	O
13. Pay attention to details.	O	O	O	O	O
14. Worry about things.	O	O	O	O	O
15. Have a vivid imagination.	O	O	O	O	O

High-dimensional data from psychometrics

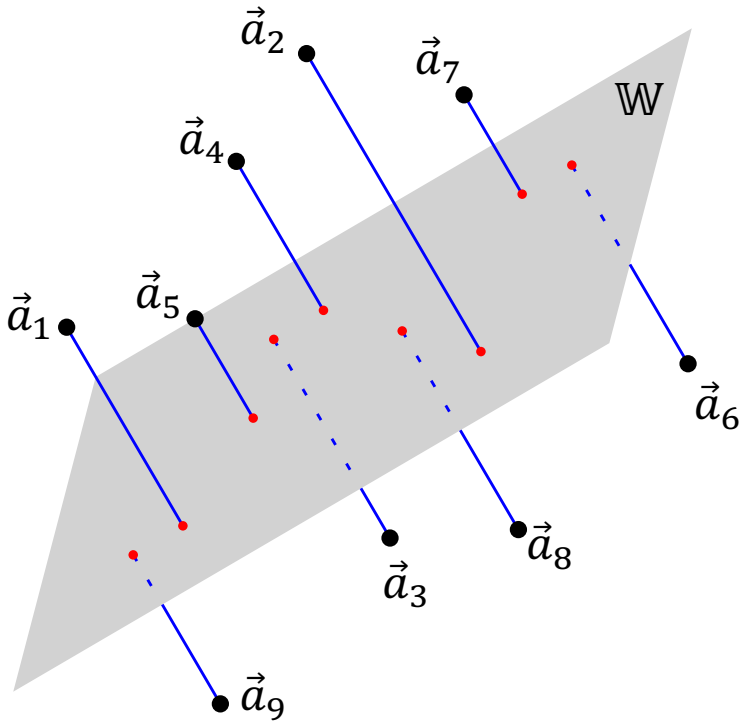
- Response vector for subject i:

$$\vec{a}_i = (0, -2, +2, -1, 0, +1, -2, -1, -2, +2, 0, -1, +1, \dots) \in \mathbb{R}^d$$

  One value per "International Personality Item" ($d = 3320$)
- **Hypothesis**: a few underlying "personality traits" can explain variability in responses of subjects
  - E.g., "Big Five": openness to experience, conscientiousness, extraversion, agreeableness, neuroticism
- **Linear algebraic hypothesis**: subjects' responses $\vec{a}_1, \dots, \vec{a}_n$ are "close" to a low-dimensional subspace of $\mathbb{R}^d$

Best fitting subspace problem

- **Input**: Data $\vec{a}_1, \ldots, \vec{a}_n \in \mathbb{R}^d$, target dimension k
- **Goal**: Find k-dim. subspace W of $\mathbb{R}^d$ that is "closest" to the data



Quality measure

- Distance of $\vec{a}_i$ to subspace W given by

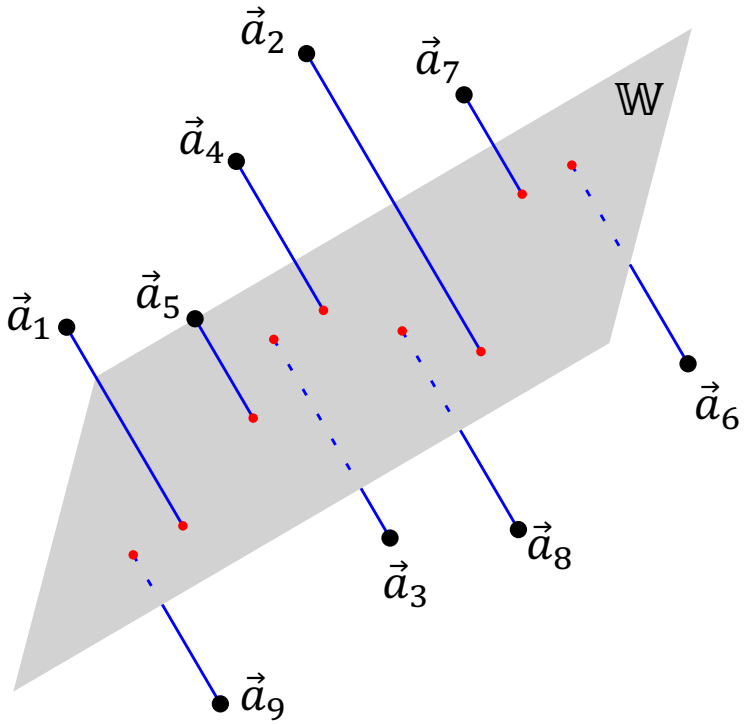
$$\|\vec{a}_i - P_W \vec{a}_i\|$$

  (In this lecture, always use P_W to denote orthoprojector for W)
- There are n "distances" that we want to be small, one per data point
- **Our choice of quality measure**: sum of squares

$$\text{cost}(W) = \|\vec{a}_1 - P_W \vec{a}_1\|^2 + \cdots + \|\vec{a}_n - P_W \vec{a}_n\|^2$$

Best fitting subspace problem (again)

- **Input**: Data $\vec{a}_1, \ldots, \vec{a}_n \in \mathbb{R}^d$, target dimension k
- **Goal**: Find k-dim. subspace W of $\mathbb{R}^d$ of smallest cost(W)



Punchline: Solving k-BFS yields the SVD

1. There is a simple algorithm for k-dim. Best Fitting Subspace (k-BFS)
   - **Greedy algorithm**: Repeatedly solve a Best Fitting Line problem

2. Solving k-BFS for all k provides a decomposition of data matrix

$$A = \begin{bmatrix} \leftarrow & \vec{a}_1^\top & \rightarrow \\ & \vdots & \\ \leftarrow & \vec{a}_n^\top & \rightarrow \end{bmatrix} = \sigma_1 \vec{u}_1 \vec{v}_1^\top + \cdots + \sigma_r \vec{u}_r \vec{v}_r^\top$$

Singular Value Decomposition (SVD) of A

where

- r is rank of A
- $(\vec{u}_1, \ldots, \vec{u}_r)$ is ONB for $\mathrm{CS}(A)$ (called left singular vectors of A)
- $(\vec{v}_1, \ldots, \vec{v}_r)$ is ONB for $\mathrm{CS}(A^\top)$ (called right singular vectors of A)
- $\sigma_1, \ldots, \sigma_r > 0$ (called singular values of A)

Reformulating cost

- **Pythagorean identity**: For any subspace W and vector $\vec{v}$

$$\|\vec{v}\|^2 = \|P_W\vec{v}\|^2 + \|\vec{v} - P_W\vec{v}\|^2$$

- Therefore:

$$\text{cost}(W) = \|\vec{a}_1\|^2 + \cdots + \|\vec{a}_n\|^2 - \underbrace{(\|P_W\vec{a}_1\|^2 + \cdots + \|P_W\vec{a}_n\|^2)}_{\text{gain}(W)}$$

Reformulating gain

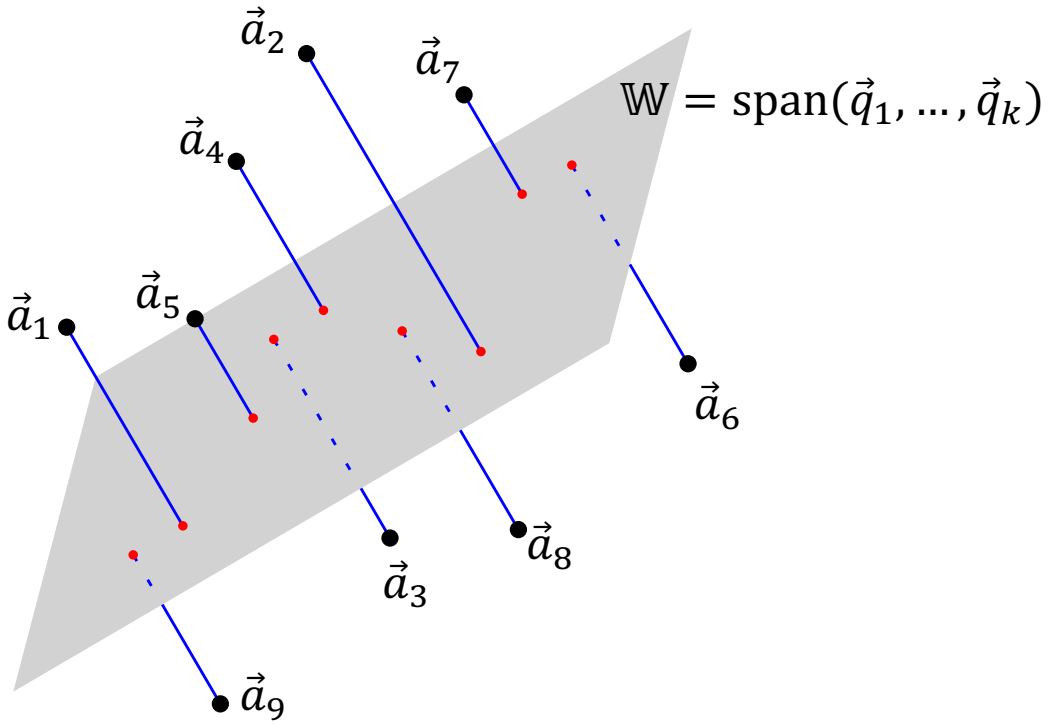
- Identify k-dim. subspace W with ONB $(\vec{q}_1, \dots, \vec{q}_k)$ for W:

$$\text{gain}(\vec{q}_1, \dots, \vec{q}_k) = \sum_{i=1}^{n} \|P_W \vec{a}_i\|^2 = \sum_{i=1}^{n} \sum_{j=1}^{k} \langle \vec{a}_i, \vec{q}_j \rangle^2 = \sum_{j=1}^{k} \underbrace{\sum_{i=1}^{n} \langle \vec{a}_i, \vec{q}_j \rangle^2}_{\text{gain}(\vec{q}_j)}$$

(by Parseval's identity)

Best fitting subspace problem (yet again)

- **Input**: Data $\vec{a}_1, \ldots, \vec{a}_n \in \mathbb{R}^d$, target dimension k
- **Goal**: Find ONB $(\vec{q}_1, \ldots, \vec{q}_k)$ of highest $\text{gain}(\vec{q}_1) + \cdots + \text{gain}(\vec{q}_k)$



Greedy algorithm for k-BFS

- **Input**: Data $\vec{a}_1, \dots, \vec{a}_n \in \mathbb{R}^d$, target dimension k
- **For** $j = 1,2, \dots, k$ **do**
  - $S_{j-1} = \mathrm{span}(\{\vec{v}_1, \dots, \vec{v}_{j-1}\})$
  - Let $\vec{v}_j$ be unit vector $\vec{x} \in S_{j-1}^{\perp}$ of highest gain$(\vec{x})$ (★)
- **Return** ONB $(\vec{v}_1, \dots, \vec{v}_k)$ for $S_k = \mathrm{span}(\{\vec{v}_1, \dots, \vec{v}_k\})$

Sample run of 2-BFS greedy algorithm

- Data ($n = 4$ data points in $\mathbb{R}^d$ for $d = 4$)

$$A = \begin{bmatrix} \leftarrow & \vec{a}_1^\top & \rightarrow \\ \leftarrow & \vec{a}_2^\top & \rightarrow \\ \leftarrow & \vec{a}_3^\top & \rightarrow \\ \leftarrow & \vec{a}_4^\top & \rightarrow \end{bmatrix} = \begin{bmatrix} 3 & 1 & 2 & 0 \\ -1 & -3 & 0 & -2 \\ 0 & 2 & 1 & 3 \\ -2 & 0 & -3 & -1 \end{bmatrix}$$

- Step $j = 1$

$$S_0 = \{\vec{0}\}, \qquad S_0^\perp = \mathbb{R}^4, \qquad \vec{v}_1 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$$

achieves $\text{gain}(\vec{v}_1) = 3^2 + (-3)^2 + 3^2 + (-3)^2 = 36$

Sample run of 2-BFS greedy algorithm (2)

- Data ($n = 4$ data points in $\mathbb{R}^d$ for $d = 4$)

$$A = \begin{bmatrix} \leftarrow & \vec{a}_1^\top & \rightarrow \\ \leftarrow & \vec{a}_2^\top & \rightarrow \\ \leftarrow & \vec{a}_3^\top & \rightarrow \\ \leftarrow & \vec{a}_4^\top & \rightarrow \end{bmatrix} = \begin{bmatrix} 3 & 1 & 2 & 0 \\ -1 & -3 & 0 & -2 \\ 0 & 2 & 1 & 3 \\ -2 & 0 & -3 & -1 \end{bmatrix}$$

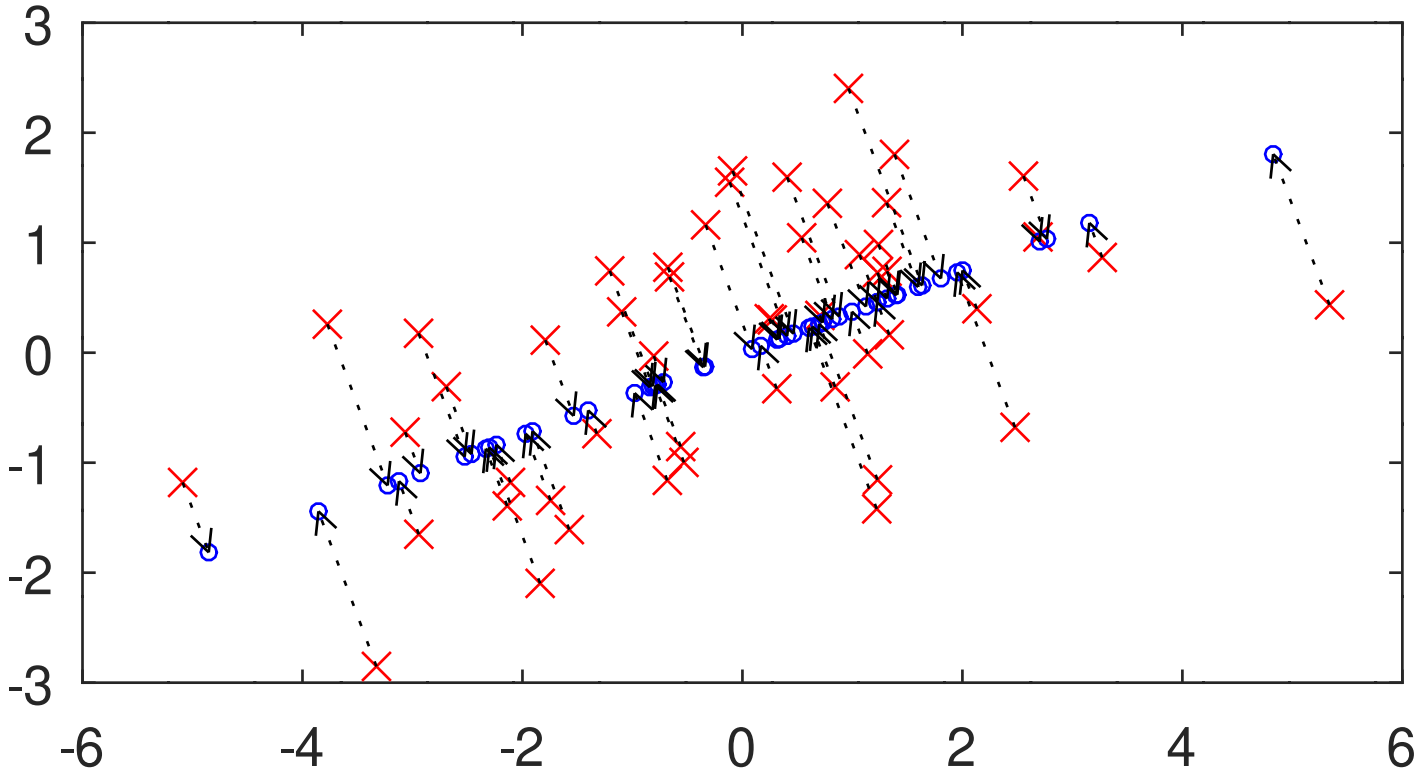
- Step $j = 2$

$$S_1 = \text{span}(\vec{v}_1), \qquad S_1^\perp = \text{NS}(\vec{v}_1^\top), \qquad \vec{v}_2 = \left(\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}\right)$$

achieves $\text{gain}(\vec{v}_2) = 2^2 + 2^2 + (-2)^2 + (-2)^2 = 16$

Best fitting line (★)

- Step j of k-BFS greedy algorithm: Solve "Best Fitting Line" problem restricted to orthogonal complement of $S_{j-1} = \text{span}(\vec{v}_1, \ldots, \vec{v}_{j-1})$



Optimality of k-BFS greedy algorithm

Theorem: Consider any dataset $\vec{a}_1, \ldots, \vec{a}_n$ and any k

1. k-BFS greedy algorithm finds k-dim. subspace S_k such that

$$\text{gain}(S_k) \geq \text{gain}(W)$$

   for all k-dim. subspaces W

2. Non-negative values $\sigma_j = \sqrt{\text{gain}(\vec{v}_j)}$ satisfy

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k$$

More properties of k-BFS greedy algorithm

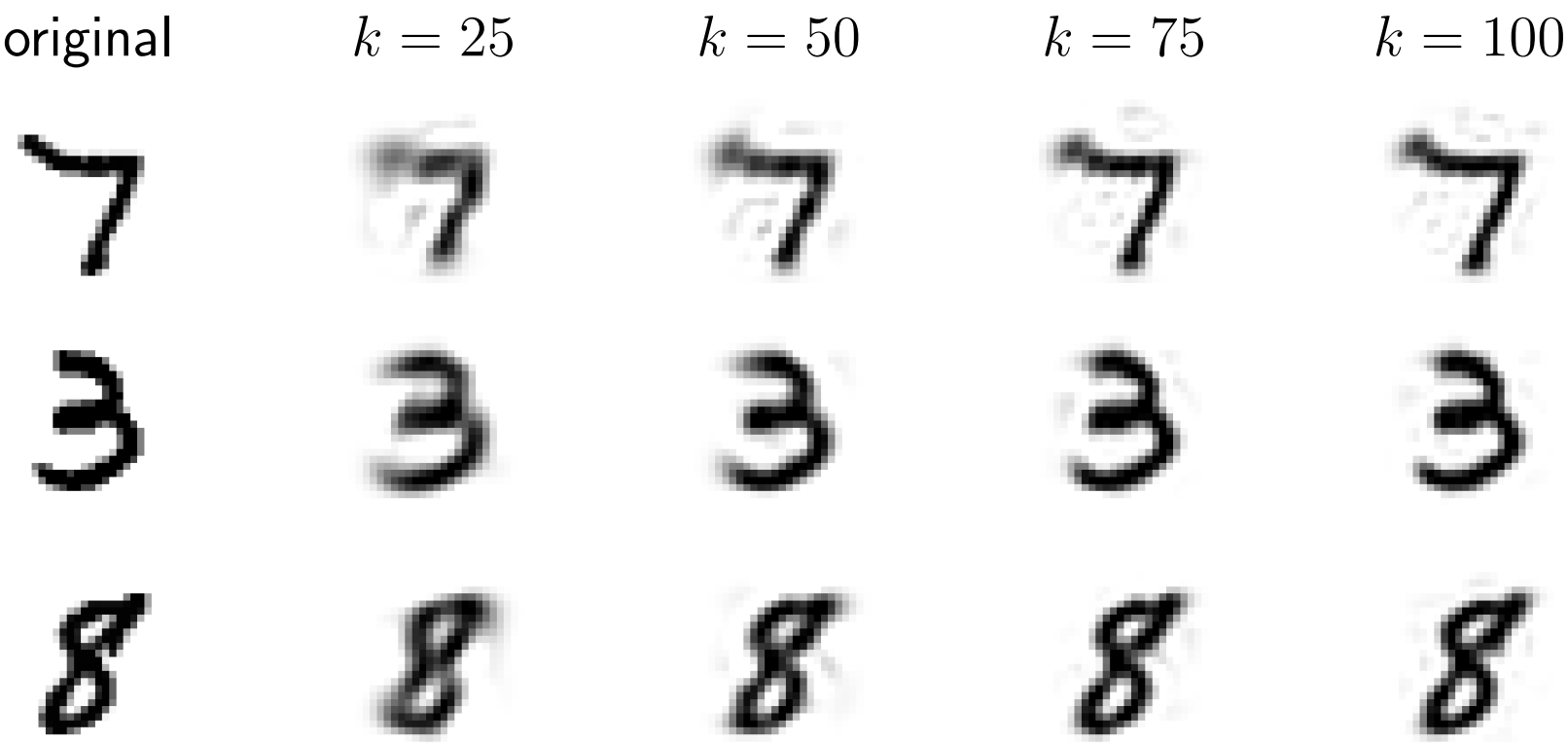
Theorem: Let A be $n \times d$ matrix with rows $\vec{a}_1^\top, \ldots, \vec{a}_n^\top$; let $r = \text{rank}(A)$

1. $S_r = \text{span}(\{\vec{a}_1, \ldots, \vec{a}_n\}) = \text{CS}(A^\top)$
2. $\sigma_j = \left\|A\vec{v}_j\right\| > 0$ for all $j \in \{1, \ldots, r\}$
3. $\text{cost}(S_k) = \sigma_{k+1}^2 + \cdots + \sigma_r^2$
4. If $\vec{u}_j = \frac{1}{\sigma_j} A\vec{v}_j$ for $j \in \{1, \ldots, r\}$, then

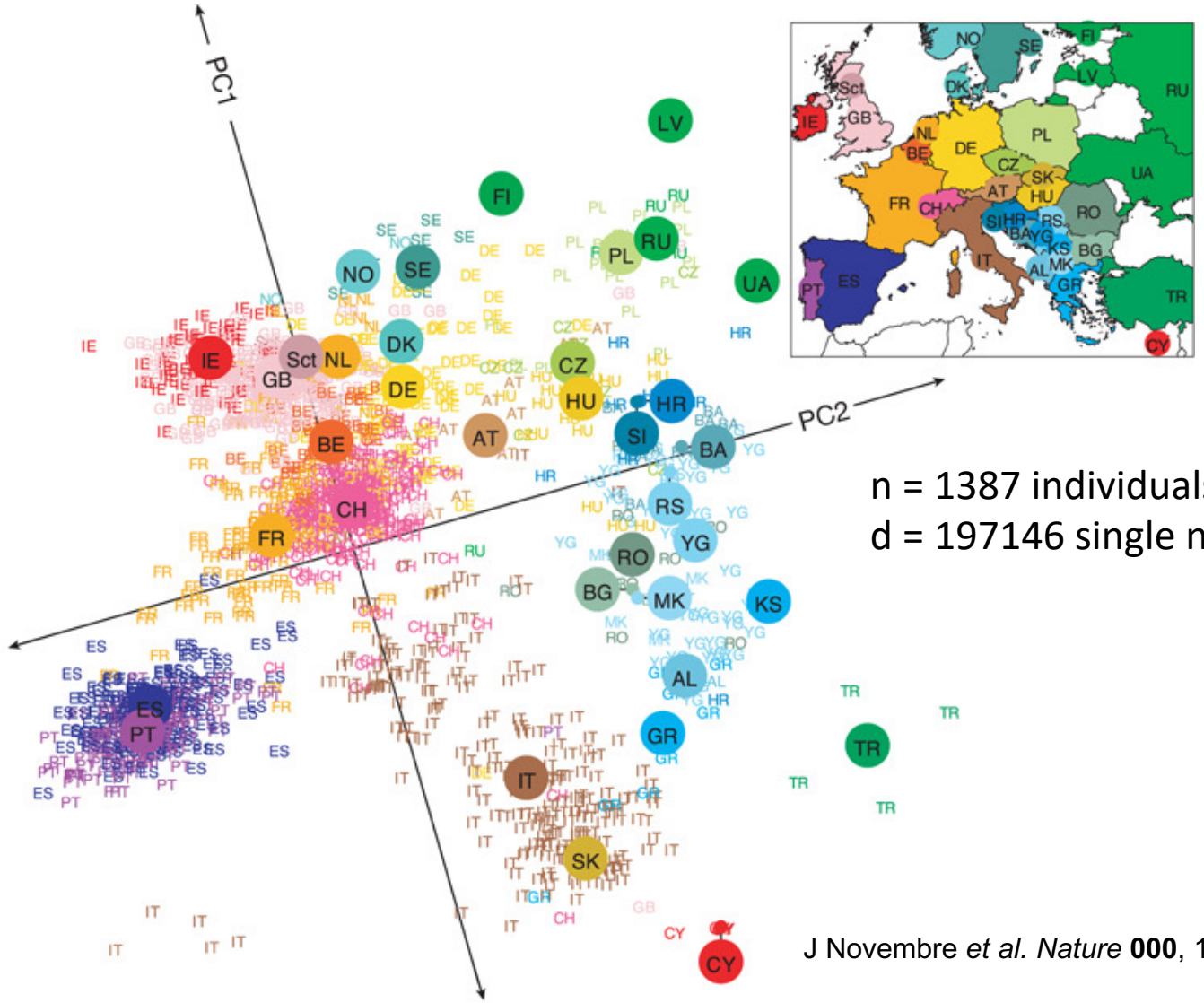
$$A = \sigma_1 \vec{u}_1 \vec{v}_1^\top + \cdots + \sigma_r \vec{u}_r \vec{v}_r^\top$$

and $\vec{u}_1, \ldots, \vec{u}_r$ are orthonormal

MNIST



Population structure within Europe



n = 1387 individuals
d = 197146 single nucleotide polymorphisms

nature

J Novembre *et al. Nature* **000**, 1-4 (2008) doi:10.1038/nature07331

2. Singular value decomposition

SVD existence theorem

SVD theorem: For any $n \times d$ matrix A of rank r, there exist

- ONB $(\vec{u}_1, \ldots, \vec{u}_r)$ for $\mathrm{CS}(A)$ (called left singular vectors of A),
- ONB $(\vec{v}_1, \ldots, \vec{v}_r)$ for $\mathrm{CS}(A^\top)$ (called right singular vectors of A),
- positive numbers $\sigma_1 \geq \cdots \geq \sigma_r > 0$ (called singular values of A)

such that

$$A = \sigma_1 \vec{u}_1 \vec{v}_1^\top + \cdots + \sigma_r \vec{u}_r \vec{v}_r^\top$$

This decomposition is called a singular value decomposition (SVD) of A

Proof: By construction, using k-BFS greedy algorithm

SVD as matrix factorization

- Compact SVD: Given SVD $A = \sigma_1 \vec{u}_1 \vec{v}_1^\top + \cdots + \sigma_r \vec{u}_r \vec{v}_r^\top$, can write

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

where

$$U = \begin{bmatrix} \uparrow & & \uparrow \\ \vec{u}_1 & \cdots & \vec{u}_r \\ \downarrow & & \downarrow \end{bmatrix}, \qquad \Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}, \qquad V = \begin{bmatrix} \uparrow & & \uparrow \\ \vec{v}_1 & \cdots & \vec{v}_r \\ \downarrow & & \downarrow \end{bmatrix}$$

- Notice that $U^\top U = I_n$, $V^\top V = I_d$, and Σ is invertible $r \times r$ diagonal matrix
- See reading for description of "Full SVD" variant
- Compact SVD of transpose is $A^\top = V \Sigma U^\top$

SVD in NumPy

`numpy.linalg.svd`

- External call to `_gesdd` from LAPACK ("Linear Algebra Package")
- Pay attention to the options (e.g., `full_matrices`)
- Time complexity: $O(nd \min\{n, d\})$

3. Covariance matrices and PCA

Multivariate data

- Data points $\vec{a}_1, \ldots, \vec{a}_n \in \mathbb{R}^d$, e.g.,

$$\vec{a}_i = (0, -2, +2, -1, 0, +1, -2, -1, -2, +2, 0, -1, +1, \ldots) \in \mathbb{R}^d$$

  - Each $\vec{a}_i$ contains d measurements (a.k.a. "variables" or "feature" values)
- It is typical to center the data: subtract the mean vector

$$\vec{\mu} = \frac{1}{n} \sum_{i=1}^{n} \vec{a}_i$$

from each data point $\vec{a}_i$

Covariance matrix

- If data points $\vec{a}_1, \ldots, \vec{a}_n \in \mathbb{R}^d$ are already centered, then their (empirical) covariance matrix is the $n \times d$ matrix

$$C = \frac{1}{n} \sum_{i=1}^{n} \vec{a}_i \vec{a}_i^{\top}$$

- The $(i, j)^{\text{th}}$ component of C is the (empirical) covariance between the i^{th} "feature" and the j^{th} "feature"
  - $C_{i,i}$ is the variance of the i^{th} "feature"
- If data points are stacked as rows in $n \times d$ matrix A, then $C = \frac{1}{n} A^{\top} A$

Principal components analysis

- Principal components analysis (PCA): linearly transform (centered) data $\vec{a}_1, \ldots, \vec{a}_n \in \mathbb{R}^d$ by an $d \times r$ matrix Q

$$\vec{b}_i = Q^\top \vec{a}_i$$

  so that "features" in new data set $\vec{b}_1, \ldots, \vec{b}_n$ are
  - Uncorrelated (zero covariance between different features)
  - Ordered so that variance is non-increasing (e.g., feature 1 has highest variance, feature 2 has next highest variance, …)
- Desired matrix Q is given by **right singular vectors** of A, with order given by corresponding singular values of A (from largest to smallest)
  - Variances are the squares of the singular values of A (divided by n)

Eigenvectors and eigenvalues

PCA usually defined by eigenvectors of covariance matrix $C = \frac{1}{n} A^\top A$

If SVD factorization of A is $A = U\Sigma V^\top$, then matrix V diagonalizes C:

$$V^\top C V = \frac{1}{n}\Sigma^2$$

- Eigenvectors of C are right singular vectors of A
- Eigenvalues of C are squared singular values of A (divided by n)
- j^{th} largest eigenvalue of C is variance of the data set in direction $\vec{v}_j$

Bivariate normal distribution

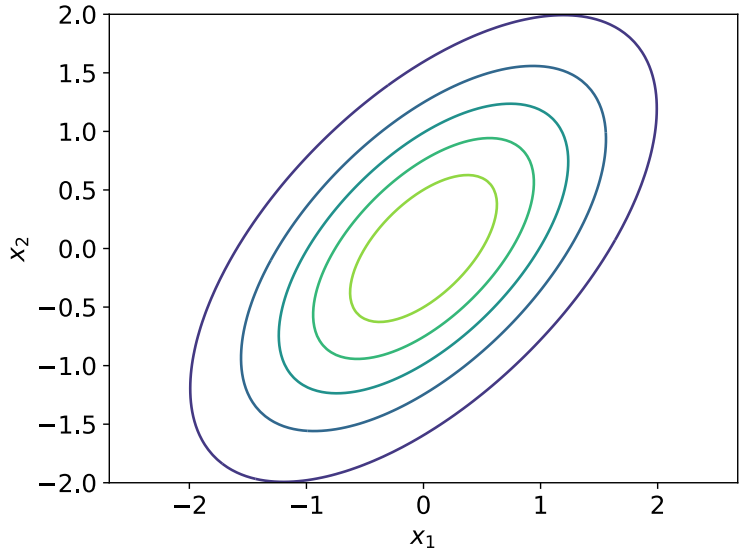
- Example: (infinitely many) data from bivariate normal distribution

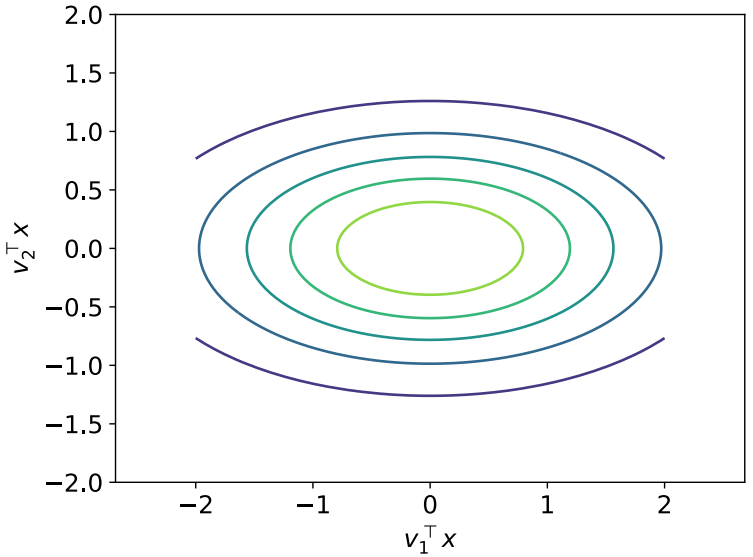
$$\mathrm{N}\left(0, \begin{bmatrix} 10/9 & 2/3 \\ 2/3 & 10/9 \end{bmatrix}\right)$$

- (Population) covariance matrix has eigenvalues/eigenvectors

$$\lambda_1 = \frac{16}{9}, \qquad \vec{v}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

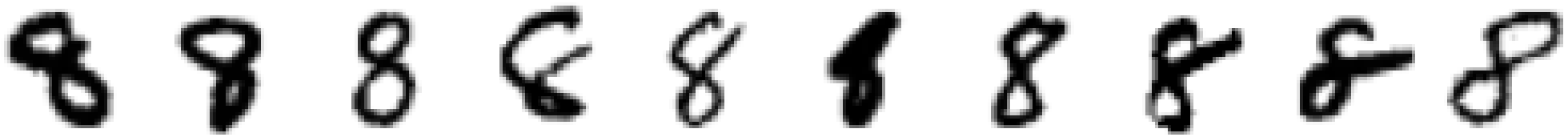
$$\lambda_2 = \frac{4}{9}, \qquad \vec{v}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$





MNIST

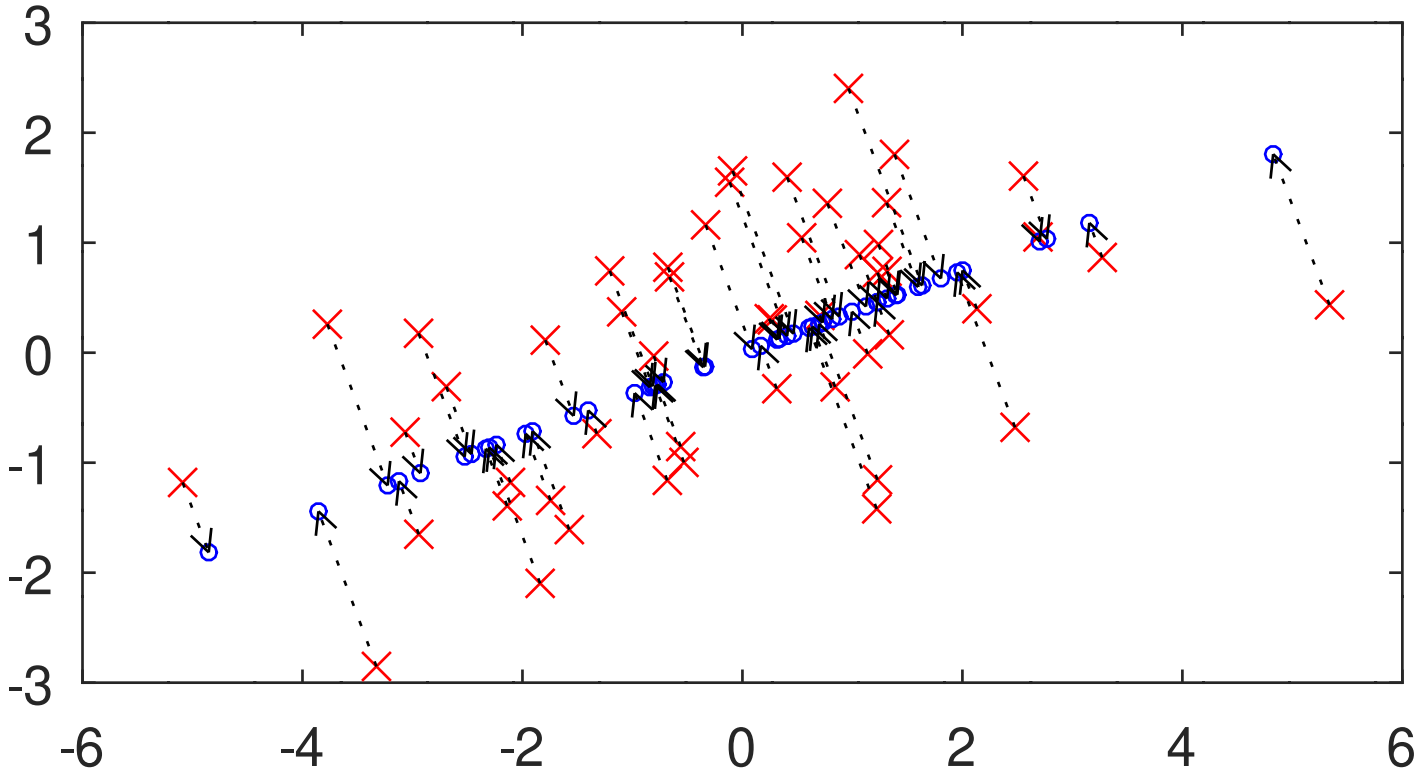
- Example: MNIST (just the 8's), $n = 6000$, $d = 784$
  - 10 images sorted by $\vec{x}^{\top}\vec{v}_1$



4. Power method

Best fitting line (★)

- Step j of k-BFS greedy algorithm: Solve "Best Fitting Line" problem restricted to orthogonal complement of $S_{j-1} = \text{span}(\vec{v}_1, \ldots, \vec{v}_{j-1})$

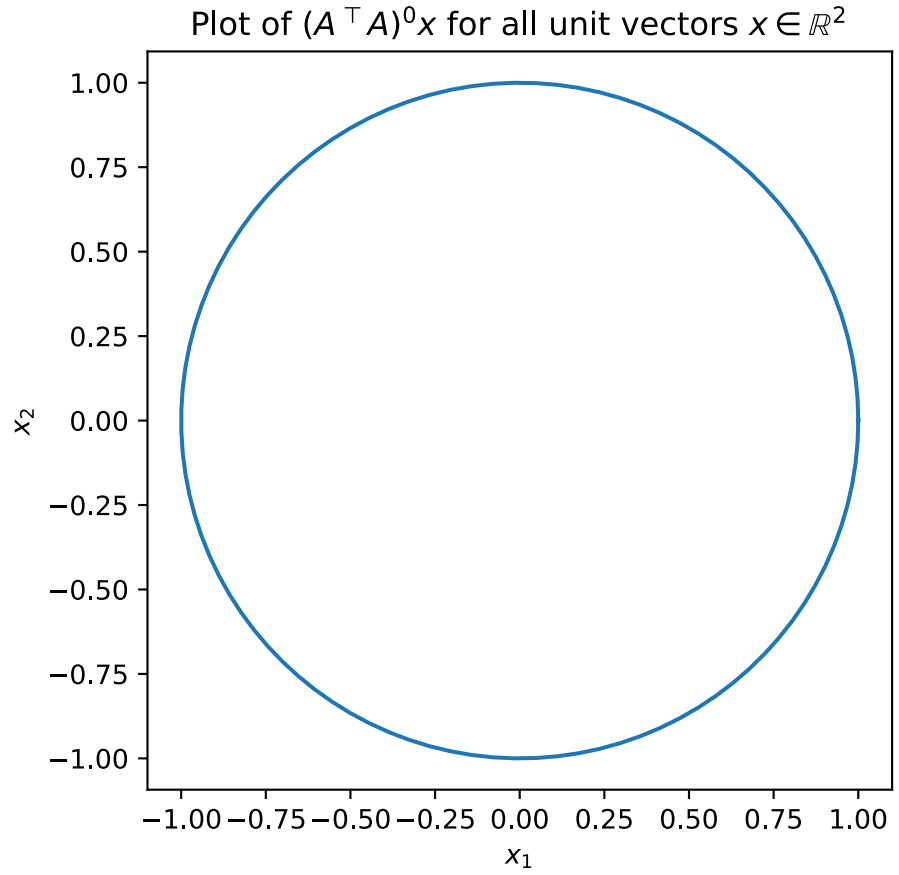


Powers of a positive semidefinite matrix

- If SVD factorization of A is $A = U\Sigma V^\top$, then V diagonalizes $A^\top A$:
$$V^\top (A^\top A) V = \Sigma^2$$
- V also diagonalizes $(A^\top A)^2$:
$$V^\top (A^\top A)^2 V = \Sigma^4$$
- V also diagonalizes $(A^\top A)^3$:
$$V^\top (A^\top A)^3 V = \Sigma^6$$
- …
- So, for any t:
$$(A^\top A)^t = \sigma_1^{2t} \vec{v}_1 \vec{v}_1^\top + \sigma_2^{2t} \vec{v}_2 \vec{v}_2^\top + \cdots + \sigma_r^{2t} \vec{v}_r \vec{v}_r^\top$$

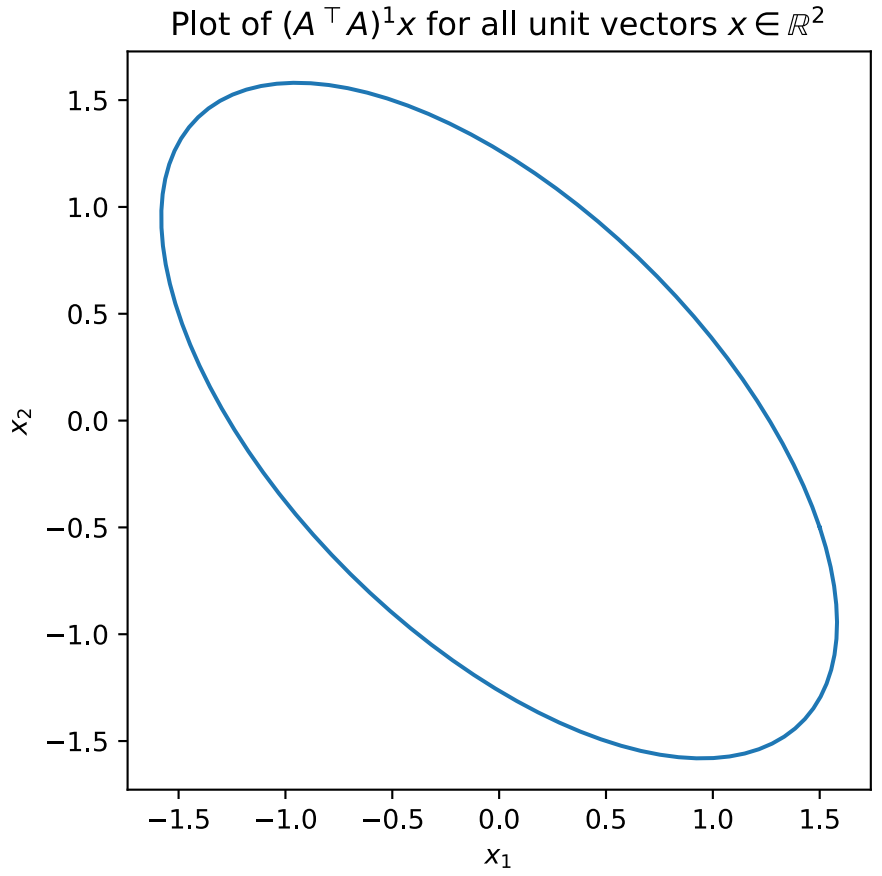
Powers of $A^\top A$

$$A = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1 & -1 \end{bmatrix}$$



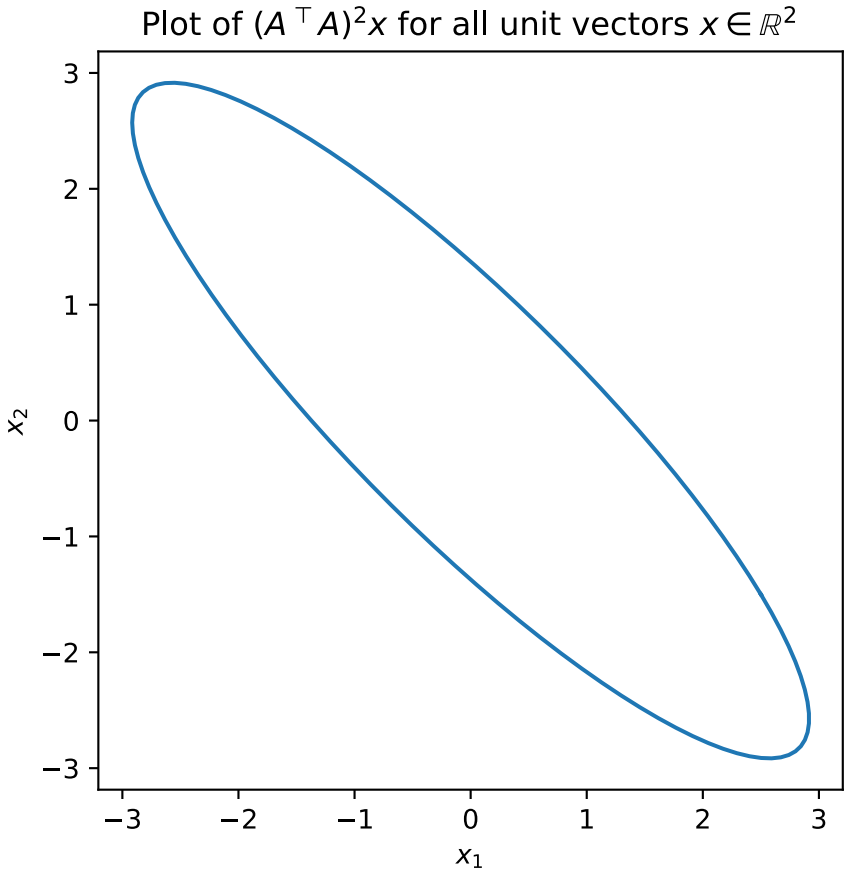
Powers of $A^\top A$

$$A = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1 & -1 \end{bmatrix}$$



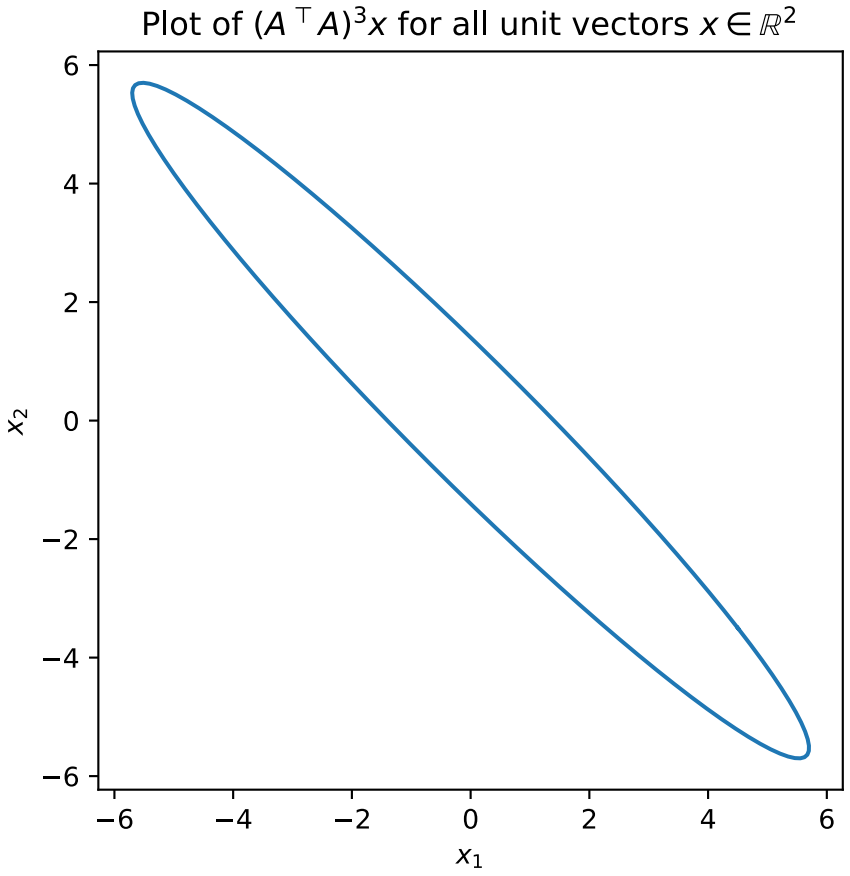
Powers of $A^\top A$

$$A = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1 & -1 \end{bmatrix}$$



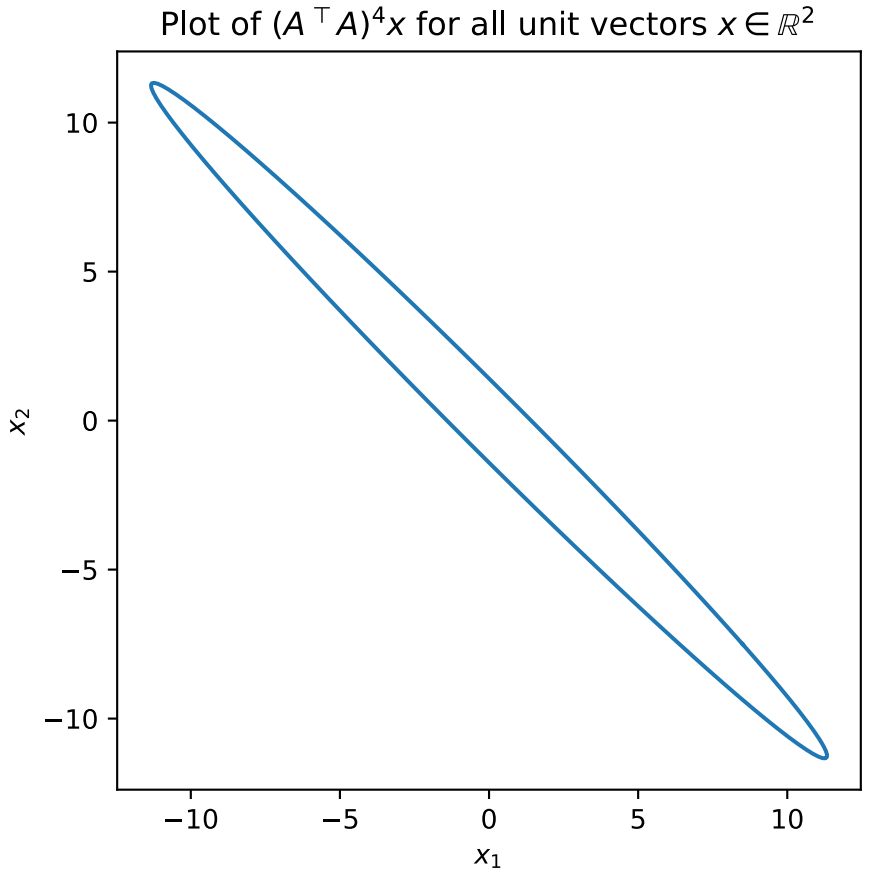
Powers of $A^\top A$

$$A = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1 & -1 \end{bmatrix}$$



Powers of $A^\top A$

$$A = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1 & -1 \end{bmatrix}$$



Multiplying powers of $A^\top A$ by a vector

- For any t:

$$(A^\top A)^t = \sigma_1^{2t}\left(\left(\frac{\sigma_1}{\sigma_1}\right)^{2t}\vec{v}_1\vec{v}_1^\top + \left(\frac{\sigma_2}{\sigma_1}\right)^{2t}\vec{v}_2\vec{v}_2^\top + \cdots + \left(\frac{\sigma_r}{\sigma_1}\right)^{2t}\vec{v}_r\vec{v}_r^\top\right)$$

- If $\sigma_1 > \sigma_2$, then for almost all unit vectors $\vec{x} \in \mathbb{R}^d$ and large enough t,

$$(A^\top A)^t\vec{x} \approx \sigma_1^{2t}\vec{v}_1$$

  - If $\sigma_1 = \cdots = \sigma_p > \sigma_{p+1}$, then get vector (approximately) in span of $\vec{v}_1, \ldots, \vec{v}_p$ (all such vectors are equally good!)

- Also: can compute $(A^\top A)^t\vec{x}$ iteratively

Power method

- **Input**: $n \times d$ matrix A
- Randomly pick a d-dimensional unit vector $\vec{x}$
- **For** $j = 1,2,\ldots,t$ **do**
  - $\vec{x} = \frac{1}{\|(A^\top A)\vec{x}\|}(A^\top A)\vec{x}$
- **Return** $\vec{x}$

Use in k-BFS greedy algorithm

- Step j of k-BFS greedy algorithm: Solve "Best Fitting Line" problem restricted to orthogonal complement of $S_{j-1} = \text{span}(\vec{v}_1, \ldots, \vec{v}_{j-1})$
- Q: How to ensure we restrict to $S_{j-1}^{\perp}$?
- A: Run power method on data $P_W \vec{a}_1, \ldots, P_W \vec{a}_n$ for $W = S_{j-1}^{\perp}$
  - Then every $\vec{v} \in S_{j-1}$ has $\langle \vec{v}, P_W \vec{a}_i \rangle = 0$ for all i

Use in implementations

- LAPACK implementation of SVD does not use power method
- However, main idea of power method is used in some large-scale algorithms needed when, e.g., cannot even load A into memory

5. Low-rank approximations

Low-rank matrix approximation

- **Problem**: Given $n \times d$ matrix A and non-negative integer k, find a $n \times d$ matrix B of rank k so that

$$\sum_{i=1}^{n}\sum_{j=1}^{d}\left(A_{i,j} - B_{i,j}\right)^2$$

  is as small as possible.
- **Motivation**: Rank k matrix can be represented using $(n + d)k$ numbers (whereas a $n \times d$ matrix requires nd numbers)
  - Space savings if $k < nd/(n + d)$

Truncated SVD

- Rank-k truncated SVD of $n \times d$ matrix A of rank r (for $k \leq r$):

$$\hat{A} = \sigma_1 \vec{u}_1 \vec{v}_1^\top + \cdots + \sigma_k \vec{u}_k \vec{v}_k^\top$$

  - Drop the $r - k$ terms in SVD corresponding to $r - k$ smallest singular values
  - Matrix $\hat{A}$ also called the rank-k SVD approximation of A

Eckart-Young Theorem

- **Eckart-Young Theorem**: For any $n \times d$ matrix A of rank r, any $k \leq r$, and any $n \times d$ matrix B of rank k,

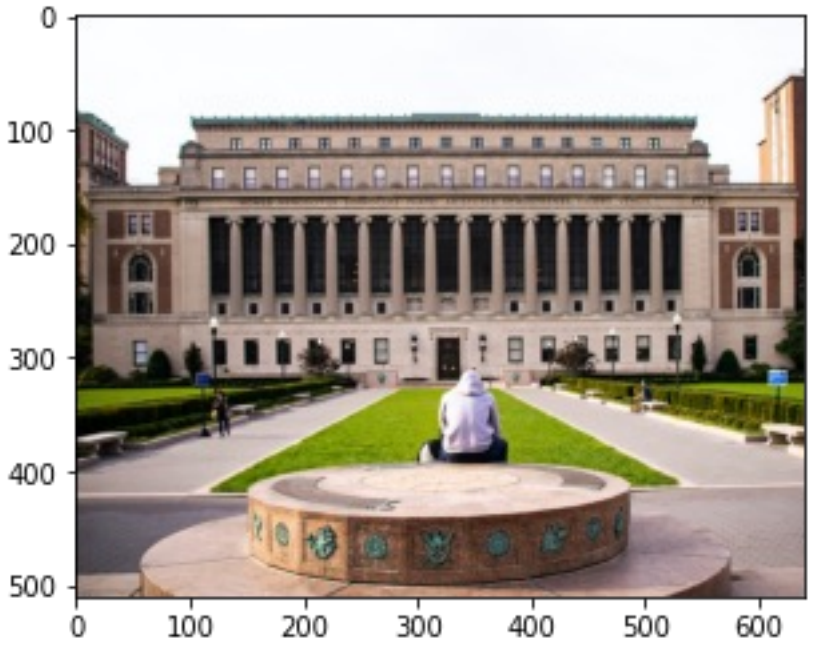
$$\sum_{i=1}^{n}\sum_{j=1}^{d}\left(A_{i,j} - B_{i,j}\right)^2 \geq \sum_{i=1}^{n}\sum_{j=1}^{d}\left(A_{i,j} - \hat{A}_{i,j}\right)^2 = \sigma_{k+1}^2 + \cdots + \sigma_r^2$$

  where $\hat{A}$ is rank-k SVD approximation of A, and $\sigma_1 \geq \cdots \geq \sigma_r$ are the singular values of A

- This is a corollary of the k-BFS greedy algorithm's guarantees

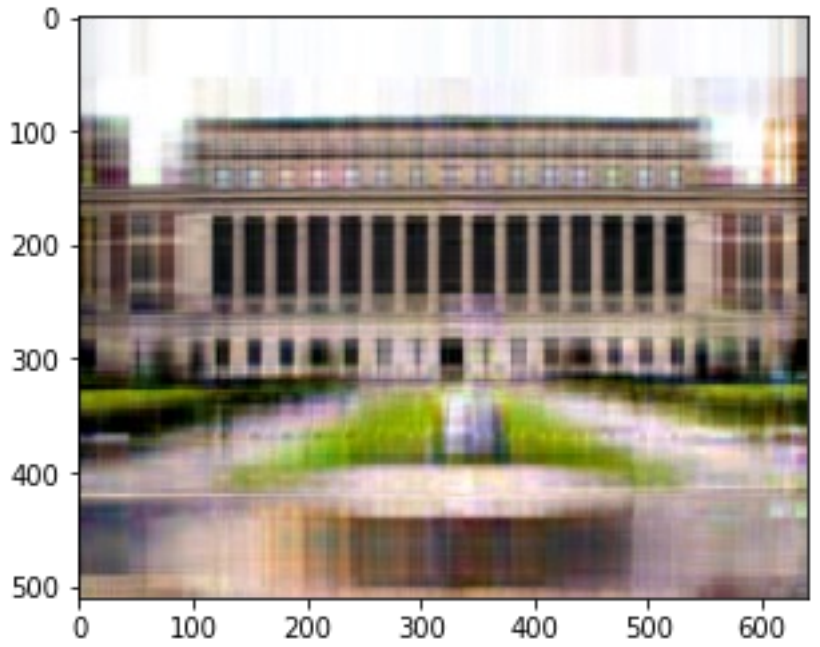
Example: image compression

510x640 pixel image: 3 separate 510×640 matrices, one per RGB color



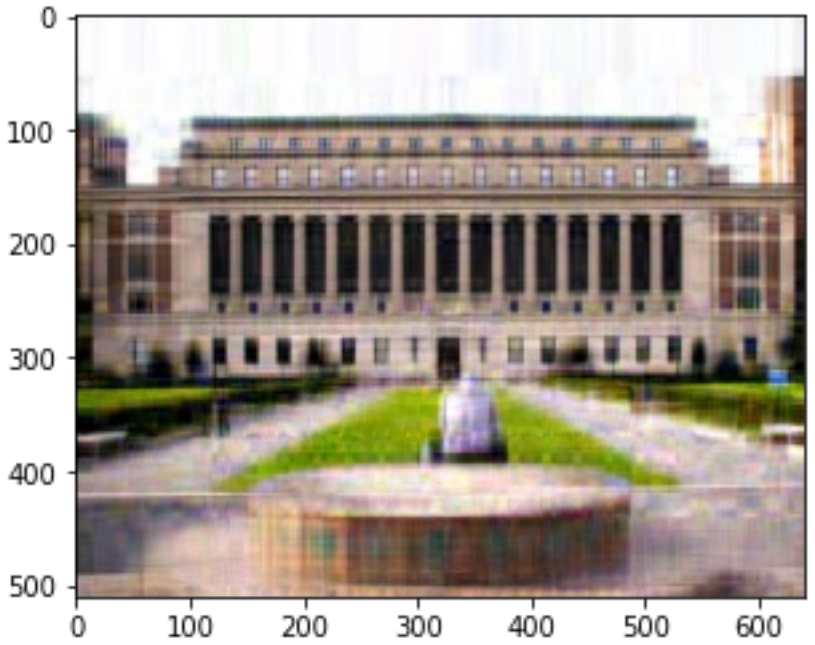
Rank 8 SVD approximations

- Replace each matrix by rank 8 SVD approximation



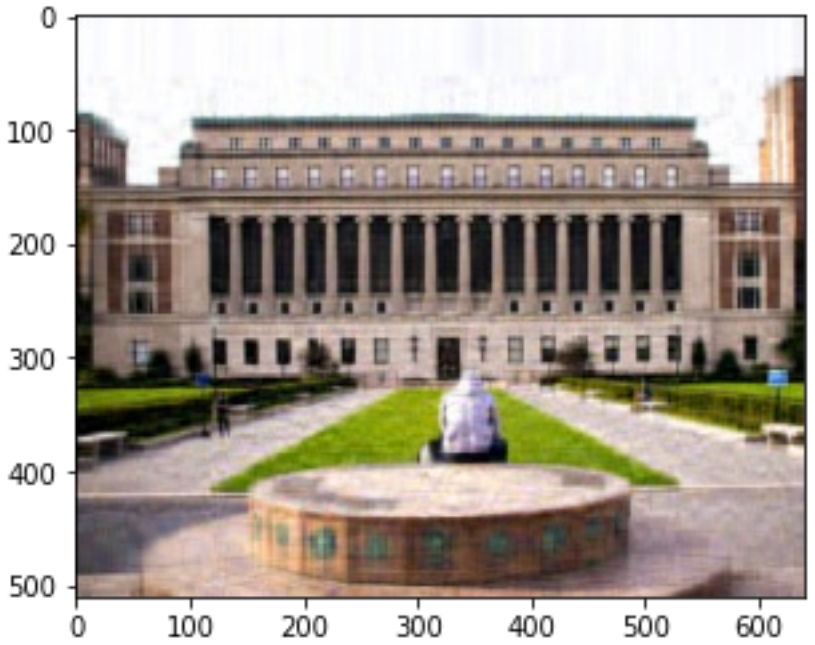
Rank 16 SVD approximations

- Replace each matrix by rank 16 SVD approximation



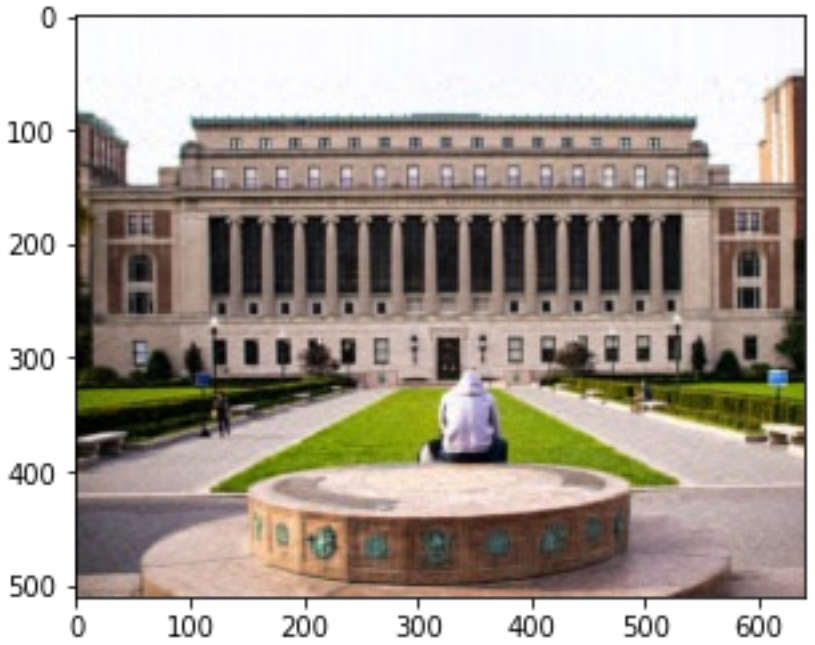
Rank 32 SVD approximations

- Replace each matrix by rank 32 SVD approximation



Rank 64 SVD approximations

- Replace each matrix by rank 64 SVD approximation



Statistical application

- Consider the following statistical model: A is $n \times d$ matrix of independent random variables, where

$$A_{i,j} \sim \mathrm{N}\left(H_{i,j}, \sigma^2\right)$$

  - Here, H is a $n \times d$ matrix of rank k, regarded as the model parameter
- Maximum likelihood estimator for H: rank-k truncated SVD of A
- **Intuitive interpretation**: truncated SVD attempts to remove the noise

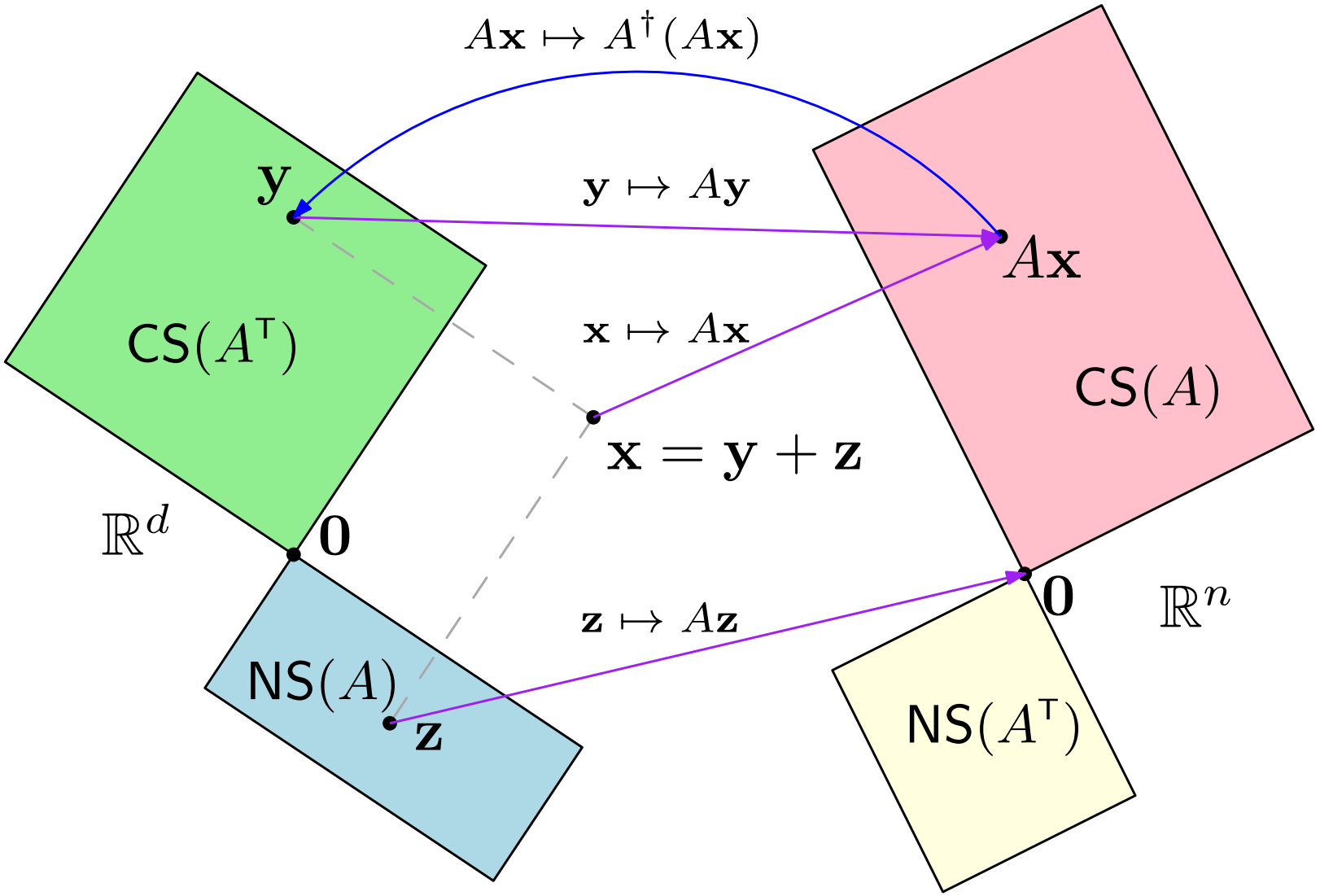
6. Moore-Penrose pseudoinverse

Moore-Penrose pseudoinverse

- For $A = \sigma_1 \vec{u}_1 \vec{v}_1^\top + \cdots + \sigma_r \vec{u}_r \vec{v}_r^\top$, Moore-Penrose pseudoinverse is

$$A^\dagger = V \Sigma^{-1} U^\top$$

- Useful fact #1: Fundamental subspaces of $A^\dagger$ are same as that of $A^\top$
- Useful fact #2: When restricting attention to $\mathrm{CS}(A^\top)$ and $\mathrm{CS}(A)$, the linear maps A and $A^\dagger$ are inverses of each other (in the usual sense)


$A\mathbf{x} \mapsto A^\dagger(A\mathbf{x})$
$\mathbf{y}$
$\mathbf{y} \mapsto A\mathbf{y}$
$A\mathbf{x}$
$\mathbf{x} \mapsto A\mathbf{x}$
$\mathsf{CS}(A^\mathsf{T})$
$\mathsf{CS}(A)$
$\mathbf{x} = \mathbf{y} + \mathbf{z}$
$\mathbb{R}^d$
$\mathbf{0}$
$\mathbf{0}$
$\mathbb{R}^n$
$\mathbf{z} \mapsto A\mathbf{z}$
$\mathsf{NS}(A)$
$\mathbf{z}$
$\mathsf{NS}(A^\mathsf{T})$

Relation to orthogonal projections

- Useful fact #3:

$$AA^{\dagger} = U\Sigma V^{\top} V\Sigma^{-1} U^{\top} = UU^{\top} = P_{\mathrm{CS}(A)}$$

- How to think about this:
  - Write $b \in \mathbb{R}^n$ as $b = b_{\|} + b_{\perp}$ for $b_{\|} \in \mathrm{CS}(A)$ and $b_{\perp} \in \mathrm{NS}(A^{\top})$
  - Recall $\mathrm{NS}(A^{\top}) = \mathrm{NS}(A^{\dagger})$
  - Then $A^{\dagger}b = A^{\dagger}\left(b_{\|} + b_{\perp}\right) = A^{\dagger}b_{\|}$, so $A^{\dagger}b$ is unique vector $y \in \mathrm{CS}(A^{\top})$ such that $Ay = b_{\|}$

Application: solving normal equations

- Normal equations:

$$A^\top A w = A^\top b$$

- Solution in row space of A:

$$\hat{w} = A^\dagger b$$

- Why is $\hat{w}$ a solution?

$$A\hat{w} = AA^\dagger b = P_{\mathrm{CS}(A)} b$$

- Fact: If $w \neq \hat{w}$ also solves the normal equations, then $\|w\| > \|\hat{w}\|$
  - Can always make w shorter by removing the part of w in $\mathrm{NS}(A)$
  - There is only one solution to $Aw = P_{\mathrm{CS}(A)} b$ in $\mathrm{CS}(A^\top)$

7. Latent semantic analysis

Document-term matrix

- n documents in a corpus
- d words in vocabulary
- Create $n \times d$ matrix A of word counts per document

	aardvark	abacus	abalone	⋯
Document 1	3	0	0	⋯
Document 2	7	0	4	⋯
Document 3	2	4	0	⋯
⋮	⋮	⋮	⋮	⋱

Latent semantic analysis (LSA)

- Rank-k truncated SVD of A:

$$\hat{A} = \hat{U}\hat{\Sigma}\hat{V}^\top$$

where

$$\hat{U} = \begin{bmatrix} \uparrow & & \uparrow \\ \vec{u}_1 & \cdots & \vec{u}_k \\ \downarrow & & \downarrow \end{bmatrix}, \quad \hat{\Sigma} = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix}, \quad \hat{V} = \begin{bmatrix} \uparrow & & \uparrow \\ \vec{v}_1 & \cdots & \vec{v}_k \\ \downarrow & & \downarrow \end{bmatrix}$$

- Use i^{th} row of $\hat{U} \in \mathbb{R}^{n \times k}$ as representation of i^{th} document
- Use j^{th} column of $\hat{\Sigma}\hat{V}^\top \in \mathbb{R}^{k \times d}$ as representation of j^{th} vocabulary word

Topic modeling with LSA

- Each row of A is approximated by a linear combination of the k rows of $\hat{\Sigma}\hat{V}^\top \in \mathbb{R}^{k \times d}$
  - Interpret each row of $\hat{\Sigma}\hat{V}^\top$ as a representation of a "topic"
  - i^{th} row of $\hat{U}$: "weights" that i^{th} document puts each of the k topics
- Alternatives to LSA (e.g., "Latent Dirichlet Allocation"):
  - Also give low-rank approximations to A
  - But also have probabilistic interpretations

Word embeddings

- **Word embeddings**: vector representations of vocabulary words
- **Default ("one-hot") word embeddings**:
  - i^{th} vocabulary word: $\vec{e}_i = (0, \dots, 0, \overbrace{1}^{i^{\text{th}}\ \text{position}}, 0, \dots, 0)$
  - Geometry: all words are orthogonal to each other
- **Word embeddings from LSA**:
  - Geometry: reflects co-occurrence statistics in documents
  - Hope that truncated SVD removes corpus-specific "noise"
- **Using word embeddings in machine learning**:
  - Get word embeddings (by LSA or other means) from a large corpus of documents (not just the training data for "downstream" application)
  - Represent words using the embeddings in data for downstream application