భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad
॥ नवोन्मेषेण देश निर्माणम् ॥

Matrix Theory

23 Truncated SVD and PCA

Dr. Konda Reddy Mopuri
Dept. of AI, IIT Hyderabad
July-Nov 2026

Eckart–Young–Mirsky

For any $k < r$,

$$\min_{\operatorname{rank} B \leq k} \|A - B\| \;=\; \|A - A_k\|\,,$$

for the spectral norm (Eckart–Young, 1936) and indeed for every unitarily invariant norm, including Frobenius (Mirsky, 1960).

Eckart–Young–Mirsky

For any $k < r$,

$$\min_{\operatorname{rank} B \le k} \|A - B\| \;=\; \|A - A_k\|\,,$$

for the spectral norm (Eckart–Young, 1936) and indeed for every unitarily invariant norm, including Frobenius (Mirsky, 1960).

Why this matters. Truncating the SVD is not merely *a* good low-rank approximation — it is *the best possible one*. No cleverer rank-k matrix exists. Among all the compression schemes you could invent that output a rank-k matrix, this one wins, and you get it by sorting.

Eckart–Young–Mirsky

For any $k < r$,

$$\min_{\operatorname{rank} B \le k} \|A - B\| \;=\; \|A - A_k\|\,,$$

for the spectral norm (Eckart–Young, 1936) and indeed for every unitarily invariant norm, including Frobenius (Mirsky, 1960).

Proof sketch (spectral norm). If $\operatorname{rank} B \le k$ then $\dim \operatorname{Null}(B) \ge n - k$, so $\operatorname{Null}(B)$ meets the $(k+1)$-dimensional space $\operatorname{span}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k+1}\}$ in some unit vector $\boldsymbol{w}$. Then $\|A - B\|_2^2 \ge \|(A - B)\boldsymbol{w}\|^2 = \|A\boldsymbol{w}\|^2 = \sum_{j \le k+1} \sigma_j^2 (\boldsymbol{v}_j^{\mathsf{T}} \boldsymbol{w})^2 \ge \sigma_{k+1}^2$.

What “low rank” buys us

Storing a general $m \times n$ matrix costs mn numbers. Storing A_k costs

$$\underbrace{km}_{U_k} + \underbrace{k}_{\sigma\text{'s}} + \underbrace{kn}_{V_k} = k(m+n+1) \text{ numbers.}$$

What "low rank" buys us

Storing a general $m \times n$ matrix costs mn numbers. Storing A_k costs

$$\underbrace{km}_{U_k} + \underbrace{k}_{\sigma\text{'s}} + \underbrace{kn}_{V_k} = k(m+n+1) \text{ numbers.}$$

Compression ratio

$$\rho = \frac{k(m+n+1)}{mn}.$$

Multiplying by a vector drops from $O(mn)$ to $O(k(m+n))$ flops as well:

$$A_k \boldsymbol{x} = U_k \Sigma_k V_k^T \boldsymbol{x} = \sum_{j=1}^{k} \sigma_j \, \boldsymbol{u}_j \, (\boldsymbol{v}_j^{\mathsf{T}} \boldsymbol{x}).$$

Setting it up

A grayscale image is literally a matrix: $A \in \mathbb{R}^{m \times n}$, $a_{ij} \in [0, 255]$ the intensity of pixel (i, j).

Setting it up

A grayscale image is literally a matrix: $A \in \mathbb{R}^{m \times n}$, $a_{ij} \in [0, 255]$ the intensity of pixel (i, j).

- No statistics, no learning. Just linear algebra on the pixel array.
- **Why does it work at all?** Natural images are highly redundant: neighbouring rows look like one another. So A is close to a low-rank matrix even though its exact rank is usually full.
- Equivalently: the singular values decay fast.

Setting it up

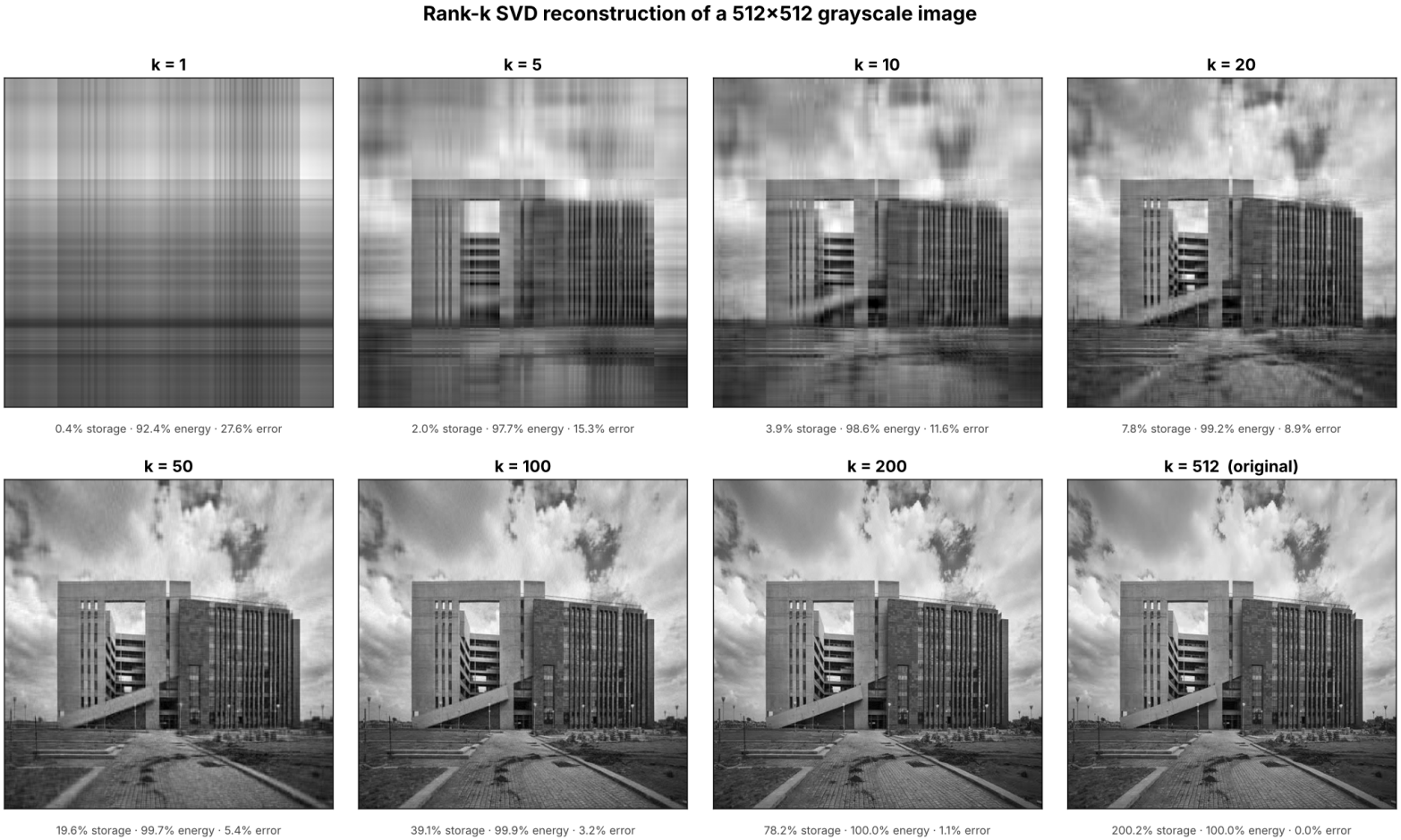
A grayscale image is literally a matrix: $A \in \mathbb{R}^{m \times n}$, $a_{ij} \in [0, 255]$ the intensity of pixel (i, j).

- No statistics, no learning. Just linear algebra on the pixel array.
- **Why does it work at all?** Natural images are highly redundant: neighbouring rows look like one another. So A is close to a low-rank matrix even though its exact rank is usually full.
- Equivalently: the singular values decay fast.

Recipe.

1. Compute $A = U\Sigma V^{\mathsf{T}}$.
2. Choose k; form $A_k = U_k \Sigma_k V_k^{\mathsf{T}}$.
3. Clip to $[0, 255]$ and display.

Example 512×512 image



Numbers for a 512×512 image

Full storage: $512 \times 512 = 262{,}144$ values.

k	values stored $k(m+n+1)$	ratio ρ	visual impression
5	5,125	2.0%	blobs; only gross layout
20	20,500	7.8%	recognisable, heavy ringing
50	51,250	19.6%	good; minor artefacts
100	102,500	39.1%	near-indistinguishable
200	205,000	78.2%	indistinguishable

Numbers for a 512×512 image

Full storage: $512 \times 512 = 262{,}144$ values.

k	values stored $k(m+n+1)$	ratio ρ	visual impression
5	5,125	2.0%	blobs; only gross layout
20	20,500	7.8%	recognisable, heavy ringing
50	51,250	19.6%	good; minor artefacts
100	102,500	39.1%	near-indistinguishable
200	205,000	78.2%	indistinguishable

Break-even

$k(m+n+1) < mn \iff k < \dfrac{mn}{m+n+1}$. For 512×512 that is $k < 255.7$. Beyond that, you are *inflating* the file.

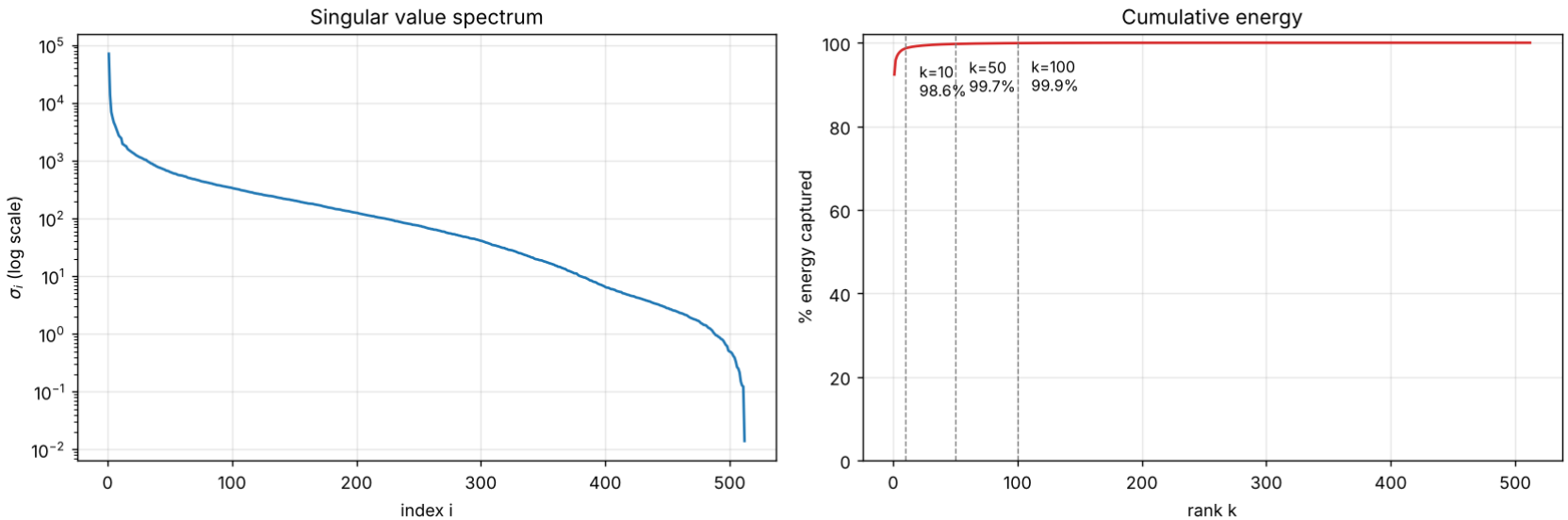
How much energy did we keep?

Define the *captured energy*

$$E_k = \frac{\sum_{j \le k} \sigma_j^2}{\sum_{j \le r} \sigma_j^2} = 1 - \frac{\|A - A_k\|_F^2}{\|A\|_F^2}.$$

A common rule of thumb: pick the smallest k with $E_k \ge 0.95$ or 0.99. Note that energy is a *proxy* — the eye is not a Frobenius norm.

Singular value decay and Energy



Comparison with JPEG

SVD compression is a good example to understand the k-rank approximation. It is not what our camera does.

	Truncated SVD	JPEG
basis	adapted to *this* image	fixed (DCT)
must store basis?	yes (U_k, V_k)	no
cost	$O(mn \min(m,n))$	$O(mn)$
quantisation	none	perceptually weighted
entropy coding	none	Huffman/arithmetic
typical ratio at good quality	$\sim 5\times$	~ 10–$20\times$

Observation

The SVD's optimality is with respect to the *Frobenius norm*, not human perception, and it pays the price of shipping its own basis.

Where truncated SVD genuinely is the right tool

- **Noise removal.** If signal is low-rank and noise is full-rank with small σ, truncation is denoising.
- **Matrix completion / recommender systems.** Netflix-style user $\times$ item matrices are modelled as approximately low rank.
- **Latent semantic analysis.** Truncate a term $\times$ document matrix; documents sharing no words can end up close.
- **Model order reduction.** Replace a huge dynamical system with a projection onto the leading k singular directions.
- **Background subtraction in video.** Static background $=$ low rank, moving objects $=$ sparse.

In all of these the point is not saving bytes; it is that the leading singular subspace *is* the signal.

The question PCA answers

Data: n observations of p variables, stored in $X \in \mathbb{R}^{n \times p}$ (rows = samples).

Goal

Find a small number of directions in $\mathbb{R}^p$ along which the data varies the most, and describe each sample by its coordinates in those few directions.

The question PCA answers

Data: n observations of p variables, stored in $X \in \mathbb{R}^{n \times p}$ (rows = samples).

Goal

Find a small number of directions in $\mathbb{R}^p$ along which the data varies the most, and describe each sample by its coordinates in those few directions.

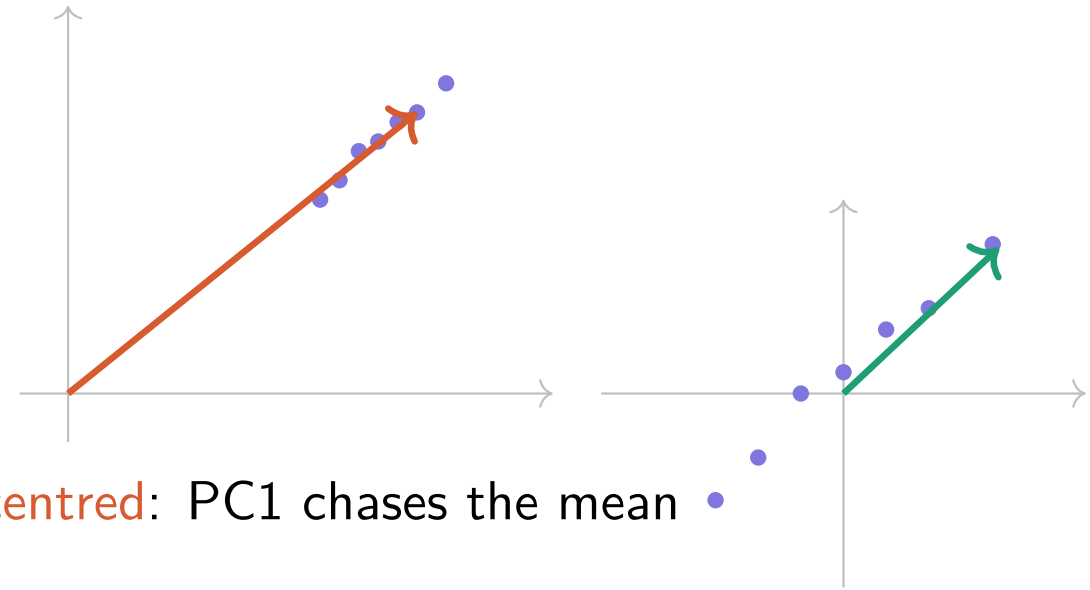
Reasons to want this:

- **Visualisation:** project 50 variables down to 2 for a scatter plot.
- **Decorrelation:** produce uncorrelated features.
- **Compression / denoising:** same low-rank logic as images.
- **Conditioning:** drop near-collinear directions before regression.

Step 0: centre the data (non-negotiable)

Let $\bar{\boldsymbol{x}} = \frac{1}{n}\sum_i \boldsymbol{x}_i$ be the column mean vector, and set

$$\tilde{X} = X - \mathbf{1}\bar{\boldsymbol{x}}^{\mathsf{T}}.$$



uncentred: PC1 chases the mean

centred: PC1 is the direction of spread

Without centring, the first "component" mostly encodes where the cloud sits, not how it is shaped.

Step 1: the optimisation problem

Find the unit vector $\boldsymbol{w} \in \mathbb{R}^p$ maximising the variance of the projected data:

$$\max_{\|\boldsymbol{w}\|=1} \frac{1}{n-1} \sum_{i=1}^{n} \left(\tilde{\boldsymbol{x}}_i^{\mathsf{T}} \boldsymbol{w}\right)^2 = \max_{\|\boldsymbol{w}\|=1} \frac{1}{n-1} \left\|\tilde{X}\boldsymbol{w}\right\|^2 = \max_{\|\boldsymbol{w}\|=1} \boldsymbol{w}^{\mathsf{T}} S\, \boldsymbol{w}$$

where

$$S = \frac{1}{n-1} \tilde{X}^{\mathsf{T}} \tilde{X} \in \mathbb{R}^{p \times p}$$

is the sample covariance matrix — symmetric and positive semidefinite.

Step 1: the optimisation problem

Find the unit vector $\boldsymbol{w} \in \mathbb{R}^p$ maximising the variance of the projected data:

$$\max_{\|\boldsymbol{w}\|=1} \frac{1}{n-1} \sum_{i=1}^{n} \left(\tilde{\boldsymbol{x}}_i^{\mathsf{T}} \boldsymbol{w}\right)^2 = \max_{\|\boldsymbol{w}\|=1} \frac{1}{n-1} \left\|\tilde{X}\boldsymbol{w}\right\|^2 = \max_{\|\boldsymbol{w}\|=1} \boldsymbol{w}^{\mathsf{T}} S\, \boldsymbol{w}$$

where

$$S = \frac{1}{n-1} \tilde{X}^{\mathsf{T}} \tilde{X} \in \mathbb{R}^{p \times p}$$

is the sample covariance matrix — symmetric and positive semidefinite.

Find the next best directions

$\max_{\boldsymbol{w} \neq \boldsymbol{0}} \frac{\boldsymbol{w}^{\mathsf{T}} S \boldsymbol{w}}{\boldsymbol{w}^{\mathsf{T}} \boldsymbol{w}} = \lambda_{\max}(S)$, attained at the top eigenvector. Successive maxima over directions orthogonal to the previous ones give $\lambda_2, \lambda_3, \ldots$.

Step 1: the optimisation problem

Find the unit vector $\boldsymbol{w} \in \mathbb{R}^p$ maximising the variance of the projected data:

$$\max_{\|\boldsymbol{w}\|=1} \frac{1}{n-1} \sum_{i=1}^{n} \left(\tilde{\boldsymbol{x}}_i^{\mathsf{T}} \boldsymbol{w}\right)^2 = \max_{\|\boldsymbol{w}\|=1} \frac{1}{n-1} \left\|\tilde{X}\boldsymbol{w}\right\|^2 = \max_{\|\boldsymbol{w}\|=1} \boldsymbol{w}^{\mathsf{T}} S\, \boldsymbol{w}$$

where

$$S = \frac{1}{n-1} \tilde{X}^{\mathsf{T}} \tilde{X} \in \mathbb{R}^{p \times p}$$

is the sample covariance matrix — symmetric and positive semidefinite.

Find the next best directions

$\max_{\boldsymbol{w} \neq \boldsymbol{0}} \frac{\boldsymbol{w}^{\mathsf{T}} S \boldsymbol{w}}{\boldsymbol{w}^{\mathsf{T}} \boldsymbol{w}} = \lambda_{\max}(S)$, attained at the top eigenvector. Successive maxima over directions orthogonal to the previous ones give $\lambda_2, \lambda_3, \ldots$.

So, the principal components are the eigenvectors of the covariance matrix.

Step 2: do it with the SVD instead

Take the SVD of the *centred* data matrix:

$$\tilde{X} = U\Sigma V^{\mathsf{T}}.$$

Then $S = \frac{1}{n-1}\tilde{X}^{\mathsf{T}}\tilde{X} = \frac{1}{n-1}V\Sigma^2 V^{\mathsf{T}}$, so the eigenvectors of S are the columns of V.

PCA object	SVD object
principal directions (loadings)	columns of V
variance along PC_k	$\lambda_k = \dfrac{\sigma_k^2}{n-1}$
scores (coordinates of samples)	$Z = \tilde{X}V = U\Sigma$
rank-k reconstruction	$\tilde{X}_k = U_k\Sigma_k V_k^{\mathsf{T}}$

Use the SVD of $\tilde{X}$, not the eigen decomposition of $\tilde{X}^{\mathsf{T}}\tilde{X}$. Forming the covariance matrix squares the condition number.

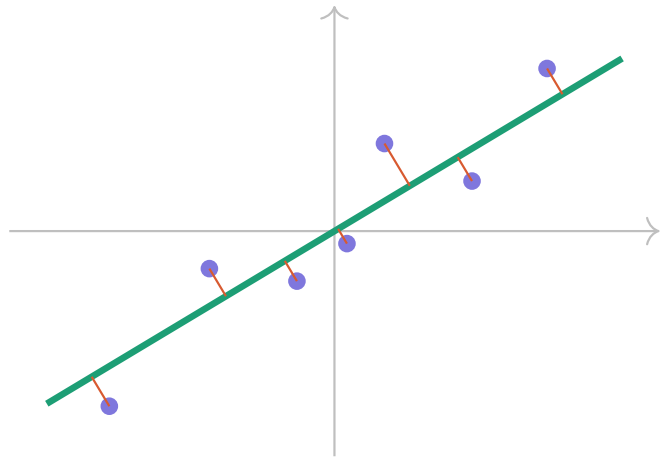
Two equivalent characterisations

Maximum variance

PC_1 is the direction in which the projected data is most spread out.

Minimum reconstruction error

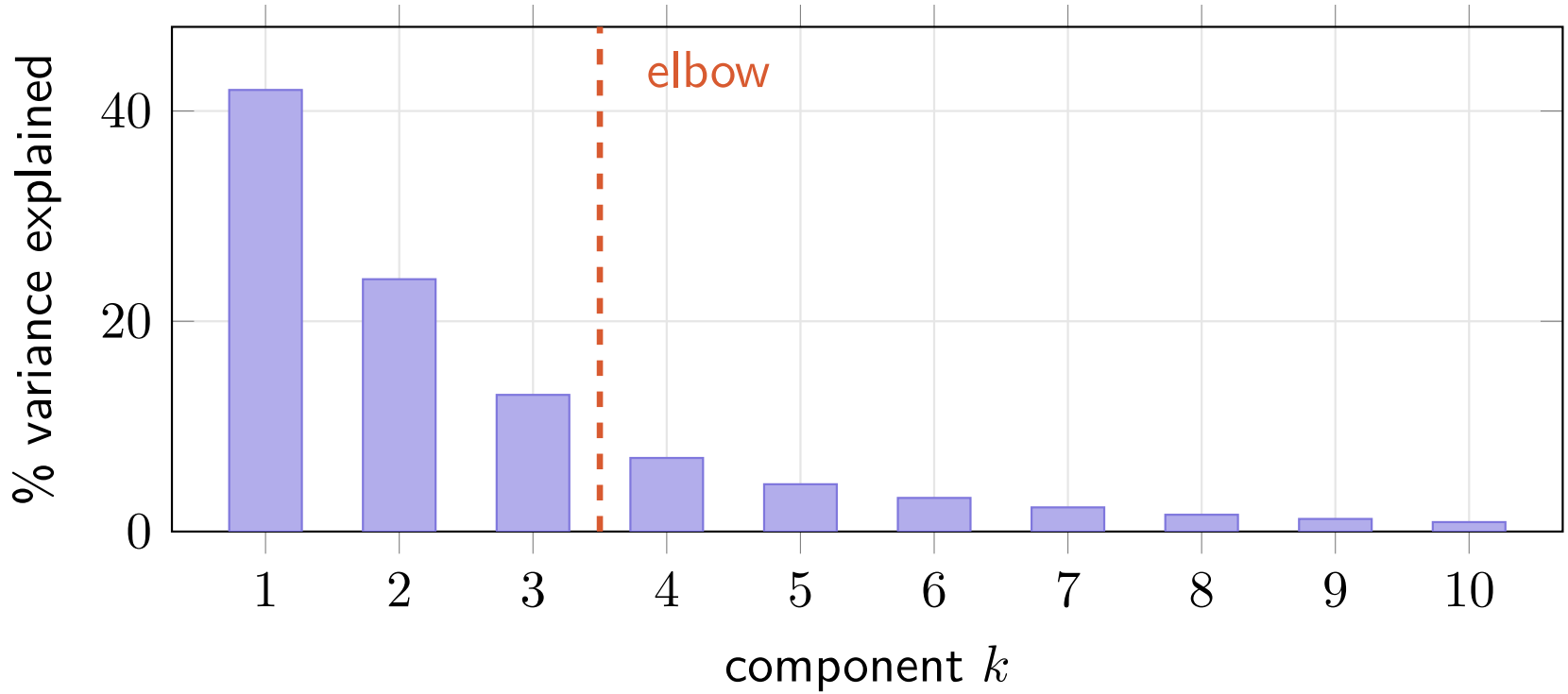
PC_1 spans the line minimising the sum of squared *perpendicular* distances from the points.



PCA is not linear regression

Least-squares regression minimises *vertical* residuals and treats y as special. PCA minimises *perpendicular* distance and treats all variables symmetrically. They give different lines.

How many components? The scree plot



Common criteria: the elbow; cumulative variance $\geq 90\,\%$ or $95\,\%$; Kaiser's rule ($\lambda_k > 1$ on standardised data, widely criticised); cross-validation on the downstream task — usually the only one that is defensible.

To standardise or not

S depends on the units of each variable. A variable measured in millimetres has 10^6 times the variance of the same variable in kilometres.

Covariance PCA (centre only)

Appropriate when all variables share units and comparable scales — e.g. pixel intensities, spectra, repeated measurements of one quantity.

Correlation PCA (centre *and* scale to unit variance)

Appropriate when units differ — e.g. age (years), income (currency), height (cm). This is the usual default for heterogeneous data.

Consequence

"The PCA" of a dataset is not well defined until you state the preprocessing. Changing units changes the answer.

Image compression and PCA are the same computation

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad
॥ नवोन्मेषेण देश निर्माणम् ॥

	Image compression	PCA
matrix	pixel array A	centred data $\tilde{X}$
preprocessing	none	centre (and maybe scale)
decomposition	$A = U\Sigma V^{\mathsf{T}}$	$\tilde{X} = U\Sigma V^{\mathsf{T}}$
keep	top k terms	top k terms
optimality	Eckart–Young in $\|\cdot\|_F$	max variance / min reconstruction
σ_k means	energy in a layer	$\sqrt{(n-1)\lambda_k}$, a standard deviation

PCA *is* truncated SVD applied to a centred data matrix. The difference is interpretation and preprocessing, not mathematics.

Pitfalls worth stating out loud

1. **PCA is unsupervised.** The direction of greatest variance need not be the direction that predicts your label. A tiny-variance direction can be the discriminative one.
2. **Components are not features you chose.** PC1 is a dense linear combination of *all* p variables. PCA is dimension reduction, not variable selection. (Use sparse PCA or LASSO if you need selection.)
3. **Interpretation is risky.** Naming PC2 "socioeconomic status" is a story you are telling, not a result. Rotation within a near-tied eigenvalue block is arbitrary.
4. **Linearity.** PCA finds linear subspaces. Data on a curved manifold (a spiral, a circle) will not be captured. Consider kernel PCA, diffusion maps, UMAP.
5. **Outliers dominate.** Variance is a squared quantity; one bad point can define PC1. Consider robust PCA.
6. **Leakage.** Fit the centring, scaling and V on the training set only, then apply to test data.

The one slide to remember

$$A = \sum_{k=1}^{r} \sigma_k \, \boldsymbol{u}_k \boldsymbol{v}_k^{\mathsf{T}}, \qquad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$$

- Exists for every matrix; both bases orthonormal; the σ_k unique.
- Truncating at k gives the *provably best* rank-k approximation (Eckart–Young).
- Rank, norms, condition number, four subspaces, pseudoinverse: all read off the factors.
- Applied to a pixel array $\Rightarrow$ image compression. Applied to a centred data matrix $\Rightarrow$ PCA.

"*The SVD is the Swiss Army knife of matrix decompositions.*" — Diane O'Leary