Lecture 09

Clustering Aggregation

Clustering aggregation is a technique where you combine multiple clustering results into a single, consolidated clustering that is considered “better” or more robust.

Instead of relying on one clustering algorithm or one run, you aggregate several clustering’s to:

- Reduce variability
- Improve stability
- Leverage different perspectives or algorithms

Think of it like taking a vote among multiple clustering’s to decide the final cluster assignment for each data point. **But why?**

- Many clustering algorithms (like K-Means) are **sensitive to initialization**.
- Different algorithms or runs may produce **different results**.
- Aggregating multiple clusterings gives a **more reliable** solution, especially on noisy or high-dimensional data.

Steps of Clustering Aggregation

Step 1: Generate multiple clusterings

- Run the **same algorithm multiple times** with different initializations.
- Or run **different algorithms** (e.g., K-Means, GMM, hierarchical) on the same dataset.

Step 2: Build a co-association matrix

- For N data points, create an $N \times N$ matrix where entry (i, j) = fraction of clusterings in which points i and j belong to the same cluster.

Example:

Point Pair	Times in same cluster	Co-association
$(x1, x2)$	8 / 10 runs	0.8
$(x1, x3)$	3 / 10 runs	0.3

Step 3: Apply consensus clustering

- Treat the co-association matrix as a **similarity matrix**.
- Run **any clustering algorithm** (often hierarchical clustering) on this matrix to get final clusters.

Comparing Clusters

Given two clusterings of the same dataset, say:

- $C1 = \{C_1^1, C_1^2, \dots, C_1^{k_1}\}$
- $C2 = \{C_2^1, C_2^2, \dots, C_2^{k_2}\}$

We want a **distance measure** d(C_1, C_2) such that:

- $d = 0$ $\rightarrow$ clusterings are identical
- $d = 1\ (or\ max)$ $\rightarrow$ completely different

Disagreement Distance

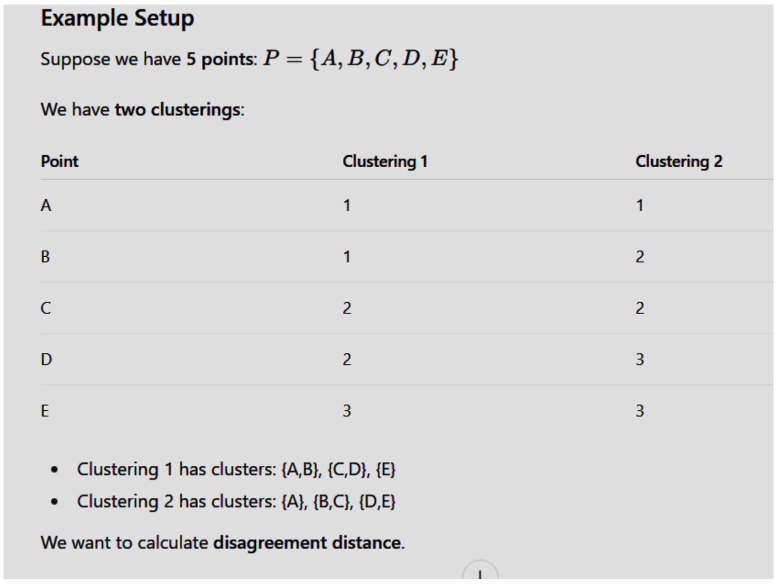
Used in clustering aggregation. It is based on the pairwise co-clustering of points:

1. For every pair of points (i, j), check whether they are in the **same cluster** in C_1 and C_2.
2. Count the number of **pairs that disagree**:
   - Agree if both in same cluster **or** both in different clusters
   - Disagree if one clustering puts them together and the other does not

$$Disagreement\ Distance = \frac{Number\ of\ disagreeing\ pairs}{\binom{N}{2}}$$

Where, N = number of points,

- Range: $[0,1]$
- 0 $\rightarrow$ clusterings are identical
- 1 $\rightarrow$ every pair disagrees



Example Setup

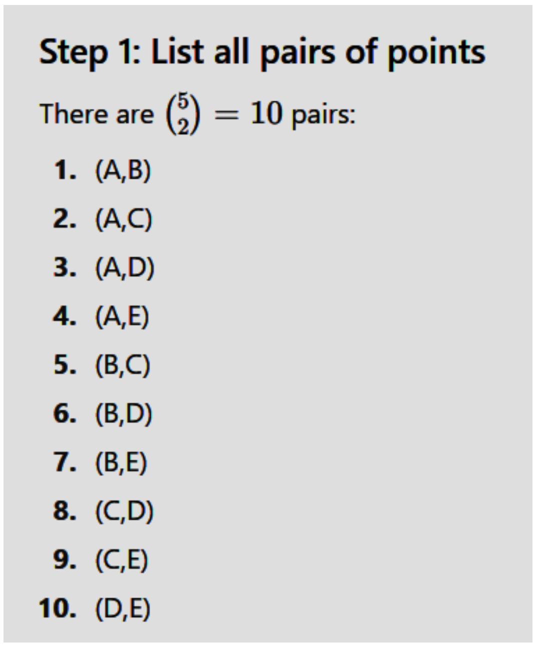
Suppose we have **5 points**: $P = \{A, B, C, D, E\}$

We have **two clusterings**:

Point	Clustering 1	Clustering 2
A	1	1
B	1	2
C	2	2
D	2	3
E	3	3

- Clustering 1 has clusters: {A,B}, {C,D}, {E}
- Clustering 2 has clusters: {A}, {B,C}, {D,E}

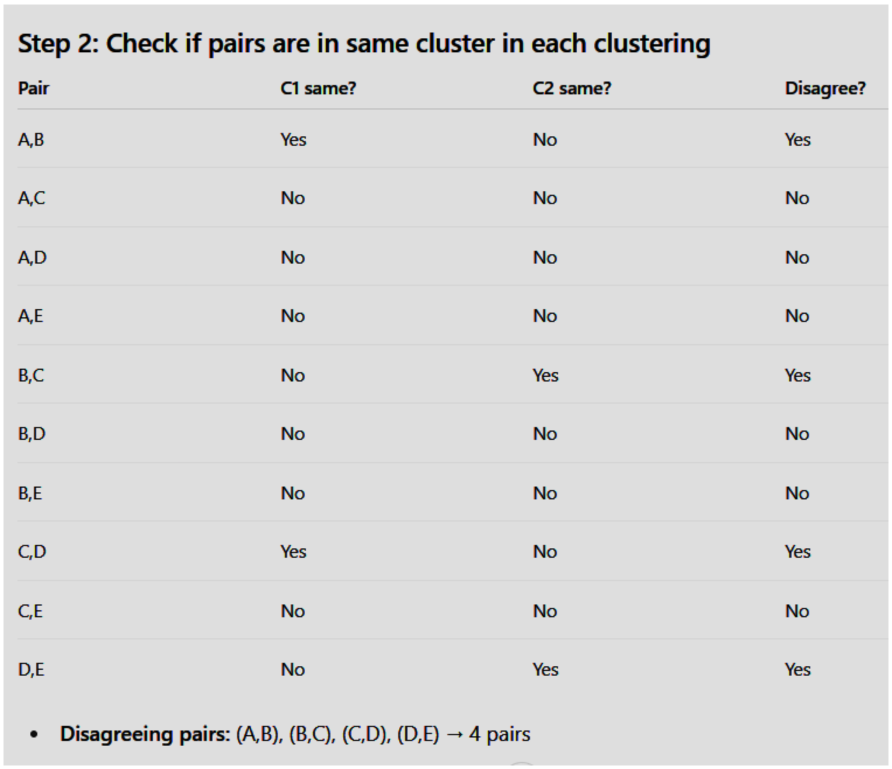
We want to calculate **disagreement distance**.



Step 1: List all pairs of points

There are $\binom{5}{2} = 10$ pairs:

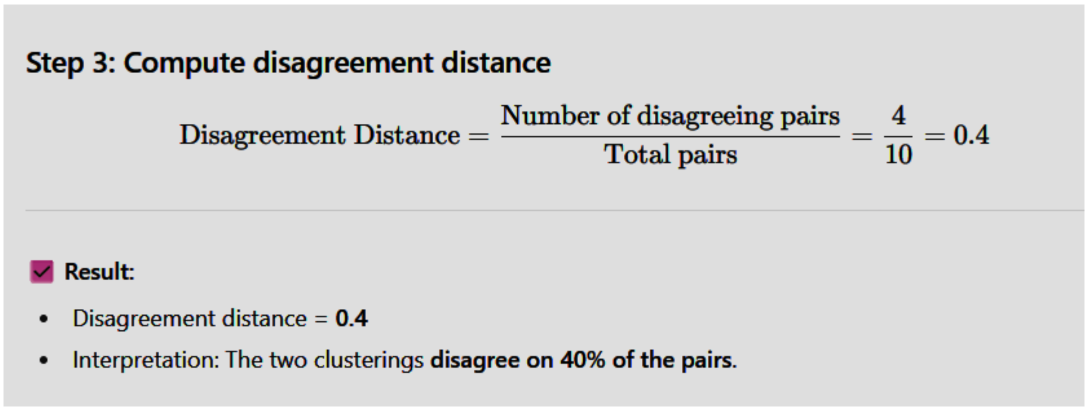
1. (A,B)
2. (A,C)
3. (A,D)
4. (A,E)
5. (B,C)
6. (B,D)
7. (B,E)
8. (C,D)
9. (C,E)
10. (D,E)



Step 2: Check if pairs are in same cluster in each clustering

Pair	C1 same?	C2 same?	Disagree?
A,B	Yes	No	Yes
A,C	No	No	No
A,D	No	No	No
A,E	No	No	No
B,C	No	Yes	Yes
B,D	No	No	No
B,E	No	No	No
C,D	Yes	No	Yes
C,E	No	No	No
D,E	No	Yes	Yes

- **Disagreeing pairs:** (A,B), (B,C), (C,D), (D,E) → 4 pairs



Step 3: Compute disagreement distance

$$\text{Disagreement Distance} = \frac{\text{Number of disagreeing pairs}}{\text{Total pairs}} = \frac{4}{10} = 0.4$$

✅ **Result:**

- Disagreement distance = **0.4**
- Interpretation: The two clusterings **disagree on 40% of the pairs**.

Singular Value Decomposition:

For any real $m \times n$ matrix A SVD factorizes it as:

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

Where,

Matrix	Shape	Properties
U	$m \times m$	Orthogonal ($U^T U = I_m$) — columns are left singular vectors
Σ	$m \times n$	Diagonal (only non-negative entries σ_i called singular values)
V	$n \times n$	Orthogonal ($V^T V = I_n$) — columns are right singular vectors

SVD **rotates and scales** your space:

- V^T rotates the coordinate system.
- Σ scales along principal directions.
- U rotates the output space.

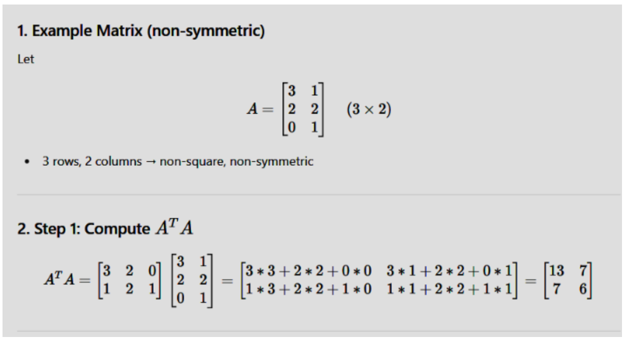
Think of it as a **generalization of eigen decomposition** for non-square matrices.

Rank K approximation

To approximate A using only the top k components (largest singular values):

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T = U_k \Sigma_k V_k^T$$

Where, k is first k columns of respective notation.



1. Example Matrix (non-symmetric)

Let

$$A = \begin{bmatrix} 3 & 1 \\ 2 & 2 \\ 0 & 1 \end{bmatrix} \quad (3 \times 2)$$

- 3 rows, 2 columns → non-square, non-symmetric

2. Step 1: Compute $A^T A$

$$A^T A = \begin{bmatrix} 3 & 2 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 2 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 3*3+2*2+0*0 & 3*1+2*2+0*1 \\ 1*3+2*2+1*0 & 1*1+2*2+1*1 \end{bmatrix} = \begin{bmatrix} 13 & 7 \\ 7 & 6 \end{bmatrix}$$

3. Step 2: Compute eigenvalues of A^TA

- Solve $\det(A^TA - \lambda I) = 0$:

$$\begin{vmatrix} 13-\lambda & 7 \\ 7 & 6-\lambda \end{vmatrix} = (13-\lambda)(6-\lambda) - 49 = 0$$

$$(13-\lambda)(6-\lambda) - 49 = 78 - 19\lambda + \lambda^2 - 49 = \lambda^2 - 19\lambda + 29 = 0$$

- Solve $\lambda^2 - 19\lambda + 29 = 0$

$$\lambda = \frac{19 \pm \sqrt{361-116}}{2} = \frac{19 \pm \sqrt{245}}{2} \approx \frac{19 \pm 15.652}{2}$$

$$\lambda_1 \approx \frac{19+15.652}{2} \approx 17.326, \quad \lambda_2 \approx \frac{19-15.652}{2} \approx 1.674$$

4. Step 3: Compute singular values

$$\sigma_i = \sqrt{\lambda_i}$$

$$\sigma_1 \approx \sqrt{17.326} \approx 4.16, \quad \sigma_2 \approx \sqrt{1.674} \approx 1.29$$

- Σ will be:

$$\Sigma = \begin{bmatrix} 4.16 & 0 \\ 0 & 1.29 \\ 0 & 0 \end{bmatrix} \quad (3 \times 2)$$

5. Step 4: Compute right singular vectors V

- Solve $(A^TA)v = \lambda v$

For $\lambda_1 = 17.326$:

$$\begin{bmatrix} 13 & 7 \\ 7 & 6 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 17.326 \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

- First equation: $13v_1 + 7v_2 = 17.326v_1 \implies 7v_2 = 4.326v_1 \implies v_2 = 0.618v_1$
- Normalize: $\|v\| = \sqrt{v_1^2 + v_2^2} = \sqrt{1 + 0.618^2} \approx 1.175$

$$v_1 = \frac{1}{1.175} \approx 0.851, \quad v_2 \approx 0.851 * 0.618 \approx 0.526$$

- So first right singular vector:

$$V_1 = \begin{bmatrix} 0.851 \\ 0.526 \end{bmatrix}$$

- Similarly, compute V_2 (orthogonal to V_1):

$$V_2 \approx \begin{bmatrix} -0.526 \\ 0.851 \end{bmatrix}$$

6. Step 5: Compute left singular vectors U

$$u_i = \frac{Av_i}{\sigma_i}$$

First left singular vector u_1:

$$Av_1 = \begin{bmatrix} 3 & 1 \\ 2 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.851 \\ 0.526 \end{bmatrix} = \begin{bmatrix} 3 * 0.851 + 1 * 0.526 \\ 2 * 0.851 + 2 * 0.526 \\ 0 * 0.851 + 1 * 0.526 \end{bmatrix} = \begin{bmatrix} 3.079 \\ 2.754 \\ 0.526 \end{bmatrix}$$

- Divide by σ_1 = 4.16 →

$$u_1 \approx \begin{bmatrix} 0.741 \\ 0.662 \\ 0.126 \end{bmatrix}$$

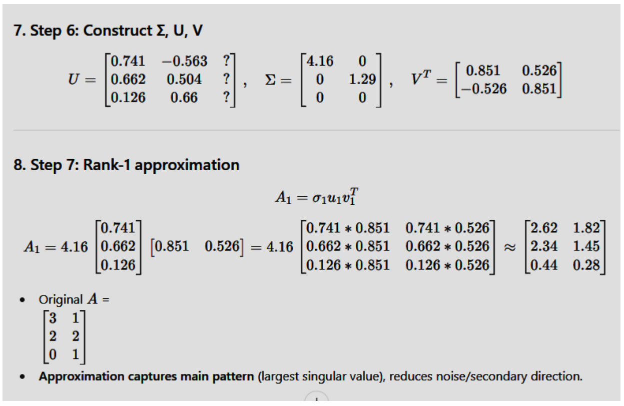
Second left singular vector u_2:

$$Av_2 = \begin{bmatrix} 3 & 1 \\ 2 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -0.526 \\ 0.851 \end{bmatrix} = \begin{bmatrix} -0.727 \\ 0.650 \\ 0.851 \end{bmatrix}$$

- Divide by σ_2 = 1.29 →

$$u_2 \approx \begin{bmatrix} -0.563 \\ 0.504 \\ 0.66 \end{bmatrix}$$

- If needed, complete U to 3×3 with a vector orthogonal to u1 and u2.



7. Step 6: Construct Σ, U, V

$$U = \begin{bmatrix} 0.741 & -0.563 & ? \\ 0.662 & 0.504 & ? \\ 0.126 & 0.66 & ? \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 4.16 & 0 \\ 0 & 1.29 \\ 0 & 0 \end{bmatrix}, \quad V^T = \begin{bmatrix} 0.851 & 0.526 \\ -0.526 & 0.851 \end{bmatrix}$$

8. Step 7: Rank-1 approximation

$$A_1 = \sigma_1 u_1 v_1^T$$

$$A_1 = 4.16 \begin{bmatrix} 0.741 \\ 0.662 \\ 0.126 \end{bmatrix} \begin{bmatrix} 0.851 & 0.526 \end{bmatrix} = 4.16 \begin{bmatrix} 0.741 * 0.851 & 0.741 * 0.526 \\ 0.662 * 0.851 & 0.662 * 0.526 \\ 0.126 * 0.851 & 0.126 * 0.526 \end{bmatrix} \approx \begin{bmatrix} 2.62 & 1.82 \\ 2.34 & 1.45 \\ 0.44 & 0.28 \end{bmatrix}$$

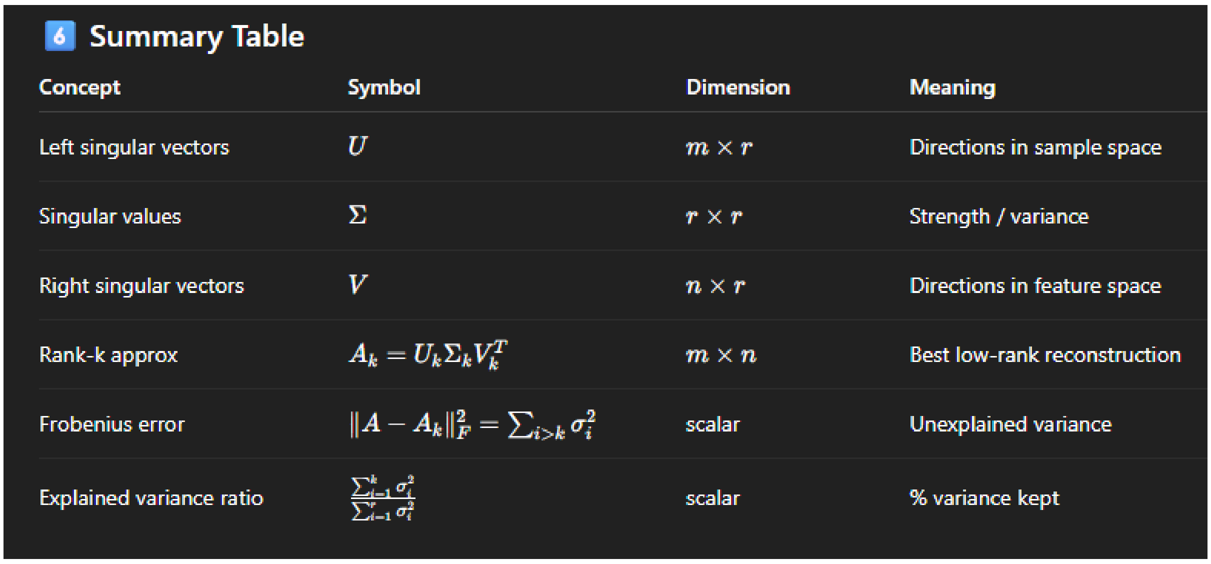
- Original A = $\begin{bmatrix} 3 & 1 \\ 2 & 2 \\ 0 & 1 \end{bmatrix}$
- **Approximation captures main pattern** (largest singular value), reduces noise/secondary direction.

PCA

Principal component analysis (PCA) is a dimensionality reduction technique that transforms a data set into a set of orthogonal components called principal components which capture the maximum variance in the data. PCA simplifies complex data sets while preserving their most important structures.

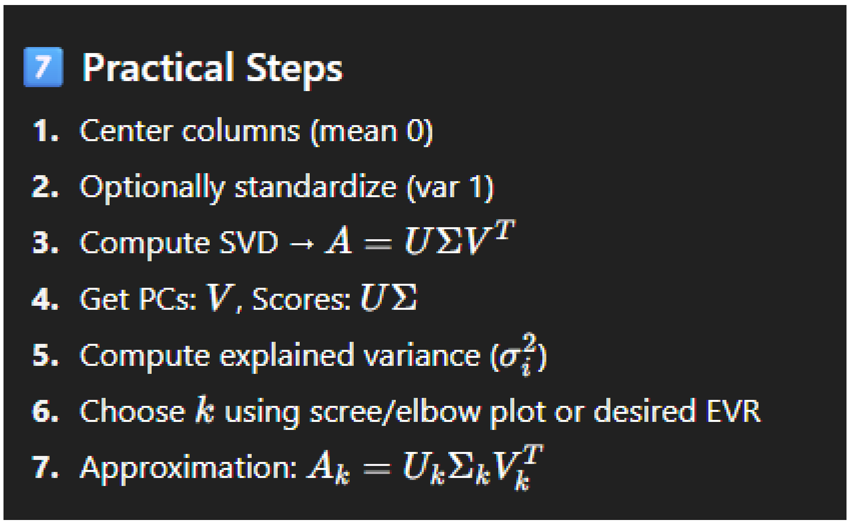
Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (i.e., principal components) are uncorrelated and most of the information within the initial variables is squeezed or compressed into the first components. So, the idea is 10-dimensional data gives you 10 principal components, but PCA tries to put maximum possible information in the first component, then maximum remaining information in the second and so on.

- 1st PC: axis of maximum variance
- 2nd PC: orthogonal to 1st, captures remaining variance



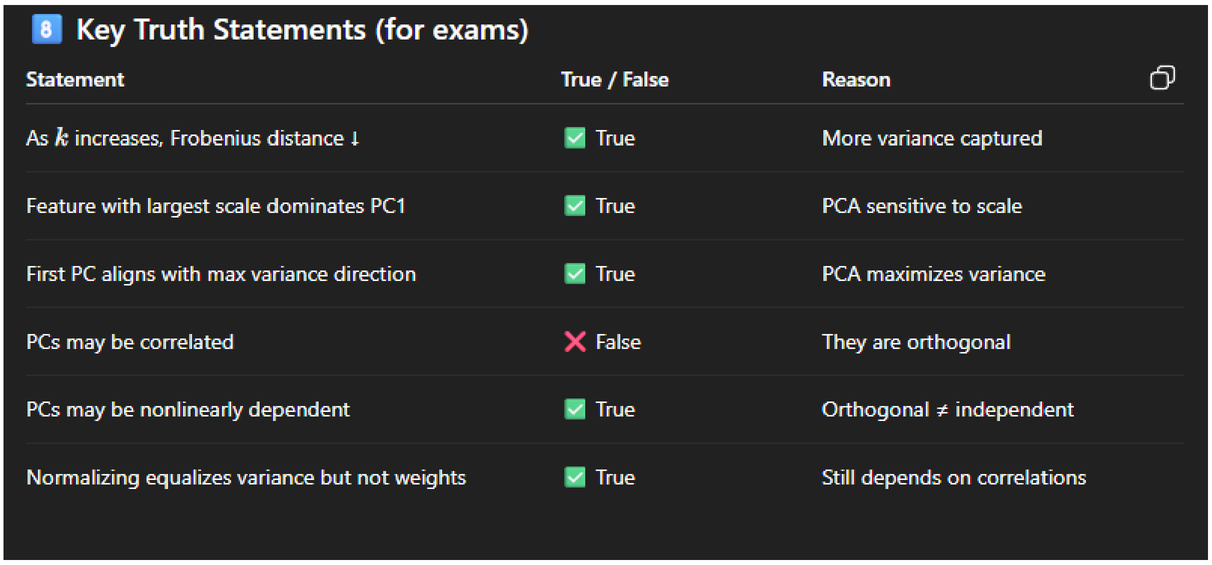
6️⃣ Summary Table

Concept	Symbol	Dimension	Meaning
Left singular vectors	U	$m \times r$	Directions in sample space
Singular values	Σ	$r \times r$	Strength / variance
Right singular vectors	V	$n \times r$	Directions in feature space
Rank-k approx	$A_k = U_k \Sigma_k V_k^T$	$m \times n$	Best low-rank reconstruction
Frobenius error	$\|A - A_k\|_F^2 = \sum_{i>k} \sigma_i^2$	scalar	Unexplained variance
Explained variance ratio	$\frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}$	scalar	% variance kept



7️⃣ Practical Steps

1. Center columns (mean 0)
2. Optionally standardize (var 1)
3. Compute SVD → $A = U\Sigma V^T$
4. Get PCs: V, Scores: $U\Sigma$
5. Compute explained variance (σ_i^2)
6. Choose k using scree/elbow plot or desired EVR
7. Approximation: $A_k = U_k\Sigma_k V_k^T$



8️⃣ Key Truth Statements (for exams)

Statement	True / False	Reason
As k increases, Frobenius distance ↓	✅ True	More variance captured
Feature with largest scale dominates PC1	✅ True	PCA sensitive to scale
First PC aligns with max variance direction	✅ True	PCA maximizes variance
PCs may be correlated	❌ False	They are orthogonal
PCs may be nonlinearly dependent	✅ True	Orthogonal ≠ independent
Normalizing equalizes variance but not weights	✅ True	Still depends on correlations

Covariance Formula:

$$Cov(X) = \frac{1}{n-1} X_c^T X_c$$

So, each entry means

$$Cov(X) = \begin{bmatrix} Var(f1) & Cov(f1, f2) \\ Cov(f2, f1) & Var(f2) \end{bmatrix}$$

Where,

$$Var(f) = \frac{1}{n-1} \sum (x_i - \bar{x}\,)^2$$

And

$$Cov(f1, f2) = \frac{1}{n-1} \sum (x_{i1} - \overline{x_1}\,)(x_{i2} - \bar{x}_2\,)$$

Steps:

1. Take dataset and compute mean of each feature / column
2. Center the data (column – mean)
3. Create Covariance matrix C from the columns in step 2
4. Find Eigen values (λ) of Covariance matrix by (solve $det(C - \lambda I) = 0$)
5. You will get $\lambda_!, \lambda_2 \dots \lambda_n$
6. Find Eigen vector using λ sub in equation $det(C - \lambda I)v = 0$
    a. $v = [v1\ v2]^T$
    b. solve using linear equations
    c. normalize and do for all λ
    d. assume you get two sets of $[v_1 1\ v_1 2]^T$ and $[v_2 1\ v_2 2]^T$
    e. Therefore $V = \begin{bmatrix} v_1 1 & v_2 1 \\ v_1 2 & v_2 2 \end{bmatrix}$
7. Compute singular values $\sigma_i = \sqrt{\lambda_i (n-1)}$
8. Thus $\sum$ becomes $\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$
9. So $U = X_c V \Sigma^{-1}$
    a. Where each column u_i is represented as $\frac{1}{\sigma_i} X_c v_i$
10. Rank-1 formula: $\hat{X}_c^{(1)} = \sigma_1 u_1 v_1^T$