Applications of Singular Value Decomposition (SVD)

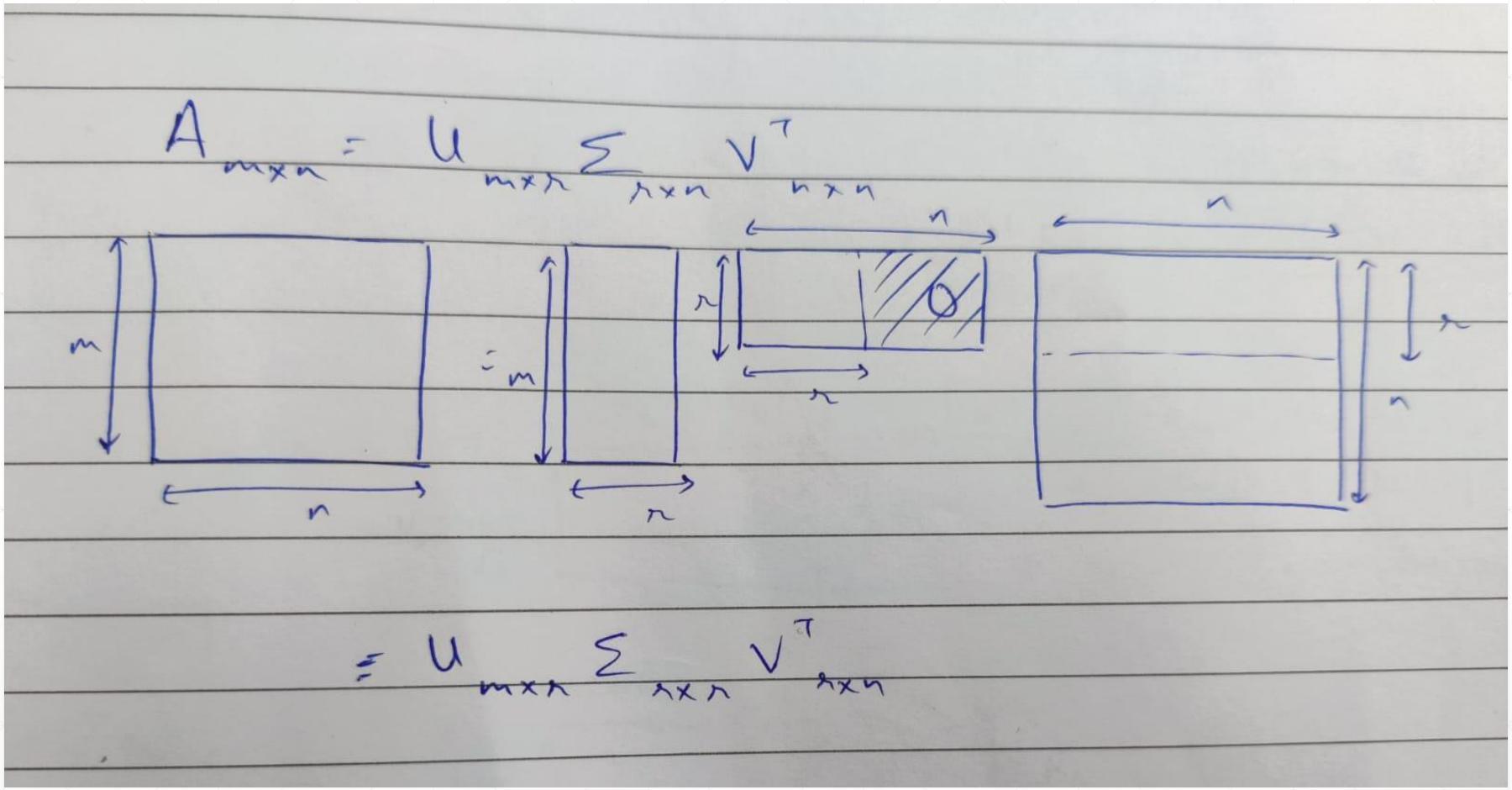
MTQ315 Presentation : Preksha Mishra , 2022ES11849

Singular Value Decomposition Definition

Singular Value Decomposition: Any m by n matrix A can be factored into

$$A = U\Sigma V^{\mathrm{T}} = (\textbf{orthogonal})(\textbf{diagonal})(\textbf{orthogonal}).$$

The columns of U (m by m) are eigenvectors of AA^{T}, and the columns of V (n by n) are eigenvectors of $A^{\mathrm{T}}A$. The r singular values on the diagonal of Σ (m by n) are the square roots of the nonzero eigenvalues of both AA^{T} and $A^{\mathrm{T}}A$.


$A_{m \times n} = U_{m \times r} \Sigma_{r \times r} V^T_{r \times n}$
m
n
r
r
r
r
n
n
r
$= U_{m \times r} \Sigma_{r \times r} V^T_{r \times n}$

Noise Filtering

Vulnerability of sensors to pick up noise makes this topic is useful in areas like image processing, signal processing, and data compression.

1. **SVD Decomposition**:

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

2. **Sort Singular Values**:

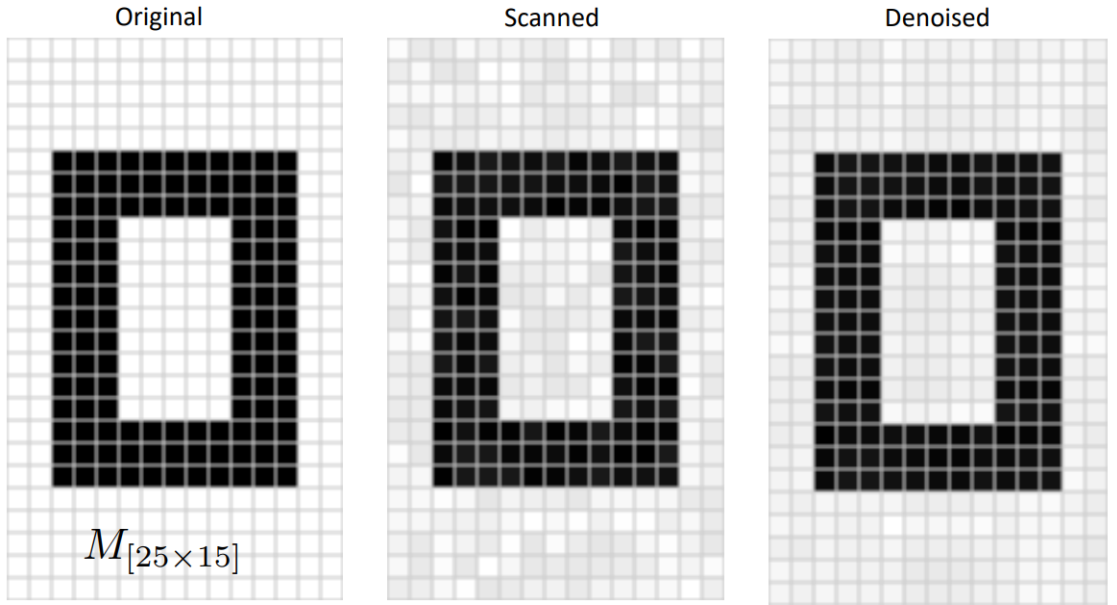
$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$$

3. **Select Threshold**: Retain the top k singular values:

$$\Sigma_k = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k)$$

4. **Reconstruct Filtered Data**:

$$X_{\text{filtered}} = U_k \Sigma_k V_k^T$$



$$M_{[25\times15]} \approx u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T + u_3\sigma_3 v_3^T$$

$\sigma_1 = 14.15$
$\sigma_2 = 4.67$
$\sigma_3 = 3.00$

$\sigma_4 = 0.21$
$\sigma_5 = 0.19$
...
$\sigma_{15} = 0.05$

Dimension Reduction

We only preserve dimensions with large variance (thus smallest error), ensuring easier visualization while still retaining the most significant patterns in the data. This also reduces the risk of overfitting due to noise in the lesser important dimensions.

The U matrix contains the principal components (eigenvectors of the covariance matrix in PCA), which represent the directions of maximum variance in the original data.

1. **Formulate Input Matrix A:**

$$A = \text{data matrix}$$

2. **Compute Covariance Matrix C:**

$$C = \frac{1}{\text{m} - 1} A^T A$$

3. **Perform SVD on C:**

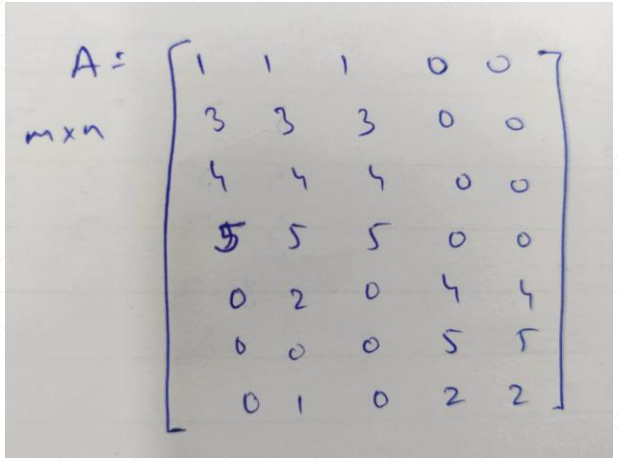
$$[U, S, V] = \text{svd}(C)$$

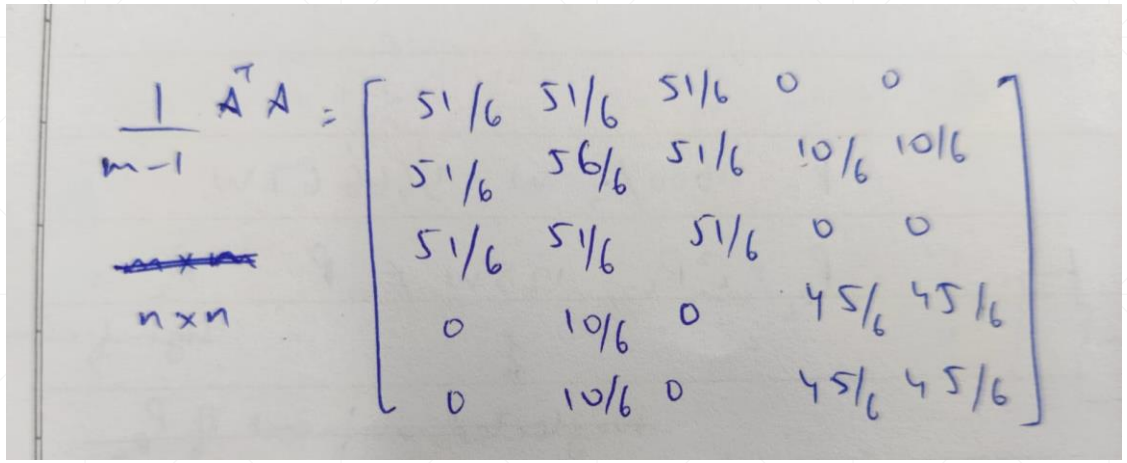
4. **Select Top k Principal Components:**
   - Choose the top k columns of U.

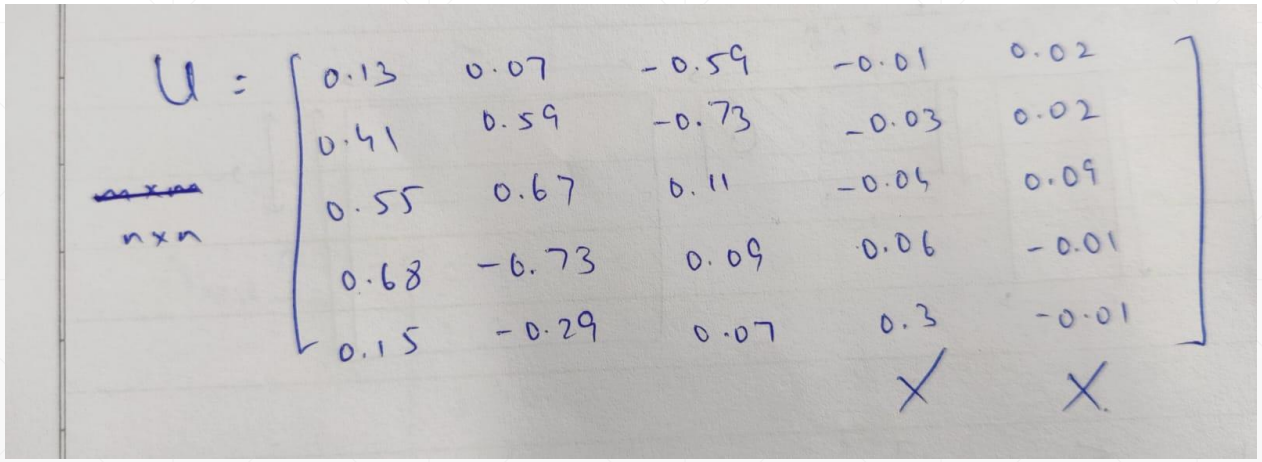
5. **Project Data onto Selected Components:**

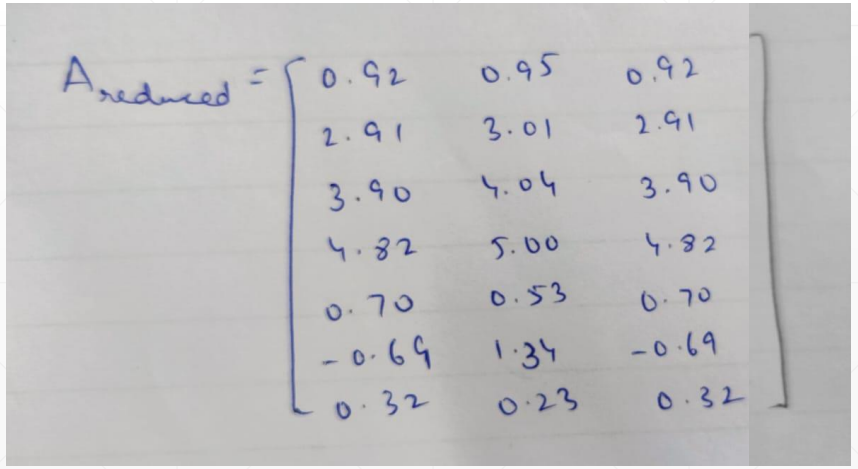
$$A_{\text{reduced}} = AU_k$$

Dimension Reduction









Latent Semantic Analysis (LSA)

This is a technique for uncovering hidden semantic structures in text data by reducing dimensionality.
Process:

1. Create a Term-Document Matrix A: Each row represents a term, each column a document, with values as term frequencies.
2. Apply Singular Value Decomposition (SVD): Decompose A into $A=U\Sigma V^T$, where
   U: Term-concept matrix, with terms positioned in the semantic space.
   Σ: Diagonal matrix of singular values, showing the importance of each concept.
   V: Document-concept matrix, positioning documents in the semantic space.
3. Dimensionality Reduction: Use only the top k singular values and vectors in Uk and Vk, capturing main concepts while filtering noise.
4. Interpreting results:
   Each column in Uk corresponds to a latent topic. The terms with high weights in each column of Uk represent key words for a specific latent topic. Terms with similar weights are semantically related and cluster around the same concept (e.g., "cat," "dog," "pet" might cluster around the "animals" topic).
   Documents with similar values in Vk are closely related in meaning, sharing the same dominant topics. These documents can be grouped based on their latent semantic structure
5. Using Cosine Similarity to Quantify Similarity: Cosine similarity measures the angle between two document vectors in V, capturing how similarly they align with each latent topic. A cosine similarity value close to 1 indicates a strong alignment in themes.

Latent Semantic Analysis (LSA)

Index Words	Titles								
	T1	T2	T3	T4	T5	T6	T7	T8	T9
book			1	1					
dads						1			1
dummies		1						1	
estate							1		1
guide	1					1			
investing	1	1	1	1	1	1	1	1	1
market	1		1						
real							1		1
rich						2			1
stock	1		1					1	
value				1	1				

	C1	C2	C3
book	0.15	-0.27	0.04
dads	0.24	0.38	-0.09
dummies	0.13	-0.17	0.07
estate	0.18	0.19	0.45
guide	0.22	0.09	-0.46
investing	0.74	-0.21	0.21
market	0.18	-0.30	-0.28
real	0.18	0.19	0.45
rich	0.36	0.59	-0.34
stock	0.25	-0.42	-0.28
value	0.12	-0.14	0.23

3.91	0	0
0	2.61	0
0	0	2.00

T1	T2	T3	T4	T5	T6	T7	T8	T9	
0.35	0.22	0.34	0.26	0.22	0.49	0.28	0.29	0.44	C1
-0.32	-0.15	-0.46	-0.24	-0.14	0.55	0.07	-0.31	0.44	C2
-0.41	0.14	-0.16	0.25	0.22	-0.51	0.55	0.00	0.34	C3

Image Compression

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

image = mpimg.imread("cat.jpg")

gray_image = np.mean(image, axis=2)

U, S, Vt = np.linalg.svd(gray_image, full_matrices=False)

k = 100
U_k = U[:, :k]
S_k = np.diag(S[:k])
Vt_k = Vt[:k, :]
```

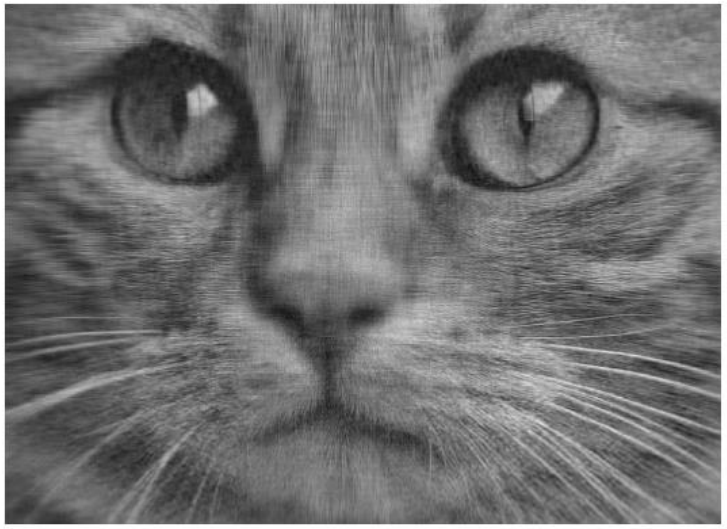
```
compressed_image = np.dot(U_k, np.dot(S_k, Vt_k))

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(gray_image, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(compressed_image, cmap='gray')
plt.title(f'Compressed Image (k={k})')
plt.axis('off')

plt.show()
```

Image Compression

Original Image

Compressed Image (k=100)

Compressed Image (k=50)



Compressed Image (k=10)

Thank You

MTQ315 Presentation : Preksha Mishra , 2022ES11849