Tut_1_TUT_2 (1)_merged.pdf
PDF

generate the solution to this

Alright. Let's solve this cleanly and exam-ready. No drama. Just marks secured. 💪

I'll go section by section.

✅ Tutorial – 1

Q1–Q10 (MCQs)

1. **(C)** Represents spread around the mean
2. **(B)** 75% (Chebyshev: ≥ 1 − 1/k² = 1 − 1/4 = 0.75)
3. **(A)** Covariance matrix
4. **(A)** Mean
5. **(A)** Always real and orthogonal (real symmetric matrix property)
6. **(B)** All eigenvalues ≥ 0
7. **(B)** Sum of eigenvalues
8. **(B)** $A^{-1} = A^T$
9. **(C)** Direction of maximum variance
10. **(C)** Improving classification accuracy directly

Q11 (2 marks)

Geometric intuition of PCA

PCA finds new orthogonal axes (directions) such that:

- First PC → captures **maximum variance**
- Second PC → maximum variance **orthogonal** to first
- And so on…

Geometrically:
We are rotating the coordinate system to align with directions where data spreads most.

Why first PC important?

Because it captures the largest variation in the dataset → most information → best 1D summary.

Q12 (4 marks)

Matrix:

$$A = \begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix}$$

Step 1: Eigenvalues

$$|A - \lambda I| = \begin{vmatrix} 4-\lambda & 1 \\ 1 & 3-\lambda \end{vmatrix}$$

$$= (4-\lambda)(3-\lambda) - 1$$

$$= \lambda^2 - 7\lambda + 11$$

Solve:

$$\lambda = \frac{7 \pm \sqrt{49 - 44}}{2} = \frac{7 \pm \sqrt{5}}{2}$$

So eigenvalues:

$$\lambda_1 = \frac{7+\sqrt{5}}{2}, \quad \lambda_2 = \frac{7-\sqrt{5}}{2}$$

Step 2: Eigenvectors

For each λ solve:

$$(A - \lambda I)v = 0$$

You'll get eigenvectors proportional to:

$$v_1 = \begin{bmatrix} 1 \\ \frac{\sqrt{5}-1}{2} \end{bmatrix}, \quad v_2 = \begin{bmatrix} 1 \\ -\frac{\sqrt{5}+1}{2} \end{bmatrix}$$

Normalize them.

Spectral Decomposition

$$A = Q\Lambda Q^T$$

Where:

- Q = orthonormal eigenvectors
- Λ = diagonal eigenvalue matrix

That's your final form.

Q13 (2 marks)

Center + Covariance manually

Python

```python
import numpy as np

X = np.array([[1, 2],
              [3, 4],
              [5, 6]])

# Step 1: Compute mean
```



```python
# Step 2: Center data
X_centered = X - mean

# Step 3: Covariance matrix
n = X.shape[0]
cov_matrix = (X_centered.T @ X_centered) / (n - 1)

print(cov_matrix)
```

Output:

$$\begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix}$$

Q14 (2 marks)

Pandas preprocessing

Python

```python
import pandas as pd

data = pd.DataFrame({
'age': [22, 25, 47, 52],
'income': [20000, 25000, 47000, 52000],
'city': ['Delhi', 'Mumbai', 'Delhi', 'Chennai']
})

# Min-max scaling on age
data['age'] = (data['age'] - data['age'].min()) / \
              (data['age'].max() - data['age'].min())

# Z-score on income
data['income'] = (data['income'] - data['income'].mean()) / \
                 data['income'].std()

# One-hot encoding
data = pd.get_dummies(data, columns=['city'])

print(data)
```

✅ Tutorial – 02

Q1

(B) Columns of U are orthonormal

Q2

(A) 3, 5

(Singular values = absolute diagonal values)

Q3

Total variance:

$$4.5 + 2 + 1 + 0.5 = 8$$

90% of 8 = 7.2

Cumulative:

- 4.5
- 6.5
- 7.5 ✅

Minimum PCs = **3**

Q4

$$\frac{8000}{9600} = 0.8333$$

Answer: **83.33%**

Q5

True

If all non-zero singular values kept → exact reconstruction.

Q6

(C) C is orthogonal ❌

Q7

(B) Columns of V are eigenvectors of $A^T A$

Q8

Matrix:

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \\ 3 & 1 \end{bmatrix}$$

Mean:

$$\left(\frac{2+0+3}{3}, \frac{0+2+1}{3}\right) = \left(\frac{5}{3}, 1\right)$$

Q9

(B)

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{n} \lambda_j}$$

Q10

$$\frac{5^2 + 3^2}{5^2 + 3^2 + 1^2} = \frac{25 + 9}{25 + 9 + 1} = \frac{34}{35} = 0.9714$$

≈ **97.14%**

Q11

(D) Efficient sorting of numerical arrays

Q12 (SVD Reconstruction Code)

Python

```python
import numpy as np

A = np.array([[1, 2],
              [3, 4]], dtype=float)

U, s, VT = np.linalg.svd(A, full_matrices=False)
Sigma = np.diag(s)

A_hat = U @ Sigma @ VT
err = np.linalg.norm(A - A_hat, ord='fro')

print("U=\n", U)
print("Sigma=\n", Sigma)
print("VT=\n", VT)
print("Frobenius error =", err)
```

Error ≈ **0** (numerical precision ~1e-15)

Q13 (sklearn PCA)

Python

```python
import numpy as np
from sklearn.decomposition import PCA

np.random.seed(0)
X = np.random.randn(10, 3)

pca = PCA(n_components=2)
Z = pca.fit_transform(X)

print("Explained variance ratio:",
      pca.explained_variance_ratio_)
```

🔥 Final Advice

If this comes in exam:

- MCQs = theory clarity
- Numericals = just cumulative sums
- Coding = center → multiply → divide by (n−1)
- PCA = eigenvalues = variance

No overthinking. No philosophy. Just execution.

If you want, I can now give you a **10-minute rapid revision sheet** to memorize before exam.