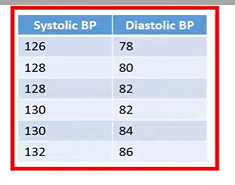
**Data analysis tools for Experimental Research 2023**

**Principal component analysis:**

**Read the instructions carefully**

1. Please label your x-axis and y-axis for all the questions. If Axis labels are not present, the answers will not be evaluated.
2. No discussion is allowed during the tutorial session.

**Question 1, 2D-1D**: Suppose following are the measurement of blood pressure for 6 patients. And you want to see them in one dimension. Decompose the data using eigenvalue and eigenvector calculations.



1. Center the data with respect to mean.
2. Compute the covariance matrix.
3. Calculate the eigenvalues and Eigen vectors.
4. Find the principal component and covert the original data matrix, 2D data to 1D.

Question 2: Computation steps: ***Principal component analysis***

org\_data = np.array([[1, 0, 2],

[0, 2, 1],

[2, 1, 0]])

### Calculate the covariance matrix (dissimilarity matrix) through calculation. [ paper and pencil]

### Also print in python command window.

### Calculate the eigenvalues and eigenvectors for the covariance matrix.

### (calculate the eigenvalues and calculate the eigenvector using python function)

### Choose the two principal components and compute the final features (2D). (Show the matrix multiplication calculation and find the reduced matrix).

### Load all the relevant libraries.

# We are using pandas, numpy and matplotlib

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

### Calculate the covariance matrix (dissimilarity matrix) and print in python command window.

# Subtract mean from data

mean = np.mean(org\_data, axis=0)

print("Mean ", mean.shape)

mean\_data = org\_data - mean

print("Data after subtracting mean ", mean\_data.shape, "\n")

# Compute covariance matrix

cov = np.cov(mean\_data.T)

cov = np.round(cov, 2)

print("Covariance matrix ", cov.shape, "\n")

print(cov)

### Calculate the eigenvalues for the covariance matrix

# Perform eigen decomposition of covariance matrix

eig\_val, eig\_vec = np.linalg.eigh(cov)

print("Eigen vectors \n", eig\_vec)

print("\n\nEigen values ", eig\_val, "\n")

### Sort the eigenvalues and plot eigenvalues in decreasing order

# Sort eigen values and corresponding eigen vectors in descending order

indices = np.argsort(eig\_val)[::-1]

eig\_val = eig\_val[indices]

eig\_vec = eig\_vec[:,indices]

# Plot the eigenvalues

plt.figure(figsize=(6, 4))

plt.plot(range(1, len(eig\_val) + 1), eig\_val, marker='o')

plt.xlabel('Eigenvalue Index')

plt.ylabel('Eigenvalue')

plt.title('Eigenvalues in Decreasing Order')

plt.grid(True)

plt.show()

### Choose the feature vector for two principal components

# Sort the eigenvalues and eigenvectors in decreasing order

sorted\_indices = np.argsort(eigenvalues)[::-1]

sorted\_eigenvalues = eigenvalues[sorted\_indices]

sorted\_eigenvectors = eigenvectors[:, sorted\_indices]

# Choose the feature vector for two principal components

num\_components = 2

feature\_vector = sorted\_eigenvectors[:, :num\_components]

print("Feature Vector:")

print(feature\_vector)

### Find the final reduced data matrix with two dimension and plot them in 2 dimension

# Project the original data onto the feature vector

reduced\_data = np.dot(org\_data.T, feature\_vector)

print("Reduced Data Matrix:")

print(reduced\_data)

# Plot the reduced data points in 2D space

plt.figure(figsize=(6, 6))

plt.scatter(reduced\_data[:, 0], reduced\_data[:, 1])

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('Reduced Data Points in 2D Space')

plt.grid(True)

plt.show()