# Department of Information Technology A.Y. 2024-2025

Class: TY BTech-IT, Semester: VI Subject: Big Data Lab

NAME: Anish Sharma SAP:60003220045

## Experiment – 10

1. **Aim:** To implement SON/CUR algorithm.

**CODE:**

**SON:**

```
import numpy as np

def son(A, num_columns):
    """
    Perform SON (Subsampled Orthogonalization and Normalization)
    Args:
        A: Input matrix (m x n)
        num_columns: Number of columns to select
Returns:
        A matrix C (m x num_columns) representing selected columns
"""
    # Step 1: Subsample the columns randomly
m, n = A.shape
    selected_columns = np.random.choice(n, num_columns, replace=False)
C = A[:, selected_columns]

    # Step 2: Orthogonalization (using Gram-Schmidt or QR factorization)
Q, R = np.linalg.qr(C)

    # Step 3: Normalize the selected columns
    C_normalized = Q

    return C_normalized

# Example usage:
A = np.array([[1, 2, 3, 4],
              [5, 6, 7, 8],
              [9, 10, 11, 12]])
```

```
C_son = son(A, num_columns=2) print("SON
Resulting Columns:\n", C_son)
```

CUR:

```
def cur_decomposition(A, num_columns, num_rows):
    """
    Perform CUR Decomposition
    Args:
        A: Input matrix (m x n)
        num_columns: Number of columns to select
num_rows: Number of rows to select      Returns:
        C: Submatrix of selected columns
        U: Middle matrix that approximates relationships
        R: Submatrix of selected rows
    """     m, n =
A.shape

    # Step 1: Select columns randomly (subsampling)
    selected_columns = np.random.choice(n, num_columns, replace=False)
    C = A[:, selected_columns]

    # Step 2: Select rows randomly (subsampling)
    selected_rows = np.random.choice(m, num_rows, replace=False)
    R = A[selected_rows, :]

    # Step 3: Calculate U using the pseudo-inverse of C and R
    U = np.linalg.pinv(C) @ A @ np.linalg.pinv(R)

    return C, U, R

# Example usage:
A = np.array([[1, 2, 3, 4],
              [5, 6, 7, 8],
              [9, 10, 11, 12]])

C_cur, U_cur, R_cur = cur_decomposition(A, num_columns=2, num_rows=2)
A_approx = C_cur @ U_cur @ R_cur

print("Original Matrix A:\n", A) print("CUR
Decomposition - C:\n", C_cur) print("CUR
Decomposition - U:\n", U_cur) print("CUR
Decomposition - R:\n", R_cur) print("CUR
Approximation of A:\n", A_approx)
```

2. **Requirements:** PC, Internet

**OUTPUT**: SON:

## SON Algorithm (Subsampled Orthogonalization and Normalization):

- **Resulting Columns (C):**

$$\begin{bmatrix} -0.0967 & 0.9077 \\ -0.4834 & 0.3157 \\ -0.8701 & -0.2763 \end{bmatrix}$$

These are the **orthogonalized and normalized columns** selected from the matrix $A$.

CUR:

### CUR Decomposition:

- **Selected Columns (C):**

$$\begin{bmatrix} 4 & 3 \\ 8 & 7 \\ 12 & 11 \end{bmatrix}$$

- **Middle Matrix (U):**

$$\begin{bmatrix} 1.375 & -0.375 \\ -1.5 & 0.5 \end{bmatrix}$$

- **Selected Rows (R):**

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

- **CUR Approximation of $A$:**

$$\begin{bmatrix} 1. & 2. & 3. & 4. \\ 5. & 6. & 7. & 8. \\ 9. & 10. & 11. & 12. \end{bmatrix}$$

3. **Conclusion:**

Thus, in this experiment, we implemented SON/CURE algorithms.