Chennai Mathematical Institute

CUR Matrix Decomposition For Improved Data Analysis

Pranav Pothan MDS202429

Riya S Huddar MDS202431

Raja S MDS202430

A report presented for the course of

Linear Algebra and its Applications

Under the guidance of
Prof. Kavita Sutar

Year: **2025**

Abstract

Traditional Singular Value Decomposition (SVD) is effective for data analysis but often lacks interpretability, as its components are linear combinations of all data points. CUR matrix decomposition offers a more interpretable alternative by selecting a subset of actual rows and columns from the original data matrix.

In this report, we present a CUR decomposition method that selects rows and columns based on statistical leverage scores, which identify the most influential data points. This results in improved relative-error approximations and makes the method particularly well-suited for Exploratory Data Analysis (EDA).

Contents

1. Acknowledgement

This report is based on the work "CUR Matrix Decompositions for Improved Data Analysis" by Michael W. Mahoney and Petros Drineas. We would like to acknowledge their significant contribution to the development of CUR matrix decomposition, particularly the randomized column sampling approach, which provided the foundation for our study.

We would also like to express our sincere gratitude to Prof. Kavita Sutar for her continuous support, insightful guidance, and encouragement throughout this project. Her mentorship has been invaluable in shaping the direction and success of this work.

2. Work Contribution

This project was a collaborative effort, with each member contributing their unique strengths to different aspects of the work:

- **Raja S** contributed to identifying the motivation and broader significance of CUR decomposition, helping to frame its relevance in modern data analysis contexts.
- **Pranav Pothan** focused on the mathematical foundations of the algorithm, including theoretical derivations and formal analysis.
- **Riya S Huddar** led the algorithmic implementation and experimental design, including visualization, interpretation of results, and dataset-driven insights.

All team members engaged deeply with the source literature and participated actively in discussions, writing, and finalizing the report.

3. Introduction

Modern datasets are often represented by large matrices of dimensions $m \times n$, where m represents the number of objects, and n represents the number of features. For example, datasets may consist of documents, genomes, or other forms of multivariate data. In many cases, an important step in the analysis of such data is to construct a compressed representation of the data matrix A that may be easier to analyze and interpret.

One popular method for dimensionality reduction is Singular Value Decomposition (SVD), which decomposes a matrix into three components: left singular vectors, singular values, and right singular vectors. The SVD of a general matrix A can be expressed as:

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

Where U represents the left singular vectors, Σ is a diagonal matrix with singular values, V represents the right singular vectors.

This decomposition is widely used in data analysis, often via methods such as PCA Principal Component Analysis (PCA), where truncating the SVD to a smaller number of components provides the best rank-k approximation of the data matrix A. Choosing k such that it provides a low-rank approximation while preserving as much variance in the data as possible is an essential part of this process.

Although Singular Value Decomposition (SVD) is effective for data analysis, it is often difficult to interpret because its components are linear combinations of almost all the data points. To address this, CUR decomposition was developed in the research paper, which explicitly expresses the data in terms of a small number of actual columns and/or rows of the data matrix. This improves both the interpretability and effectiveness of the data analysis.

4. Background

For example, in the Iris dataset, which contains 150 measurements of 4 features (sepal and petal length and width) for 3 distinct species of Iris, is a popular multivariate dataset often used in SVD applications. The first five rows of the dataset are shown below:

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2

Figure 1: Iris Dataset: Measurements of the three Iris species across 4 features.

After performing reduced SVD on the Iris dataset, we observe that the top singular vectors capture a significant amount of variance. However, these singular vectors are abstract mathematical constructs that do not easily correspond to physical or easily interpretable features of the data.

This lack of interpretability can be a challenge when the goal is to extract actionable insights from the data. To address this issue, we can use CUR matrix decomposition, a low-rank matrix decomposition that expresses the data matrix A in terms of a small number of actual rows and columns from the original matrix. Specifically, for an $m \times n$ matrix A, CUR decomposition expresses A as the product of three matrices:

$$A = CUR$$

Where C is a small number of actual columns of A, R is a small number of actual rows of A, U is a carefully constructed matrix that guarantees the product CUR approximates A.

U (150 x 4)

0	1	2	3
-0.061617	0.129611	0.002139	0.001638
-0.058071	0.111020	0.070672	0.051757
-0.056763	0.117966	0.004343	0.009557
-0.056653	0.105308	0.005925	-0.041644
-0.061230	0.131090	-0.031881	-0.032215

Figure 2: Left Singular Vectors (U) after performing SVD.

The extent to which the matrix A is approximated by CUR depends on the choice of C and R, as well as the construction of U. Because C and R consist of actual data elements, they are more interpretable and can be directly related to the field from which the data

V (4 x 4)

0	1	2	3
-0.751108	-0.380086	-0.513009	-0.167908
0.284175	0.546745	-0.708665	-0.343671
0.502155	-0.675243	-0.059166	-0.537016
0.320814	-0.317256	-0.480745	0.751872

Figure 3: Right Singular Vectors (V) after performing SVD.

originates. This is in contrast to the abstract nature of the components derived from SVD.

Thus, CUR matrix decompositions offer a more interpretable alternative to traditional SVD, providing a better understanding of the data through actual rows and columns from the dataset.

5. CUR Decomposition

5.1 Algorithm

To address this interpretability issue in Data Analysis, CUR matrix decompositions, i.e., low-rank matrix decompositions that are explicitly expressed in terms of a small number of actual columns and/or actual rows of the original data matrix.

The CUR decomposition approximates a matrix $A \in \mathbb{R}^{m \times n}$ as $A \approx CUR$, where:

- $C \in \mathbb{R}^{m \times c}$ contains c columns selected from A
- $R \in \mathbb{R}^{r \times n}$ contains r rows selected from A
- $U \in \mathbb{R}^{c \times r}$ is a carefully constructed matrix that guarantees that the product CUR is "close" to A (often C^+AR^+)

This, C and/or R can be used in place of the eigen columns and eigen rows, since they consist of actual data elements they will be interpretable in terms of the field from which the data are drawn.

The extent to which $A \approx CUR$, depends sensitively on the choice of C and R, as well as on the construction of U. Hence to compute this C and R there are certain works have already been performed.

5.2 Prior Works

Frieze, Kannan, and Vempala randomly sampled columns of A according to a probability distribution that depends on the Euclidean norms of those columns.

Worst-case additive-error guarantees of the form,

$$\|A - \text{PCA}\|_F \leq \|A - A_k\|_F + \epsilon \|A\|_F$$

Drineas, Kannan, and Mahoney have chosen the columns and rows simultaneously based on the same probability distribution used above and constructed CUR matrix decomposition.

Worst-case additive-error guarantees of the form,

$$\|A - CUR\|_F \leq \|A - A_k\|_F + \epsilon \|A\|_F$$

But these additive-error matrix decompositions are, however, quite coarse in the worse case. Moreover, the insights provided by their sampling probabilities into the data are limited—the probabilities are often uniform due to data preprocessing.

6. Improved CUR Decomposition

To construct C, we compute an "importance score" for each column of A. A small number of columns are randomly sampled from A, using those scores as an importance sampling probability distribution.

We aim to choose columns that exert a disproportionately large influence on the best low-rank fit, so that CUR closely approximates A_k and captures the dominant spectral components of A.

Statistical Leverage Scores Computation

We know from the Singular Value Decomposition (SVD):

$$A = U\,\Sigma\,V^\top$$

For a rank-k approximation, we truncate to:

$$A_k = U_k\,\Sigma_k\,V_k^\top$$

To build intuition, we examine the matrix product $U\Sigma$:

$$U\,\Sigma = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1r} \\ u_{21} & u_{22} & \cdots & u_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & \cdots & u_{mr} \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \end{bmatrix}$$

$$= \begin{bmatrix} \sigma_1 u_{11} & \sigma_2 u_{12} & \cdots & \sigma_r u_{1r} \\ \sigma_1 u_{21} & \sigma_2 u_{22} & \cdots & \sigma_r u_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_1 u_{m1} & \sigma_2 u_{m2} & \cdots & \sigma_r u_{mr} \end{bmatrix}$$

Right singular vectors are given by V and its transpose:

$$V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1r} \\ v_{21} & v_{22} & \cdots & v_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nr} \end{bmatrix}, \quad V^\top = \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{n1} \\ v_{12} & v_{22} & \cdots & v_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1r} & v_{2r} & \cdots & v_{nr} \end{bmatrix}$$

Putting it all together, we reconstruct A:

$$A = \begin{bmatrix} \sigma_1 u_{11} & \sigma_2 u_{12} & \cdots & \sigma_r u_{1r} \\ \sigma_1 u_{21} & \sigma_2 u_{22} & \cdots & \sigma_r u_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_1 u_{m1} & \sigma_2 u_{m2} & \cdots & \sigma_r u_{mr} \end{bmatrix} \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{n1} \\ v_{12} & v_{22} & \cdots & v_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1r} & v_{2r} & \cdots & v_{nr} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{\xi=1}^{r} \sigma_\xi u_{1\xi} v_{1\xi} & \sum_{\xi=1}^{r} \sigma_\xi u_{1\xi} v_{2\xi} & \cdots \\ \sum_{\xi=1}^{r} \sigma_\xi u_{2\xi} v_{1\xi} & \sum_{\xi=1}^{r} \sigma_\xi u_{2\xi} v_{2\xi} & \cdots \\ \vdots & \vdots & \vdots \\ \sum_{\xi=1}^{r} \sigma_\xi u_{m\xi} v_{1\xi} & \sum_{\xi=1}^{r} \sigma_\xi u_{m\xi} v_{2\xi} & \cdots \end{bmatrix}$$

We now take a column-wise view of the matrix A:

$$A^j = \sum_{\xi=1}^{r} \sigma_\xi u^\xi v_j^\xi$$

$$A^j = \sum_{\xi=1}^{k} \sigma_\xi u^\xi v_j^\xi$$

Thus, the j^{th} column of A is a linear combination of the left singular vectors, weighted by the singular values and the components of the j^{th} column of $V^\top$.

Thus normalized statistical leverage scores can be computed from V_j's i.e. the right singular vectors, A:

$$\pi_j = \frac{1}{k} \sum_{\xi=1}^{k} (v_j^\xi)^2$$

7. Mathematical Analysis

7.1 Column Sampling Techniques

7.1.1 Deterministic Column Sampling

Intuitive idea: We should select columns in the decreasing order of π_j.

Problems:

1. **Loss of diversity:** The chance of selecting redundant columns similar to each other increases. Columns that align similarly with a dominant singular vector will have similar leverage scores, so they'll both be selected, even though they contribute very little additional information.
2. **Outlier sensitivity:** No room or flexibility to adjust to noisy columns. Outliers with high leverage may dominate the selection.

7.1.2 Randomized Column Sampling

Key idea: Some columns might have low leverage scores but are still important for capturing variability or diversity in the dataset.

Advantages:

1. **Diverse selection:** Rather than the top-ranked columns, we select a diverse subset of columns.
2. **Provable guarantees on low error approximation:** With high probability, the matrix C (formed from sampled columns) will approximate the span of the top k singular vectors of A within a small relative error—often a $(1+\varepsilon)$-approximation.

7.2 Properties of π_j

We observe that π_j has the following properties:

1. $\pi_j \geq 0$
2. $\sum_{j=1}^{n} \pi_j = 1$

Explanation:

$$\sum_{j=1}^{n} \pi_j = \sum_{j=1}^{n} \frac{1}{k} \sum_{\xi=1}^{k} (v_{\xi j})^2 = \frac{1}{k} \sum_{\xi=1}^{k} \sum_{j=1}^{n} (v_{\xi j})^2$$

Each right singular vector v_ξ is a unit vector:

$$\sum_{j=1}^{n}(v_{\xi j})^2 = \|v_\xi\|^2 = 1$$

Therefore:

$$\sum_{j=1}^{n}\pi_j = \frac{1}{k}\cdot k = 1$$

Hence, π_j forms a probability distribution over the n columns.

We want our algorithm to choose columns based on this probability.

7.3 Choosing the Number of Columns

Let c be an upper bound on the number of columns to be sampled.

Factors to be considered:

1. **Error ε:** As the bandwidth of the allowed error decreases, we need to choose more columns.
$$c \propto \frac{1}{\varepsilon} \quad \text{or} \quad c \propto \frac{1}{\varepsilon^2}$$
2. **Rank of the approximation k:** As the rank of the approximation increases, we need more columns.
$$c \propto k \quad \text{(but in practice, more than } k \text{ columns are usually needed)}$$

It can be shown that by selecting

$$c = O\left(\frac{k\log k}{\varepsilon^2}\right)$$

we can approximately span the top k singular vectors of A within $(1+\varepsilon)$ relative error.

Since sampling is randomized, let c' be the actual number of columns selected. We want:

$$\mathbb{E}[c'] \leq c$$

Let p_j be the probability of choosing column j. We choose:

$$p_j = \min\{1, c\pi_j\}$$

Define indicator random variable X_j for each column j:

$$X_j = \begin{cases} 1 & \text{if column } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

Then:

$$c' = \sum_{j=1}^{n} X_j$$

Since X_j is a Bernoulli random variable with parameter p_j, we have:

$$\mathbb{E}[X_j] = p_j = \min\{1, c\pi_j\}$$

Thus:

$$\mathbb{E}[c'] = \sum_{j=1}^{n} \mathbb{E}[X_j] = \sum_{j=1}^{n} \min\{1, c\pi_j\} \leq c$$

Case 1: All $c\pi_j < 1$

Then $\boldsymbol{p_j = \min(1, c\pi_j) = c\pi_j}$

$$\mathbb{E}[\boldsymbol{c'}] = \sum_{j=1}^{\boldsymbol{n}} \boldsymbol{c\pi_j} = \boldsymbol{c} \sum_{j=1}^{\boldsymbol{n}} \boldsymbol{\pi_j} = \boldsymbol{c} \cdot \frac{\boldsymbol{k}}{\boldsymbol{k}} = \boldsymbol{c}$$

Case 2: Some $c\pi_j \geq 1$

Then some $\boldsymbol{p_j}$ become $\mathbf{1}$:

$$\Rightarrow \mathbb{E}[\boldsymbol{c'}] < \boldsymbol{c}$$

From this we can conclude that:

$$\mathbb{E}[\boldsymbol{c'}] \leq \boldsymbol{c}$$

7.4 Implications

This algorithm computes $\boldsymbol{p_j}$ for all columns and chooses columns in the following way:

Case 1: $c\pi_j \geq 1$

$$\boldsymbol{p_j = \min(1, c\pi_j) = 1}$$

These form a small number of columns with high leverage scores ($\boldsymbol{\pi_j}$) and are always chosen.

Case 2: $c\pi_j < 1$

$$\boldsymbol{p_j = \min(1, c\pi_j) = c\pi_j}$$

These columns are chosen with probability $\boldsymbol{c\pi_j}$, but are not always selected.

7.5 Relative Error Calculation

C = Matrix containing selected columns
R = Matrix containing selected rows
$P_C = CC^+$ where C^+ is the pseudoinverse of C
$P_R = R^+R$ where R^+ is the pseudoinverse of R
ε = Approximation error tolerance
A_k = Best rank-k approximation to A

With high probability (at least 99%), we have:

$$\|A - P_C A\|_F \leq \left(1 + \frac{\varepsilon}{2}\right) \|A - A_k\|_F$$

$$\|A - AP_R\|_F \leq \left(1 + \frac{\varepsilon}{2}\right) \|A - A_k\|_F$$

7.5.1 Proof

Let:

$$A = A_k + A_\perp$$

where:

- A_k is the best rank-k approximation of A
- $A_\perp = A - A_k$ is orthogonal to the top-k subspace

Let P_C be a projection matrix onto the column space of a sampled matrix C.

We want to bound the squared projection error:

$$\|A - P_C A\|_F^2$$

Decompose and apply linearity:

$$\|A - P_C A\|_F^2 = \|(I - P_C)(A_k + A_\perp)\|_F^2 = \|(I - P_C)A_k + (I - P_C)A_\perp\|_F^2$$

Use approximate orthogonality (Pythagorean-like inequality):

$$\|A - P_C A\|_F^2 \leq \|(I - P_C)A_k\|_F^2 + \|(I - P_C)A_\perp\|_F^2$$

Now, bound each term:

- If C captures the top-k space well:

$$\|(I - P_C)A_k\|_F^2 \leq \varepsilon \|A - A_k\|_F^2$$

- Since $A_\perp$ is orthogonal to the top-k space:

$$\|(I - P_C)A_\perp\|_F \leq \|A_\perp\|_F = \|A - A_k\|_F \Rightarrow \|(I - P_C)A_\perp\|_F^2 \leq \|A - A_k\|_F^2$$

Add both bounds:

$$\|A - P_C A\|_F^2 \leq (\varepsilon + 1)\|A - A_k\|_F^2$$

Now take square root:

$$\|A - P_C A\|_F \leq \sqrt{1 + \varepsilon}\|A - A_k\|_F$$

Using the inequality $\sqrt{1 + \varepsilon} \leq 1 + \frac{\varepsilon}{2}$ for small ε, we get:

$$\boxed{\|A - P_C A\|_F \leq \left(1 + \frac{\varepsilon}{2}\right) \|A - A_k\|_F}$$

Let $A = CUR$, where $U = C^+ AR^+$. Then:

$$\|A - CUR\|_F = \|A - CC^+ AR^+ R\|_F$$

Add and subtract $CC^+ A$, and apply the triangle inequality:

$$\begin{aligned}
\|A - CUR\|_F &= \|A - CC^+ A + CC^+ A - CC^+ AR^+ R\|_F \\
&\leq \|A - CC^+ A\|_F + \|CC^+ A - CC^+ AR^+ R\|_F \\
&\leq \|A - CC^+ A\|_F + \|A - AR^+ R\|_F \\
&= \|A - P_C A\|_F + \|A - AP_R\|_F \\
&\leq \left(1 + \frac{\varepsilon}{2}\right) \|A - A_k\|_F + \left(1 + \frac{\varepsilon}{2}\right) \|A - A_k\|_F \\
&= (2 + \varepsilon)\|A - A_k\|_F
\end{aligned}$$

Thus, with 98% probability:

$$\|A - CUR\|_F \leq (2 + \varepsilon)\|A - A_k\|_F$$

This follows because the probability of both the column selection and the row selection failing is at most $1\% + 1\% = 2\%$.

7.6 Column Selection Without Scaled π_j

To obtain a rank-k approximation of a matrix A, we must sample at least k columns from A. When sampling is done *without scaling*, we sample k columns with replacement, where each column j is selected with probability π_j (typically based on leverage scores).

The expected number of times column j is selected is:

$$\mathbb{E}[X_j] = k \cdot \pi_j$$

Thus, the expected total number of selected columns is:

$$\mathbb{E}[c'] = \mathbb{E}\left[\sum_{j=1}^{n} X_j\right] = \sum_{j=1}^{n} \mathbb{E}[X_j] = \sum_{j=1}^{n} k \cdot \pi_j = k$$

7.6.1 Implications of Sampling Without Scaled π_j

1. **k-column choice:** The selected k columns may not correspond to the top singular vectors, which provide the best rank-k approximation of A.
2. **Error bounds:** The approximation quality no longer satisfies the typical guarantee:
   $$\|A - CUR\|_F \leq (2 + \varepsilon)\|A - A_k\|_F$$
   due to lack of scaling and structured randomness.
3. **Biased approximation:** Without proper scaling, the resulting CUR decomposition may be biased. Columns with higher leverage scores might dominate the approximation, leading to distortion in the reconstructed matrix.

8. Algorithmic Implementation

8.1 Overview of the CUR Decomposition

The CUR matrix decomposition approximates a matrix $A \in \mathbb{R}^{m \times n}$ using a small number of its actual columns and rows. Specifically, it expresses A as:

$$A \approx CUR$$

Here, C is formed from actual columns of A, R from actual rows, and U is a small linking matrix typically computed via pseudoinverses.

8.2 Tools and Libraries Used

The implementation was done in Python 3.10, using the following libraries:

- **NumPy** – for matrix operations, pseudoinverse computation via `np.linalg.pinv()`
- **Pandas** – for data loading, preprocessing, and manipulation
- **Matplotlib/Seaborn** – for optional visualizations

8.3 Algorithm Steps

The CUR matrix decomposition was implemented using leverage score-based probabilistic sampling of columns and rows. The steps are outlined as follows:

1. **Input:** A matrix $A \in \mathbb{R}^{m \times n}$, target rank k, and error parameter ϵ.
2. **Compute SVD:** Perform SVD on A to obtain $A = U\Sigma V^T$.
3. **Compute Leverage Scores:**
   - Take the top-k rows of V^T, denoted V_k^T.
   - Compute the leverage scores for each column j as:

$$\pi_j = \frac{1}{k} \sum_{i=1}^{k} V_{ij}^2$$

4. **Determine Sampling Probability:**
   - Calculate scaling factor $c = \frac{k \log k}{\epsilon^2}$

- Selection probabilities:

$$p_j = \min(1, c \cdot \pi_j)$$

5. **Column Selection:** Select each column j of A independently with probability p_j to form matrix C.
6. **Row Selection:** Apply the same procedure to A^T (using V from SVD of A^T) to select rows for matrix R.
7. **Intersection Matrix:** Use pseudoinverses of C and R to compute matrix U:

$$U = C^+ A R^+$$

   where C^+ and R^+ denote the Moore-Penrose pseudoinverses.
8. **Output:** The CUR approximation $A \approx CUR$

9. Results and Inference

To evaluate the performance of the CUR decomposition algorithm, we conducted experiments on randomly generated matrices with varying configurations. The approximation quality of CUR was compared against the optimal rank-k approximation obtained from Singular Value Decomposition (SVD).

9.1 Approximation Error Comparison

We first applied the CUR decomposition on a randomly generated matrix $A \in \mathbb{R}^{100\times100}$ using $k = 2$ and $\epsilon = 0.5$. The CUR approximation was compared to the best rank-k SVD approximation in terms of Frobenius norm error:

- **CUR Approximation Error:** 33.80
- **SVD Rank-k Approximation Error:** 27.98

We checked whether the CUR error obeys the theoretical upper bound of $(2 + \epsilon)$ times the optimal SVD error:

`Is CUR error bounded by` $(2+\epsilon)\times$ `SVD error?` **True**

9.2 Robustness over Multiple Trials

To assess the stability of the CUR approximation, we repeated the experiment 1000 times on a random matrix $B \in \mathbb{R}^{100\times100}$ with entries sampled from a standard normal distribution scaled by 10. In each trial, the CUR error was compared with the SVD error using the same theoretical bound.

- **CUR failed to satisfy the bound in 0 out of 1000 trials.**
- **Failure rate:** 0.00%

This demonstrates that the CUR approximation is highly reliable and consistently adheres to the expected error bounds, even under randomness in both the input matrix and the sampling.

9.3 Leverage Score Visualization

To further understand the column sampling mechanism, we analyzed the scaled leverage scores $c \cdot \pi_j$ for a matrix $A \in \mathbb{R}^{250 \times 250}$ and visualized them for $k = 2$ and $\epsilon = 0.1$.

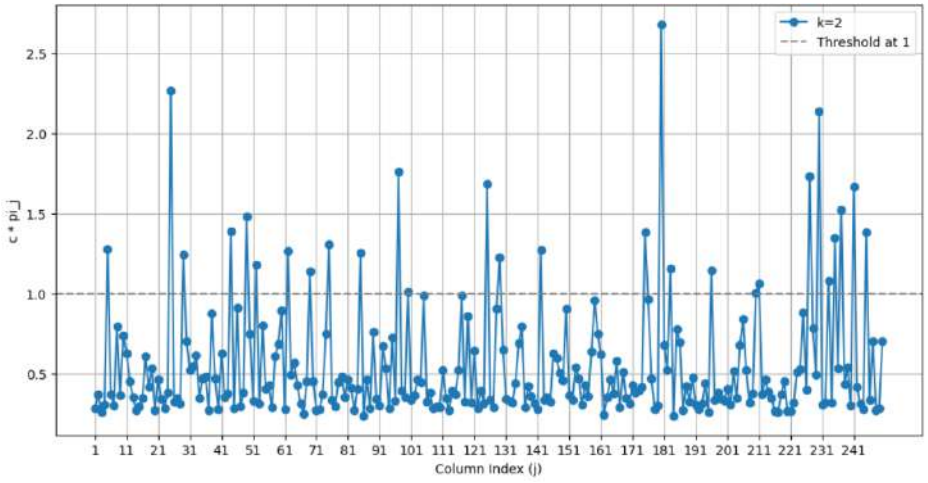

Figure 4: Plot of scaled leverage scores $c \cdot \pi_j$ versus column index for $k = 2$

From the plot, we observe the following:

- A significant number of columns have $c \cdot \pi_j > 1$, and these columns will always be selected. This occurs because the probability of selecting a column j, denoted by p_j, is given by the formula:

  $$p_j = \min(1, c \cdot \pi_j)$$

  When the scaled leverage score $c \cdot \pi_j$ exceeds 1, the probability of selecting the column is capped at 1, meaning these columns will always be chosen for the CUR decomposition. This behavior ensures that the most significant columns, based on leverage scores, are consistently selected, potentially leading to a more accurate approximation of the matrix.

- The distribution of leverage scores is not uniform, emphasizing the importance of data-driven sampling.
- The threshold line at 1 clearly distinguishes columns with high versus low inclusion probability.

These results validate the role of leverage scores in focusing sampling on "important" directions in the data, thus justifying the use of randomized sampling in CUR without substantial loss of approximation quality.

9.4 Column Norms vs Scaled Leverage Scores

In this section, we examine the relationship between the normalized column norms of matrix A and the scaled leverage scores $c \cdot \pi_j$, where π_j is the leverage score of the j-th column. While it may be tempting to assume that columns with larger norms are more "important", leverage scores offer a more principled way of identifying significant columns based on the matrix's low-rank structure.

Methodology

- **SVD Calculation:** We computed the Singular Value Decomposition (SVD) of matrix A and extracted the top k right singular vectors. These were used to calculate the leverage scores π_j for each column.
- **Scaling Factor c:** We scaled each π_j by $c = \frac{k \log k}{\epsilon^2}$ to obtain $c \cdot \pi_j$, which defines the sampling probabilities in the CUR decomposition.
- **Column Norms:** Column norms were calculated and normalized so that their sum equals 1, enabling fair visual comparison across all columns.

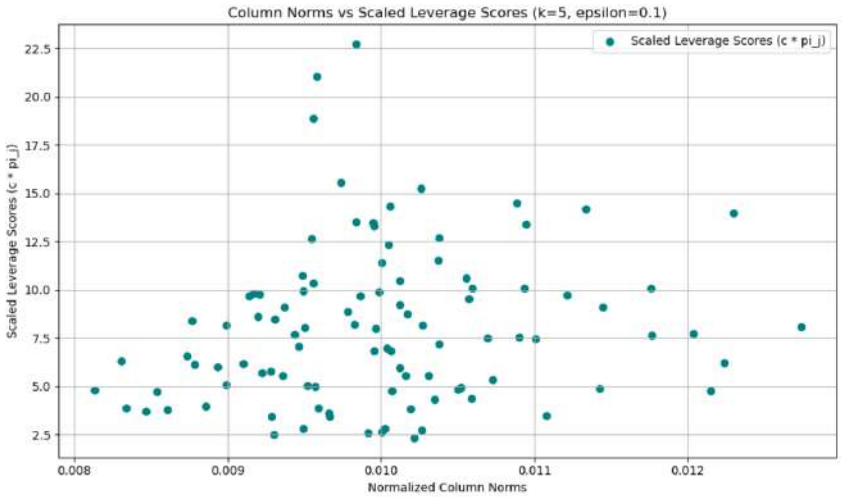

Figure 5: Scatter plot of normalized column norms vs scaled leverage scores $c \cdot \pi_j$. Each point represents a column of matrix A.

Plot Observations Interestingly, the scatter plot reveals 'no significant correlation' between column norms and scaled leverage scores. This implies that:

- **Column Magnitude $\neq$ Column Importance:** Columns with high norms do not necessarily have high leverage scores. Hence, selecting columns based solely on norm can be misleading.
- **Leverage Scores Capture Structural Importance:** Unlike norms, leverage scores take into account the geometry of the matrix's top k singular vectors, leading to a more informed column selection strategy.
- **Non-trivial Selection Patterns:** Several columns with moderate or even low norms end up with high scaled leverage scores (possibly exceeding 1), suggesting these are structurally significant despite appearing "small".

Key Insight The lack of correlation between column norms and leverage scores highlights the need for smarter sampling techniques like those based on leverage scores in CUR decomposition. Relying on norms alone could miss structurally important columns and degrade approximation quality.

9.5 Visualization of Column Selection Probabilities

To better understand the stochastic nature of column selection in CUR decomposition, we visualize the selection probabilities computed from the leverage scores.

Given:

- Input matrix $A \in \mathbb{R}^{250\times 250}$
- Target rank $k = 2$
- Error tolerance $\varepsilon = 0.5$
- Random seed: 42 (to ensure reproducibility)

The top k right singular vectors of A are used to compute the leverage scores π_j, for each column j. These are defined as:

$$\pi_j = \frac{1}{k}\sum_{i=1}^{k}(v_{ij})^2$$

where v_{ij} are entries from the top k rows of $V^\top$ in the SVD of A.

The selection probabilities are then given by:

$$p_j = \min(1, c\cdot\pi_j), \quad \text{where} \quad c = \frac{k\log k}{\varepsilon^2}$$

For our values, $c = \frac{2\cdot\log 2}{0.5^2} \approx 5.55$.

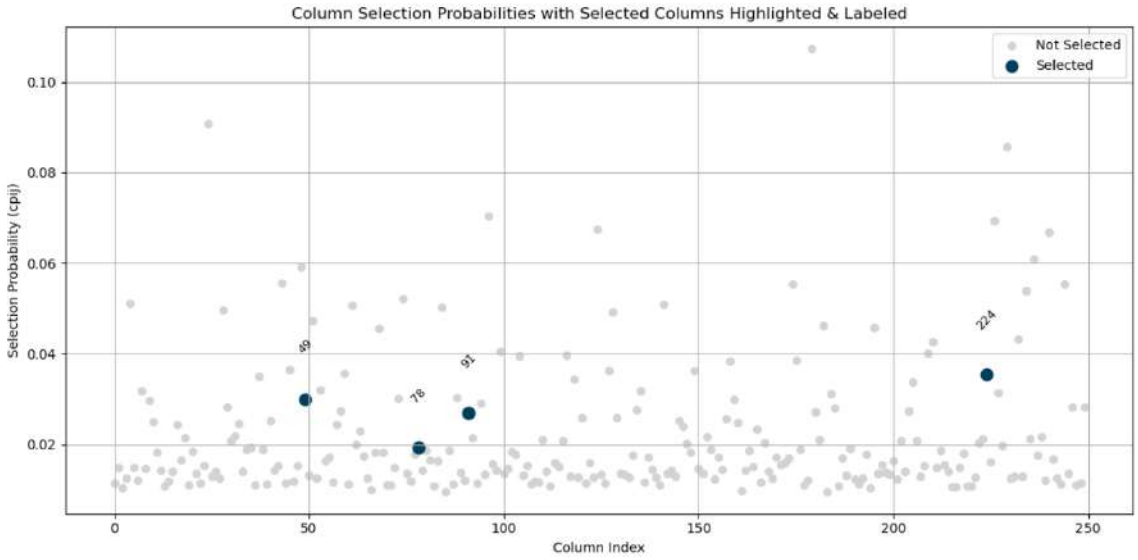

Figure 6: Selection probabilities p_j for all $j = 1, \ldots, 250$. Selected columns are highlighted and annotated.

In Figure 6, we observe that columns with high leverage scores correspond to higher selection probabilities, making them more likely to be selected. However, due to the randomized sampling process, even columns with moderate probabilities can be selected. This visualization confirms that our implementation of leverage score-based sampling is both probabilistically sound and practically interpretable.

9.6 Scaled Leverage Scores with Varying k Values

We now explore the behavior of the scaled leverage scores $c \cdot \pi_j$ for different values of k, the target rank, in the context of CUR decomposition. The leverage scores are computed from the top k right singular vectors of the input matrix A. The scaling factor c is calculated as:

$$c = \frac{k \log k}{\varepsilon^2}$$

where $\varepsilon = 0.1$ is a fixed error tolerance and we set the matrix dimensions as $m = n = 250$. In our experiment, we vary k over the values $k = 2, 5, 10, 20$ and visualize the corresponding scaled leverage scores.

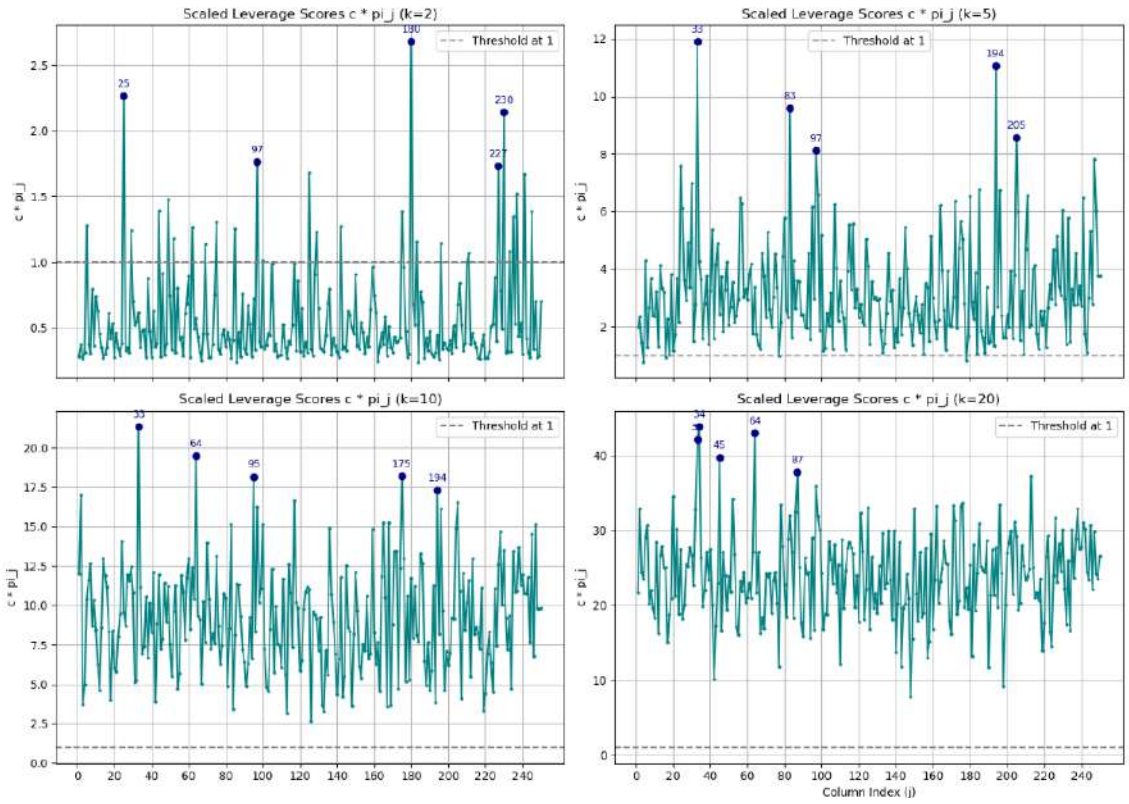

Figure 7: Scaled leverage scores $c \cdot \pi_j$ for different values of k. The top 5 columns with the highest scores are highlighted and annotated. The threshold at 1 is marked with a dashed gray line.

Each subplot in Figure 7 shows the scaled leverage scores for a particular value of k. As k increases, the scaling factor c increases, making the leverage scores larger. The top 5 columns with the highest leverage scores are annotated in navy on each plot. These columns represent those that contribute the most to the matrix approximation and are thus most likely to be selected in the CUR decomposition.

9.7 Scaled Leverage Scores for Different ε Values

In this section, we investigate the effect of varying ε values on the scaled leverage scores $c \cdot \pi_j$, where c is the scaling factor given by:

$$c = \frac{k \log k}{\varepsilon^2}$$

For the experiment, we set the matrix dimensions as $m = n = 250$, and $k = 10$ for comparison. The values of ε used are: $\varepsilon = 0.01, 0.1, 0.3, 0.4, 0.5, 1.0$. For each value of ε, we compute the scaled leverage scores $c \cdot \pi_j$ and plot them.

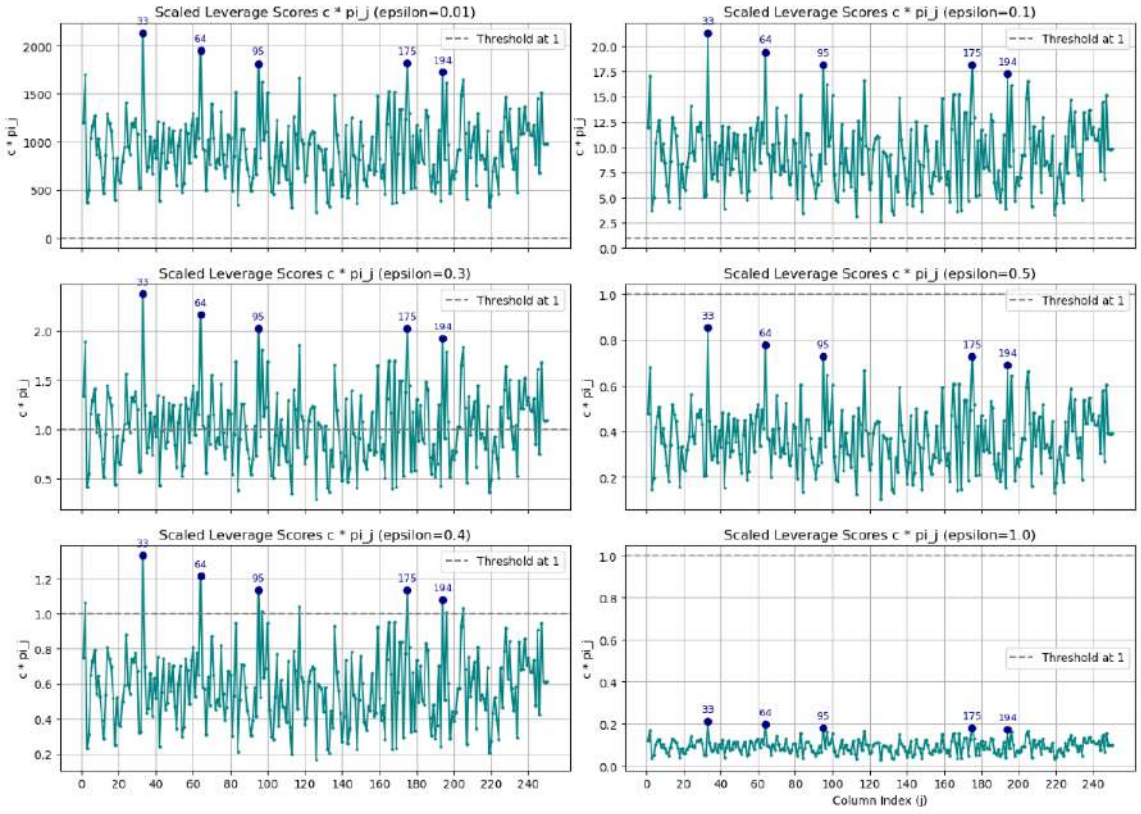

Figure 8: Scaled leverage scores $c \cdot \pi_j$ for different ε values. The top 5 columns with the highest scores are highlighted and annotated. The threshold at 1 is marked with a dashed gray line.

In Figure 8, each subplot corresponds to a different value of ε. As ε decreases, the scaling factor c increases, causing the scaled leverage scores $c \cdot \pi_j$ to become larger. The columns with the highest scores are annotated in navy color on each plot. These columns are the most influential for the matrix approximation and are more likely to be selected in the CUR decomposition.

9.8 CUR Decomposition Visualization

In this subsection, we present the visualization of the CUR decomposition of a sample matrix A with dimensions 10×10. The rows and columns selected by the CUR decomposition are color-coded to distinguish between the rows, columns, and their overlap. The visualization was generated using the CUR decomposition algorithm with a chosen rank $k = 2$ and epsilon $\epsilon = 0.5$. Since the algorithm is probabilistic, the selected rows and columns may change each time the decomposition is run, resulting in variations in the color coding.

In the plot:

- The light purple color represents the rows selected by the CUR decomposition.
- The soft green color represents the columns selected by the CUR decomposition.

- The lavender color represents the overlap, where a row and a column are both selected at the same position.

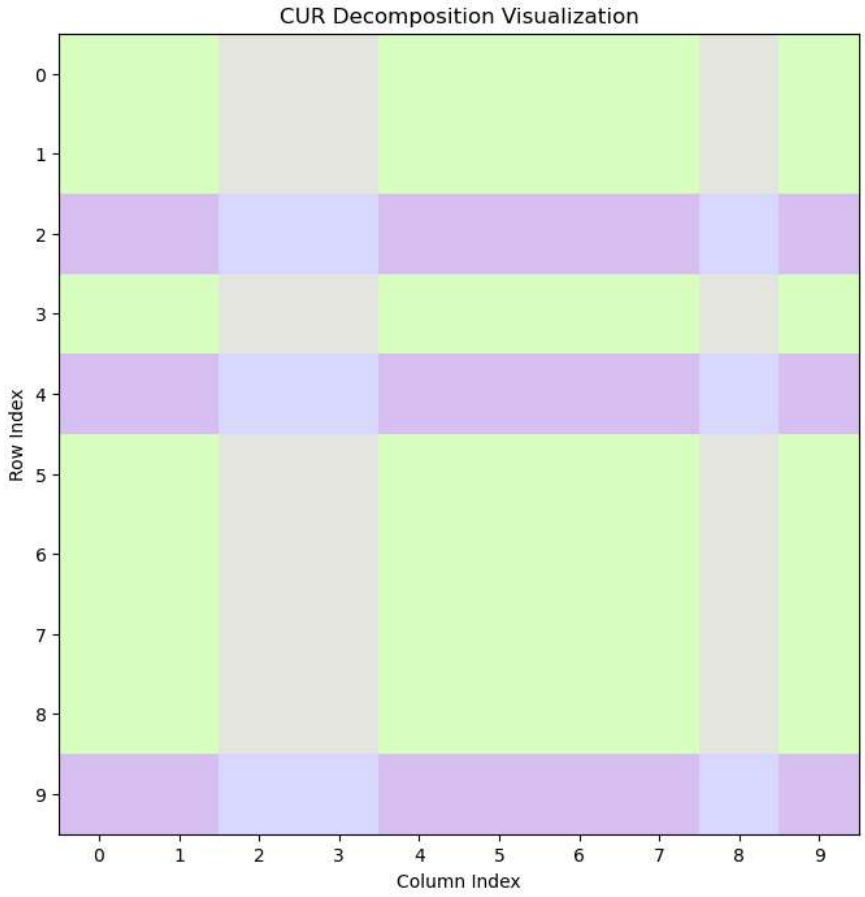

Figure 9: CUR Decomposition Visualization for a sample matrix A. The light purple color represents the selected rows, the soft green represents the selected columns, and the lavender color indicates the overlap where both a row and column are selected. The dimensions of the matrix are 10×10, and the selection of rows and columns can vary with each run due to the probabilistic nature of the CUR decomposition algorithm.

Since CUR decomposition is probabilistic, the rows and columns selected may vary with each run. This makes the results inherently dynamic, providing different insights depending on the random selections made during each execution of the algorithm.

9.9 Column Selections in CUR Decomposition Over Multiple Runs

In this experiment, we applied the CUR decomposition algorithm to a matrix A with dimensions 250×250, a rank parameter $k = 3$, and a tolerance parameter $\epsilon = 0.5$. The goal was to investigate the number of columns selected in the decomposition over multiple runs.

We ran the CUR decomposition algorithm 100 times, each time randomly selecting a set of columns based on the probabilistic procedure.

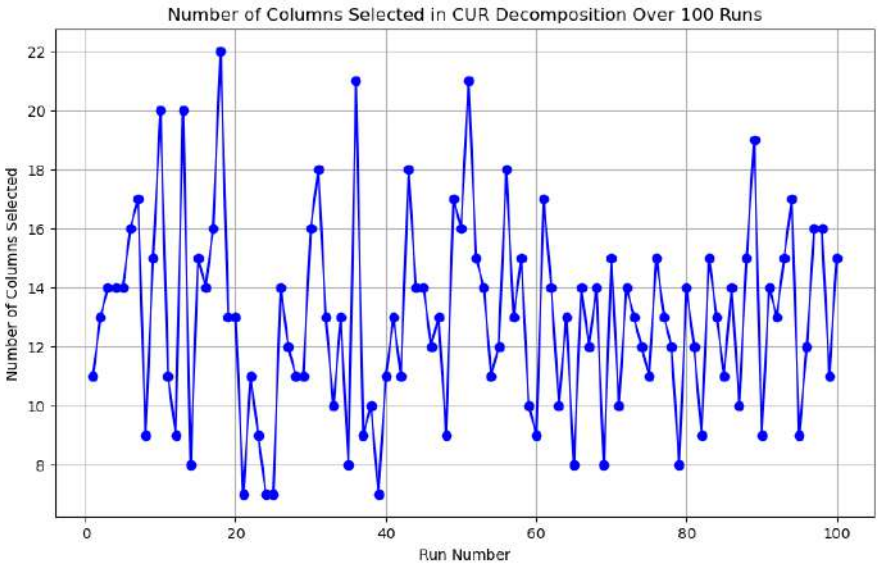

Figure 10: Number of Columns Selected in CUR Decomposition Over 100 Runs. The plot shows the number of columns selected in each run, demonstrating the variation across different runs. Despite fluctuations, the number of selected columns is concentrated around its expected value.

The plot above shows the number of columns selected in each of the 100 runs of CUR decomposition. As shown, the number of columns selected fluctuates but is tightly concentrated around a certain value (14.00).

In CUR decomposition, the matrix C contains c columns, where c is the number of columns selected during the decomposition. From the results of the experiment, we find that the expected number of columns selected, denoted $c_{\text{in expectation}}$, is approximately $c_{\text{in expectation}} = 14.00$, as determined by the rank parameter k and error tolerance $\epsilon = 0.5$. In practice, c remains tightly concentrated around its expectation, i.e., $c \leq c_{\text{in expectation}}$, even with probabilistic variability in each run.

As shown in the plot, the number of columns selected varies slightly with each run, but it stays close to its expected value of 14, consistent with the probabilistic nature of the algorithm. This demonstrates that despite random column selections, the procedure tends to converge to a stable number of selected columns, which is crucial for the stability and efficiency of the CUR decomposition.

9.10 GIST Dataset

The dataset used in this study is obtained from the Gene Expression Omnibus (GEO) under the accession number **GSM77772**. It contains dual-channel microarray expression data derived from a study on gastrointestinal stromal tumors (GIST). Specifically, it profiles mRNA from a primary tumor sample (STT646) harboring a mutation in exon 11.

Channel 1 (Cy3) contains mRNA from a control or reference sample (CRF), while Chan-

nel 2 (Cy5) includes mRNA from the tumor tissue. The expression values are reported for approximately 24,192 probes, each corresponding to specific genes or gene fragments. The dataset includes background-corrected and normalized signal intensities for both channels, and expression changes are calculated as the log base 2 ratio of Cy5 to Cy3 intensities, i.e., $\log_2(\text{Cy5}/\text{Cy3})$.

This sample was processed using the Stanford Microarray Database and the microarray platform **GPL2935**. It provides a valuable resource for investigating gene expression alterations associated with specific genetic mutations in GIST tumors.

9.10.1 Data Preprocessing and Subsetting

The raw dataset obtained from GEO contains multiple metadata fields, control probes, and missing values that are not directly relevant to matrix decomposition analysis. Therefore, a data preprocessing pipeline was implemented using Python and the `pandas` library.

First, all metadata lines prefixed with the `#` symbol were excluded during loading. The first column, labeled `ID_REF`, containing probe identifiers, was also removed to retain only numerical expression values. Subsequently, all columns containing missing values (`NaN`) were discarded to ensure matrix integrity for downstream matrix factorization algorithms.

The resulting cleaned data matrix, denoted as $A \in \mathbb{R}^{m \times n}$, was then subset to retain only the first 18 columns for computational efficiency. This submatrix was labeled `Gene_dataset_subset` and used as the input for the CUR matrix decomposition algorithm.

```
# Example code snippet
df = pd.read_csv("sample_dataset.txt", sep="\t", comment="#")
df.drop(columns=['ID_REF'], inplace=True)
df.dropna(axis=1, inplace=True)
Gene_dataset = df.values
Gene_dataset_subset = Gene_dataset[:, :18]
```

The matrix `Gene_dataset_subset` was subsequently factorized using the CUR decomposition algorithm with rank parameter $k = 2$ and approximation threshold $\varepsilon = 0.3$. This yielded three matrices C, U, and R such that $A \approx CUR$. The selected rows and columns were printed for interpretability.

Results for one run:

```
Selected Columns Indices: [0 1 4 5]
Selected Rows Indices: [8 13 16 18 21 24 26]
```

9.10.2 CUR Decomposition and Frequency Analysis of Selected Rows and Columns on the subset of GIST dataset

To analyze the frequency with which rows and columns are selected during multiple runs of the CUR decomposition, we performed 10,000 iterations on the gene expression dataset subset. By tracking the frequency of selection for each row and column, we can identify which parts of the dataset are most influential.

Methodology: In this analysis:

- Two arrays, `column_counts` and `row_counts`, were initialized to track the selection frequency of each column and row.
- For each iteration, the CUR decomposition was applied, and the selected rows and columns were recorded.
- After completing 10,000 iterations, we calculated the frequency of selection for each row and column.

Results: The following plots display the frequency with which each column and row were selected across the 10,000 iterations. The first plot represents the selection frequency of each column, and the second plot represents the selection frequency of each row.

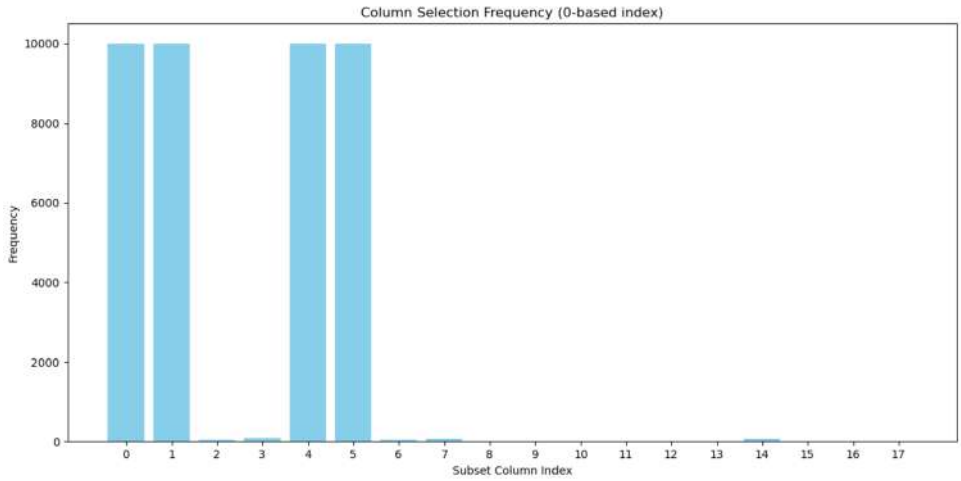

Figure 11: Frequency of Column Selection across 10,000 Iterations

Interpretation: From the plots, it is evident that certain rows and columns were selected significantly more often than others. This indicates that these rows and columns play a more central role in the dataset's structure and may contain more informative or influential features.

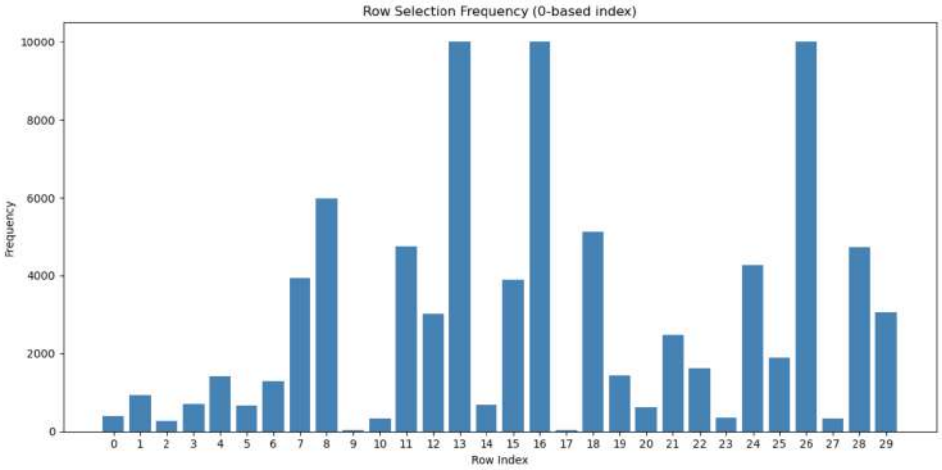

Figure 12: Frequency of Row Selection across 10,000 Iterations

Subset Column Index	Original Column Index	Column Name
0	1	CH1I_MEAN
1	2	CH2I_MEAN
4	5	CH1D_MEAN
5	6	CH2D_MEAN

Table 1: Mapping of subset column indices to original dataset columns

Next Steps: Based on the frequency of selection, we can refine our analysis by focusing on the rows and columns that were selected more frequently, as they are likely to be more significant. Rows and columns with low selection frequencies can potentially be excluded from further analysis, which would reduce the complexity and improve the efficiency of subsequent analyses.

The plots provided offer valuable insights into the parts of the dataset that contribute most to the CUR decomposition, guiding further exploration and modeling.

10. Conclusion

- CUR decomposition is an effective and interpretable method for low-rank matrix approximation, relying on leverage score-based sampling to select significant columns.
- Contrary to expectations, there is no consistent correlation between column norms and their corresponding leverage scores. This indicates that even columns with low or moderate norms could have high leverage scores, suggesting their importance in the matrix approximation process.
- The probabilistic column selection through leverage score sampling ensures that CUR decomposition is a robust approach for approximating matrices effectively.
- In an experiment varying the target rank k and error tolerance ϵ, we investigated how the number of columns selected changes across multiple iterations.
- Another experiment focused on the number of iterations required for a fixed value of k and ϵ. We observed that the number of selected columns stabilizes around its expected value as the iterations progressed ,further validating the practical applicability and efficiency of CUR decomposition.
- The CUR decomposition method was applied to a gene expression dataset from gastrointestinal stromal tumors (GIST). Frequency analysis of row and column selections revealed the most influential features in the dataset, providing valuable insights into its structure.
- Overall, CUR decomposition, particularly with leverage score-based sampling, is a powerful technique for matrix approximation that reveals hidden insights from complex datasets.
- Future work could focus on further refining the selection process by adapting it to the structural properties of the data to improve efficiency and accuracy.

11. References

1. Mahoney, M.W. and Drineas, P., 2009. CUR matrix decompositions for improved data analysis. Proceedings of the National Academy of Sciences, 106(3), pp.697-702.

2. Nielsen TO, West RB, Linn SC, Alter O et al. Molecular characterisation of soft tissue tumours: a gene expression study. Lancet 2002 Apr 13;359(9314):1301-7. PMID: 11965276

3. https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15750-s20/www/notebooks/SVD-irises-clustering.html

4. Mahoney, M. W., Maggioni, M., & Drineas, P. (2006). Tensor-CUR decompositions for tensor-based data. *Proceedings of the 12th Annual ACM SIGKDD Conference*, Association for Computing Machinery, New York, pp. 327–336.

5. Paschou, P., et al. (2007). Intra- and interpopulation genotype reconstruction from tagging SNPs. *Genome Research*, 17, 96–107.

6. Drineas, P., Mahoney, M. W., & Muthukrishnan, S. (2008). Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30, 844–881.

7. Stewart, G. W., & Sun, J. G. (1990). *Matrix Perturbation Theory.* Academic Press, New York.

8. Nashed, M. Z. (Ed.). (1976). *Generalized Inverses and Applications.* Academic Press, New York.

9. Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York.

10. Hoaglin, D. C., & Welsch, R. E. (1978). The hat matrix in regression and ANOVA. *The American Statistician*, 32, 17–22.

11. Chatterjee, S., & Hadi, A. S. (1988). *Sensitivity Analysis in Linear Regression.* Wiley, New York.

12. Chatterjee, S., & Hadi, A. S. (1986). Influential observations, high leverage points, and outliers in linear regression. *Statistical Science*, 1, 379–393.

13. Velleman, P. F., & Welsch, R. E. (1981). Efficient computing of regression diagnostics. *The American Statistician*, 35, 234–242.