TIG CUR Decomposition Challenge

TIG Labs

January, 2026

1 Introduction

In the era of big data, analysts and practitioners routinely work with matrices containing tens or even hundreds of thousands of rows and columns. This scale is computationally taxing; however, the data can contain redundancy. Isolating the informative low-dimensional structure has therefore become a critical first step in any analysis. Matrix factorizations-—and, in particular, low-rank decompositions and low-rank approximation-—sit at the heart of modern data science and numerical analysis, providing the theoretical backbone for everything from principal-component analysis and recommender systems to fast solvers in scientific computing. Within this landscape, CUR approximations have gained popularity because they preserve the actual rows and columns of the original matrix, yielding interpretable, sparsity-aware low rank approximations. CUR approximations have been used in a variety of areas including genetic data [1], imaging [2], and recommender systems [3].

2 Problem Description

Given an $m \times n$ data matrix A, a *low–rank CUR approximation* approximates A by

$$A \approx C\,U\,R,$$

where

- $C \in \mathbb{R}^{m\times c}$ contains c **actual columns** of A;
- $R \in \mathbb{R}^{r\times n}$ contains r **actual rows** of A;
- $U \in \mathbb{R}^{c\times r}$ is called the linking matrix.

The rank of the CUR approximation is the rank of CUR.

3 The TIG Challenge

Currently there are constants l_i which still need to be fixed.

The TIG version of the CUR approximation problem is summarised in three steps:

- **Step 1. Generating.** Given the difficulty parameters m, n (outlined in Section 5) a rank k is fixed as $k = \lfloor \min\{m, n\}/2 \rfloor$, l_1 matrices $A_1, \dots A_{l_1}$ are generated (by the procedure outlined in Section 4), they are all rank k.
- **Step 2. Solving.** Three target ranks are chosen as target_ranks $= [\frac{k}{l_2}, \frac{k}{l_3}, \frac{k}{l_4}]$. The TIG framework loops over all l_1 matrices and all target_ranks, at each iteration $i \in \{1 \dots, l_1\}$, target_rank $\in$ target_ranks the Innovators submission algorithm is called :

  $$CUR_{i,\text{target\_rank}} \leftarrow \text{Innovator\_Submission}(A_i, \text{target\_rank})$$

  returning a CUR approximation of A_i with at most target_rank columns and target_rank rows.

- **Step 3. Verification.** For each $i \in \{1, 2, \ldots, l_1\}$, we score the CUR approximation (with rank target_rank) as

$$\frac{\kappa\|A_i - SVD_{i,\text{target\_rank}}\|_F - \|A_i - CUR_{i,\text{target\_rank}}\|_F}{(\kappa - 1)\|A_i - SVD_{i,\text{target\_rank}}\|_F}$$

where $SVD_{i,\text{target\_rank}}$ is the target_rank SVD approximation to A_i. Note that this is of the form

$$\frac{\text{baseline} - \text{solution}}{\text{baseline} - \text{optimal}} \tag{1}$$

with baseline $= \kappa \cdot$optimal. This a natural way to asses the quality of the solution, if it is optimal we get a score of 1, if the solution is the same as the baseline we get a score of 0. We will need to calibrate $\kappa > 0$.

We take the geometric mean of the scores

$$\text{geom\_mean}_i = \left(\prod_{\text{target\_rank}} \frac{\kappa\|A_i - SVD_{i,\text{target\_rank}}\|_F - \|A_i - CUR_{i,\text{target\_rank}}\|_F}{(\kappa - 1)\|A_i - SVD_{i,\text{target\_rank}}\|_F} \right)^{1/3},$$

We used geometric means over the target ranks since we note that the relative error increases as the target rank is increased. The geometric means are then averaged for the final quality score of the instance :

$$\text{quality} = \frac{1}{l_1} \sum_{i=1}^{l_1} \text{geom\_mean}_i \tag{2}$$

The asymmetric nature of the verification is discussed in Section 6.

4 Random Instance Generation

TIGs instance generation procedure generates matrices in $\mathbb{R}^{m \times n}$ with low k-rank structure. It does this by constructing the matrices through a k-rank SVD decomposition.

For a single instance of the TIG CUR challenge we need to generate l_1 matrices, each of size $m \times n$ and rank k. This is done as follows

1. Sample $G_U \in \mathbb{R}^{m \times k}$ as a Gaussian matrix (random matrix taking iid $N(0, 1)$ entries), similarly sample $G_V \in \mathbb{R}^{n \times k}$. Scale the columns of the random matrices $G_U D$ and $G_V D$, for a diagonal matrix D with entries d_j such that $\max_j d_j = \Delta$ and $\min_j d_j = 1$, with the other d_j linearly decaying diagonal entries from Δ to 1[1].

2. Generate $U \in \mathbb{R}^{m \times k}, V \in \mathbb{R}^{n \times k}$ via QR decomposition $G_U D = UR$ and $G_V D = VR$.

3. We generate k singular values as follows : for $j \in \{1, \ldots, k\}$, $\sigma(j)$ is set to be

$$\sigma(j) = \exp\left(-\frac{l_5\sqrt{j}}{\sqrt{k}} \right) \tag{3}$$

[1]This allows us to control the so-called coherence [4],to tune the problem's difficulty. The coherence is defined as the largest leverage score; the leverage scores are the squared row norms of the leading (in this case k) singular vectors. Incoherent matrices (e.g. randomly generated U, V) makes the subset selection easy. On the other hand, coherent matrices are considered difficult as finding a good set of indices is challenging; however, the benefit of doing so is enormous.

Note I stuck in a $\sqrt{k}$ to stop the majority of σ being 0 as k increased. Then we take l_1 random relabalings of the singular values : for $i \in \{1, \ldots, l_1\}$ let π_i be a permutation on $\{1, \ldots, k\}$, set $\sigma_i(j) = \sigma_{\pi_i(j)}$. This gives l_1 matrices $\Sigma_i = \text{diag}(\sigma_i)$ of singular values.

4. Generate l_1 matrices A_i by singular value decompositions

$$A_i = U\Sigma_i V^T. \tag{4}$$

5 Challenge Tracks

Within each challenge, there are various challenges tracks. These can range over instance size and/or type. Currently, the challenge supports varying sizes of matrices generated in the instances. We use the following tracks:

- **Track 1:**
- **Track 2:**
- **Track 3:**
- **Track 4:**
- **Track 5:**

The coherence Δ is fixed but could be varied in the future to adapt the difficulty. Moreover we could introduce sparsity too. As well as the rank in proportion to the matrix size.

6 Asymmetry Characteristics

To maintain the asymmetry characteristics essential to the TIG protocol the time an algorithm takes to solve an instance must be significantly longer than the time it takes to generate the instance plus the time it takes to verify a solution (i.e. compute the error (2)) :

- For the instance generation, we construct l_1 matrices relatively quickly: we only compute U, V once, and we do the matrix multiplications (4) l_1 times.

- The calculation of the error (2) is straightforward, for a given target_rank, the error $\|A_i - SVD_{i,\text{target\_rank}}\|_F$ is given by $\left(\sum_{j=1+\text{target\_rank}}^{k} \sigma_j^2\right)^{1/2}$, where note the sum is from $j = 1 + \text{target\_rank}$ to $j = k$ and the σ_j are those generated in (3) (before any permutation is done).

References

[1] András Bodor, István Csabai, Michael W Mahoney, and Norbert Solymosi. rcur: an r package for cur matrix decomposition. *BMC bioinformatics*, 13(1):103, 2012.

[2] Muhammad AA Abdelgawad, Ray CC Cheung, and Hong Yan. Efficient blind hyperspectral unmixing framework based on cur decomposition (cur-hu). *Remote Sensing*, 16(5):766, 2024.

[3] Michael W Mahoney, Mauro Maggioni, and Petros Drineas. Tensor-cur decompositions for tensor-based data. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 327–336, 2006.

[4] Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, and David P Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13(1):3475–3506, 2012.